SESSION

GENETIC + EVOLUTIONAY ALGORITHMS

Chair(s)

TBA

Exploring Inevitable Convergence for a Genetic Algorithm Persistent FPGA Placer

Peter Jamieson
Miami University, Oxford, OH, 45056
Email: jamiespa@muohio.edu

Abstract— *Persistent CAD algorithms offer the potential to optimize power consumption for programmable chips post development and deployment. The basic idea is that algorithms continue to search for better design solutions and as better solutions are found, the design is deployed to programmable chips resulting in better performance. In this work, we further study a persistent placement algorithm for FPGAs and investigate a number of algorithm improvements attempting to delay premature convergence. Our results show that these techniques create some divergence in the solutions, but in all cases what we call "inevitable convergence occurs" in the first 2 hours of execution.*

Keywords: GA, Placement, FPGA, Persistent

1. Introduction

Field-Programmable Gate Arrays (FPGAs) are programmable Integrated Chips (ICs) that continue to gain popularity due to the challenges and costs associated with creating an Application-Specific Integrated Circuit (ASIC). One aspect of the FPGA, the programmability, means that these devices, similar to general purpose processors, can be reprogrammed. This means that even when the chip is deployed in the field, with the appropriate functionality, these devices can be updated with bug fixes, new designs, and as this work examines, more efficient designs. The last of these updates is part of what we call persistent Computer Aided Design (CAD) where optimization algorithms, which map designs to FPGAs, are run post chip deployment to try and find more optimal implementations of the design. These more optimal implementations focus on improving the power consumption of the FPGA.

We first introduced a persistent genetic algorithm (GA) for placement on FPGAs in [1]. This work showed how a genetic algorithm finds improved solutions early and the rate of these improvements slows as time increases. Our window of measurement for a relatively small benchmark was only one hour. In this work, we increase the time window of observation and use a variety of modifications to the GA to attempt to maintain diversity in the population and to delay what we call "inevitable convergence". Inevitable convergence, like premature convergence [2] [3], is the lack of diversity in a population such that new offspring are not sufficiently diverse, therefore, resulting in suboptimal solutions. In the case of persistent CAD, convergence is inevitable and is likely suboptimal. The goal, therefore, is to identify convergence (or measure population diversity [4], [5]) as well as avoid convergence. In this work we attempt techniques to deal with the later and leave the identification of convergence as future work.

To study inevitable convergence for persistent placement we implement the following algorithmic variations:

- Parallel algorithmic threads [6]
- A partial mapped crossover breeding operator [3]
- A partial mapped crossover breeding operator with mutations

Our results show that these techniques all tend to converge in about 2 hours. In all cases, when compared against the baseline random solution it is evident that the large gap between random solutions and these techniques means that it is highly unlikely that they will ever leave their local minimums.

The remainder of this paper is organized as follows. Section 2 briefly describes FPGAs, CAD for FPGAs, and the persistent FPGA placement problem. Section 3 describes our implementation of genetic algorithm placer. Section 4 describes our experimental setup and shows results for one MCNC benchmarks. Finally, Section 5 concludes this work.

2. Background

FPGAs are programmable ICs that can implement any digital design. These devices consist of programmable logic blocks and a programmable routing [7] where the programmable routing consists of wire segments that are connected to either logic blocks or other wire segments via programmable switches. The logic blocks are also called clusters (which is the term we will use throughout this work) where these clusters commonly consist of a combination of Look-up Tables (LUTs), flip-flops, and internal programmable routing. The most important aspect of this architecture for the placement problem is the cluster, and the placement algorithm maps design clusters onto the FPGA, which, itself, consists of an array of these clusters.

Our open source CAD flow used by VPR 5.0 [8], which is an academic FPGA tool that allows us to experimentally test algorithms and FPGA architectures consists of Odin II [9] (high-level synthesis), ABC [10] (logic optimization

and technology mapping) and tv-pack [11] (clustering). First, a digital design is created in Verilog HDL [12] and used as the input to this CAD flow, and a series of CAD flow stages convert the design to a programmable bit-stream that can be uploaded to the FPGA to implement the digital design.

This work focuses on the FPGA placement step, which maps the design clusters onto the FPGA. This is the second last stage and is implemented in VPR 5.0. VPR 5.0, originally, used simulated annealing (SA) for this placement, and we have implemented a genetic algorithm within this software framework.

2.1 Details of FPGA Placement

FPGA placement algorithms try to place the clusters, representing the digital design, onto the array of FPGA clusters such that the critical path (the longest path from either a primary input to a primary output, a primary input to flip-flop, flip-flop to flip-flop, or flip-flop to primary output) is minimized, the power consumption of the programmable routing is minimized, and the overall wire-length of the mapped circuit is minimized. This problem has been shown to be NP-complete to solve optimally, and a number of popular algorithms have been used to solve this problem including simulated annealing ([13], [7]), which is the algorithm used in VPR 5.0 [8], min-cut ([14], [15], [16]), analytic ([17], [18]), and genetic algorithms (GAs) [19], [20], and [21].

We focus on SA and GA algorithms in this work since our GA is built off the SA in VPR. The SA uses a cooling schedule to control the acceptance of randomly selected swaps between clusters on an FPGA. Each swap of clusters will either improve or degrade the critical path (as well as other metrics), and initially, all swaps are accepted regardless if they improve the optimizations metrics or not. As the temperature cools, only swaps that improve the critical path are accepted. In this way, the early phases of the cooling schedule is used to allow hill climbing that will, hopefully, avoid local minimums in this optimization problem [7].

The two most relevant aspects of the annealer as a placement algorithm for FPGAs are the scheduling of the cooling and the cost function. The scheduling of the annealer determines if a random swap is accepted and determines the maximum Manhattan distance of the cluster swaps. As the algorithm continues, swapping of clusters that don't improve the cost function are not accepted, and the distance between the swaps is reduced.

The distance of a random swap of clusters on a X by Y array is based on the term R_{limit}. Given a 5 by 5 FPGA, R_{limit} can have a maximum value of 5 meaning that a cluster located at the x coordinate 0 and y coordinate 0 could be swapped with another cluster located at x coordinate 4 and y coordinate 4. As R_{limit} is reduced by the annealer's scheduler, the distance for a swap is reduced, and this represents the stabilizing of the placement algorithm (the cooling and lower excitation of the molecules in a metal).

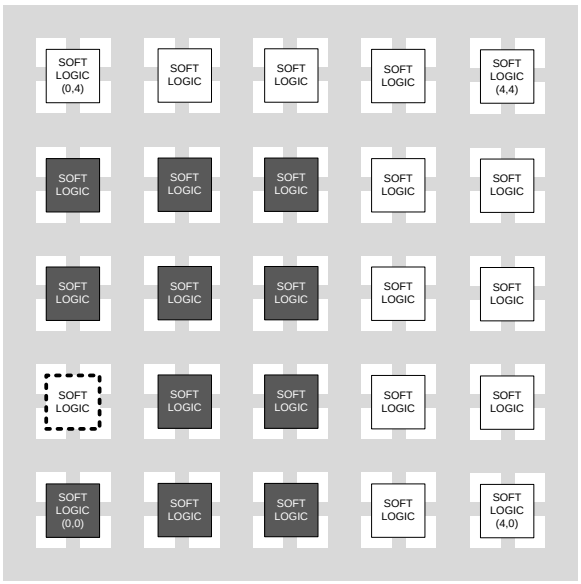


Figure 1: Shows how the R_{limit} factor affects the distance of swaps

For example, Figure 1 shows a 5 by 5 FPGA where random swaps could happen for the digital logic at x = 0, y = 1 with an $R_{limit} = 2$ (The candidate cluster to swap is surrounded by a thick dotted line, and the clusters it can swap with are shaded in darker grey).

The second aspect of the annealer is the cost function used estimate the quality of the placement. The cost function for SA in VPR 5.0 with power [22] consists of three components defined in [7]. First is the sum of the bounding box dimensions of all nets which estimates the total amount of wire needed to implement the circuit (also know as wire-length). Given N nets, $bb_x(i)$ and $bb_y(i)$ are the x and y dimensions of a bounding box for each $net(i)$, and $q(i)$ as a scaling factor for better wire-length estimates, then the first component of the cost function is defined as:

$$WiringCost = \sum_{i=1}^{N} q(i) \cdot [bb_x(i) + bb_y(i)] \qquad (1)$$

The second component of the cost function evaluates the timing cost of a placement where,

$$TimingCost = \sum_{\forall i,j \in circuit} Delay(i,j) \cdot Criticality(i,j)^{(CE)} \qquad (2)$$

where CE is a constant, Delay(i,j) is the delay of the connection from source i to sink j, and Criticality(i,j) is a measure of how close the given i, j path is to the global critical path. The power component is defined as:

$$PowerCost = \sum_{i=1}^{N} q(i) \cdot [bb_x(i) + bb_y(i)] \cdot Activity(i) \qquad (3)$$

where $Activity(i)$ is the switching activity on a particular net, and by reducing this component, the power consumed over long and power hungry programmable routing lines is reduced. The new cost function with this component is the following:

The perceived change in the cost function for each placement change is:

$$Cost = \lambda \cdot \frac{TimingCost}{PreviousTimingCost} + (1-\lambda) \cdot \left[(1-\gamma) \cdot \frac{WiringCost}{PreviousWiringCost} + \gamma \cdot \frac{PowerCost}{PreviousPowerCost} \right] \quad (4)$$

where the previous costs are used to normalize the two components of the cost function, and the λ parameter is used to weight the optimization importance of each of the two components and the γ factor is used to control the importance of the power optimization component.

As described, the parameters γ and λ are used to control the weighting of the cost function, or how much the cost function cares about optimizing for a particular metric. Previous research has shown that for a cost function that attempts to optimize for power has a γ equal to 0.8 and a λ equal to 0.5 [23]. In our previous research [21] we confirm that these parameters are also suitable for our GA.

3. GA Framework and Modifications for FPGA Placement

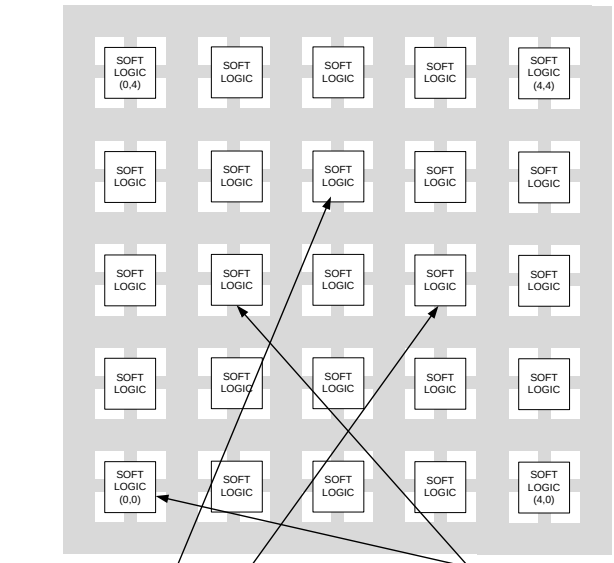
We previously built our genetic algorithm FPGA placer that includes power optimizations and describe the details of this algorithm here [21]. In this section we review some of these details, but do not cover all the details. In particular, we will describe genetic strings, the mutation operator, the crossover operator, and the parameters in the GA framework.

3.1 The Genome for Placement

Genetic algorithms (GA) and evolutionary programming algorithms have been previously implemented and explored for FPGA placement. We use a genome similar to the implementation by Venkatraman *et. al.* [19] in which they implemented a GA based placer in VPR 4.3 (the predecessor to VPR 5.0). In their work, each cluster's location on the FPGA array is a gene, and the 2-D location of each of the clusters forms an individuals genome. Figure 2 shows how a genome for a design consisting of 20 elements is represented.

3.2 Parameters in our GA Framework

Similar to other GA implementations of FPGA placement, our GA placement algorithm framework creates a genome based on the x and y coordinates of each cluster in the design (see Figure 2). In addition to how the genome is represented, we define a number of parameters within the framework



Design Logic	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Location	(0,1)	(2,3)	(4,1)	(4,2)	(0,4)	(2,4)	(3,1)	(3,0)	(3,2)	(0,2)	(0,3)	(1,2)	(1,4)	(4,0)	(0,0)	(2,2)	(1,1)

Figure 2: Sample genome for 20 elements on a 5x5 FPGA

that control the GA. The size of a population is defined by σ. Using this number we define the parameters ω, α, and β as percentages where $\omega + \alpha + \beta = 100\%$, and $\omega\%$ of the population is the number of individuals from the current generation to maintain as parents in the next generation, $\alpha\%$ of the population are the children of the parents, and $\beta\%$ of the population is randomly created new individuals.

Our GA measures an individuals fitness based on the cost function shown previously in equation 4 and both λ and γ control the weighting of this cost function.

3.3 Mutation Operator

One of the operators in our GA framework uses a mutation operator to create new individuals in a population. Random swaps of clusters on the FPGA are the mutation operations for our GA framework and this is similar to how the SA works, and therefore, this mutation is related to the term R_{limit}, which controls the distance between clusters for a random swap. The number of mutations per new individual is defined by the parameters $local\_swaps\%$ and $global\_swaps\%$ where $local\_swaps\%$ multiplied by the number of clusters in a circuit defines the number of mutations (or swaps) to try where R_{limit} is equal to one, and $global\_swaps\%$ multiplied by the number of clusters in a circuit is the number of mutations/swaps to try where R_{limit} is scheduled to be between 1 and the maximum size of the FPGA array in one dimension.

R_{limit} is a parameter that changes over time. In the SA algorithm the parameter is decreased when a current set of swaps does not result in any improvement. In our GA, we

Table 1: Configurable parameters for the GA

Parameter	Description of parameter
ω	The percentage of the fittest individuals in the population to use as parents
α	The percentage of the population created from the fittest individuals
β	The percentage of the population that is randomly created
σ	The number of individuals in the population
R_limit	The distance between swaps on the FPGA array
$global\_swaps$	The percentage of the number of clusters that defines the number of global mutations for a new individual
$local\_swaps$	The percentage of the number of clusters that defines the number of local mutations for a new individual
λ	A cost function parameter to weight timing optimization importance
γ	A cost function parameter to weight power optimization importance

also schedule R_{limit} in a similar fashion, except that our algorithm is a persistent algorithm. Therefore, once R_{limit} equals 1 we then reset it to the maximum size of the FPGA array in one dimension.

3.4 Crossover Operator

The crossover operator within our GA framework is the partially mapped crossover (PMX) originally proposed by Goldberg for the traveling salesman problem [3]. This operator fits well with our placement algorithm since our string of clusters requires that each cluster appears only once in the genome string.

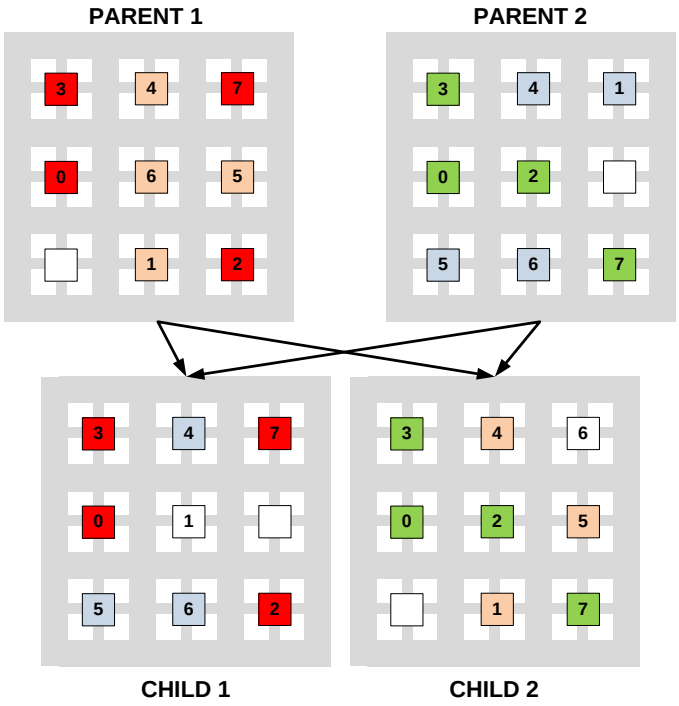


Figure 3: Sample PMX mutation

Figure 3 shows a sample crossover mutation for 2 parents generating 2 children. The figure has been color coded to show how the parent genes are crossed for the example picked genes 0, 2, 3, and 7. Note how in child 1 the gene 1 is not colored and child 2 the gene 6 is not color coded. In both of these instances, these gene locations are mapped by a series of remappings that the PMX operator achieves using a remapping list.

In our current implementation of the GA framework, two parents are chosen at random from the most fit percent of the population as specified by parameter ω to generate 2 children as part of the α new population. Within the genome, 50% of the clusters are randomly selected to stay constant from parent 1 to child 1 and parent 2 to child 2, respectively (in Figure 3 these clusters are in red. Then the PMX mapping is done on the remaining clusters to map parent 2's clusters to child 1 and parent 1's clusters to child 2 (as seen in Figure 3 in the orange and blue squares). The random swapping of components is not necessarily the best choice, and this is an area to study in the future.

4. Algorithmic Modification Results

To observe how different modifications to our persistent genetic algorithm for placement helps delay inevitable convergence we run the following experiment.

To attempt to maintain diversity we test the following modifications:

- Parallel algorithmic threads [6] - we introduce 4 threads initialized by different random seeds and execute these threads in parallel
- A partial mapped crossover breeding operator [3] - we use the PMX crossover to generate new individuals
- A partial mapped crossover breeding operator with mutations - we use the PMX crossover to generate new individuals and mutate these new individuals

Note that the β parameter is also a diversity factor, but this factor was already introduced in our first study of persistent placement with genetic algorithms, and it had very little impact. We do, however, maintain a 10% value for this parameter in our experiments.

Table 2 shows the parameters for our experiments. The first column shows the parameter, the second column shows the parameters value for the parallel GA threads, and column three shows the parameter values for the PMX crossover with

Table 3: The FPGA architectural parameters

Parameter	W	N	K	F_{cin}	F_{cout}	F_s	routing	transistor sizing
Value	144	10	5	0.18	0.1	3	uni-directional	27mwt

Table 2: Parameters for the GA parallel threads and Crossover

Parameter	Values for GA thread	Values for GA crossover
σ	200	200
ω	20% of σ	20% of σ
α	70% of σ	70% of σ
β	10% of σ	10% of σ
R_limit	Variable	Variable
$global\_swaps$	30%	30% and 0%*
$local\_swaps$	0%	0%
PMX	No	Yes
λ	0.5	0.5
γ	0.8	0.8

and without (* in table) mutations. Note that λ and γ have been experimentally determined based on out previous work.

The FPGA architectural parameters that describe the FPGA we are mapping to are shown in Table 3. For a more detailed explanation of these parameters please consult [7], but for the sake of space and unnecessary details, we do not describe these parameters here. The transistor size for these experiments is based on our results in [22]. For FPGA architects, note that W (channel width) is fixed to 144. This is done since we are only using one benchmark and the constant value considerably increases the speed of the routing algorithm. Instead of performing a binary search and increasing W by 20% (as is normally done in CAD experiments) we believe this fixed sized W is reasonable for this experiment considering that persistent CAD will, in reality, have W as a fixed parameter.

These experiments are run using the largest MCNC benchmarks [24] where these benchmarks have been converted to a netlist of clusters using an academic CAD flow. This benchmark, clma, is passed into VPR 5.0 for the same FPGA as described in Table 3. VPR 5.0 outputs the current best placement every 30 minutes and executes for 2 days. Once the algorithm is done, we then use these output placement files and run them through VPR's routing algorithm to find the final speed and power consumption metrics, which we report in the next section.

All the variations in the algorithms are run in Linux on Intel Xeon 2.4 Ghz cores.

4.1 Results

Figure 4 shows the results for our experiment. The x-axis shows the time over 2 days where we sampled the progress of the persistent placement algorithms every 30 minutes. On the y-axis, energy consumption is shown in terms of joules per clock. This metric is not truly reflective of instantaneous power consumption, but at present we do not have the capability to set the critical path in VPR 5.0.

In the figure, the upper line reflects the current best random placement result. We use this as a baseline noting that there is a significant difference between a random solution and the GA versions. Next, the orange line stabilizes after the first 30 minutes is the genetic algorithm with crossover and no mutation. Interestingly enough, in the first 30 minutes, diversity is completely eliminated since there are no mutations to introduce diversity back into the population. Depending on the random seed this happens at a different point.

The next collection of results are for the parallel threads. First, note that the lines (particularly "ga seed 99") sometimes seem to increase in energy consumption over the persistent exploration. The reason for this is due to the estimation models used to calculate the cost or fitness function at the placement level. These models are not as accurate as the models used after performing a complete placement and routing of a design (which are the results reported in this graph). In future work, it might be valuable to perform what we call a deep fitness evaluation at intervals to confirm population fitness at the placement level.

In general, the parallel approach shows that each thread finds solutions that are all in the similar energy range, but each thread itself seems to have converged relatively early in the search. Finally, the crossover and mutation operator performs similar to each of the parallel threads. The population eventually becomes dedicated towards a local minimum and cannot exit this area.

4.2 Discussion

A recent research paper by Mingjie and Wawrzynek [25] looked at the concept of tunneling to low energy (local optimal points) in SA for FPGA placement. Their claim is that once an SA finds a low energy area, no hill climbing technique can escape the localized search space. In our examples, the parallel threads and crossover mutations are in these low energy regions when we compare them relative to the random results, and our tunneling capabilities are essentially different start points (seeds).

To further persistent CAD for placement and maintain diversity longer, we will need to address the second part of the inevitable convergence problem, which is to identify when a population has converged. With this capability we can then explore if island model [26] for GAs or another solution will better suit are persistent CAD.

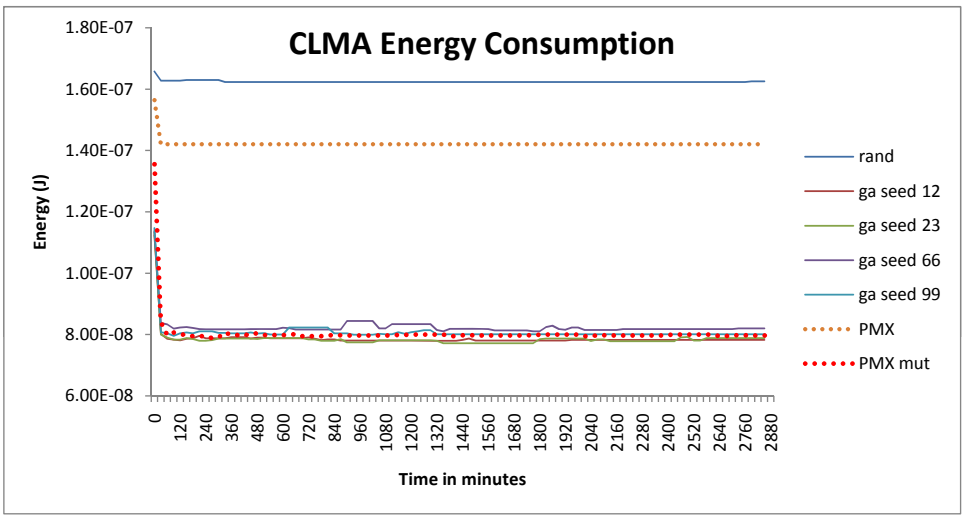


Figure 4: Energy consumption of CLMA benchmark over a 2 day execution

5. Conclusion

In this paper we explored various techniques for maintaining a divergent population in GA for persistent FPGA placement. Our results show that adding both parallel threads of GA populations and including a PMX crossover operator do not significantly impact the diversity of the population in this domain as in all cases the populations converge in a low energy search space. In future, we hope to take these techniques and incorporate them into a more complex system that will maintain divergence by searching in multiple islands of solutions and then use our techniques to mix these populations.

References

[1] P. Jamieson, "Persistent cad for in-the-field power optimization," in *ERSA*, 2010, pp. 267–270.

[2] M. Rocha and J. Neves, "Preventing premature convergence to local optima in genetic algorithms via random offspring generation," in *Proceedings of the 12th international conference on Industrial and engineering applications of artificial intelligence and expert systems: multiple approaches to intelligent systems*, 1999, pp. 127–136. [Online]. Available: http://portal.acm.org/citation.cfm?id=341506.341546

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Wesley Professional, January 1989. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0201157675

[4] Y. Leung, Y. Gao, and Z. Xu, "Degree of population diversity - a perspective on premature convergence in genetic algorithms and its markov chain analysis," *IEEE Transactions on Neural Networks*, vol. 8, pp. 1165–1176, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.3087

[5] E. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: an analysis of measures and correlation with fitness," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 1, pp. 47 – 62, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1266373

[6] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, no. 6-7, pp. 619 – 632, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/B6V12-4GMBN2V-2/2/e78d51ea3dd47a6e6be1e93a23edead6

[7] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[8] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, and J. Rose, "VPR 5.0: FPGA CAD and Architecture xploration Tools with Single-Driver Routing, Heterogeneity and Process Scaling," in *ACM/SIGDA International Symposium on FPGAs*, Feb 2009. [Online]. Available: http://doi.acm.org/10.1145/1508128.1508150

[9] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, "Odin II - An Open-source Verilog HDL Synthesis tool for CAD Research," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, 2010, pp. 149–156. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/FCCM.2010.31

[10] A. Mishchenko, S. Chatterjee, and R. K. Brayton, "Improvements to technology mapping for LUT-based FPGAs," *IEEE Transactions on CAD*, vol. 26, no. 2, pp. 240–253, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.68.9003

[11] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," in *ACM/SIGDA International Symposium on FPGAs*, Monterey, CA, 1999, pp. 37–46. [Online]. Available: http://doi.acm.org/10.1145/296399.296426

[12] *Verilog Hardware Description Reference*, Open Verilog International, March 1993.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[14] J. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," *IEEE Transactions on Computer-Aided Design*, vol. 10, no. 3, pp. 356–365, Mar. 1991.

[15] D. Huang and A. Khang, "Partitioning-Based Standard-cell global placement with an Exact Objective," in *International Symposium on Physical Design*, Napa Valley, CA, 1997, pp. 18–25.

[16] A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, vol. 4, no. 1, pp. 92–98, Jan. 1985.

[17] B. M. Riess and G. G. Ettelt, "Speed: Fast and Efficient Timing Driven Placement," in *IEEE International Symposium on Circuits*, 1995, pp. 377–380.

[18] K. Vorwerk, A. Kennings, and A. Vannelli, "Engineering details of a

stable force-directed placer," in *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, 2004, pp. 573–580.

[19] R. Venkatraman and L. M. Patnaik, "An evolutionary approach to timing driven fpga placement," in *GLSVLSI '00: Proceedings of the 10th Great Lakes symposium on VLSI*, 2000, pp. 81–85.

[20] Y. Meng, A. E. A. Almaini, and W. Pengjun, "Fpga placement optimization by two-step unified genetic algorithm and simulated annealing algorithm," *Journal of Electronics (China)*, vol. 23, no. 4, pp. 632–636, 2007.

[21] P. Jamieson, "Revisiting genetic algorithms for the fpga placement problem," in *GEM*, 2010, pp. 16–22.

[22] P. Jamieson, W. Luk, S. J. Wilton, and G. A. Constantinides, "An energy and power consumption analysis of fpga routing architectures," in *International Conference on Field-Programmable Technology*, 2009, pp. 324–327. [Online]. Available: http://www.users.muohio.edu/jamiespa/html_papers/fpt_09.pdf

[23] J. Lamoureux and S. J. E. Wilton, "On the interaction between power-aware fpga cad algorithms," in *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, 2003, p. 701.

[24] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," 1991, tech. Report. Microelectronics Centre of North Carolina. P.O. Box 12889, Research Triangle Park, NC 27709 USA.

[25] M. Lin and J. Wawrzynek, "Improving fpga placement with dynamically adaptive stochastic tunneling," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 12, pp. 1858 –1869, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5621041&tag=1

[26] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evol. Comput.*, vol. 1, pp. 127–149, June 1993. [Online]. Available: http://dx.doi.org/10.1162/evco.1993.1.2.127

Using Genetic Algorithms for Subset Selection for Partial Fault Tolerance in Reconfigurable Logic

David L. Foster
Electrical and Computer Engineering Department, Kettering University, Flint, MI, USA

Abstract- *As field programmable gate arrays and other reconfigurable logic devices are increasingly used for aerospace and terrestrial applications, fault tolerance methodologies have been developed to improve reliability. Applying fault tolerance to an entire design may incur prohibitive area and energy penalties, and a need exists for techniques that can trade fault tolerance for lower area penalties. However, selecting a circuit subset that minimizes both the area overhead and the vulnerability to faults remains an open topic. Area-Constrained Partial Fault Tolerance (ACPFT) is a unique approach that explicitly accepts the device resources as an input and attempts to find a maximally fault-tolerant subset within this constraint. While previous options in ACPFT determined this subset from different heuristics, this paper presents an extension called ACPFT-GA that uses genetic algorithms for subset selection. Testing shows an improvement of up to 31.92% more coverage than the previous version.*

Keywords: Genetic algorithms, partial fault tolerance, reconfigurable logic, field programmable gate array

1 Introduction

Genetic algorithms are an established approach to solving difficult optimization problems. This makes them an ideal candidate for applying partial fault-tolerance to field programmable gate arrays (FPGA). An FPGA is a reconfigurable logic device that has the flexibility of a general purpose processor and can approach the computational power of an application-specific integrated circuit (ASIC). They are also off-the-shelf devices and therefore do not have the lead times of ASICs. They are often used in aerospace applications and are increasing used in terrestrial systems. As reconfigurable devices, they may contain millions of bits to store their current configurations. This makes them more susceptible to faults caused by electromagnetic radiation than other types of devices. Errors due to single event upsets (SEUs) are already a concern in aerospace, and as transistor feature sizes continue to shrink, they are becoming increasingly important in terrestrial uses[1, 2].

Many applications implemented with FPGAs are not safety-critical. Thus, failures can be tolerated, but reducing the failure rate would certainly be beneficial. Currently, a circuit design targets an off-the-shelf FPGA containing at least enough cells to implement the circuit. Since the circuit does not fully occupy the FPGA, some logic cells will be unallocated. These unused resources can be leveraged to provide redundancy and reduce the failure rate.

The problem of applying partial fault-tolerance can then be formulated as follows. The circuit's logic cells must be partitioned into a protected and a non-protected subset such that fault-coverage is maximized given a limited amount of additional logic cells. With circuits containing thousands of logic cells, this presents an incredible challenge with an enormous solution space.

The author has previously proposed Area-Constrained Partial Fault Tolerance (ACPFT) as a solution based on triple modular redundancy [3]. This paper presents an extension to that work called ACPFT with Genetic Algorithms (ACPFT-GA) that generates the protected subset using the well-known evolutionary functions of mutation, crossover, and selection.

This paper is organized as follows. Section 2 reviews existing methods for applying partial fault-tolerance to FPGAs. Section 3 describes how ACPFT is cast as a genetic algorithm. Section 4 presents the experimental results, and Section 5 concludes the paper.

2 Related Work

Triple modular redundancy (TMR) remains the standard fault-tolerance method for FPGAs [4]. It is universally applicable and typically adds only a small delay. Circuits protected by TMR are almost completely protected against a single fault, although voting logic may still be susceptible. However, it more than triples the circuit's size with a corresponding increase in power use. [5]. TMR is often the basis for partial fault-tolerance.

A method called partial error masking is presented in [6]. Based on TMR, it only applies to circuits implemented with discrete logic gates. It uses a consistency check reduction that does not give precise control of the area overhead.

The BYU-LANL Partial TMR Tool applies TMR in order of logic cell function in a course-grained approach [7]. This tool first protects feedback logic, logic feeding the feedback structures, then remaining logic.

Selective TMR (STMR) uses a sensitivity threshold P_{th} to partition a circuit into sensitive and non-sensitive subsets. [8]. TMR is applied to the sensitive subset, and reported results show that there is no clear relationship between P_{thr} and the resulting area or level of fault mitigation. Reduced TMR (RTMR) is similar to STMR, but it focuses strictly on look-up table-based designs [9].

What could be considered a form of an evolutionary algorithm, automatic insertion of Partial TMR applies TMR to a subset of circuit registers [10]. It initially separates the registers into a protected and an unprotected set, implements and simulates the design to measure fault-mitigation, then updates the subset according to the Fiduccia-Mattheyses algorithm. It is computationally intense and can lead to an unacceptable amount of time even for a design with only 54 flip-flops.

Area-Constrained Partial Fault Tolerance (ACPFT) applies partial TMR and/or concurrent error detection (CED) to a net list [3]. One noteworthy consideration is that majority voters are not considered ideal and are included in the sensitive section of the device. ACPFT generates a cell priority for each logic cell, and it protects cells according to their priorities until extra resources are exhausted. Several priority methods, such as cell fan-out and logic depth, are reported in [11]. Several heuristics have also been tested, such as backtracking from circuit outputs, a course-grained method that creates pockets of protected cells around highly-prioritized seed cells, and minimizing voters inserted into a propagation path.

Since ACPFT is the basis for ACPFT-GA, it will be helpful to understand its basic principles in more detail. In many fault-tolerant methods using TMR, the majority voters are considered ideal in that they do not experience faults. This assumption is used for simplification and because when a large subset of the circuit is being protected, the cross-sectional area of the majority voters is significantly smaller than the tripled area, perhaps by several orders of magnitude. However, in a fine-grained approach when the protected area has a comparable area to the majority voter, this assumption is completely invalid. In fact, applying TMR to small subsets is useless, since the added sensitivity of the majority voter nullifies the added protection.

ACPFT avoids this assumption by modifying how TMR is applied to a small subset. Instead of tripling only the target cell and applying a majority voter to the three cells' outputs, it triples the target cell and triples all cells that use its outputs. These output cells are then connected to majority voters. This seems to impose an extremely large area overhead for the reduction of a single sensitive cell. However, protecting cells that are connected to this subset is more efficient since some of the logic has already been tripled. Because of this interaction, selecting the subset of cells to protect is challenging and crucial to maximizing the fault-tolerance.

ACPFT seems suitable for a genetic algorithm since its partitioning problem is similar to the well-known knapsack problem. In knapsack, there is a set of items, each with a weight and a value, and a knapsack that can hold a fixed weight. The combinatorial problem is to select a subset of items that can be carried in the knapsack with the maximum total value. The additional area of the FPGA relates to the knapsack, and the logic cells and their areas and sensitivities relate to items with weight and value. However, the fault-tolerance problem is more complicated since the additional area required by each logic cell is not a constant value. It is a function of the other cells being protected.

3 Implementing ACPFT as a Genetic Algorithm

3.1 Algorithm Format

Formulating ACPFT as a genetic algorithm is straightforward. The genome is a binary string corresponding to an ordered list of all cells in the target circuit. A "0" in the gene represents an unprotected cell, and a "1' represents a protected cell. Mutation simply requires flipping a bit. Single-point crossover swaps substrings beginning from a randomly selected bit. The fitness function used is the total number of "1"s in the chromosome, indicating the number of cells that have been made fault-tolerant using ACPFT's TMR scheme.

The complexity lies in the constraint that the represented circuit must fit in the target FPGA. Therefore, a constraint-satisfaction check is performed for each new chromosome in addition to calculating the fitness value. The constraint check copies the new chromosome into a temporary, integer-valued structure. The value of each gene represents a single copy of the logic cell, three copies corresponding to a protected, tripled cell, or three copies and a majority voter.

ACPFT-GA begins with the chromosome having only single and tripled cells. Using a net list that details the connectivity of the circuit, the tripled cells are processed to determine which single cells must be converted to tripled and voted cells. This ensures that the resulting circuit has the appropriate form of partial TMR according to ACPFT's principles. The net list, which also contains the type of each logic cell, and the modified chromosome are used to calculate the total numbers of logic cells of each type required. If all sums are within the limits of the target FPGA, the binary chromosome is valid, and it is added to the next generation. If the chromosome is invalid, the original chromosome from the previous generation is copied into the next generation without modification. Thus, further attempts to find a new valid version may be found in future generations instead of iterating over the chromosome in the same generation.

3.2 Software Implementation

ACPFT was implemented in Perl, since Perl is inherently proficient at working on text files. The main input was a net list of the circuit in Electronic Design Interchange Format (EDIF), which is a very common, vendor neutral file format. This net list contained a unique name for each logic cell, its cell type, and to which cells it connects. ACPFT was added by modifying this script to output several simple text files containing a condensed version of the net list and a listing of

the general cell types used. A C++ program read these intermediate files and performed the genetic evolution. The best chromosome was converted into a list of discrete cell priorities, "1" to protect and "0" to not protect. The ACPFT script then produced a modified version of the original circuit in an EDIF file. The modified EDIF could then immediately be used to implement the partially-protected circuit on an FPGA.

4 Testing and Results

ACPFT-GA was tested using the *pdc* circuit from the ACM/SIGDA "Big 20" benchmarks. This circuit was chosen since it was of significant complexity with 1272 logic cells. It had also been used previously by the author to test ACPFT's heuristic techniques, so there was existing data for comparison.

The test used a set of 4096 chromosomes. Each initial chromosome was initialized to a string of "0"s, representing a fully unprotected circuit. The mutation factor was 1/1272, the reciprocal of the number of logic cells. In each generation, the top 256 chromosomes were carried over into the next generation. 1920 were selected using elitism for mutation, and each gene was checked for a mutation using the above probability. The remaining 1920 chromosomes were generated by selecting a pair of chromosomes and crossing them at a random gene. Mutations were not applied to chromosomes created through crossover.

The algorithm needed to be tested using differing amounts of additional circuit resources. Since the size of *pdc* was fixed and actual FPGAs come in relatively few discrete sizes, an array of hypothetical FPGAs was used, as was done when previously testing ACPFT. The number of circuit resources in *pdc* was used as a baseline "perfect-fit" device and contained one inverter, 56 multiplexors, and 1215 4-input look-up tables for internal logic cells. A larger device was emulated by increasing the number of each resource by a constant multiplier and rounding down. This method created 21 more theoretical FPGAs, from 10% to 210% additional resources increments of 10%. Note that at 210%, there were sufficient resource for full TMR, and partial fault-tolerance would no longer be necessary.

ACPFT-GA was run ten times on each of the 21 larger FPGAs. Each evolution was allowed to run for 5000 generations. Figure 1 shows the maximum, minimum, and average fitness values over the ten runs on *pdc* using 20% additional resources. The graph shows data for only the first 500 generations since no improvements were found in later generations. The C++ program tracked the times required for mutation, crossing, and calculating the fitness functions and constraint conditions. It also logged the current best chromosome whenever a new maximum fitness value was found.

The results of ACPFT-GA were compared against the Backtracking method from previous research. This algorithm showed the best performance of the heuristic methods for the *pdc* circuit. Backtracking attempted to minimize the resources used as majority voters by first selecting a logic cell that directly fed an output signal. Future selections were limited to the subset of cells that either fed an output cell or had already been protected. Essentially, it started at the outputs and "backtracked" to the inputs.

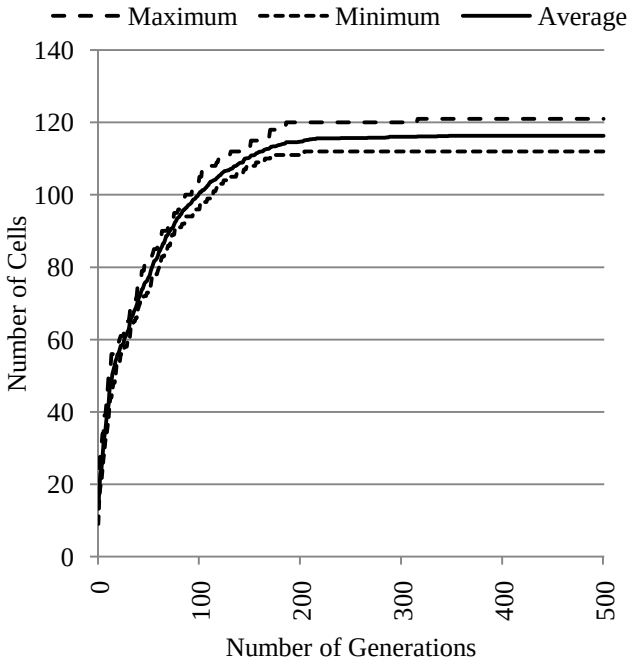


Figure 1 Cell coverage of *pdc* with 20% additional resources

Table 1 shows the cell coverage of Backtracking and the minimum, average, and maximum cell coverages over the ten runs of the genetic algorithm. The genetic versions performed significantly better with up to 40% additional resources in which even the lowest fitness values were still better than Backtracking. ACPFT-GA performed slightly better from 50% to 70% additional resources. Backtracking provided a little more coverage in the remaining tests, with noticeably better performance at 130% additional resources. This pattern can be explained by recalling that the efficiency of partial fault-tolerance is linked to the ratio of additional resources used for copying logic cells versus those used for majority voters. At low amounts of resources, the Backtracking method's selection was limited to a small subset of cells emanating from the outputs. However, the genetic algorithm was free to select cells with more efficient patterns throughout the circuit.

Most tests demonstrated that maximal fitness values were found in fewer than 1000 generations. Figure 2 gives the minimum, maximum, and average number of generations taken to reach the peak values found. As can be seen by comparing the maximums to the averages, there were a few sets of tests where one or two runs found a slight improvement in a late generation. For low to moderate amounts of resources, the number of generations required increased as the amount of resources increased.

Table 1 Coverage of ACPFT-GA compared to ACPFT with the Backtracking method

Additional Resources	ACPFT Backtracking Coverage	ACPFT-GA Coverage min/ave/max	Percent Improvement
10%	47	59/61.3/62	31.91%
20%	96	112/116.3/121	26.04%
30%	151	155/165.9/175	15.89%
40%	194	203/216.2/228	17.53%
50%	254	252/264.4/277	9.06%
60%	313	306/315.9/327	4.47%
70%	373	340/360.6/382	2.41%
80%	429	391/406.5/422	-1.63%
90%	491	442/466.5/491	0.00%
100%	558	513/530.4/551	-1.25%
110%	624	497/580.2/608	-2.56%
120%	688	563/628.6/666	-3.20%
130%	755	630/650.3/677	-10.33%
140%	816	717/723.9/736	-9.80%
150%	890	793/807.9/826	-7.19%
160%	960	865/882/900	-6.25%
170%	1024	954/966.9/985	-3.81%
180%	1092	1051/1059.2/1069	-2.11%
190%	1159	1133/1138.71149	-0.86%
200%	1217	1208/1209.7/1211	-0.49%
210%	1234	1217/1220.2/1226	-0.65%

Most tests demonstrated that maximal fitness values were found in fewer than 1000 generations. Figure 2 gives the minimum, maximum, and average number of generations taken to reach the peak values found. As can be seen by comparing the maximums to the averages, there were a few sets of tests where one or two runs found a slight improvement in a late generation. For low to moderate amounts of resources, the number of generations required increased as the amount of resources increased.

The breakdown of the algorithm's running times is shown in Figure 3. The test computer used a Core i7 processor at 2.6 GHz with 6 GB of RAM. As with many genetic algorithms, evaluating the fitness function required the majority of the time. Since validating the constraints in ACPFT-GA is related to the number of cells that have been protected, the evaluation time was expected to increase slightly as the amount of additional resources increased.

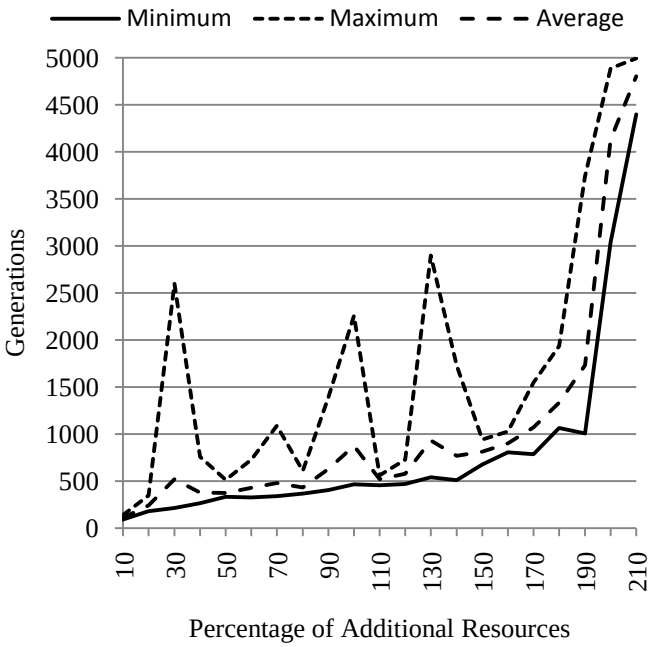


Figure 2 Number of generations required to reach maximal fitness value

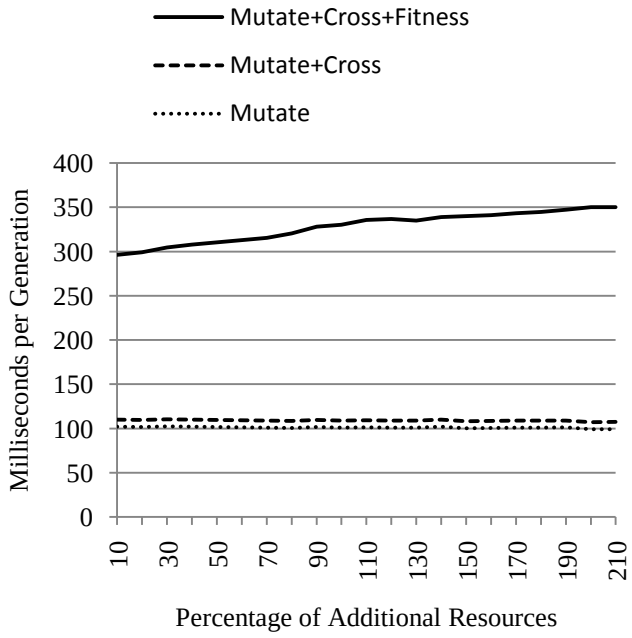


Figure 3 Time required for primary functions per generation with a population size of 4096

Looking at both Figure 2 and Figure 3, it can be seen that the total time required to find the maximal values ranged from less than one minute at 10% additional resources to about 30 minutes with 210% additional resources.

In evaluating the overall performance of ACPFT-GA, it is important to remember the context of partial fault-tolerance in FPGAs. A designer typically selects the smallest FPGA that has sufficient resources for the original, unprotected circuit to minimize the amount of unused resources on the chip and hence the cost of the device. Having 100% or more unused resources often indicates that the FPGA is over-sized and a smaller device would suffice. Although tests were performed up to 210%, in practice it is far more likely for partial fault-tolerance algorithms to have less than 100% additional resources available. In this area, ACPFT-GA generally showed better coverage than previous methods. Also, the run times for genetic evolution in this region were usually less than 5 minutes, making the method very practical.

5 Conclusions

This research applied a genetic algorithm to partition a logic design into protected and non-protected subsets for partial fault tolerance with ACPFT. Experimental results demonstrated a significant improvement in coverage for cases with small to moderate amounts of additional resources, which is the expected environment for this type of fault-tolerance.

6 References

[1] J. Lach, *et al.*, "Efficiently Supporting Fault Tolerance in FPGAs," presented at the ACM International Symposium on FPGAs, 1998.

[2] F. L. Kastensmidt, *et al.*, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," in *Design, Automation and Test in Europe*, 2005, pp. 1290-1295.

[3] D. Foster and D. Hanna, "Partial Fault Tolerance with Explicit Area Constraints," *International Journal of Embedded Systems*, December, 2009 2009.

[4] H. Quinn, *et al.*, "Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs," *IEEE Transactions on Nuclear Science*, vol. 54, pp. 2037-2043, 2007.

[5] F. Lima, *et al.*, "Designing fault tolerant systems into SRAM-based FPGAs," presented at the Design Automation Conference, 2003.

[6] K. Mohanram and N. A. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *Test Conference, 2003. Proceedings. ITC 2003. International*, 2003, pp. 893-901.

[7] B. Pratt, *et al.*, "Improving FPGA Design Robustness with Partial TMR," presented at the 12th NASA Symposium on VLSI Design, Coeur d'Alene, Idaho, 2005.

[8] P. K. Samudrala, *et al.*, "Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," *Nuclear Science, IEEE Transactions on*, vol. 51, pp. 2957-2969, 2004.

[9] N. Vijaykrishnan, *et al.*, "Reduced Triple Modular Redundancy for Tolerating SEUs in SRAM-based FPGAs," Pennsylvania State University2005.

[10] O. Ruano, *et al.*, "A Methodology for Automatic Insertion of Selective TMR in Digital Circuits Affected by SEUs," *IEEE Transactions on Nuclear Science*, vol. 56, pp. 2091-2102, August 2009 2009.

[11] D. L. Foster and D. M. Hanna, "Maximizing Area-Constrained Partial Fault Tolerance in Reconfigurable Logic," presented at the Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays, Monterey, California, USA, 2010.

Using Evolutionary Imperialist Competitive Algorithm (ICA) to Coordinate Overcurrent Relays

Farzad Razavi, Vahid Khorani, Ahsan Ghoncheh, Hesamoddin Abdollahi
Azad University, Qazvin Branch
Electrical Engineering Department
Qazvin, Iran
farzad.razavi@qiau.ac.ir

Iman Askarian
Amirkabir University of technology
Electrical Engineering Department
Tehran, Iran

Abstract—**This paper studies the ability of evolutionary Imperialist Competitive Algorithm (ICA) to coordinate overcurrent relays. Also, in order to show its greater power in optimization, the ICA is compared to the Genetic Algorithm (GA). For this purpose, the two algorithms are used to coordinate overcurrent relays, with the main optimization parameters being similar. The coordination of overcurrent relays by these two algorithms is implemented on a six-bus transmission system. More specifically, the algorithms were compared in terms of the mean convergence speed, mean convergence time, convergence reliability, and the tolerance of convergence speed in obtaining the absolute optimum point. This paper shows that at the first stage of optimization where getting close to the absolute optimum point is of importance the ICA is more powerful, while the GA shows greater power at the second stage where obtaining the exact absolute optimum point is the key question.**

Keywords- imperialist competitive algorithm, genetic algorithm, Power sytem protection, relay

I. INTRODUCTION (HEADING 1)

Accurate setting and coordination of overcurrent relays is vital for power systems. Researchers have described various methods of optimizing overcurrent relay settings [1]. Due to the complexity of the techniques used in nonlinear optimization, the traditional methods of optimizing overcurrent relays were usually performed through linear programming techniques, such as simplex [2, 3], two-phase simplex [4] and dual-simplex methods [1, 5]. It is difficult to solve the problem of coordinating protection relays, which is constrained by discrete optimization, through conventional optimization techniques [6]. Thanks to advances in the development of processors in recent years, optimization problems have made extensive use of various methods which are based on artificial intelligence and random search.

Of all intelligent algorithms, the Imperialist Competitive Algorithm (ICA), proposed by [7], leads to the best results in optimization. Algorithms such as Genetic Algorithm (GA), IGA, and PSO and their combinations have been repeatedly used in optimization problems. Also, the GA has been improved through various operations. In contrast, the potentiality of the young ICA has yet to be studied in its entirety.

A GA-based method for the optimization of the relay coordination [8] had two problems. One was lack of coordination and the other was that there was no solution for discrete Time Setting Multiplier (TSM) or Time Division Setting (TDS) [9]. The next algorithm used for this purpose was Evolutionary Algorithm, which had the same problems. However, its only advantage was that it made concrete the discrete TSM or TDS being made [6, 10, 11].

D. Birla et. Al. made some attempts to obtain additional constraints in coordinating directional overcurrent relays so that problems such as sympathy trips could be solved. The previous objective function was further improved which resulted in better coordination. In other words, lack of coordination for concrete and discrete TSM or TDS has been handled by introducing a new parameter and adding a new term to the existing objective function [9]..

In the method used in [10], relay coordination is optimized by Evolutionary GA. However, ICA is more preferable because of its fast operation. ICA starts with an initial population in which two sets of countries are included, colonies and imperialists. Each imperialist takes possession of some colonies to form an empire. Competition between the empires forms the basis of Evolutionary GA. During this competition, the weakest empire gives one colony to the most powerful empire. In the long run, a powerful empire is created whose imperialist shows the optimum point. Fig. 1 is the flowchart of this algorithm [7].

Here is a summary of the innovations in this paper:

- Solving the problem of relay coordination with evolutionary ICA.
- Comparing the operation of ICA with that of GA by considering the mean of convergence speed, mean of convergence time, convergence reliability, and the tolerance of convergence speed to reach the optimum point.
- Combining these two algorithms and proposing a new algorithm called ICA-GA capturing the best points of each algorithm.

- Providing a method to find the best point for switching from ICA to GA in combinational algorithm.
- Analyzing the operation of these algorithms by changing their initial population and countries.
- Analyzing the Imperialist Competitive Algorithm as a new method of producing population in evolutionary algorithms.
- Improving ICA by adding a new term to the formula used to determine the power of empires.
- Proposing a new method to determine the number of colonies possessed in any iteration of ICA.

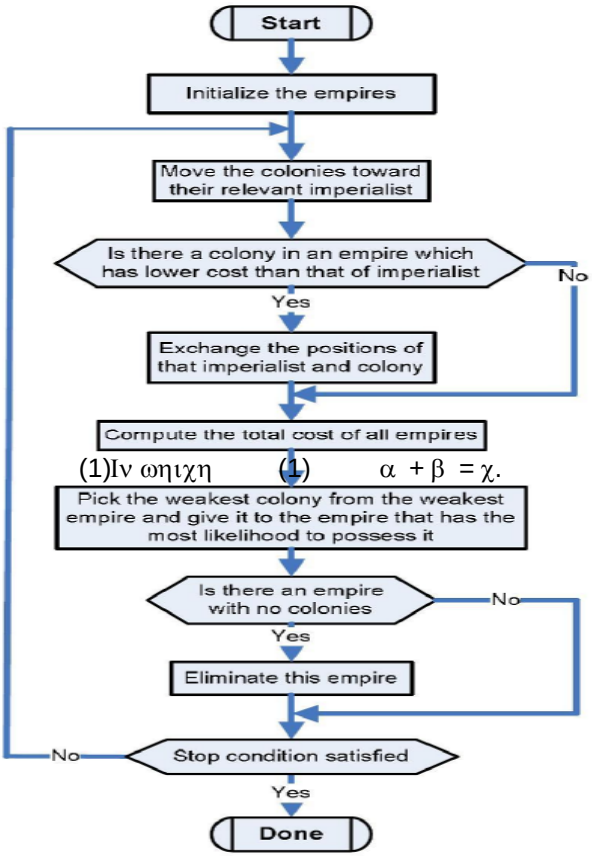


Fig. 1. Flowchart of Imperialist Competitive Algorithm

II. COMPARISON THEORY

A. Description of ICA as it relates to relay coordination

The variables of relay coordination are TSM or TDS. So, the time setting of the relays is taken as a parameter to determine the power of a country in ICA. The initial population of the countries is created randomly. In order to create a country, a vector of random numbers is created in which the rows are equal in number to time settings and each arrays represents a time setting. In order to determine the power of each country, the value of its time settings is placed in the objective function. Afterwards, a number of powerful countries are selected as imperialists and the rest are called colonies of these imperialists. All the colonies are randomly divided among the imperialists. More powerful imperialists take possession of more colonies. An imperialist together with its colonies is called an empire. In each iteration, these colonies start moving toward their imperialist country. This movement is done in a special way which is the main character of this algorithm. Then, the power of each empire is calculated again and the most powerful empire takes control of a colony from the weakest empire. After a number of iterations, weak empires collapse and eventually there will be a single powerful empire whose imperialist arrays are those time settings which are used to optimize the objective function.

B. Comparison factors

The intelligent methods of GA and ICA are used in relay coordination. In this paper, convergence speed of algorithms is based on the number of iterations which any algorithm requires in order to obtain the absolute optimum point. Although convergence speeds are ranked, at the end of this paper, according to the length of time required to get the absolute optimum point, this factor depends on the program used. Accordingly, it is more advisable to base the ranking on the number of iterations.

Equation below determines the tolerance of convergence speed:

$$S = \frac{\sqrt{\sum_{i=1}^{N}\left(X_i - \bar{X}\right)^2}}{\bar{X}} \quad (1)$$

In which

$$\bar{X} = \frac{\sum_{i=1}^{N}\left(X_i\right)}{N} \quad (2)$$

where X_i is the convergence speed of any implementation, is the total number of implementations, and $\bar{X}$ is the mean of convergence speed in various implementations. In this paper, 40 iterations are done for each algorithm. The ratio of successful to unsuccessful implementation is called convergence reliability.

III. TEST RESULTS

A. General discussion

The main factors taken into consideration in the tests are as follows:

- The end point in both algorithms is obtaining the optimum point.
- In the ICA, the algorithm is stopped if there is no result after 15000 iterations. This happens in the GA algorithm after 15000 generations.

- The number of countries in ICA and the population size in GA is set to be 5000.
- For bigger iterations, absolute optimum point is proved (TABLE I) but in testing the algorithms, the problem is studied in following two states in order to have reasonable results for convergence reliability; The algorithm is regarded convergent when OF 2.8102 and the algorithm is regarded convergent when OF 2.8110.

TABLE I. OBTAINED RESULTS FOR TSMS IN ABSOLUTE OPTIMIZATION (OF=2.8101)

Relay Number	TSM Before Latest Rounding	Rounded TSM
TSM_1	0.0958	0.10
TSM_2	0.0765	0.08
TSM_3	0.1145	0.11
TSM_4	0.1404	0.14
TSM_5	0.1421	0.14
TSM_6	0.0715	0.07
TSM_7	0.1228	0.12
TSM_8	0.0538	0.05
TSM_9	0.1059	0.11
TSM_{10}	0.1086	0.11
TSM_{11}	0.0884	0.09
TSM_{12}	0.0500	0.05
TSM_{13}	0.0500	0.05
TSM_{14}	0.0751	0.08

B. *Applied objective function*

The objective function used here is proposed in [9].

$$OF = \alpha_1 \sum_{i=1}^{N} (t_i)^2 + \alpha_2 \sum_{k=1}^{P} \left(\Delta t_{mbk} - \beta\left(\Delta t_{mbk} - |\Delta t_{mbk}|\right)\right)^2 \quad (3)$$

where:

$\Delta t_{mbk} = t_{bk} - t_{mk} - CTI$

OF: Objective function

Δt_{mbk}: Operation time difference and coordination time interval for the kth pair relay

t_i: operation time of the i'th relay to close the breaking circuit of the i'th relay when a fault occurs

t_{bk} and t_{mk}: operation time of main and backup relays to close the breaking circuit of the main relay.

N: number of relays

P: number of P/B pairs

K: is used to represent each P/B pair and varies from 1 to P

i: is used to represent each relay and varies from 1 to N

CTI: is coordination time and can be set to be 0.3 or 0.4 depending on the accuracy of the system.

β: the parameter of lack of coordination

α_1 and α_2: used to determine the weight of the two terms.

C. *The network under study*

Fig. 2 illustrates the network studied in this paper. This network includes 7 lines, 6 buses and one transformer. The relays of this network are assumed to be of a normal inverse type and their specification is calculated from the relation below:

$$\frac{t}{TSM} = a_1 + \frac{a_2}{M-1} + \frac{a_3}{(M-1)^2} + \frac{a_4}{(M-1)^3} + \frac{a_5}{(M-1)^4} \quad (4)$$

where M is the ratio of the relay's current to the pickup's current. M is the ratio of the relay's current to the pickup's current. a_1, a_2, a_3, a_4, and a_5 are scalar values which identify the characteristics of modeled relay and are assumed as below:

$$\begin{cases} a_1 = 1.98772, a_2 = 8.57922 \\ a_3 = -0.46129, a_4 = 0.036446 \\ a_5 = -0.000319901 \end{cases} \quad (5)$$

The network data are presented in TABLE II to TABLE IV. R(pu) and X(pu) are per-unit values based on 100MVA and 150KV. The data of P/B relays are given in TABLE V. TSM relays are assumed to be discrete and vary between 0.05 and 1.3 at intervals of 0.01. The TSMs of the relay are first calculated as concrete and then are converted to discrete values.

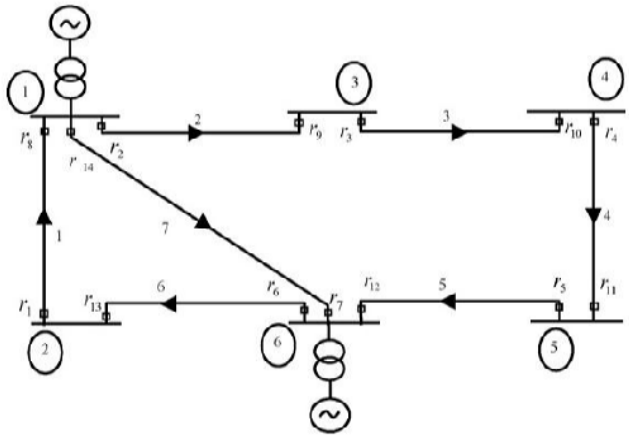


Fig. 2. Sample network

TABLE II. LINE INFORMATION

Line	*R* (pu)	*X* (pu)	*V* (kV)
1	0.0018	0.0222	150
2	0.0018	0.0222	150
3	0.0018	0.02	150
4	0.0022	0.02	150
5	0.0022	0.02	150
6	0.0018	0.02	150
7	0.0022	0.0222	150

TABLE III. GENERATOR INFORMATION

Generator	*R* (pu)	*X* (pu)	*V* (kV)
1	0.000001	0.1	10

TABLE IV. TRANSFORMATION INFORMATION

Transformer	*R* (pu)	*X* (pu)
1	0.000001	0.026666

TABLE V. P/B PAIR INFORMATION

Main Relay	Backup Relay	Primary Relay SC Current	Backup Relay SC Current
8	9	4961.7704	410.8226
8	7	4961.7704	1520.8911
2	7	5362.2983	1528.0660
2	1	5362.2983	804.8782
3	2	3334.5191	3334.5191
4	3	2234.3306	2234.3308
5	4	1352.8751	1352.8751
6	5	4695.0442	411.3675
6	14	4695.0442	1522.9084
14	1	4232.7243	794.0920
14	9	4232.7243	407.2292
1	6	2682.4959	2682.4959
9	10	1443.6699	1443.6699
10	11	2334.6515	2334.6515
11	12	3480.7511	3480.7511
12	14	5365.0609	1529.3638
12	13	5365.0609	805.5618
13	8	2490.7454	2490.7454
7	5	4232.6340	407.2472

IV. DISCUSSION

A. A comparison of convergence speed and other parameters in the two algorithms in obtaining the absolute optimum point 2.8102

This comparison is drawn in order to obtain the absolute optimum point 2.8102 and prove that ICA is more powerful than GA in achieving the absolute optimum point. The diagram for algorithm convergence for this network is given in Fig. 3. This diagram shows the objective function for the number of iterations. This figure shows the result of 40 implementations of both algorithms and selecting the nearest case to the mean of convergence speed calculated for these 40 implementations. In these implementations, the initial number of countries in ICA and corresponding initial population in GA is assumed to be 5000. ICA obtains the optimum point through 5683 iterations; GA through 6720 generations. This convergence diagram can be used to compare GA and ICA and to show the precedence of ICA.

Fig. 4 gives a schematic representation of convergence speed of the two algorithms in various implementations. The tolerance of convergence speed in ICA is not suitable to find the absolute optimum point. TABLE VI presents the mean of convergence speed, mean of convergence time, convergence speed reliability, and the tolerance of convergence speed for both algorithms in order to obtain the optimum point. As it is obvious in this table, there is not a significant difference between the two algorithms in terms of convergence time. However, this is not a suitable comparison factor due to different programming methods used in the two algorithms. If the optimum point is not obtained after more than 10000 iterations or generations, the algorithm will be regarded divergent. TABLE VII shows the optimum point obtained in this sample.

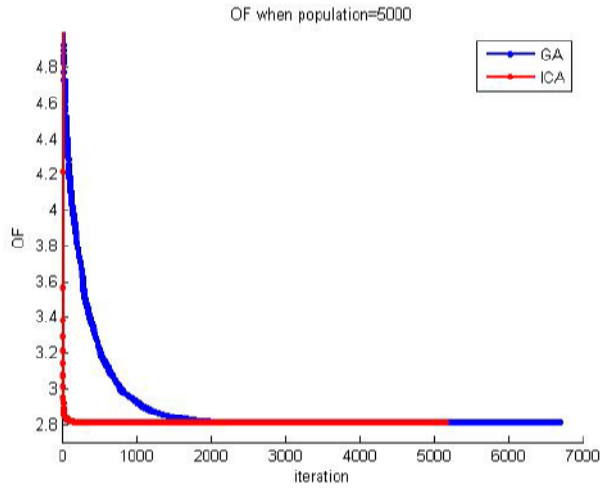


Fig. 3. Algorithm convergence in relay coordination (2.8102)

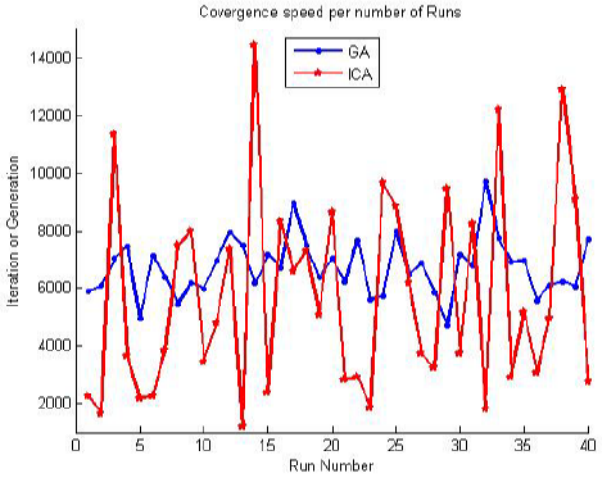


Fig. 4. Diagram of algorithm convergence speed (2.8102)

TABLE VI. OBTAINED RESULTS FOR OPTIMIZATION FROM 10 RUNS (OF=2.8102)

	ICA	GA
Mean Convergence Speed	5683	6720
Mean Convergence Time (Sec)	601	612
Convergence Speed Reliability	0.9	1
Convergence Speed Tolerance	61%	15%

TABLE VII. OBTAINED RESULTS FOR TSMS IN ABSOLUTE OPTIMIZATION (OF=2.8102)

Relay Number	TSM Before Latest Rounding		Rounded TSM
	ICA	*GA*	
1	0.0950	0.0965	0.10
2	0.0750	0.0771	0.07
3	0.1125	0.1152	0.11
4	0.1382	0.1411	0.14
5	0.1398	0.1430	0.14
6	0.0707	0.0720	0.07
7	0.1212	0.1233	0.12
8	0.0536	0.0540	0.05
9	0.1059	0.1063	0.11
10	0.1079	0.1089	0.11
11	0.0884	0.0886	0.09
12	0.0500	0.0500	0.05
13	0.0500	0.0501	0.05
14	0.0784	0.0756	0.07

Comparison of the two algorithms in obtaining the absolute optimum point is categorized below:

- Convergence speed in ICA is more than in GA.
- Consumed time for convergence in the two algorithms is approximately equivalent.
- The tolerance of convergence speed in GA is better than in ICA.
- Convergence speed reliability of GA is better than that of ICA.

B. A comparison of the two algorithms in terms of convergence speed and other parameters in obtaining the relative optimum point 2.8110

This comparison is drawn to find the relative optimum point 2.8110 in order to prove that ICA is much more powerful and its convergence reliability enhances. According to all considerations, this algorithm is the best option for real-time optimization. Fig. 5 is the diagram of algorithm convergence. This diagram illustrates the objective function according to the number of iterations, and is the result of implementing the two algorithms 40 times and then selecting the nearest case to the mean of convergence speed which is calculated from these 40 implementations. In these implementations, the initial number of countries in ICA and the initial population in GA is assumed to be 5000. GA obtains the optimum point through 2777 iterations; ICA through 455 iterations. Fig. 6 illustrates the convergence speed for the algorithms in various implementations. This figure shows that the mean convergence speed in ICA is lower than in GA but that the ICA's variation of convergence speed is comparable to the GA's in various implementations. TABLE VIII presents the mean convergence speed, mean convergence time, convergence speed reliability, and convergence speed tolerance of the two algorithms in finding the relative optimum point. Implementations that last for more than 3000 iterations or generations are assumed to be divergent. It can be seen that ICA is better than in GA in all respects except in the tolerance of convergence speed.

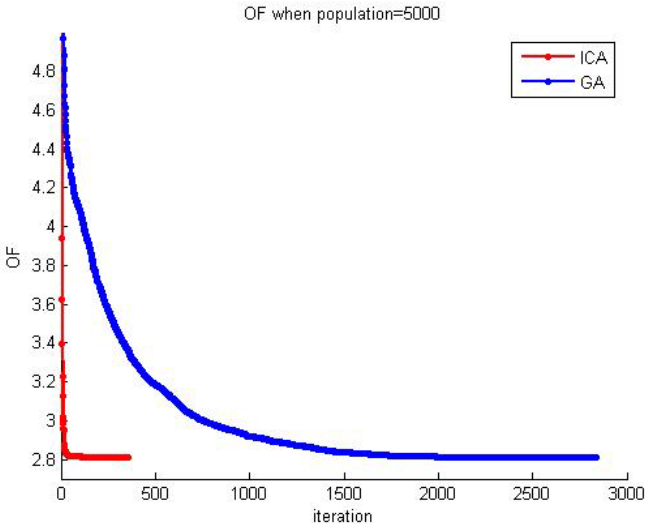


Fig. 5. Convergence diagram (2.8110)

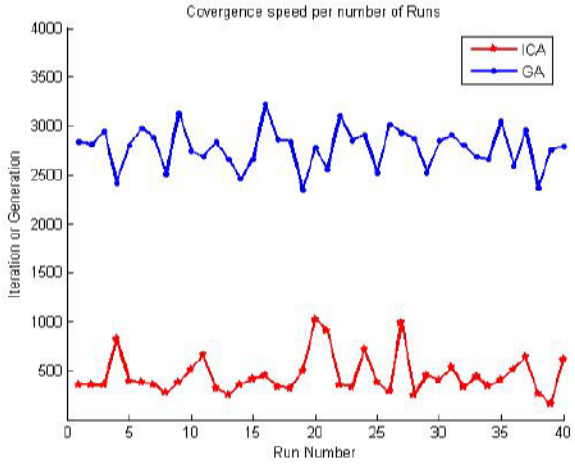


Fig. 6. Diagram of algorithm convergence speed (2.8110)

TABLE VIII. OBTAINED RESULTS FOR OPTIMIZATION FROM 10 RUNS (OF=2.8110)

	ICA	GA
Mean Convergence Speed	455	2777
Mean Convergence Time (Sec)	44	253
Convergence Speed Reliability	1	0.88
Convergence Speed Tolerance	44%	7%

V. CONCLUSION

In this paper, the two algorithms of ICA and GA were used to optimally coordinate overcurrent relays and the results were compared. The mean convergence speed, mean convergence time, convergence speed reliability, and the tolerance of convergence speed were analyzed and compared for the two algorithms. It was proved that ICA is much more powerful than GA in the first stage of optimization which is finding the approximate location of the optimum point. However, in the proximity of the optimum point, where the absolute optimum

point should be accurately located, GA operates more powerfully. It was also proved that in the first stage of optimization, convergence speed, convergence time, and convergence reliability was better in ICA. It was also shown that the tolerance of convergence speed is better in GA than in ICA.

REFERENCES

[1] D. Biral, R. Prakash Maheshwari, and H. Om Gupta, "Time-overcurrent relay coordination: A review," International Journal of Emerging Electric Power Systems, vol. 2, Iss. 2, 2005.

[2] A. J. Urdaneta, R. Nadira, and L. G. Perez Jimenez, "Optimal coordination of directional overcurrent relays in interconnected power systems," Power Delivery, IEEE Transactions on, vol. 3, Iss. 3, pp. 903-911, 1988.

[3] A. J. Urdaneta, H. Restrepo, S. Marquez, and J. Sanchez, "Coordination of directional overcurrent relay timing using linear programming," Power Delivery, IEEE Transactions on, vol. 11, Iss. 1, pp. 122-129, 1996.

[4] B. Chattopadhyay, M. S. Sachdev, and T. S. Sidhu, "An on-line relay coordination algorithm for adaptive protection using linear programming technique," Power Delivery, IEEE Transactions on, vol. 11, Iss. 1, pp. 165-173, 1996.

[5] H. Askarian Abyaneh, and R. Keyhani, "Optimal co-ordination of overcurrent relays in power system by dual simplex method," in AUPEC Conference, Perth, Australia, vol. 3, pp. 440-445, 1995.

[6] C. W. So, and K. K. Li, "Overcurrent relay coordination by evolutionary programming," Elsevier, Electric Power Systems Research vol. 53, Iss. 2, pp. 83–90, 2000.

[7] E. Atashpaz Gargari, and L. Caro, "Imperialist competitve algorithm: An algorithm for optimization inspired by imperialistic competition," in 2007 IEEE Congress on Evolutionary Computation (CEC 2007), 2007.

[8] C. W. So, K. K. Li, K. T. Lai, and K. Y. Fung, "Application of genetic algorithm for overcurrent relay coordination," in Developments in Power System Protection, Sixth International Conference on (Conf. Publ. No. 434), pp. 66-69, 1997.

[9] F. Razavi, H. Askarian Abyaneh, M. Al-Dabbagh, R. Mohammadi, and H. Torkaman, "A new comprehensive genetic algorithm method for optimal overcurrent relays coordination," Elsevier, Electric Power Systems Research, vol. 78, Iss. 4, pp. 713-720, 2007.

[10] C. W. So, and K. K. Li, "Time coordination method for power system protection by evolutionary algorithm," Industry Applications, IEEE Transactions on, vol. 36, Iss. 5, pp. 1235-1240, 2000.

[11] C. W. So, and K. K. Li, "Intelligent method for protection coordination," in Electric Utility Deregulation, Restructuring and Power Technologies, 2004. (DRPT 2004). Proceedings of the 2004 IEEE International Conference on, vol. 1, pp. 378-382, 2004.

A methodology to find clusters in the data based on Shannon's Entropy and Genetic Algorithms

Edwin Aldana-Bobadilla[1], **Angel Kuri-Morales**[2]
[1]Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas, Universidad Nacional Autónoma de México, Mexico City, Mexico
[2]Departamento Académico de Computación, Instituto Tecnológico Autónomo de México, Mexico City, Mexico

Abstract - *The most common clustering methods are based on metrics that allow the determination of the similarity between elements of a given data set. This similarity allows us to divide the data set into subsets (clusters) that contain "highly similar" elements. The use of a metric imposes two constraints. First, the shape of the found clusters is generally hyper-spherical (in the space of the metric) due to the fact that each element in a cluster lies within a radial distance relative to a given center. Second, the metric may be sensitive to the probability density function of the data set. Following this fact several methods based on statistical approaches have become an attractive and powerful option. These involve the estimation of the probability density function (pdf) of the data set which minimizes an optimality criterion. Generally this is a highly non-linear and usually non-convex optimization problem which disallows the use of traditional optimization techniques. In this paper we propose a statistical method based on Shannon's Conditional Entropy which uses a rugged genetic algorithm to find the optimal pdf. Each individual of the Genetic Algorithm is a possible solution of a clustering problem. The fitness of an individual is determined by Shannon´s entropy encoded in its genome and an additional constraint related to the "quality" of this solution. The "quality" is measured through a validity index of the clustering process. A novel and important aspect of our method is the form of representation of the objects of the data set in order to reduce the computational complexity due to the high dimensionality. We show that our proposal has high effectiveness relative to methods as k-means, fuzzy c-means and Kohonen Maps with a synthetic data set.*

Keywords: Clustering, Information Theory, Genetic Algorithms, Bayesian Classifier, Data Mining.

1 Introduction

The clustering process is an optimization problem that maximizes the similarity between objects or elements that belong to same cluster and minimizes the similarity between elements of different clusters. The effectiveness of a clustering method is given by several factors such as the metric and the desired number of clusters.

Particularly, the use of a metric imposes some constraints on the shape of clusters found. These shapes generally are hyperspherical (in the space of the metric) due to the fact that each element in a cluster lies within a radial distance relative to a given center. In other words the elements of a cluster tend to group around a single mean value (center) which sometimes disallows the extraction of hidden patterns in the data set.

In this paper we propose an alternative method based on a statistical approach. Our proposal does not use explicitly a metric to determine the elements that belong to given cluster. Overall, this proposal is an iterative search of a partition model of the data set in which the entropy (uncertainty) is minimized. In order to determine the entropy of the data set for a particular partition model, the estimation of its probability density function (pdf) is necessary. This estimation can be achieved statistically from three different methods: parametric, semi-parametric and non-parametric [15]. Unlike parametric and semi-parametric methods, the non-parametric methods do not make any assumption about of the pdf of the data set. The Parzen window [5] is among the most widely-used non-parametric density estimation method.

Different clustering methods have been proposed around these non-parametric methods and minimum entropy principle [9], [15],[16]. These methods can be seen as an iterative search of an optimal pdf of the data set such that the entropy is minimal. However, depending on the dataset the search may be unfeasible or may yield local optimal solutions. Thus, this is a highly non-linear and usually non-convex optimization problem which disallows the use of traditional optimization techniques or pdf estimation methods.

We propose a method which uses a rugged genetic algorithm (the so-called Vasconcelos's GA [12]). Each individual of GA is a possible solution of a clustering problem which represents a pdf of the data set. The fitness of an individual is based on the minimum entropy principle and an additional constraint related to the "quality" of the solution. The "quality" is measured through an validity indices of the clustering process. Several validity indices have been developed and introduced [4],[8], [11]. A novel and important aspect of our proposal is the form of representation of the objects of the data set. Generally the properties of each object are represented as real values of vector in a Euclidean space. The dimensionality of this vector is given by the number of such properties. Its value is an important element of the computational complexity of a clustering algorithm.

In order to reduce the dimensionality, statistical techniques as such as Pearson's correlation analysis [3] and/or principal components' analysis [17] have been used. In many cases, however, these techniques are sensitive to the data distribution and impair the effectiveness of the clustering process. To avoid this fact we map the *n*-dimensional vector space of the data set to the space of all possible strings (words) that can be built using the symbols of an alphabet $\sum$. This transformation allows us to represent an object of data set as a word of length *n* (for a *n*-dimensional space) and a cluster as a subset of words with some "degree of similarity". The entropy of a cluster is determined by the probability distribution of all words that belong to that cluster.

Our work begins with an account of several concepts which are needed to expose our method. Then, we expound the fundamental process of our proposal. Finally we show several numerical results and the respective conclusions.

2 Theoretical Aspects

In what follows we make a very brief mention of the theoretical aspects having to do with the proper understanding of our proposal. The reader may find more details in the references.

2.1 Minimum Entropy Principle

Shannon's entropy [20] allows us to measure the uncertainty associated with a random variable *X*. Mathematically, Shannon's entropy of *X* with a probability mass function *p(x)* is defined as:

$$H(X) = -\sum p(x)\log(p(x)) \qquad (1)$$

The possible values of a random variable *X* occur with certain probability *p(X=x)* or simply *p(x)*. When *p(x)* is uniformly distributed we say that the uncertainty is greatest or that the process represented by the random variable *X* has a highest degree of "disorder". Figure 1 represents the entropy for two possible values of *X* with probabilities *p* and *1- p*; when *p=0.5* the entropy is maximum.

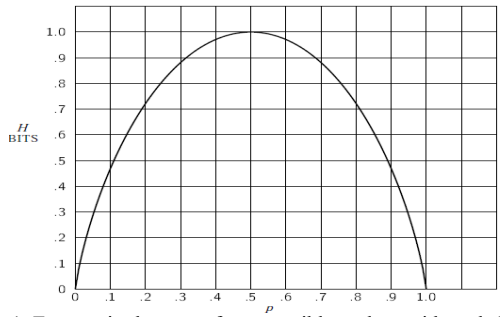


Figure 1. Entropy in the case of two possibles values with probabilities *p* and *(1-p)*

In the context of the clustering problem we assume that a cluster is a subset of the data set which has minimum entropy. It means that a cluster is a partition of data set with minimum degree of "disorder". The entropy of a cluster is directly related to its elements. In terms of probability, the entropy of the cluster depends of the pdf of its elements. In what follows we expound on this fact.

Let *D* be the data set with *K* partitions (clusters) and *x* an element that belong to *D*. Then the pdf of *x* is given by:

$$p(x) = \sum_{i}^{K} p(x|i)\,p(i) \qquad (2)$$

where *p(i)* is the prior probability for the *i-th* partition and *p(x|i)* is the prior probability of *x* given the *i-th* partition. In Figure 2 we show an intuitive representation of the probabilities *p(x|1)* and *p(x|3)*, the probability *p(x|2)* is zero.

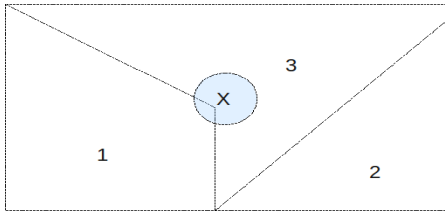


Figure 2. Probability space of a data set with three partitions. The element *x* belongs to partition 1 and 2 with a probability greater than zero.

However we would like to know the dependence of pdf of the *i-th* partition with respect to *x*. This dependence is given by Bayes Theorem [10] :

$$p(i|x) = \frac{p(x|i)\,p(i)}{p(x)} \qquad (3)$$

When *p(i|x)* is uniformly distributed for all i, we can say that the element *x* belongs to any partition and thus the uncertainty is maximum (see Figure 3a.). On the other hand if all *p(i|x)* but one are zero (one having the value unity) then we are certain of the partition to which *x* belongs (see Figure 3b).

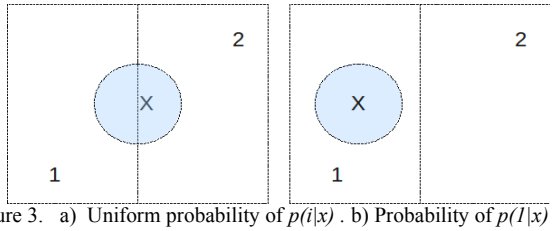


Figure 3. a) Uniform probability of *p(i|x)* . b) Probability of *p(1|x)*

Now, let *C* be a random variable whose possible values are *1,2,..K* which represent the partitions of *D*. Let *X* be a random variable whose possible values are all elements *x* that belong to *D*. Then the entropy of *C* given *X* is:

$$H(C|X) = -\sum_{i=1}^{K} p(i|x)\log(p(i|x)) \qquad (4)$$

where $p(i|x)$ is a posteriori pdf. Thus, our goal is to find this function such that $H(C|X)$ is minimum. For reasons already mentioned we use a genetic algorithm. The entropy given by Equation 4 is called *Conditional Entropy* [1].

2.2 Genetic Algorithms

Genetic Algorithms (an interesting introduction to GA's and other evolutionary algorithms may be found in [2]) are optimization algorithms which are frequently cited as "partially simulating the process of natural evolution". Although this a suggestive analogy behind which, indeed, lies the original motivation for their inception, it is better to understand them as a kind of algorithms which take advantage of the implicit (indeed, unavoidable) granularity of the search space which is induced by the use of the finite binary representation in a digital computer. In such finite space, numbers originally conceived as existing in R^n actually map into B^m space. Thereafter it is simple to establish that a genetic algorithmic process is a finite Markov chain (MC) whose states are the populations arising from the so called genetic operators: (typically) selection, crossover and mutation [19]. As such they display all of the properties of a MC. From this fact one may prove that:

1. The final results of the evolutionary process are independent of the initial population and

2. A GA preserving the best individual arising during the process will converge to the global optimum (albeit the convergence process is not bounded in time).

Their most outstanding feature is that, as opposed to other more traditional optimization techniques, the GA iterates simultaneously over several possible solutions. Then, other plausible solutions are obtained by combining (crossing over) the codes of these solutions to obtain hopefully better ones. The solution space (SS) is, therefore, traversed stochastically searching for increasingly better plausible solutions. In order to guarantee that the SS will be globally explored some bits of the encoded solution are randomly selected and changed (a process called mutation). The main concern of GA-practitioners (given the fact that well designed GAs, in general, will find the best solution) is to make the convergence as efficient as possible. The work of Forrest et al. has determined the characteristics of the so-called Idealized GA (IGA) which is impervious to GA-hard problems [6].

2.2.1 Vasconcelos's Genetic Algorithm

The implementation of the IGA is unattainable in practice. However, a practical approximation called the Vasconcelos's GA (VGA) has been repeatedly tested and proven to be highly efficient [12]. The VGA, therefore, turns out to be an optimization algorithm of broad scope of application and demonstrably high efficiency. A statistical analysis was done by minimizing a large number of functions and comparing the relative performance of six optimization methods of which five are GAs[1]. The ratio of every GAs absolute minimum (with probability $p = 0.95$) relative to the best GAs absolute minimum may be found in Table 1 under the column "Relative Performance". The number of functions which were minimized to guarantee the mentioned confidence level is shown under "Number of Optimized Functions". It may be seen that VGA, in this study, was the best of all the analyzed variations. Interestingly the CGA (the classical or "canonical" genetic algorithm) comes at the bottom of the list with the exception of the random mutation hill climber (RHC) which is not an evolutionary algorithm. According to these results, the minimal found with VGA are, in the worst case, more than 25% better than those found with the CGA. Due to its tested efficiency, we now describe in more detail VGA.

As opposed to the CGA, VGA selects the candidate individuals deterministically picking the two extreme (ordered according to their respective fitness) performers of the generation for crossover. This would seem to fragrantly violate the survival-of-the-fittest strategy behind evolutionary processes since the genes of the more apt individuals are mixed with those of the least apt ones. However, VGA also retains the best n individuals out of the $2n$ previous ones.

Table 1: Relative Performance of Different Breeds of Genetic Algorithms

Algorithm	Relative Performance	Number of Optimized Functions
VGA	1.000	2,736
EGA	1.039	2,484
TGA	1.233	2,628
SGA	1.236	2,772
CGA	1.267	3,132
RHC	3.830	3,600

The net effect of this dual strategy is to give variety to the genetic pool (the lack of which is a cause for slow convergence) while still retaining a high degree of elitism. This sort of elitism, of course, guarantees that the best solutions are not lost. On the other hand, the admixture of apparently counterpointed plausible solutions is aimed at avoiding the proliferation of similar genes in the pool. In nature as well as in GAs variety is needed in order to ensure the efficient exploration of the space of solutions. As stated before, all elitist GAs will eventually converge to a global optimum. The VGA does so in less generations. Alternatively we may say that VGA will outperform other GAs given the same number of generations. Besides, it is easier to program because we need not simulate a probabilistic process. Finally, VGA is impervious to negative fitness's values. We, thus, have a tool which allows us to identify the best values for a set of predefined metrics possibly reflecting complementary goals. For these reasons we use in our work VGA as the optimization method. In what follows we explain our proposal based in the concepts mentioned above.

[1]VGA: Vasconcelos' GA; EGA: Eclectic GA; TGA: Elitist GA; SGA: Statistical GA; CGA: Canonical (or Simple) GA; RMH: Random Mutation Hill Climber.

3 Methodology

We begin our explanation by discussing the preprocessing of the data set. It will allow us to change the vector representation of the data in order to facilitate subsequent calculations. Second, we show the details of the genome's encoding in the context of the clustering problem. Finally we show the way to evaluate each solution or individual in order to find the best.

3.1 Preprocessing of the data set.

Let Σ be an alphabet and w a string that contains symbol of Σ. Let D be a data set. Let $x_i = \{a_1, a_2, ...a_n\}$ be an n-dimensional vector such that $x_i \in D$ where $a_i \in \boldsymbol{R}$ and $D \in \boldsymbol{R^n}$.

Let $\perp a_k$, $\top a_m$ be the minimal and maximal value $\forall\, a_i \in D$. Let Δ be the difference between $\top a_m$ and $\perp a_k$, then we assign to every symbol of Σ an interval value as following:

Table 2: Assigning values to symbols of Σ

Symbol	Interval Value
s_o	$\left[.\perp a_k, \perp a_k + \frac{\Delta}{\lvert\Sigma\rvert}\right]$
s_1	$\left[s_{0\,max}, s_{0max} + \frac{\Delta}{\lvert\Sigma\rvert}\right]$
...	...
s_m	$\left[s_{m-1\,max}, s_{m-1\,max} + \frac{\Delta}{\lvert\Sigma\rvert}\right]$

Where $s_{i\,max}$ is the maximum interval value of S_i and $|\Sigma|$ is the cardinality of Σ ($m=|\Sigma|$). Now we assume that Σ is conformed by the letters of the English alphabet and $\top a_m=1$ and $\perp a_k=0$. In accordance with Table 2 we can determine the interval values of Σ as shown in Table 3.

Table 3: Possible assignment of values for letters of the English alphabet

Symbol	Symbol Value
A	$\left[0,0+\frac{1}{26}\right]$
B	$\left[\frac{1}{26}, \frac{1}{26}+\frac{1}{26}\right]$
...	...
Z	$\left[\frac{25}{26}, 1\right]$

Moreover, if we assume a data set D in R^3 such that some x=[0.038,0.022,0.99]. Then x may be represented by w=AAZ. Thus, $\forall x \in D$, $\exists w \in \Sigma^*$. We represent the set of all strings or words w as D'. For practical reasons we use the English Alphabet although the method described does not depend on any particular symbol set. However this method will be affected by the cardinality of Σ. For example, if $|\Sigma| = 1$ we have that all elements of the data set are represented by the same word regardless of their degree of similarity. Otherwise when the value of $|\Sigma|$ is higher we will have more precision but the performance will be affected.

3.2 Encoding of the genome.

The individuals of the algorithm have been encoded as follows. a) The length of the genome is equal to the cardinality of D'. b) Each gene is associated with a word of D'. The value of each gene corresponds to a label (for practical purposes we use $1,2,...K$) of the cluster to which the word belongs. Thus, the i-th gene represent the cluster to which the i-th word belongs. Figure 4 exemplifies a genome for $K=3$.

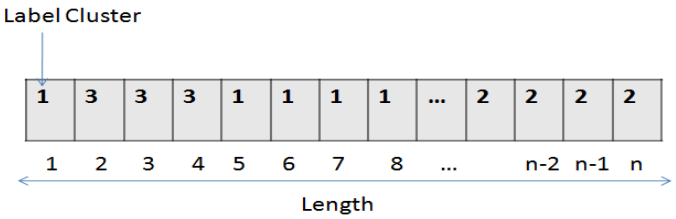


Figure 4. Genome of the individual (K=3)

3.3 Fitness

Each individual is a possible solution of a clustering problem which is evaluated through a fitness function. In what follows we explain how this function is defined in the context of our method.

Suppose that D'={AAA, ACA, MOM, NPM, ADE, UVT, VXT, NQP, VWV} and $K=3$. Let I_i be the i-th individual of the population whose genome are shown in Figure 5.

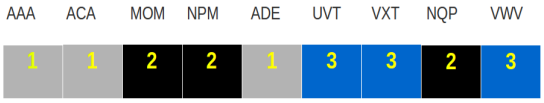


Figure 5. Possible solution given by an Individual for $K=3$. Here are shown the words associated to each gene.

As discussed above we use the Minimum Entropy Principle. In Equation 4 X is a random variable whose set of possibles values belongs to D. Thus, if the data set D is transformed to set D' (conformed by words w) then Equation 4 may be rewritten as:

$$H(C|W) = -\sum_{i=1}^{K} p(i|w)\log(p(i|w)) \qquad (5)$$

Where W is a random variable whose possibles values are strings of the Σ alphabet. We can calculate $H(C|W)$ for all individuals based on their genomes. This entropy may be expressed as the sum of entropies for each cluster as follows:

$$H(C|W)=\sum_{i=1}^{K} H(i|W) \quad (6)$$

Where $H(i|W)$ is the entropy of cluster i. The idea is to minimize the entropy for each cluster. However, this fact involves a multi-objective optimization problem because minimizing the entropy of a cluster affects the entropy of any other. To resolve this problem we apply Pareto's Efficiency [18]. Our objective function may be written as:

$$min\left[H(1|W),H(2|W)\ldots H(K|W)\right] \quad (7)$$

So, the GA must find the individuals that minimize this function which is represented as a vector of K dimensions. In what follows this vector is called *Entropy Vector*. Each individual has a Entropy Vector whose values are given by its genome. In order to determine the individual with the best vector, we apply the principle of *Pareto Dominance* [18]. The Pareto Dominance says that a Y vector dominates to Y^* if $\forall y_i \in Y$, $y_i \leq y_i^*$ and $\exists y_p$ such that $y_p < y_p^*$. In the context of VGA, a solution vector X of an Individual will dominate other solution vectors. The number of vectors dominated by X are called the *dominance value.* Thus, individuals with higher dominance value will be the best. The result of the evolutionary process yields a *Pareto Front*[18]. The fitness function for *i-th* individual (I_i) may be written as:

$$f(I_i)=dom_i \quad (8)$$

Where dom_i is the dominance value of the i^{th} individual. However this function does not always assure that an individual with maximal dominance value is the best solution to the clustering problem. We, therefore propose a quality measure.

Our quality measure is based on the concept of *Mutual Information (MI)* [21]. It is a symetric measure that quantifies the mutual dependence between two random variables or the information that these share. In the context of our problem, the MI between two cluster u and v is given by:

$$I(u,v)=\sum_{i=1}^{R}\sum_{j=1}^{S} p(w_i,w_j)\log\frac{p(w_i,w_j)}{p(w_i)p(w_i)} \quad (9)$$

where R and S are $|u|$ and $|v|$ respectively and $p(w_i, w_j)$ is the probability that the words w_i and w_j are similar. This probability is given by:

$$p(w_i,w_j)=\frac{|w_i\cap w_j|}{length(w_i)} \quad (10)$$

where the intesection between two word is given by their common symbols. Clearly, all words of D' have the same length.

If $u \neq v$ then the value of $I(u,v)$ will be called *Mutual Information Intercluster* (MI_{Inter}) . Otherwise this value will be called *Mutual Information Intracluster* (MI_{Intra}). A lower value of MI_{Inter} and higher value of MI_{Intra} means better clusters. So, we propose a quality measure given by:

$$Q=\frac{\sum_{i=1}^{K} MI_{Intra}(i,i)}{\sum_{i,j\leq K, i\neq j} MI_{Inter}(i,j)} \quad (11)$$

An individual with higher value of Q means a better solution. Therefore the fitness function of the i^{th} individual may be defined as:

$$f(I_i)=dom_i Q_i \quad (12)$$

However, we observe that an individual with a "good" fitness value does not always represent a global optimum. Thus, we assume that each individual must be subject to the following constraint : *The probability for all partition (cluster) of D must be greater than zero. Mathematically p(i)>0* $\forall$ *i=1,2,..K (see Equation 2 and Equation 3).*

This constraint ensures that the individuals consists of non-empty clusters whose entropy is minimal. Otherwise the solutions will be outside of the feasible region. To encourage reproduction of feasible individuals (which represents feasible solutions) in every generation of VGA, we appeal to an penalty method [14] whose goal is to punish unfeasible individuals.

Here the penalty for unfeasible individual I_i is given by:

$$P(I_i)=J-\sum_{i=1}^{s}\frac{J}{m} \quad (13)$$

where J is a large constant [$O(10^9)$], m is the number of constraints and s is the number of these which have been satisfied.

4 Numerical Experiment

In what follows we briefly describe how the test data set was generated. Subsequently we show several parameters and features of the performed tests. Finally we show the results. We call our proposal has been called *Entropic Evolutionary Clustering* (EEC).

4.1 The data set

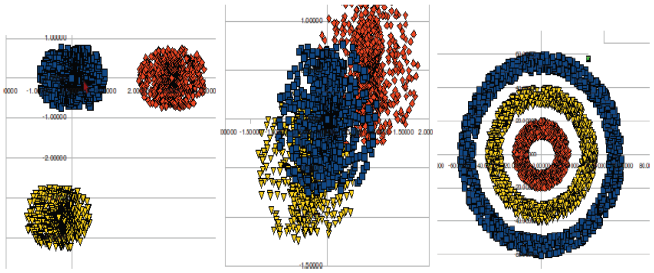
Three data sets are analyzed in this work. We shall call them "*A*", "*B*" and "*C*" respectively. Every set is composed of vectors (in a *3D* space) that belong to three different spheres which we call sphere 1, 2 and 3 respectively. There are 10,000 vectors in each one of the spheres. They were generated from.

$$x = x_0 + r \sin\theta \cos\phi \quad (14)$$

$$y = y_0 + r \sin\theta \sin\phi \quad (15)$$

$$z = z_0 + r \cos\theta \quad (16)$$

from uniformly distributed values for $r \in [0,1)$, $(0 \le \phi \le 2\pi$ and $0 \le \theta \le \pi)$. For set *A* the three centers of the spheres were chosen so that the spheres would not intersect. In set *B*, the chosen centers yield partially overlapping data. Finally, in set *C*, the spheres shared a common center. However, in the last set for sphere 1 $r \in [0,1)$; for sphere 2 $r \in [0, 0.666)$; for sphere 3 $r \in [0, 0.333)$. In this case, then, spheres 1, 2 and 3 share the same space where the density of 2 is larger than that of 1 and the density of 3 is larger than the other two. Our intent is to choose vectors in set A, B and C whose distribution is not uniform but Gaussian. To achieve this, we determined to divide the space of probabilities of a Gaussian curve in 20 equally spaced intervals. The area under the curve for a normal distribution with $\mu = 0$ and $\sigma = 1$ between -4 and +4 is very closely equal to one. Therefore, it is easy to see that 5%, of the observations will be between −4 and −1.654; 5%, will be between −1.654 and −1.280, etc. The required normal behavior may be approximated by selecting 50 of the uniformly distributed values from the interval $[-4,-1,654)$; another 50 from the interval $[-1.654,-1.280)$, etc. In all we will end up with 1000 vectors for every sphere. These vectors will now be very closely Gaussian. When data is normally distributed, a Bayesian classifier is optimal. The behavior of one such classifier will serve as a base point. To stress: when the distribution of the data set to classify is Gaussian, a Bayesian classifier yields the best theoretical results (by minimizing the probability of classification error independently of the degree of overlap between the distributions of the clusters) [7]. Hence, we resorted to Gaussian distributed data in order to establish a behavior relative to the best theoretical one when measuring the performance of non-traditional methods. Our claim is that, if the methods perform satisfactorily when faced with Gaussian data, they will also perform reasonably well when faced with other possible distributions. That is, we wish to show that the results obtained with non-traditional methods are close to those obtained with a Bayesian classifier for the same data set. This would mean that these results correspond to an efficient algorithm. The data sets are illustrated in Figure 6.



(a) Disjoint data set (b) Overlapped data set (c) Overlapped data set

Figure 6. Types of data set

5 Results

The values of the parameters of VGA are given in Table 4. These values were determined experimentally. As mentioned above we use the English Alphabet to transform the original data set. The VGA was run 20 times (with different seeds of the pseudo random number generator) per data set. The same data sets was tested with *K-Means* [22], *Kohonen Maps* [23] and *Fuzzy C-Means* [4]. Since it may be proven that a Bayesian Classifier is optimal when the data's pdf is Gaussian [7], we include a comparison with such a Bayesian Classifier. The results obtained with disjoint clusters are shown in Table 5. This allows us to see that the results of EEC are similar to those given by some alternative algorithms. The high effectiveness in all cases is due to the spatial distribution of data set. The results obtained with overlapping clusters are shown in Table 6 where we can see that the effectiveness decreases significantly in general.

Table 4: Parameters Test

Parameter Name	**Values**
N (Number of Individuals)	50
G (Generations)	4000
pm (Mutation probability)	0.00
pc (Crossover Probability) 0.99	0.99

However EEC showed better results than traditional methods and close results to Bayesian Classifier. The results obtained in the two last cases (overlapping and concentric clusters) are due to the fact that it is not possible to find a simple separable boundary. Therefore, the boundary decision is unclear and the vast majority of the clustering methods yield poor solutions. The closeness of the results obtained so far relative to a Bayesian Classifier, tells us that our approach is quite efficient. In future works we will report on experiments encompassing a wider range of data sets.

Table 5: Results obtained with disjoint clusters data set

Algorithm	**Average Effectiveness**
EEC	0.99
K-Means	0.98
Kohonen Maps	0.99
Fuzzy C-Means	0.98
Bayesian Classifier Effectiveness	0.99

Table 6: Results obtained with overlapping clusters data set

Algorithm	**Average Effectiveness**
EEC	0.87
K-Means	0.45
Kohonen Maps	0.72
Fuzzy C-Means	0.15
Bayesian Classifier Effectiveness	0.89

Table 7: Results obtained with concentric clusters data set

Algorithm	**Average Effectiveness**
EEC	0.71
K-Means	0.36
Kohonen Maps	0.38
Fuzzy C-Means	0.15
Bayesian Classifier Effectiveness	0.72

6 Conclusion

Following the minimum entropy principle we employed a genetic algorithm so that we were able to explore the solution space of the clustering problem. This approach resulted a better effectiveness with different data sets respect to *K-Means*, *Kohonen Maps* and *Fuzzy C-Means*. If we consider that Bayesian Classifier represents a theoretical limit then the most interesting result is the nearness of EEC respect this classifier. Our method promises to be a feasible alternative to find non-spherical clusters due to the results obtained with the concentric clusters of the data set C. However, we require testing several data sets that allow us to statistically ascertain that our method is good. We will report on these issues shortly. Additionally, data preprocessing proved to be a good alternative to reduce the computational complexity when the dimensionality of the data set is fairly high.

7 References

[1] Arndt, C., Information measures: information and its description in science and engineering, *Springer Verlag,* 2001

[2] Bäck, Th., Evolutionary Algorithms in Theory and Practice, *Oxford University Press*, 1996

[3] Cohen, J., Applied multiple regression/correlation analysis for the behavioral sciences, *Lawrence Erlbaum*, 2003

[4] Dunn, J. C., A Fuzzy Relative of the ISODATA Process and Its Use in Detecting CompactWell-Separated Clusters, *Journal of Cybernetics 3*, volume 3, 32–57, 1973

[5] Parzen E.: On the estimation of a probability density function and the mode. *Annals of Math. Stats.*, 33:1065-1076, 1962.

[6] Forrest, and Mitchell, What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation, MACHLEARN: *Machine Learning 13*, volume 13, 1993

[7] Haykin, Simon, *Neural networks: A comprehensive foundation*, MacMillan,1994

[8] Halkidi, Maria, Batistakis, Yannis, and Vazirgiannis, Michalis, On Clustering Validation Techniques, J. Intell. *Inf. Syst. 17(2-3)*, volume 17, 107–145, 2001

[9] Jenssen, R., Hild, KE, Erdogmus, D., Principe, J.C., and Eltoft, T., Clustering using Renyi's entropy, Neural Networks, 2003. *Proceedings of the International Joint Conference on*, volume 1, 523–528, 2003

[10] Joyce, James, *Bayes Theorem, The Stanford Encyclopedia of Philosophy*, Fall 2008 edition, Eds: Zalta, Edward N., 2008

[11] Kovacs, Ferenc, and Ivancsy, Renata, *A novel cluster validity index: variance of the nearest neighbor distance*, *WSEAS Transactions on Computers*, volume 3, 477-483, March 2006

[12] Kuri-Morales, Angel, A Methodology for the Statistical Characterization of Genetic Algorithms, MICAI, volume 2313, *Springer*, 79–88, Eds: Coello, Carlos A. Coello, de Albornoz, Alvaro, Sucar, Luis Enrique, and Battistutti, Osvaldo Cairó, 2002

[13] Kuri-Morales, Angel, and Aldana-Bobadilla, Edwin, Finding Irregularly Shaped Clusters Based on Entropy, ICDM, volume 6171, *Springer*, 57–70, Eds: Perner, Petra, 2010

[14] Kuri-Morales, Angel and Gutiérrez-García, Jesús, Penalty Function Methods for Constrained Optimization with Genetic Algorithms: A Statistical Analysis, *MICAI, volume 2313, Springer*, 108–117, Eds: Coello, Carlos A. Coello, de Albornoz, Alvaro, Sucar, Luis Enrique, and Battistutti, Osvaldo Cairó, 2002

[15] Lee, Y., and Choi, S., Minimum entropy, k-means, spectral clustering, Neural Networks, 2004. *Proceedings IEEE International Joint Conference on*, volume 1, 2005.

[16] Li, H., Zhang, K., and Jiang, T., Minimum entropy clustering and applications to gene expression analysis, *IEEE Computer Society*, 2004.

[17] Pearson, K., Principal components analysis, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 6(2)*, volume 6, 559, 1901

[18] Podinovskii, VV, and Nogin, VD, Pareto-Optimal Solutions of Multicriteria Problems, *Nauka*, Moscow, 1982

[19] Rudolph, G., Convergence Analysis of Canonical Genetic Algorithms, *IEEE Transactions on Neural Networks 5(1)*, volume 5, 96–101, January 1994

[20] Shannon, C. E., and Weaver, W., The Mathematical Theory of Communication, *Scientific American*, July 1949

[21] Vinh, N.X., Epps, J., and Bailey, J., Information theoretic measures for clusterings comparison: is a correction for chance necessary?, *Proceedings of the 26th Annual International Conference on Machine Learning*, 1073–1080, 2009

[22] McQueen, J. B., Some Methods of Classification and Analysis of Multivariate Observations, *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281–297, Eds: Cam, L. M. Le, and Neyman, J., 1967

[23] Kohonen Teuvo, Self-organizing maps, *Springer-Verlag,* New York, Inc., 199

Bézier Parameterization for Optimal Control by Differential Evolution

T. Rogalsky
Mathematics, Canadian Mennonite University, Winnipeg, Manitoba, Canada

Abstract - *Direct solution methods for optimal control problems treat them from the perspective of global optimization: perform a global search for the control function that optimizes the required objective. Increasingly, Differential Evolution is being recognized as a powerful global optimizer for optimal control. A parameterization technique is required, which can represent control functions using a small number of real-valued parameters. Typically, direct methods using Differential Evolution parameterize control functions with a piecewise constant approximation. In this paper a new parameterization is introduced, using Bézier curves, and is combined with Differential Evolution into a new evolutionary direct method for optimal control. The effectiveness of the new method is demonstrated by solving a range of optimal control problems.*

Keywords: Control Vector Parameterization, Differential Equations, Evolutionary Algorithms, Optimal Control

1 Introduction

In many mathematical models, the dynamics are described by a system of ordinary differential equations for a set of dependent functions, $\mathbf{x}(t)$. When these systems are also controlled by a second set of independent functions, $\mathbf{u}(t)$, an obvious goal is to find $\mathbf{u}(t)$ that optimizes, in some sense, the dynamical system. This type of problem is known as optimal control, or sometimes, dynamic optimization. Mathematically, the problem can be stated as follows:

$$\min_{\mathbf{u}} F(\mathbf{u}) = \int_{t_0}^{t_f} f(t, \mathbf{x}(t), \mathbf{u}(t))dt, \quad \text{subject to } \begin{cases} \mathbf{x}'(t) = g(t, \mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases}, \tag{1}$$

where t_0 and t_f are the initial and final times, and f and g depend on the particular model. The dependent functions $\mathbf{x}(t)$ are known as state functions, and the independent $\mathbf{u}(t)$ as control functions.

For example, in a model of an epidemic disease, the state functions might be the populations, at time t, of those who are susceptible to the disease, those who are immune from the disease, and those who are recovered from the disease. Control functions might include a vaccination rate and a quarantine rate, both functions of time. One possible goal would then be to find a public health policy represented by control functions that minimize both the number of infectious persons, and the cost of implementing the policy.

There are two general approaches to optimal control. These are often labeled as direct and indirect methods. An indirect method transforms the problem into a boundary value problem (BVP), which can then be solved analytically or numerically using well-known techniques for differential equations. An excellent introduction to this method can be found in a recent text by Suzanne Lenhart and John Workman [1].

In a direct method, optimal control is seen as a standard optimization problem: perform a search for the control function $\mathbf{u}(t)$ that optimizes the objective functional. Evolutionary algorithms, such as Differential Evolution (DE) [2], are powerful global optimizers, but they do not operate on infinite-dimensional spaces. So before optimizing, a parameterization method is required, whereby the functions of continuous time can be approximated by a discrete set of parameters.

Typically, direct methods using DE simply discretize the control function space. That is, control functions are approximated using a piecewise constant parameterization. When the optimal solution is continuous, then, one must choose between accuracy and efficiency. A large number of parameters will converge slowly to an accurate approximation of the true solution, while a small number will converge quickly to a poor approximation.

In this paper, a new direct method is developed for optimal control, using DE in conjunction with Bézier curves to parameterize the control functions. The new method is designed to be both accurate and efficient simultaneously. Part 2 of the paper examines evolutionary direct methods in general. In Part 3 the Bézier parameterization is developed for use with DE. Part 4 applies the method to several optimal control problems. The focus here is to confirm that this new direct method is effective and efficient for a broad range of problems. In each case, the examples used can be solved analytically by an indirect method. This permits comparison of the two solutions, and validates the method. Finally, Part 5 looks ahead to future implementations and applications.

2 Direct methods for optimal control

2.1 Direct vs. indirect methods

There are certain mathematical advantages to using an indirect method, including existence and uniqueness results, exact solutions when the BVP can be solved analytically, and error estimates when it is solved numerically [1]. There are also several limitations which can be overcome by a direct method.

First among the limitations of an indirect method is that each solution is problem-specific. A separate mathematical transformation must be derived for each distinct optimal control problem, and in some cases the mathematics can be rather complicated. A direct method, on the other hand, is a more universal solution, which can be easily and quickly applied to a new control problem.

Second, in an indirect method, the transformation requires that the optimal control problem be formulated with a single objective functional. When there are multiple objectives, they must be collected into one. By contrast, direct evolutionary methods can use a multiobjective global optimizer. One numerical run can produce a range of solutions that can be considered mutually optimal in some sense [3].

Third, applied optimal control problems often have multiple, complicated constraints. In the indirect method, these are be difficult to impose, and can lead to intractability [4n]. In a direct evolutionary method, constraints are easily imposed with a penalty function. Examples are given below.

Fourth, because the indirect method relies on variational calculus, it is of necessity a local optimization method. But complicated systems sometimes have multimodal landscapes with many local optima. In these cases, a global, evolutionary optimization scheme can be more effective [5].

2.2 Evolutionary direct methods

Evolutionary Algorithms (EAs) are powerful, global optimizers, that treat optimization from the perspective of natural evolution: an initial population of feasible solutions evolves into a population of globally near-optimal solutions. There are typically two mechanisms by which new feasible solutions are formed: mutation (small perturbations in a binary- or real-valued individual) and recombination (combining the characteristics of two different individuals). Some form of natural selection is used to decide which population members "survive" to the next generation, and after many generations the population converges, to one or several near-optimal solutions.

There are two types of EA, distinguished by the way in which they represent individual feasible solutions. Genetic Algorithms (GAs) [6] use binary representation, and are thus suitable for discrete or integer optimization problems. Evolutionary Strategies (ESs) [7] use real-valued vectors, and are better suited for the kind of continuous parameter optimization required for optimal control.

DE emerged in the 1990s as one of the most impressive ESs, converging faster and with more certainty than many other acclaimed global optimization methods [3]. In the years since, it has successfully been used in many different applied fields [2]. DE has been shown to be a robust and efficient global optimizer for an evolutionary direct approach – in a variety of specific applications [8]-[11], and more generally for optimal control problems that have multimodal control function landscapes [5].

2.3 Control Vector Parameterization

To use an ES for optimal control, a parameterization strategy is required by which control functions can be represented by the $\mathbf{R}^n$ vectors on which DE operates. This is known as Control Vector Parameterization (CVP). A wide variety of CVPs have been used with non-evolutionary optimizers, including piecewise constant [12], Chebyshev polynomials [13], Lagrange polynomials [14], and piecewise Lagrange polynomials [15].

Direct methods using DE have been less creative, relying almost exclusively on piecewise constant CVP. Each DE-based solution referenced above [5], [8]-[11] approximates the control as a piecewise constant function. The reason may be that it is the easiest parameterization to encode, or it may be that current researchers are simply following the path trod by those who first applied EAs to optimal control [16], [17]. In any case, there is room for improvement.

The limitations of a piecewise constant approximation are obvious: a very high number of parameters is needed for an accurate approximation. However, EAs are computationally expensive, and require a small number of parameters to converge to a near-optimal solution within a reasonable amount of time.

Thus, a more creative CVP is desirable for evolutionary direct methods. To be effective, the CVP should be able closely to approximate arbitrary, continuous, control functions. To be efficient, it must do so with a relatively small number of parameters. Also, CVPs that increase the nonlinearity of the objective function can lead to epistasis [18] – the nonlinear and interdependent manner in which the objective function relates to the design parameters. Small changes in several variables can result in large changes in the objective function. Epistatic functions can lead to premature convergence, because they provide so few clues as to the location of the global minimum. In general, a reduction of this nonlinear interaction, by having parameters more directly linked to the objective function, will enable the optimizer to converge more quickly.

3 Bézier parameterization for Differential Evolution

3.1 Bézier Control Parameterization

P. Bézier, of the French firm Regie Renault, pioneered the use of computer modeling of surfaces in automobile design. His UNISURF system, initiated in 1962 and used by designers since 1972, has been applied to define the outer panels of several cars marketed by Renault [19], [20]. The foundations of Bézier curves, however, go back much further. In 1926, S. Bernstein presented a constructive proof of the Weierstrass approximation theorem [21], using functions that have become

known as Bernstein polynomials. Bézier curves have a very similar form, and are sometimes referred to as Bézier-Bernstein polynomials.

Bézier curve parameterization is used regularly in engineering applications, such as shape optimization. It has been used effectively with DE to optimize turbomachinery airfoils [22]. An extensive search of the literature, however, suggests that this is the first use of a Bézier CVP for optimal control by any direct method, whether evolutionary or non-evolutionary.

An *n*th order Bézier curve, **P**(*z*), is defined parametrically using *n*+1 two-dimensional control points $\mathbf{P}_i(t_i, u_i)$, as follows:

$$\mathbf{P}(z) = \sum_{i=0}^{n} \mathbf{P}_i \frac{n!}{i!(n-i)!} z^i (1-z)^{n-i}, \; 0 \le z \le 1, \qquad (2)$$

where *z* is the parameter. Bézier curves begin at control point $\mathbf{P}_0$, end at control point $\mathbf{P}_n$, have initial slope equal to that of the line segment $\mathbf{P}_0\mathbf{P}_1$, ending slope equal to that of $\mathbf{P}_{n-1}\mathbf{P}_n$, and always lie within the convex hull formed by the control points. The curve is *n*th order continuous throughout and never oscillates wildly away from its defining control points. Thus Bézier curves can parameterize smooth, non-oscillatory functions, with minimal epistasis, using only a few parameters.

The Bézier Control Parameterization (BCP) introduced here is designed for a single control function. A fixed, regular mesh is used on the *t*-axis. This forces the curve to be single-valued, and also reduces the dimension of the optimization vectors to *n*+1. That is, the BCP $\mathbf{u} = [u_i]_{i=0}^{n}$ completely encodes a control function *u*(*t*) as the *n*th order parametric Bézier curve $u(t) = \langle t(z), u(z) \rangle$, as follows:

$$\left\{ \begin{array}{l} t(z) = \sum_{i=0}^{n} (t_0 + i\Delta t) \frac{n!}{i!(n-i)!} z^i (1-z)^{n-i} \\ u(z) = \sum_{i=0}^{n} u_i \frac{n!}{i!(n-i)!} z^i (1-z)^{n-i} \end{array} \right\}, \; 0 \le z \le 1, \; (3)$$

where $\Delta t = (t_f - t_0)/n$, t_0 is the initial time, and t_f is the final time.

The objective function, *F*(**u**), is computed as follows. The control function *u*(*t*), is found using the Bézier curve parameterization. It is stored as a set of data points, at parameters *z*=0,*h*,2*h*,…,1. A step-size of *h*=0.01 is used here, and can be refined when more accuracy is required. The IVP is then solved numerically for *x*(*t*), interpolating the data points to approximate *u*(*t*) as necessary. The differential equation solver used is MATLAB's ode45 function, an explicit Runge-Kutta (4,5) formula, with the Dormand-Prince pair. Finally, the objective integral is evaluated, again interpolating to approximate *x*(*t*) and *u*(*t*), as necessary. The numerical integration routine is MATLAB's quad function, a recursive adaptive Simpson quadrature. The value of the integral is the "cost" *F* of the vector **u**.

3.2 Differential Evolution

Optimization of the function *F*(**u**) is performed using DE, which minimizes the cost of a population of vectors **u**. The crucial difference between DE and other ESs lies in mutation. ESs normally use predetermined probability distribution functions to perturb vectors, leaving them unable to adapt the perturbation magnitude to the topology of the objective function. DE uses the "differential" of two randomly chosen population vectors, $\mathbf{u}_a$ and $\mathbf{u}_b$, to perturb a base vector $\mathbf{u}_c$, $\mathbf{u}_{new} = \mathbf{u}_c + F(\mathbf{u}_a - \mathbf{u}_b)$, where *F* is the differential weight. The perturbation magnitude is thus automatically appropriate to the given landscape, and the search is less random, being dictated by the shape of the objective function itself. This property of DE is known as self-organization. Ultimately, it results in better convergence properties as the algorithm nears the global minimum.

Two DE strategies are used here. In DE/local-to-best/1, the base vector is a combination of one randomly chosen vector and the vector with the lowest objective function value. *F*=0.85 is the recommended differential weight. This strategy tends to balance robustness with fast convergence, and has been demonstrated as one of the more effective DE strategies [23]. Usually a population size of *NP*=10*D* is effective, where *D*=*n*+1 is the dimension of the vector **u**. Occasionally, when misconvergence occurs, *NP* needs to be increased.

For small population sizes, a fast convergence strategy is DE/best/1 with jitter. Here the base vector is the best one in the population, which tends to reduce robustness. When the problem dimension and the population size are small, this loss can be balanced by jittering, the practice of generating a different value of *F* for each parameter. This results in small, random variations in both scale and orientation of the differential [2].

4 Results

Below we consider several representative problems from [1]. The purpose here is to demonstrate that the DE/BCP direct method can find accurate solutions to the standard range of optimal control problems. Thus, in addition to one problem in standard form, also considered are examples with payoff terms, with fixed state endpoints, and with bounded controls. Both minimization and maximization problems are considered. Most have a single state function, but for completeness the final example has multiple state functions. Each test case has a single control function, but the method can be extended to solve problems with multiple controls.

All problems considered have continuous optimal controls, and can be solved analytically. This allows validation of the BCP, by comparing its result with the exact solution. Details of the exact solutions can be found in [1].

4.1 Standard form

The standard form for optimal control problems is (1). In the following example, there is one control function *u*(*t*), and one state function *x*(*t*):

$$\min_u F(u) = \int_0^1 \left(3x(t)^2 + u(t)^2\right)dt,$$
$$\text{subject to } \begin{cases} x'(t) = x(t) + u(t) \\ x(0) = 1 \end{cases}. \quad (4)$$

The analytical solution is:

$$u(t) = \frac{3e^{-4}}{3e^{-4}+1}e^{2t} - \frac{3}{3e^{-4}+1}e^{-2t}$$
$$x(t) = \frac{3e^{-4}}{3e^{-4}+1}e^{2t} + \frac{1}{3e^{-4}+1}e^{-2t} \quad (5)$$

The BCP solution has n=3, representing four Bézier control points. DE parameters are DE/best/1/jitter, F=0.85, CR=1, with population size NP=15. The initial population is formed by random selection of control parameters, within the bounds [-3,0]. Optimization is terminated after 50 generations. The jitter strategy, in this case, was very effective, converging to an excellent solution in under two minutes, using a current Intel system. The control points for the BCP solution are shown in Table 1.

Table1. BCP solutions to optimal control problems of part 4. The Bézier curve control function is defined by (3) for unconstrained problems; and (3) and (11) for constrained problems.

Optimal control problem	Degree n	Bézier Control Parameterization u_0	u_1	u_2	u_3
(4)	3	-2.7774	-0.9353	-0.5457	0.0043
(7)	3	-2.0017	-1.3363	-0.9849	-0.7331
(9)	3	-0.7052	-0.5110	0.6082	1.2481
(12)	2	4.7188	-1.5990	-0.8749	
(13)	2	14.726	-0.1696	0.7300	
(15)	2	3.0036	1.4924	0.0044	

4.2 Payoff term

In some optimal control problems, there can be one objective over the entire time interval, and a second objective at a specific time, usually the final time t_f. The first is represented by an integral, and the second by a function of the time t_f. The two are typically combined into one objective functional through a weighted sum,

$$\min_{\mathbf{u}} F(\mathbf{u}) = \int_{t_0}^{t_f} f_1(t, \mathbf{x}(t), \mathbf{u}(t))dt + af_2(t_f), \quad (6)$$

where a is the weight of the second objective relative to the first. The term outside the integral, $af_2(t, \mathbf{x}(t), \mathbf{u}(t))$, is known as a payoff term. These might be necessary when, for example, a second objective is to minimize the final population.

In this test case, the integral objective depends only on the control, and the payoff term, $x(1)^2$, depends only on the state:

$$\min_u F(u) = \frac{1}{2}\int_0^1 u(t)^2 dt + x(1)^2,$$
$$\text{subject to } \begin{cases} x'(t) = x(t) + u(t) \\ x(0) = 1 \end{cases}. \quad (7)$$

The analytical solution is: $u(t) = -2e^{-t}$, $x(t) = e^{-t}$.

The BCP solution has n=3. The DE strategy used is DE/best/1/jitter, F=0.85, CR=1. Population size was increased to NP=25, to improve the global convergence. The initial population is formed by random selection of control parameters, within the bounds [-5,5]. Optimization is terminated after 50 generations. The BCP solution (Table 1) once again closely approximates the actual solution.

4.3 Fixed state endpoint

In standard optimal control problems, the state equations are initial value problems. But in some cases, the state is fixed not only at its initial point, but also its endpoint. Such is the case in the third example:

$$\min_u F(u) = \int_0^4 u(t)^2 + x(t)dt,$$
$$\text{subject to } \begin{cases} x'(t) = u(t) \\ x(0) = 0, \;\; x(4) = 1 \end{cases}. \quad (8)$$

The actual solution is: optimal control $u(t) = (2t-3)/4$ and optimal state $x(t) = (t^2 - 3t)/4$.

When finding the solution by an indirect method, the fixed endpoint can be included in the transformation to a BVP. However, the DE/BCP direct method uses a Runge-Kutta initial value problem solver, which cannot handle the fixed state endpoint. Thus it is necessary to reformulate the problem. One way to do this is to formulate the state equations as a constrained initial value problem, with the fixed state endpoint as the constraint.

Evolutionary algorithms typically deal with constraints by using a penalty function, in which a numerical penalty is added to the objective function whenever the solution doesn't meet the constraint. Penalties imposed are proportional to the extent to which the constraint is violated. The penalty function formulation of (8) is as follows:

$$\min_u F(u) = \int_0^4 u(t)^2 + x(t)dt \;+\; \mu\left|x(4) - 1\right|,$$
$$\text{subject to } \begin{cases} x'(t) = u(t) \\ x(0) = 0 \end{cases}, \quad (9)$$

where μ is a scaling constant, representing the weight of the penalty relative to the objective functional.

The BCP solution to (9) has n=3. The DE strategy used is DE/best/1/jitter, F=0.85, CR=1, NP=25, terminating after 50 generations. The penalty scaling constant was μ=10. The BCP solution (Table 1) closely approximates the actual, linear solution, and has a state endpoint of $x(4)$=0.99996. Endpoints closer to the required value of $x(4)$=1 could be achieved by increasing the scaling constant.

4.4 Constrained control

In first three sample problems, the control function was unconstrained. Many models, however, required upper or lower bounds on the control. In such cases, the typical evolutionary approach to constrained optimization is to introduce a penalty function. For example, if $u_{UB}(t)$ is an upper bound on control function $u_i(t)$, one could add an integral penalty to the objective function:

$$\min_{\mathbf{u}} F(\mathbf{u}) = \int_{t_0}^{t_f} f_1(t, \mathbf{x}(t), \mathbf{u}(t))dt + \mu \int_{u_i \geq u_{UB}} \left(u_i(t) - u_{UB}(t)\right) dt \,. \quad (10)$$

This approach is inadequate for the BCP method, because the solutions to constrained control problems are often non-differentiable. Bézier curves, on the other hand, are not only differentiable, but smooth, having continuous derivatives of all orders.

An alternate approach is simply to redefine the control, piecewise, to equal the constraint whenever the Bézier curve violates it. That is, the BCP method computes the Bézier curve, $u_{Bez}(t)$, as usual from (3), but the control function itself is defined as

$$u(t) = \begin{cases} u_{Bez}(t), & u_{LB} \leq u_{Bez}(t) \leq u_{UB} \\ u_{LB}, & u_{Bez}(t) < u_{LB} \\ u_{UB}, & u_{Bez}(t) > u_{UB} \end{cases}, \quad (11)$$

where upper and lower bounds, u_{UB} and u_{LB}, can be functions or constant.

In the following example, the control is bounded from above and below:

$$\max_u F(u) = \int_0^2 2x(t) - 3u(t) - u(t)^2 dt,$$
$$\text{subject to } \begin{cases} x'(t) = x(t) + u(t) \\ x(0) = 5 \end{cases}, \quad (12)$$
$$\text{such that } 0 \leq u(t) \leq 2.$$

Note that while DE solves minimization problems, (12) is a maximization problem. It is converted to the dual problem by minimizing $-F(u)$.

The actual solution is defined piecewise on three intervals, $I_1 = [0, 2 - \ln(4.5))$, $I_2 = [2 - \ln(4.5), 2 - \ln(2.5)]$, and $I_3 = (2 - \ln(2.5), 2]$, with:

$$u(t) = \begin{cases} 2, & t \in I_1 \\ e^{2-t} - 5/2, & t \in I_2 \\ 0, & t \in I_3 \end{cases}, \text{ and}$$

$$x(t) = \begin{cases} 7e^t - 2, & t \in I_1 \\ \left(7 - 81e^{-2}/8\right)e^t - e^{2-t}/2 + 5/2, & t \in I_2 \\ \left(7 - 7e^{-2}\right)e^t, & t \in I_3 \end{cases}.$$

For this problem, it is sufficient to use three Bézier control points for an excellent solution. However, both the population size and maximum number of generations had to be increased to find a global solution. This resulted in the DE/best/1/jitter strategy being rather inefficient. A better result was obtained with DE/local-to-best/1, *NP*=40, *F*=0.85, *CR*=1, and 100 generations. An excellent solution is obtained, shown in Fig. 1, with BCP solution given in Table 1.

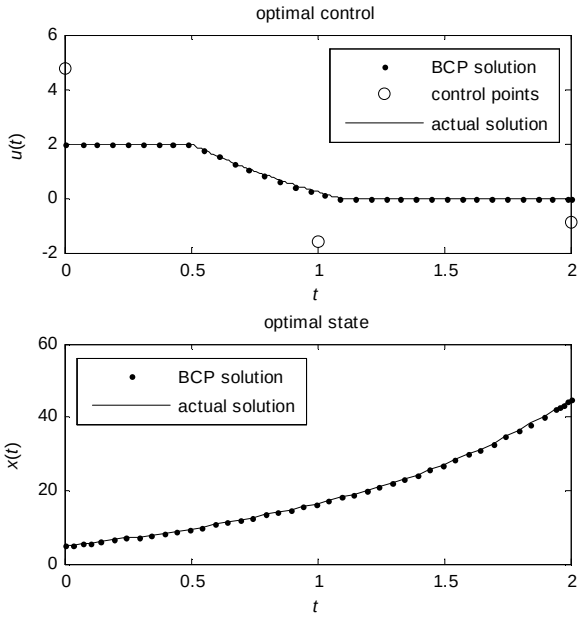


Fig. 1. Solutions of (12), an optimal control problem with constraints on the control. The BCP solution found by DE is compared to the actual, analytic solution.

Our experience generally with DE has led us to conclude that the strategy DE/local-to-best/1 is effective for a wide variety of problems, nicely balancing robustness with fast convergence. Normally a population size of *NP*=10*D* works well, but in this case, where dimension was *D*=3, *NP* had to be increased slightly to 40. In practice, when the actual solution is unknown, the results of several population sizes should be compared, to ensure a global solution.

4.5 Constrained control with payoff

In this example, we demonstrate a BCP solution for a problem with mixed constraints, both an upper bound on the control and a payoff term for the state:

$$\max_u F(u) = x(4) - \int_0^4 u(t)^2 dt,$$
$$\text{subject to } \begin{cases} x'(t) = x(t) + u(t) \\ x(0) = 0 \end{cases}, \quad (13)$$
$$\text{such that } u(t) \leq 5.$$

The actual solution has optimal control and state:

$$u(t) = \begin{cases} 5, & 0 \leq t < 4 - \ln(10) \\ e^{4-t}/2, & 4 - \ln(10) \leq t \leq 4 \end{cases},$$

$$x(t) = \begin{cases} 5e^t - 5, & 0 \le t < 4 - \ln(10) \\ -e^{4-t}/4 + \left(5 - 25e^{-4}\right)e^t, & 4 - \ln(10) \le t \le 4 \end{cases}.$$

The BCP solution uses three Bézier points, with DE strategy DE/local-to-best/1, *NP*=40, *F*=0.85, *CR*=1, and 100 generations. The result is a BCP solution (Table 1) closely approximating the actual solution (Fig. 2).

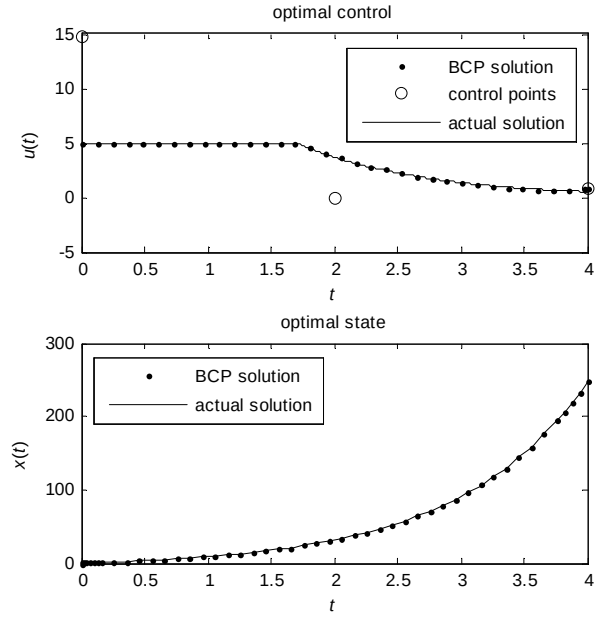


Fig. 2. Solutions of (13), an optimal control problem with mixed constraints. The BCP solution found by DE is compared to the actual, analytic solution.

4.6 Multivariable optimal control

All of the above examples have been for optimal control of a single differential equation in one state variable. The extension of the BCP/DE solution method to a system of differential equations is essentially trivial. The Runge-Kutta solver, used to solve one equation in the previous test cases, is designed for any number of differential equations. Thus no additional code is required when there are multiple state functions, as long as there remains only one control function.

For the sake of completeness, however, we demonstrate the ability of the BCP/DE technique to solve an optimal control problem with multiple state variables. In the following problem, there are two state variables, one of which has fixed initial and ending point:

$$\min_u F(u) = \int_0^1 x_2(t) + u(t)^2\, dt,$$
$$\text{subject to } \begin{cases} x_1'(t) = x_2(t), & x_1(0) = 0,\ x_1(1) = 1 \\ x_2'(t) = u(t), & x_2(0) = 0 \end{cases}. \quad (14)$$

The actual solution is $u(t) = 3 - 3t$, $x_1(t) = 3t^2/2 - t^3/2$, $x_2(t) = 3t - 3t^2/2$.

As above, the fixed endpoint for the first state variable is incorporated into the objective function with a penalty formulation, so that the problem is reformulated as follows:

$$\min_u F(u) = \int_0^1 x_2(t) + u(t)^2\, dt + \mu\left|x_1(1) - 1\right|,$$
$$\text{subject to } \begin{cases} x_1'(t) = x_2(t), & x_1(0) = 0 \\ x_2'(t) = u(t), & x_2(0) = 0 \end{cases}. \quad (15)$$

Three Bézier control points are used in the solution, with DE strategy DE/local-to-best/1, *NP*=25, *F*=0.85, *CR*=1, and 50 generations. The penalty scaling constant is $\mu = 10$. The BCP solution (Table 1) is again in excellent agreement with the actual solution (Fig. 3).

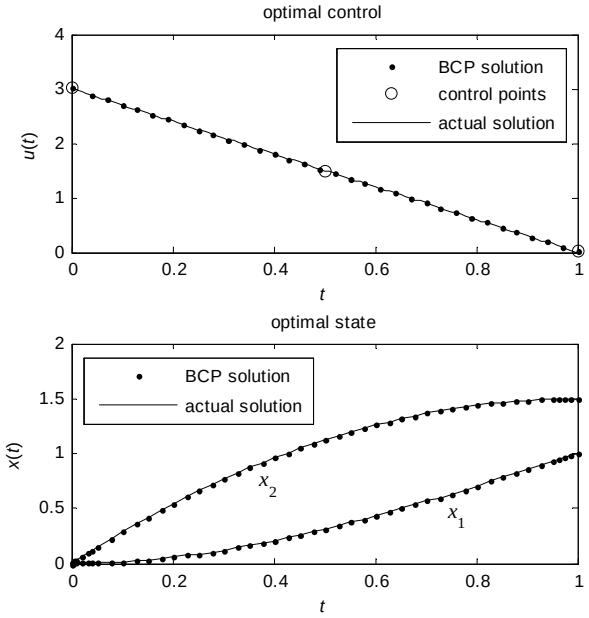


Fig. 3. Solutions of (15), an optimal control problem with multiple state functions. The BCP solution found by DE is compared to the actual, analytic solution.

5 Conclusions

The BCP method proves successful for each optimal control problem, producing an accurate approximation of the true solution, using a small number of parameters. For evolutionary direct methods, it provides a means to improve both accuracy of the final result and efficiency of the algorithm. It has been demonstrated that the technique is effective for all classes of optimal control problems.

The direct method developed here has potential to be a simple, general solution method for any optimal control problem. This can be extremely helpful in the field of epidemiological and biomedical modeling, in which researchers requiring an optimal public health policy or optimal treatment schedule may not have the mathematical skills, or the time, to solve the model indirectly. In other work, we intend to demonstrate the efficacy of the BCP method for these types of models. Many of these have multiple control functions, e.g. vaccination rate, quarantine rate, and isolation rate for an

epidemic. Thus the method will have to be extended to allow for multiple controls.

The true value of a direct evolutionary method, of course, is not to reproduce known solutions to optimal control problems, but to provide an alternate solution method for problems that are difficult or impossible to solve indirectly. Having validated the method generally, it is to these that attention can now be turned, particularly problems that are multiobjective, that are multimodal, and that have complicated constraints.

Acknowledgment

The author thanks Dr. Abba Gumel, professor of mathematics at University of Manitoba, for suggesting the problem of optimal control, motivated by his work in epidemiological modeling.

References

[1] Suzanne Lenhart and John T. Workman, *Optimal Control Applied to Biological Models*. Boca Raton: Chapman & Hall/CRC, Taylor & Francis Group, 2007.

[2] Ken Price, Rainer Storn, and Jouni Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. New York: Springer, 2005.

[3] Kalyanmov Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York: John Wiley & Sons, 2001.

[4] John McCall, "Genetic algorithms for modelling and optimization," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205-222, Dec. 2005.

[5] I.L. Lopez-Cruz, L.G. Van Willigenburg, and G. Van Straten, "Efficient differential evolution algorithms for multimodal optimal control problems," *Applied Soft Computing*, vol. 3, no. 2, pp. 97–122, Sept. 2003.

[6] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading: Addison-Wesley, 1989.

[7] Hans-Georg Beyer: *The Theory of Evolution Strategies*. New York: Springer, 2001.

[8] J.P. Chiou and F.S. Wang. "A hybrid method of differential evolution with application to optimal control problems of a bioprocess system," *Proc. IEEE International Conference on Evolutionary Computations, IEEE World Congress on Computational Intelligence*, New York, 1998, pp. 627–632.

[9] M.H. Lee, Ch. Han, and K.S. Chang, "Dynamic optimization of continuous polymer reactor using a modified differential evolution algorithm," *Ind. Eng. Chem. Res.*, vol. 38, no. 12, pp. 4825–4831, Dec. 1999.

[10] I.L. Lopez-Cruz, L.G. van Willigenburg, and G. van Straten. "Optimal control of nitrate in lettuce by a hybrid approach: differential evolution and adjustable control weight gradient algorithms," *Computers and Electronics in Agriculture*, vol. 40, nos. 1-3, pp. 179-197, Oct. 2003.

[11] M.D. Kapadi and R.D. Gudi, "Optimal control of fed-batch fermentation involving multiple feeds using differential evolution," *Process Biochemistry*, vol. 39, no. 11, pp. 1709–1721, July, 2004.

[12] C.J. Goh, K.L. Teo, "Control parametrization: a unified approach to optimal control problems with general constraints," *Automatica*, vol. 24, no. 1, pp. 3–18, Jan. 1988.

[13] J. Vlassenbroeck, "A chebyshev polynomial method for optimal control with state constraints," *Automatica*, vol. 24, no. 4, pp. 499–506, July 1988.

[14] L. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Comp. Chem. Engng.*, vol. 8, nos. 3-4, pp. 243–248, 1984.

[15] V.S. Vassiliadis, R.W.H. Sargent, and C.C. Pantelides, "Solution of a class of multistage dynamic optimization problems 1. Problems without path constraints," *I&EC Res.*, vol. 33, no. 9, pp. 2111–2122, Sept. 1994.

[16] S. Smith, "An Evolutionary Program for a class of continuous optimal control problems," *Proc. IEEE Conference on Evolutionary Computation*, vol. 1, Piscataway, pp. 418–422, 1995.

[17] N.V. Dakev, A.J. Chipperfield, and P.J. Flemming, "A general approach for solving optimal control problems using optimization techniques," *Proc. IEEE Conference on Systems, Man and Cybernetics*, part 5, Vancouver, pp. 4503–4508, 1995.

[18] Bäck, Thomas *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press, 1996.

[19] P. Bézier, *Numerical Control – Mathematics and Applications*, translated by A.R. Forrest and A.F. Pankhurst, London: John Wiley & Sons, 1972.

[20] P. Bézier, "Mathematical and Practical Possibilities of UNISURF," in *Computer-Aided Geometric Design*, R.E. Barnhill and R.F. Riesenfeld, Eds. New York: Academic Press, 1974, pp. 127-152.

[21] P.J. Davis, *Interpolation and Approximation*. New York: Blaisdell Publishing Company, 1963.

[22] T. Rogalsky, R.W. Derksen, and S. Kocabiyik, "Differential Evolution in Aerodynamic Optimization," *Canadian Aeronautics and Space Journal*, vol. 46, no. 4, pp. 183-190, Dec. 2000.

[23] A. Auger, N. Hansen, J.M. Perez Zerpa, R. Ros, M. Schoenauer. "Experimental Comparisons of Derivative Free Optimization Algorithms" in *Lecture Notes in Computer Science, Vol. 5526: Proceedings of the 8th International Symposium on Experimental Algorithms*, pp. 3-15. New York: Springer-Verlag, 2009.

The Use of Evolutionary Algorithms in the Analysis of Economics Experiments

Esmail Bonakdarian
Department of Computing Sciences and Mathematics
Franklin University
Columbus, Ohio, USA

Abstract— *We report on the application of an evolutionary algorithm in the analysis of data from experiments in economics. The algorithm was used in conjunction with regression analysis to evaluate various variable subsets and find those sets best capable of explaining the experimental outcomes. This evolutionary computation approach is offered as a supplemental method to generate optimal parsimonious subsets according to user specified criteria, or to validate subsets generated by more traditional means such as stepwise regression.*

The evolutionary algorithm is based on the Cross-generational elitist selection, Heterogeneous recombination, Cataclysmic mutation algorithm (CHC), a type of genetic algorithm. This stochastic, population-based approach offers a highly flexible, directed, user specified exploratory search guided by means of our evaluation function and offers practitioners one more selection tool for their analysis of data. We believe that this approach has the potential to find subsets otherwise missed by the more traditional and deterministic approaches.

Keywords: Genetic Algorithms, Evolutionary Algorithms, CHC, Optimization, Regression Analysis

1. Introduction

Every day researchers are confronted by large sets of survey or experimental data and faced with the challenge of "making sense" of this collection and turning it into useful knowledge. This data usually consists of a series of observations over a number of dimensions, and the objective is to establish a relationship between the variable of interest, the response or dependent variable, and other variables, the explanatory/independent variables, for purposes of prediction or exploration. In some cases researchers want to limit themselves to only a subset of the independent variables. Preferably this subset would consist of as few, but statistically significant, variables as needed to account for as much of the observed behavior of the response variable as possible. Additionally, smaller models are desirable because they lend themselves to easier interpretation.

The traditional tool for finding and establishing these relationships has been regression analysis. As long as the number of independent variables is relatively small, or the experimenter has a fairly clear idea of the possible underlying relationship, it is feasible to derive the best model using standard software packages and methodologies. However, if the number of independent variables is large, and there is no intuitive sense about the possible relationship between these variables and the dependent variable, the experimenter may have to enter an exploratory mode to discover the important and relevant independent variables. During this "fishing expedition" a number of models are fitted and compared, often through some automated processes. Two methods are commonly used: One is a form of stepwise regression that selectively adds or removes independent variables from the regression equation, each time evaluating the contribution of a given variable to the model. The other is an "all-possible" or best-subset regression in which an exhaustive search tries to evaluate all possible combinations of variables.

While stepwise regression is a computationally efficient process, it is not guaranteed to always find the best solution because the search may get trapped in a local optimum rather than finding the global optimum when dealing with many variables[1] in a high-dimensional, multi-modal search space. Additional problems may exist with multicollinearity in the data. In contrast, the best-subset regression method of executing an exhaustive search of all possible models should be able to locate the optimal solution. However, unless the quantity of independent variables is rather small, such as 30 to 40[2], it will become impractical from a computational point of view as the number of combinations quickly explodes with the increase in the number of independent variables and other selection techniques have to be considered.

Principal Components Analysis (PCA) is often used in cases of models with many coefficients. It is a data reduction method for multidimensional data that works by placing related variables into groups called components, and allows the exploration of a relationship between these components and the dependent variable. While it deals effectively with a large model by reducing its dimensionality, its ability, in the best case, lies mainly in forecasting outcomes rather than shedding light on the contribution of individual variables in the model. As the economists doing these studies were primarily interested in being able to explain behavior based on the experimental data collected, rather than prediction,

the PCA method was not a suitable way to proceed.

This paper presents the practitioner with another alternative: Instead of having to rely on a form of stepwise or best-subset regression, the use of an evolutionary algorithm is offered as a way to "evolve" the best minimal subset with the largest explanatory value. This approach offers more flexibility as the user can specify the exact search criteria (e.g., the F-statistic, the Mean Squared Error [MSE], the Adjusted R^2, and others) on which to optimize the model. The user can then examine a ranking of the top M (a user-specified parameter) models found by the system. In addition to these measures, the algorithm can also be tuned to limit the number of variables in the final model. In a second experiment we evolved a solution based on subsets of the complete model, providing an even more focused search in addition to the specific search criteria, i.e., base statistic. While not implemented in this version of the algorithm, it would be trivial to adjust the evaluation function to give certain variables extra weight and consideration and allow them to be treated specially during the search by, for instance, favoring them if the investigator decided a priori that she wanted to keep certain variables in the final model. Furthermore, the algorithm can also be used to validate results found by more traditional means.

We believe that this ability to direct the search provides flexibility to the analyst and results in models that provide additional insights.

The evolutionary computation approach was used in two economics studies described below.

This paper is structured as follows: Section 2 briefly outlines the two economics studies where this evolutionary computation approach was applied. Section 3 describes the original CHC algorithm and what distinguishes it from the standard genetic algorithm as postulated by Holland[3]. In section 4 we will outline the basic experimental setup for this study. Section 5 offers our observations and conclusions followed by some suggestions for future work.

2. Economics Studies

Both studies[4], [5] dealt with the well known classical "public goods problem"[1] from economics: When goods are provided to a larger community without required individual contributions it often results in "free-riding."

There has long been an assumption in economics that people tend to "free ride" when public goods are available. However, people also tend to show a willingness to cooperate and sacrifice for the good of the group[6]. This seemingly contradictory behavior brings up a number of interesting questions for economists studying the behavior of people under these circumstances.

[1]Public goods are goods whose use is non-excludable and non-rivalrous and that therefore pose the problem of "free-riding." For more information see [6].

The goal of the first study was to determine the difference in contributions between excludable and non-excludable public goods and whether the ability to vote on decisions would make a difference in the average contributions. The data for this study was collected in 2007 at St Lawrence University, NY. The experiment, in which 220 undergraduate students participated, yielded data for 21 variables. Limiting these data to simple interactions and squared terms resulted in a set of more than 200 variables for analysis[4]. As there was no preconceived notion regarding the relationship between the observed behavior and the very large set of independent variables, the evolutionary computation approach was used to find a small subset of statistically significant coefficients, and to eliminate redundant variables.

Using data from the same experiment mentioned above, a second experiment investigated the effect of voting and excludability on individual contributions to group projects, and both excludable and non-excludable public goods were considered. Since, like the first experiment, no preconceived notion regarding the relationship between the observed behavior and the independent variables existed, the evolutionary computation approach was used for this study as well[5].

For both of the studies, once all experimental variables, their squares and various interactions between them were taken into account, the maximum model had over 200 coefficients due to the use of the flexible functional form. An exhaustive search of this space would have been impossible since there would have been in excess of 2^{200} or more than 10^{60} different models to be considered!

Given this very large set of variables, our aim was to eliminate redundant factors and discover the statistically significant coefficients. Since enumerating all possible combinations was out of the question due to its prohibitive cost from a computational point of view, the evolutionary computation approach was chosen to tackle this problem. Our ultimate goal was to discover the simplest model with the greatest explanatory power.

3. CHC

The properties of the canonical simple genetic algorithm as first described by John Holland[3] and later David Goldberg[7] are well known: the algorithm takes a population of N potential solutions and proceeds to evaluate, select, mate and mutate its members to generate a new child population of size N, which then is subsequently destined to be evaluated, selected, mated and mutated again, and so on. In each generation, the parent population of candidate solutions is completely replaced by the new child population. The selection, recombination and mutation operations all underlie probabilistic rules, and the algorithm continues to evolve ever more fit[2] populations over a number of

[2]Elitism, always ensuring that the most fit member is passed on to the next generation, prevents any worsening of the solution, assuming the goal doesn't change.

generations until some termination condition is met, such as a minimum acceptable fitness level, or a fixed number of iterations have elapsed.

One of the major problems many evolutionary algorithms face is premature convergence as the population of candidate solutions tends to become homogeneous over time and thereby loses some of its exploratory quality. For genetic algorithms this is a natural consequence of the pressure exerted by the selection process. The mutation operation is intended to help counter this tendency to converge too quickly by maintaining a certain level of diversity in the population through randomly perturbing some of the candidate solutions during each generation. Attempting to find an effective balance between exploration and exploitation remains a challenge for evolutionary algorithms, along with some questions about the exact nature of the operations and their relation to exploration vs. exploitation[8].

The particular genetic algorithm chosen for this application, the Cross-generational elitist selection, Heterogeneous recombination, Cataclysmic mutation algorithm (CHC) as introduced by Eshelman[9] is a non-traditional genetic algorithm that addresses these problems in an effective and unique manner.

Eshelman characterizes CHC as "a conservative selection algorithm working in conjunction with a radical (i.e., highly disruptive) recombination operator." The following pseudo code illustrates the basic steps of this algorithm:

```
  CHC Pseudo Code:

  Threshold = L/4  # L = length of binary strings
  Parent_pop contains N candidate solutions

  while not Terminate:
    Child_pop = {}

    for i = 1 to N/2:
      select 2 random members p1, p2 from Parent_pop

(1)   if (Hamming_Distance(p1, p2)/2) > Threshold:
        mate p1, p2 using HUX and generate c1, c2
        Child_pop.append(c1, c2)

(2) if size(Child_pop) == 0:
      Threshold = Threshold - 1
    else:
(3)   Parent_pop = best N of Parent_pop + Child_pop

(4) if Threshold < 0:
      cataclysmic mutation # rate 35 percent
      Threshold = .35 * (1 - .35) * L
```

This algorithm differs from the standard canonical genetic algorithm in a number of ways.

First, survival-selection rather than reproduction-selection is used. Instead of probabilistically selecting more fit population members for reproduction, in CHC, members are randomly selected from the available parent pool P and *only* mated if they are sufficiently different, see (1) in pseudo code above. Eshelman refers to this heterogeneous recombination as incest prevention. Using a binary string representation, a dynamically changing threshold (2) for the Hamming distance is used as a diversity measure for the two parents. It is therefore quite possible for one generation to pass without any offspring being produced if the randomly selected parents for mating are too similar, which would be an indication of the population P becoming less diverse. In this case the next generation will work with the same population again. On the other hand, given a very diverse population, up to N new offspring may be generated in a single generation at time t. In this case the next generation is the result of selecting the best N members of the current parent $P(t-1)$ and new child population $C(t)$, see (3). In the case of the parent and child having the same fitness value, Eshelman favors the parent. This joint, cross-generational selection is reminiscent of the truncation selection used in $(\mu + \lambda)$ evolution strategies[10], [11]. This is in contrast with the generational replacement scheme in the simple genetic algorithm where the child population always takes the place of the parent population. Therefore this cross-generational, implicitly elitist selection from a pool ranging in size from N to $2N$ guarantees that the overall fitness of the population can not decrease. Mating, if it occurs between two sufficiently diverse parents, is accomplished through the use of the highly disruptive Half-Uniform Crossover (HUX) operator, which randomly swaps half of the differing bits between the parents.

Another major difference is the absence of chronic mutation in the CHC algorithm; i.e., mutation isn't part of each generational cycle. Rather, it is applied in a radical fashion to restart the search periodically when the population is deemed to have become too homogeneous as determined by the Hamming distance between randomly selected pairs of members from the current population. As mentioned above, it is possible that no offspring are created during a generation when no sufficiently different parents are found for mating and indicates that the population is converging. A threshold counter tracks the number of childless generations (2) and eventually triggers the cataclysmic mutation (4) that reseeds the population by using the best (i.e., most fit) member found so far as a template for the other $N - 1$ new members that are generated based on mutations from this elite member. By keeping a sufficiently diverse population, a certain degree of exploratory search is ensured. Therefore a mutation rate of 35% was set, along with a population size of 50, both of which were empirically established and recommended by Eshelman for the CHC, and also used for these experiments.

4. Experimental Setup

The algorithm was written in the R scripting language, which is part of the open source R statistics package[12]. As the data was already stored in a format easily accessible to R and our evaluation functions were based on the outcome of regressions, a functionality readily provided by R, we decided to use its programming environment for our

implementation. We did not view the interpretive nature of the language to be a problem as we consider the exploratory quality of this current prototype approach to be its main feature and benefit.

Each candidate solution was represented by a binary string of length c, where c was the number of coefficients in the maximum model. There was a one-to-one mapping between each coefficient and each position in the bit-string, resulting in a very close genotype to phenotype linkage, with a "1" implying inclusion in, and a "0", exclusion from the current regression equation for the given coefficient in the candidate solution. The resulting equation was fed to the native R regression function and the necessary data to compute our fitness score was extracted from the output of the regression.

While mostly true to the algorithm described by Eshelman, our implementation of the CHC algorithm differed in three minor aspects: (a) if, during the selection process of the best N members from the combined parent-child population, two members, a parent and a child, were found with the same fitness value, we did not always favor the parent as was prescribed in the original CHC algorithm, (b) we slightly varied the trigger point and mechanism responsible for starting the cataclysmic mutation to help fine-tune our algorithm in our first study, and finally (c) our re-initialization of the new threshold counter in the first study re-used the original initialization $L/4$ value rather than the modified formula that incorporated the 35% mutation rate.

4.1 Fitness Functions

The regression function formed the core of our fitness evaluation, which was embedded within our version of the CHC algorithm. We used the output of the regression and made it the basis for assessing the fitness of our candidate solutions.

In order to find the most useful and effective evaluation criteria, we implemented a variety of fitness functions, some of which took the regression output values directly without modifying them further while others manipulated the outputs to see if any qualitative improvements could be made. Our base function consisted of:

$$fitness\_val = base\_statistic + *\_modifier \quad (1)$$

where the $*\_modifier$, if used, consisted of one of these: $delta\_modifier$, $percent\_modifier$ or $delta\_percent\_modifier$ as described below.

The overall goal of our evaluation functions was to find the model with the fewest terms, yet with the highest explanatory power according to the user specified criteria. Optimizations were run over the F-statistic, the Mean Squared Error (MSE), Adjusted R^2, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and the log Likelihood (logLik) measures along with the modifiers. Additionally, the number of variables in the final model could also be limited.

4.1.1 Pure

The "pure" approach examined the regression output statistics without manipulating them any further and used them as the basis for the fitness assessment:

$$fitness\_val = base\_statistic \quad (2)$$

For the first experiment (generalized least squares estimation) we used the F-statistic, the Mean Squared Error (MSE) and the Adjusted R^2. The second experiment (mixed-effects model maximum likelihood estimation) used the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and the log Likelihood (logLik) measures as the basis of evaluation. These decisions were dictated by the statistical approach chosen for analysis and demonstrate the flexibility of the system to work with different fitness criteria.

The following modifiers took the pure statistic and introduced additional, qualitatively motivated enhancements, to help explore possibly more effective and meaningful fitness measures.

4.1.2 Delta

The purpose of the "delta" method was to exert pressure toward generating simpler models by reducing the number of variables in the model as a larger number of coefficients not only represent a challenge in terms of computation, but are also harder to interpret. This was done through the use of a constant multiplier that rewarded shorter equations. Our modifier value (delta) was computed using:

$$delta\_modifier = (MAX - terms) * DELTA \quad (3)$$

and then added to the base statistic as shown in (1). Here, and subsequently, MAX refers to the maximum number of variables in the complete model, $terms$ refers to the number of variables in the current candidate model, and $DELTA$ is a small penalty constant heuristically determined after a few trial runs to establish a sufficiently optimal value. Therefore, given two candidate models with the same base fitness value, the one with fewer coefficients would be scored higher by this penalty function.

4.1.3 Percent

The "percent" approach considered the fraction of significant coefficients in the model as defined by their p-value.

Table 1: Significance levels and weights

p-value	weight
0.001	1.0
0.01	0.05
0.05	0.01
0.1	0.001

Table 1 shows the weights associated with each coefficient's p-value as reported by the regression output.

The sum of weights resulting from evaluating the candidate model's regression output was used to compute the $percent\_modifier$ as shown below and then added to (1).

$$percent\_modifier = sum\_weight/terms \quad (4)$$

The underlying motivation was that this would favor models with a more statistically significant set of explanatory variables, yielding presumably more *meaningful* models that would therefore be of greater interest to the researcher.

4.1.4 Delta-Percent

Finally, our "delta-percent" method combined the above mentioned delta and percent approaches shown in (3) and (4) in an attempt to find models with a smaller set of variables while at the same time yielding models that contained highly statistically significant predictor variables.

$$delta\_percent\_modifier = delta\_modifier + percent\_modifier \quad (5)$$

As before, this modifier would be added to (1).

4.2 Optimization Runs

Both experiments shared much of the same runtime parameters and environment. This section summarizes what was common to them.

Including the base statistics, we had a total of 12 fitness functions available for each of the economics experiments after we applied the delta, percent, and delta-percent modifiers to the base statistic.

We ran 15 trials for each of the 12 fitness measurements. For each trial, our CHC algorithm ran for 2500 generations, with a population of 50 candidate solutions. The mutation rate was set at 35% as recommended by Eshelman. While 35% would seem like a very large mutation rate for a traditional genetic algorithm, we remind the reader that mutation in the CHC algorithm only occurs periodically to restart a stalled search, and not on an ongoing basis as with the standard genetic algorithm.

Most of the solutions converged between 1500 and 2000 generations and very little improvement was observed beyond 2200 iterations of the algorithm. Figure 1 shows one of the optimization runs. As can be seen in the figure, initially the optimization proceeds at a quick rate, and then slowly continues to improve over time until it converges. The top curve shows the optimum value found so far, while the lower, jagged line shows the current average fitness value for the population in each generation. As the search is periodically restarted in response to the population growing too homogeneous, we can clearly see that this average fitness value abruptly drops due to the cataclysmic mutation, but then recovers and leads to improvements of the best results due to the newly diversified population, until the search converges.

Next we provide information specific to each experiment.

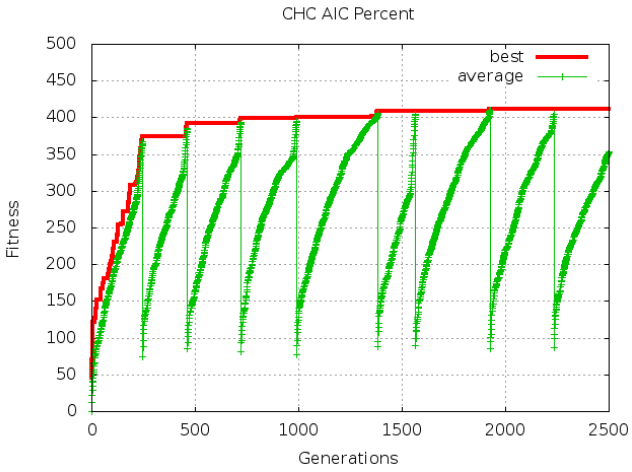


Fig. 1: Optimizing AIC-Percent

4.2.1 Experiment 1

The first economics experiment used the F-statistic, the Mean Squared Error (MSE) and the Adjusted R^2 as the base fitness measures as the economists were interested in optimizing the models according to these statistics. Accordingly our fitness functions were set up to that end. Briefly, the F-statistic evaluates the overall fit of the current model by looking at the ratio of two variances (one explained by parameters in the model, the other due to residuals), and the MSE is a measure of error between the data and the regression equation. R^2 measures the percent of variation in the dependent variable that can be accounted for by the regression, while the Adjusted R^2 assesses the fit of the model by also considering the complexity, i.e., the number of parameters in the model.

Also, while exploring various ways of fine-tuning the algorithm, we slightly modified the trigger mechanism for the cataclysmic mutation by using the average of the Hamming distance as the trigger for the reboot of the search.

At the end of the search we displayed a user specified number of the top M models found, ranked by the specific base statistic and modifier selected.

4.2.2 Experiment 2

The base fitness measures for the second experiment as specified by the economists consisted of the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and the log Likelihood (logLik) statistic. AIC and BIC are similar to each other in that they both consider the complexity of the model in their computation, but BIC will impose a larger penalty when the number of regressor variables is large. The log Likelihood measure compares the current model to a null model: the lower the log likelihood, the better the model.

In addition to the approach employed in the first exper-

iment, we optimized our search by subsets for this study. Our maximum model consisted of 240 coefficients. The economists decided to optimize along two different sets of parameters and therefore split these variables into three mutually exclusive subsets. A common base set (set_1) contained 190 variables, and sets 2_1 and 2_2 each contained 20 variables[3]. Our optimization trials combined set_1 with one of the other sets (i.e., 2_1 or 2_2) but not both, resulting in two alternative "sub-models" for optimization.

The optimizations would either be run against the common base set_1 and set 2_1, or set_1 and set 2_2. After the optimization found the optimum model using the 220 coefficients from the combined set and generated a ranking of the best M models as specified, it also proceeded to evaluate the alternative model (using the common base set, but the subset not used during the evolution of these results). That is, each of these top M candidate solutions from the final generation were also used to evaluate the alternate model consisting of the common base set and the unused subset to see if any new insights could be gained by examining these additional, alternative models.

The ability to specify different subsets of the model to help guide the search provided more flexibility to the researchers exploring their data.

5. Conclusions and Future Work

An evolutionary computation approach was used to help analyze data from two economics experiments. The main advantage this approach offers to researchers is the flexibility to direct the search for models that can explain their experimental data. This method allows the user to request optimization according to specific statistics, and to enhance these searches by taking both the size and percentage of significant variables in the candidate models into account. The results are presented in a ranking of the top models found according to the user-specified search criteria. Finally, it is also possible to optimize the search over subsets of the maximum model, offering yet more flexibility. All of this, we believe, offers the economists help with their search for meaningful regression models that might have otherwise been missed.

The evolutionary computation approach has several properties that make it an attractive option. First, the fact that these evolutionary algorithms can easily be used with different fitness criteria (i.e., evaluation functions) demonstrates the flexibility of the system as it allows the user a great deal of leeway in how to guide the (evolving) search and adjust the system to meet the user's specific needs for a given situation. For instance, by adjusting parameters and the evaluation function, the quantity and quality of results, in terms of the number of coefficients and their significance levels, can be easily adjusted to help locate the right balance between the two factors. Second, evolutionary algorithms are inherently suitable for parallel or distributed execution. Given the right platform, this would allow for the simultaneous evaluation of many candidate solutions, i.e., models, in parallel, greatly speeding up the work. Third, while it is possible for the search to get trapped in a local optimum, the stochastic nature of the search would make this unlikely with repeated runs. Finally, this approach offers an alternative to an exhaustive search that would be infeasible with a larger number of coefficients in a model as this algorithm only needs to sample a small subset of the total search space in order to reach its goal; therefore the use of a heuristic approach in the face of an exploding exponential number offers an attractive alternative even if it is unable to always guarantee an optimal solution.

While more work needs to be done to be able to generalize these findings, we believe the results were sufficiently encouraging to motivate further investigation. Future investigations of this approach may include the following:

The exploration of other run-time parameters for the CHC genetic algorithm, including dynamic self-adapting parameters during run-time, such as the mutation rate, but also the delta penalty value, and the weights assigned for the assessment of the coefficients' significances merit investigation. Additionally, the use of alternate evolutionary algorithms, such as Particle Swarm Optimization[13], [14], [15], and how they might be applied to this problem domain should be examined to determine if these can produce similar or better results, and perhaps more effectively.

The issue of how to deal with large sets of independent variables during regression analysis presents a serious challenge, especially when trying to reduce the number of variables to the smallest meaningful subset. Stepwise regression, while computationally efficient, can not always guarantee the optimum solution. An exhaustive search via "all-possible" regression is an option, especially if the number of explanatory variables is small. However, once the number of variables grows, this method quickly becomes computationally infeasible, especially considering that each additional variable doubles the search space, and other selection tools and techniques will have to be considered. The goal of analyzing experimental data is to gain an understanding of the processes that generated this data and the method outlined in this paper offers one such approach by using a flexible, directed exploratory search.

6. Acknowledgments

This paper draws from the work done with my colleagues Hans J. Czap and Natalia V. Czap in [4], [5]. I want to thank them for providing the dataset and their valuable comments on the paper. This work was also supported in part by an allocation of computing time from the Ohio Supercomputer Center. The author also thanks St. Lawrence University for the use of its computing cluster for the first study.

[3]That both of these subsets had the same cardinality is not significant.

References

[1] B. Zhang and S. Horvath, "Ridge regression based hybrid genetic algorithms for multi-locus quantitative trait mapping," *International Journal Bioinformatics Reaserch and Applications*, vol. 1, no. 3, 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*, 2nd ed. Springer, Feb. 2009. [Online]. Available: http://www.worldcat.org/isbn/0387848576

[3] J. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, USA: University of Michigan Press, 1975.

[4] E. Bonakdarian, H. Czap, and N. Czap, "Selection of minimal variable subsets with the help of genetic algorithms in economic analysis," *Journal of Interdisciplinary Economics*, vol. 21, pp. 407–425, 2009.

[5] H. Czap, N. Czap, and E. Bonakdarian, "Walk the talk? The effect of voting and excludability in public goods experiments," *Economic Research International*, vol. 2010, December 2010, doi:10.1155/2010/768546.

[6] J. Andreoni, "Cooperation in public-goods experiments: Kindness or confusion?" *American Economic Review*, vol. 85, no. 4, pp. 891–904, 1995.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Wesley Professional, 1989.

[8] A. E. Eiben and C. A. Schippers, "On evolutionary exploration and exploitation," *Fundam. Inf.*, vol. 35, pp. 35–50, August 1998. [Online]. Available: http://portal.acm.org/citation.cfm?id=297119.297124

[9] L. J. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," in *FOGA*, 1990, pp. 265–283.

[10] I. Rechenberg, *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.

[11] H.-P. Schwefel, "Evolutionsstrategie und numerische optimierung," Ph.D. dissertation, Technische Universität Berlin, 1975.

[12] R Development Core Team, *R: A Language and Environment for Statistical Computing*, http://www.R-project.org/, R Foundation for Statistical Computing, Vienna, Austria, 2010, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, August 1995, pp. 1942–1948.

[14] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998, pp. 69–73.

[15] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, ser. EP '98. London, UK: Springer-Verlag, 1998, pp. 611–616. [Online]. Available: http://portal.acm.org/citation.cfm?id=647902.739129

Algorithmic Bounded Rationality In The Iterated Prisoner's Dilemma Game

Christos A Ioannou[1] **and Ioannis Nompelis**[2]
[1]Economics Division, University of Southampton, Southampton, United Kingdom
[2]Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, Minnesota, USA

Abstract—*A genetic algorithm is used to simulate the evolution of* Moore machines *in the iterated Prisoner's Dilemma stage-game. The machines are prone to two types of errors: (a) implementation errors and (b) perception errors. We conduct computational experiments that incorporate different levels of errors in an effort to assess whether and how the distribution of machines in the population changes. In sharp contrast to previous studies, the incorporation of implementation and perception errors is sufficient to reduce cooperative outcomes. In addition, the study identifies a threshold error-level. At and above the threshold error-level, the prevailing machines converge to the open-loop machine Always-Defect. On the other hand, below the threshold, the prevailing machines are closed-loop and diverse. The diversity thus impedes our inferential projections on the superiority of a particular machine.*

Keywords: Genetic Algorithm, Automata, Prisoner's Dilemma

1. Introduction

Our objective is to use the genetic algorithm to simulate an evolving, error-prone population of agent-based strategies that plays the iterated Prisoner's Dilemma (PD) paradigm. According to the thought experiment, a group of agents is to play the PD game. The Prisoner's Dilemma payoff-matrix is provided in Table 1. Each agent is required to submit a strategy that is implemented by a type of finite automaton called a $Moore\ machine$ [1]. The machine specifies actions contingent upon the opponent's reported actions. The agents play the PD game against each other and against their twin in a round-robin structure. With the completion of all round-matches, the actual scores and machines of every agent become common knowledge. Based on this information, agents update their machines for the next generation via the genetic algorithm. Bounded rationality is introduced in the form of $implementation\ errors$ and $perception\ errors$. Implementation errors are errors in the implementation of actions. On the other hand, perception errors are errors in the transmission of information. The computational experiments conducted, incorporate different levels of errors in an effort to assess whether and how the distribution of outcomes and strategies in the population changes. In addition, behavioral patterns that fare well in the simulated environments are identified and discussed.

Table 1: Prisoner's Dilemma Matrix

	Cooperate	Defect
Cooperate	3,3	0,5
Defect	5,0	1,1

The genetic algorithm [2] is one of many search techniques developed for solving hard combinatorial optimization problems in large search spaces. Other optimization techniques include: Simulated Annealing [3], Tabu Search [4], Stochastic Hill Climbing and Compset Algorithm [5]. Axelrod [6] was the first to model the evolutionary process of the iterated PD game with a genetic algorithm. The winning strategy in his tournament was Tit-For-Tat (TFT); a strategy that starts off by cooperating and then imitates the most recent action of the opponent. Nevertheless, Axelrod's study was restricted by his use of error-free strategies whose actions were contingent to the action profiles of (only) the last three periods, and by his use of a fixed environment composed of (only) eight strategies. On the other hand, here, we circumvent these restrictions by the use of a variable environment where strategies co-evolve as the strategic population changes. In addition, we incorporate bounded rationality in the form of implementation and perception errors.

Bendor, Kramer and Stout [7] have been, to our knowledge, the first to conduct a computer tournament with random shocks. In their study, the authors re-evaluate the performance of reciprocating strategies such as TFT and identify alternative strategies that sustain cooperation in an environment with random shocks. The winning strategy in their tournament is Nice-And-Forgiving (NAF) which differs in many ways from TFT. First, NAF is nice in the sense that it cooperates as long as the frequency of cooperation of the opponent is above some threshold. Second, NAF is forgiving in the sense that although NAF retaliates if the opponent's cooperation falls below the threshold level of cooperation, it reverts to full cooperation before its opponent does, as long as certain minimal levels of cooperation are met by the opponent.

On the other hand, the results of the present study point to a very different direction from that in Axelrod [6] and Bendor, Kramer and Stout [7]. Here, we show that the evolution of cooperative machines is considerably weaker while the change in the model is ecologically plausible: errors are common in our strategic interactions. In addition, by varying the error-level, the study identifies a threshold error-level. At and above the threshold error-level, the prevailing structures converge to the one-state, open-loop machine Always-Defect: a relentless punisher. Yet, below the threshold, the prevailing machines are cooperative, closed-loop and diverse. These findings enable us to deduce that strategic simplification is a necessary condition *only* in the error-prone environments. In the presence of errors, behavior is governed by mechanisms that restrict the flexibility to choose potential actions. These mechanisms simplify behavior to less complex patterns (rules of thumb), which are easier for an observer to recognize and predict. In the absence of errors, the behavior of well-informed agents responding with flexibility to every perturbation in the environment may not produce easily recognizable patterns. The diversity thus impedes our inferential projections on the superiority of a particular machine.

The contribution of this paper is two-fold. First, the study aims to elicit an understanding of the patterns of reasoning of agent-based behaviors that emerge in adaptive systems in the presence of errors. To this extend, we discern behavioral patterns that fare well in the error-prone environments. Second, the study also contributes to a better understanding of how small error-perturbations in the agents' strategies change the set of prevailing structures.

2. Moore Machines

A finite automaton is a mathematical model of a system with discrete inputs and outputs. The system can be in any one of a finite number of internal configurations or "states". The state of the system summarizes the information concerning past inputs that is needed to determine the behavior of the system on subsequent inputs. The specific type of finite automaton used here is a Moore machine [1]. Let I denote the set of agents, A^i denote the set of $i's$ actions, A denote the cartesian product of the action spaces A^i written as $A \equiv \underset{i=1}{\overset{I}{\times}} A^i$, and $g^i : A \rightarrow \Re$ denote the real-valued utility function of i. Thus, a *Moore machine* for an adaptive agent i in a repeated game of $G = (I,\{A^i\}_{i\in I},\{g^i\}_{i\in I})$ is a four-tuple (Q^i, q_0^i, f^i, τ^i) where Q^i is a finite set of internal states of which q_0^i is specified to be the initial state, $f^i : Q^i \rightarrow A^i$ is an output function that assigns an action to every state, and $\tau^i : Q^i \times A^{-i} \rightarrow Q^i$ is the transition function that assigns a state to every two-tuple of state and other agent's action.

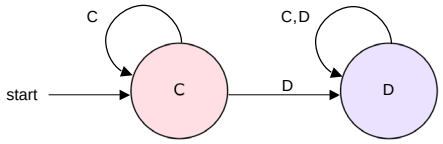


Fig. 1: Grim-Trigger Machine

$Q^i = \{q_C, q_D\}$
$q_0^i = q_C$
$f^i(q_C) = C$ and $f^i(q_D) = D$
$\tau^i(q, a^{-i}) = \{^{q_C \quad (q,a^{-i})=(q_C,C)}_{q_D \quad otherwise}$

For example the machine (Q^i, q_0^i, f^i, τ^i) in Figure 1, carries out the Grim-Trigger strategy in the context of the PD game. Thus, the strategy chooses "cooperate" so long as both agents have chosen "cooperate" in every period in the past, and chooses "defect" otherwise.

Bounded rationality is introduced in the form of random errors committed by the machines. More specifically, the study considers errors in the implementation of actions and errors in the perception of actions. Implementation and perception errors when considered in isolation lead to quite different results. For instance, the machine Contrite-Tit-For-Tat in the iterated PD game is proof against errors in implementation but not against errors in perception. The machine acts in principle as Tit-For-Tat, but enters a "contrite" state if it erroneously implements a defection rather than a cooperation. Consequently, the machine accepts the opponent's retaliation and cooperates for the next two periods but leaves the contrite state soon after. On the other hand, if the machine Contrite-Tit-For-Tat mistakenly perceives that the opponent defected, will respond with a defection without switching to the contrite state and will not meekly accept any subsequent retaliation. It is therefore crucial to formally define implementation and perception errors in the context of Moore machines.

Definition 1 *The machine of agent i in the PD game commits an implementation error with probability ϵ, when for any given state q, the machine's output function returns the action $f^i(q)$ with probability $1-\epsilon$ and draws another action "$f^i(q)$" where $f^i(q)\neq$"$f^i(q)$" otherwise.*[1]

That is, an implementation error level of ϵ indicates that with probability ϵ the course of action dictated by the particular state of the machine will be altered. For example, a cooperation dictated by the particular state will be

[1]A general definition would postulate that *the machine of agent i commits an implementation error with probability ϵ, when for any given state q, the machine's output function returns the action $f^i(q)$ with probability $1-\epsilon$ and draws another action $a^i \in A^i \setminus f^i(q)$ randomly and uniformly otherwise.* Yet, since the action space in the PD game consists of only two actions, the former definition suffices.

implemented erroneously as a defection with probability ϵ. On the other hand, perception errors are defined as follows.

Definition 2 *The machine of agent i in the PD game commits a perception error with probability δ, when for any given opponent's action a^{-i}, the machine inputs the opponent's action a^{-i} into the transition function with probability $1-\delta$ and inputs the opponent's action "a^{-i}" into the transition function* where $a^{-i} \neq$ "a^{-i}" otherwise.

Thus, a perception error level of δ indicates that with probability δ an opponent's action is reported incorrectly, while with probability $1-\delta$ the opponent's action is perfectly transmitted.

Furthermore, we consider machines that hold no more than eight internal states. The choice to keep the upper bound on the number of internal states at eight is reasonable given complexity considerations. As Rubinstein [8] indicates, agents seek to device behavioral patterns which do not need to be constantly reassessed and which economize on the number of states needed to operate effectively in a given strategic environment. A more complex plan of action is more likely to break down, is more difficult to learn, and may require more time to be executed. In fact, a number of studies (some with subjects in the laboratory) have been suggestive of the effectiveness of simple strategies over more complex ones in a wide range of environments ([9]; [10]; [11]; [12]).

3. Genetic Algorithm

The genetic algorithm is an evolutionary search algorithm that manipulates important schemata based on the mechanics of natural selection and natural genetics. Other descriptive constructs, such as replicator dynamics or evolutionary stable strategies, lack the ability to incorporate forms of innovation. The present search algorithm however, removes this restriction by allowing for innovative processess to enter the model in a tractable manner. The genetic algorithm was developed by Holland [2] for optimization problems in difficult domains. Difficult domains are those with both enormous search spaces and objective functions with many local optima, discontinuities and high dimensionality.

The search for an appropriate way to model strategic choices of agents has been a central topic in the study of game theory. The genetic algorithm is an attractive choice because it combines survival of the fittest with a structured information exchange that emulates some of the innovative flair of human search. The mechanics of the genetic algorithm involve copying strings and altering states through the operators of selection and mutation. Initially, reproduction is a process where successful strings proliferate while unsuccessful strings die off. Copying strings according to their payoff or fitness values is an artificial version of Darwinian selection of the fittest among string structures. After reproduction, selection results to higher proportions of similar successful strings. The mechanics of reproduction and selection are simple, involving random number generation, string-copying and string-selection. Nonetheless, the combined emphasis of reproduction and the structured selection give the genetic algorithm much of its power. On the other hand, mutation is an insurance policy against premature loss of important notions. Even though reproduction and selection effectively search and recombine extant notions, occasionally they may become overzealous and lose some potentially useful material. In artificial systems, mutation protects against such an irrecoverable loss. Consequently, these operators bias the system towards certain building blocks that are consistently associated with above-average performance.

4. Methodology

The genetic algorithm requires the natural parameter set of the optimization problem to be coded as a finite-length string over some finite alphabet. Each Moore machine here, is thus represented by a string of 25 elements. The first element provides the starting state of the machine. Eight three-element packets are then arrayed on the string. Each packet represents an internal state of the machine. The first bit, within an internal state, describes the action dictated by the particular state $(1 := cooperate, 0 := defect)$. The next element, within an internal state, gives the transition state if the opponent is observed to cooperate, and the final element, within an internal state, gives the transition state if the opponent is observed to defect. Given that each string can utilize up to eight states, the scheme allows the definition of any Moore machine of eight states or less.

For example, take the machine that implements TFT in Figure 2. The machine only needs to remember the opponent's last action hence utilizes only two states; the last six states are redundant as illustrated in the coding.

The genetic algorithm consists of a number of generations. Each generation starts with a given population called the $parent\ population$. A new population of the same size is then constructed called the $offspring\ population$. In this formulation, the genetic algorithm operates with a population of machines. Each machine represents an agent's strategy. Initially, a population of thirty machines is chosen at random. Then, each machine is tested against the environment (which is composed of the other machines and its twin) in a round-robin structure. The game-play occurs for 200 periods per match. Each machine, thus aggregates a raw score based on the payoffs illustrated in Table 1. The offspring population is constructed from the parent population, by selecting the machines that aggregated the top twenty scores. In addition, ten new structures are created via a process of selection and mutation. The process requires the draw of ten pairs of machines from the parent population (with the probabilities

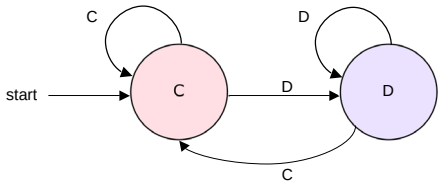


Fig. 2: Tit-For-Tat Machine

biased by their scores) and the selection of the better performer from each pair. Then, these ten machines undergo a process of mutation. Mutation occurs when an element at a random location on the selected string changes value. Each element on the string is subjected to a 4% independent chance of mutation, which implies an expectation of 1 element-mutation per string. The population is iterated for 500 generations. The adaptive plan is summarized below in the pseudocode of Figure 3 and Figure 4.[2]

```
Specify error-level
Fix max-periods = 200

Create initial population: 30 agents (seed randomly)
Initiate round-robin tournament

For t = 1 to 500 do

    For all agent-pairs do
        For p = 1 to max-periods do
            Award utils to each agent based on the PD matrix
        End loop

        Output performance score
    End loop

    Apply subroutine for the offspring-population-creation
    Store agent results

End loop
```

Fig. 3: Pseudocode of the Main Program

```
Sort agents based on performance score

Copy top 20 agents to offspring-population

Select 10 agent-pairs via probabilities biased by performance scores

For each of 10 pairs do
    Create new agent as a copy of the winner of the pair's match
    Mutate new agent by switching one element at random

End loop
```

Fig. 4: Subroutine of the Offspring-Population-Creation

5. Results

In order to assess whether and how the distribution of outcomes and structures in the population changes, we conducted four computational experiments. The computational experiments incorporate different levels of errors. In particular, in the four computational experiments conducted, the machines are subjected to a constant independent chance of implementation and perception errors of 4%, 2%, 1% and 0%, respectively. The results that follow, present the averages over all thirty members of each generation and thirty simulations conducted for each experiment.

[2] A variety of sensitivity analyses have been performed, and confirm that the results reported here, are robust to reasonable changes in these choices.

5.1 Evolution Of Payoffs

Figure 5 shows the average payoff per game-generation over all thirty members under the 4%, 2%, 1% and 0% computational experiments. In the early generations, the agents tend to use machines that defect continuously. The reason is that at the start of the evolution, the machines are generated at random. In such an environment, the best strategy is to always defect. With the lapse of a few generations though, machines in the less error-prone conditions achieve consistent cooperation which allows the payoffs to move higher. The average payoff in the last generation of the 0% treatment is 2.86 utils, whereas the average payoff in the last generation of the 1% treatment is 2.54 utils. The average payoff in the last generation of the 2% and 4% treatments is 1.95 and 1.44 utils, respectively. The paired-differences test establishes that at a 95% level of significance the means of the conditions are statistically different. The results indicate that the incorporation of errors is sufficient to alter the evolution of cooperative outcomes.

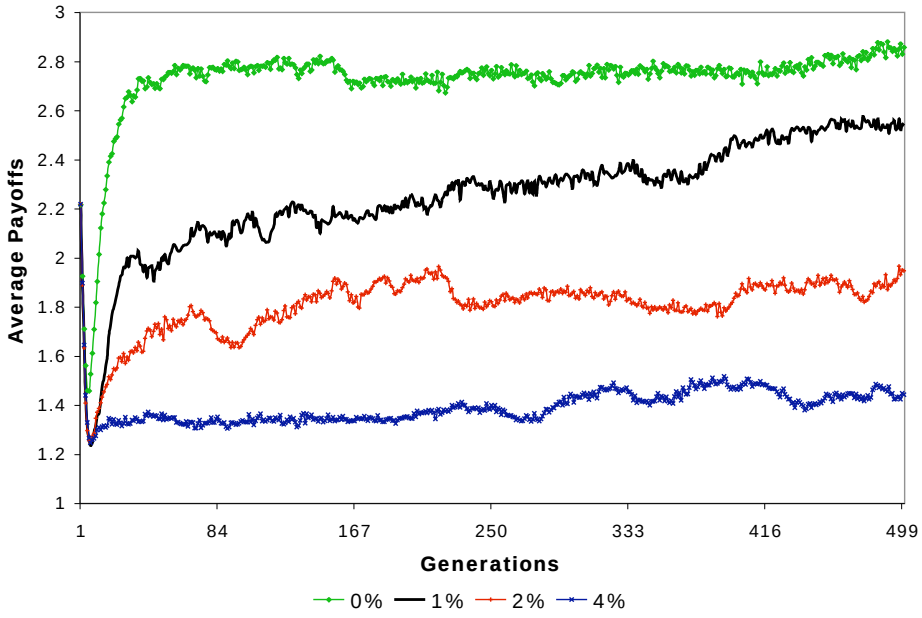


Fig. 5: Average Payoff

5.2 Prevailing Machines

The effect of errors on the structure of the machines is an important question that has not been addressed in the degree we see fit by evolutionary game theorists. Thus, here we investigate behavioral patterns that fare well in the simulated environments. This way a lot can be said about the type of machines that survive, or even the type of machines that do not survive in these environments. The clear winner in the 4% and 2% treatments was the machine Always-Defect. Always-Defect was the winner in 22 out of the 30 simulations run in the 4% treatment, and in 19 out of the 30 simulations run in the 2% treatment. The machine Always-Defect is presented in Figure 6. Always-Defect is an open-loop machine; in other words, the actions taken at any time-period do not depend on the actions of the opponent.

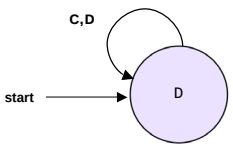


Fig. 6: Always-Defect

On the other hand, the structures that prevailed in the 1% and 0% treatments were diverse. This result halts any possible attempt to discern a particular behavioral pattern that fares well in these specific treatments. Yet, it is noteworthy that unlike the open-loop machine Always-Defect, the diverse array of machines that prevailed in the 1% and 0% treatments were all closed-loop (history-dependent). Thus, the effect of different error-levels on the structure of the machines points towards the existence of a threshold error-level at 2%.

6. Discussion

TFT was the winner in the tournaments with error-free strategies of Axelrod [9]. The performance of TFT lead Axelrod to identify some basic attributes that were necessary for the emergence and survival of cooperation. These were: (i) an avoidance of unnecessary conflict by cooperating as long as the other agent does, (ii) provocation in the face of an uncalled for defection by the other, (iii) forgiveness after responding to a provocation, and (iv) clarity of behavior so that the other agent can adapt to your pattern of action [9]. On the other hand, Bendor, Kramer and Stout [7] incorporated in their computer tournament random shocks. The winning strategy in that tournament was NAF. Yet, the success of NAF is not a robust result but is limited to the particular ecology. As Bendor, Kramer and Stout note, the generosity of NAF creates a risk: other strategies may exploit NAF's willingness to give more than it receives. In other words, NAF can be suckered by a nasty strategy that is disinterested in joint gains.

On the other hand, the results of the present study point to a very different direction from that in Axelrod [9] and Bendor, Kramer and Stout [7]. By varying the error-level, the study identifies a threshold error-level. At and above the threshold error-level, the prevailing structures converge to the open-loop machine Always-Defect. On the other hand, below the threshold, the prevailing machines are closed-loop

and diverse, which impedes our deductive power on the superiority of a particular structure. With sufficient effort though, one might be able to design the optimal strategy for these specific environments. Designing an optimal strategy is a hard problem because its effectiveness depends mostly on the strategies of the other agents involved. One possible approach for dealing with this problem is to endow the agent with the capability of adapting to other agents in the system [13]. The usage of learning techniques for adapting to other agents has received wide attention in the multi-agent system research community (for a survey, see [14]). The research in this field focuses on two central approaches: model-based learning (also known as Opponent Modelling), where an explicit model of the opponent's strategy is generated and exploited ([15]; [16]; [17]; [18]), and model-free learning, where the agent's strategy is directly adapted based on the observed behavior of the opponents [19]. This distinction is also applicable to the more general reinforcement learning problem [20] where both model-based [21] and model-free approaches [22] exist.

Recently, Markovitch and Reger [23] suggested a model-based approach where agents can greatly benefit from adapting to a particular adversary. If however, the learned model is not accurate, then using it to predict the opponent's actions may potentially harm the agent's strategy. In addition, acquiring an accurate model of a complex opponent strategy may be computationally infeasible. To contend with the complexity of learning a full opponent model, the agent learns instead only a certain aspect of the opponent's strategy: the opponent's weakness. More specifically, the agent attempts to characterize the set of states in which the opponent's performance is relatively inferior given that the opponent is a boundedly rational agent, whose quality of decision is not uniform over all domain states. In order to reduce the risk of using a faulty model, the agent uses the model only to bias his actions in a minimally-risky way. Thus, even if the model is not accurate with respect to the opponent's behavior, its use cannot harm the agent's performance significantly. In addition, the agent considers states in which the opponent suffers, but the agent's own strategy is expected to fare well. In other words, the model takes advantage of points at which the agent exhibits a relative advantage over the opponent.

7. Acknowledgements

We are grateful to Aldo Rustichini and Ket Richter for their continuous support and invaluable discussions. We are also indebted to John H. Miller for his comments.

References

[1] Moore, E. Gedanken Experiments on Sequential Machines, in Automata Studies, Princeton University Press: Princeton, New Jersey, 1956.

[2] Holland, J. Adaptation in Natural and Artificial Systems, MIT Press, 1975.

[3] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. "Optimization by Simulated Annealing," *Science* 220, 671Ð680, 1983.

[4] Glover F., and Laguna M. "Tabu Search," In Modern Heuristic Techniques for Combinatorial Problems, C.R. Reeves, 1993.

[5] Hamo, Y., and Markovitch, S. "The Compset Algorithm for Subset Selection," In Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence, 728-733, 2005.

[6] Axelrod, R. "The Evolution of Strategies in the Iterated Prisoner's Dilemma," In Lawrence Davis, Los Altos, California, Morgan Kaufmann, 1987.

[7] Bendor, J., Kramer, R. and Stout, S. "When in Doubt... Cooperation in a Noisy Prisoner's Dilemma," *Journal Of Conflict Resolution* 35, 691-719, 1991.

[8] Rubinstein, A. "Finite Automata Play the Repeated Prisoner's Dilemma," *Journal Of Economic Theory* 39, 83-96, 1986.

[9] Axelrod, R. The Evolution of Cooperation, Basic Books: New York, 1984.

[10] Rust, J., Miller, J. H., and Palmer, R. "Characterizing Effective Trading Strategies," *Journal of Economic Dynamics and Control* 18, 61-96, 1994.

[11] Selten, R., Mitzkewitz, G., and Uhlich, R."Duopoly Strategies Programmed By Experienced Traders," *Econometrica* 65, 517-555, 1997.

[12] Ioannou, C. "Bounded Rationality in Finite Automata." Discussion Papers in Economics and Econometrics, 1019. Southampton: University of Southampton, 2010.

[13] Sen, S., and Weiss, G. Adaptation and Learning in Multi-agent Systems: Lectures Notes in Articial Intelligence, Vol. 1042. Springer-Verlag, 1996.

[14] Sen, S., and Weiss, G. "Learning in Multi-agent Systems," In G. Weiss (ed.): Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press: Cambridge, Massachusetts, 259-298, 1999.

[15] Carmel, D., and Markovitch, S. "Model-based Learning of Interaction Strategies in Multi-agent Systems," *Journal of Experimental and Theoretical Artificial Intelligence* 10(3), 309-332, 1998.

[16] Carmel, D., and Markovitch, S. "Learning Models of Intelligent Agents," In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, 62-67, 1996c.

[17] Freund, Y., Kearns, M., Mansour, D., and Rubinfeld, R. "Efficient Algorithms for Learning to Play Repeated Games Against Computationally Bounded Adversaries," In Proceedings of the 36th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, California, 332-341, 1995.

[18] Stone, P., Riley, P., and Veloso, M. "Defining and Using Ideal Teammate and Opponent Agent Models," In Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), Menlo Park, CA, 1040-1045, 2000.

[19] Uther, W. T. B., and Veloso, M. "Generalizing Adversarial Reinforcement Learning," In Proceedings of the AAAI Fall Symposium on Model Directed Autonomous Systems, 1997.

[20] Kaelbling, L. P., Littman, M. L., and Moore, A.P. "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research* 4, 237-285, 1986.

[21] Moore, A. W., and Atkeson, C. G. "Prioritized Sweeping: Reinforcement Learning With Less Data and Less Time," *Machine Learning* 13, 103-130, 1993.

[22] Watkins, C. J., and Dayan, P. "Q-Learning," *Machine Learning* 8, 279-292, 1992.

[23] Markovitch, S., and Reger, R. "Learning and Exploiting Relative Weaknesses of Opponent Agents," *Autonomous Agents and Multi-agent Systems* 10, 103-130, 2005.

An hybrid genetic algorithm for two-dimensional cutting problems using guillotine cuts

Hamza Gharsellaoui[1], Hamadi Hasni[2]
[1]National Institute of Applied Sciences and Technologies, INSAT, Tunisia
Gharsellaoui.hamza@gmail.com
[2]National School of Computer Science, ENSI, University of Manouba, Tunisia
hamadi.hasni@ensi.rnu.tn

ABSTRACT
The paper deals with the purpose of one hybrid approach for solving the constrained two-dimensional cutting (2DC) problem. We study this hybrid approach that combines the genetic algorithm and the Tabu search method. For this problem, we assume a packing of a whole number of rectangular pieces to cut, and that all cuts are of guillotine type in one sheet of a fixed width and an infinite height. Finally, we undertake an extensive experimental study with a large number of problem instances extracted from the literature by the Hopper's Benchmarks in order to support and to proove our approach and to evaluate the performance.

Key words: guillotine cutting and packing problem, Tabu search, genetic algorithm, guillotine constraint, Hybrid approach.

1 INTRODUCTION

The packing problems have been widely studied during the last three decades, as they are often faced in industry. The rectangular pieces packing problem, cutting also from rectangular board, is one particular case of this set of problems. The aim is often to achieve the minimum trim loss [13]. We had done some studies on packing problems in [14]. In this paper we propose a combination tabu search / genetic algorithms approach to construction of optimization algorithms for problems such as packing problems whose involve constructing an arrangement of items that minimizes the total space required by the arrangement. This is mainly due to the constraints imposed by the industrial applications, e.g. textile, wood, steel and metal industry. A recent survey on packing problems is given in [15]. In this paper, we specifically consider the two-dimensional (2D) rectangular strip packing problem based on a new hybrid approach, named hybrid genetic algorithm. The input is a list of n rectangles with their dimensions (length and width). The goal is to pack the rectangles without overlap into a single rectangle of width W and minimum height H. We further restrict ourselves to the oriented, orthogonal variation, where rectangles must be placed parallel to the horizontal and vertical axes, and the rectangles can be rotated. Further, for our test cases, all dimensions are integers. Like most packing problems, 2D rectangular strip packing (even with these restrictions) is NP-hard. Finally, our algorithm naturally solves a more general problem: given a set of rectangles and a target rectangle, find a packing of a subset of those rectangles which gives an optimal packing of the target. Numerical examples also showed the superiority of the proposed algorithm compared with two classical methods in the literature (pure genetic algorithm and hopper's results). This paper is organized as follows. In Section 2, we provide the problem description and the literature review. In Section 3, we present the resolution methods which uses the bottom left algorithm and the guillotine constraint. In Section 4, we show how our hybrid algorithm can be adapted for solving the general 2DC problem. In Section 5, we

undertake a comparative study of our proposed algorithm and evaluate its performance for the 2DC problem using benchmark problems from the literature. Finally, in Section 6, we summarize the contributions of this paper and explain its possible extensions.

2 PROBLEM DESCRIPTION AND LITERATURE REVIEW

2.1 Definitions

The goal of our research problem is to minimize the waste and the total used space in a set of rectangles and a target rectangle, e.g find a packing of a subset of those rectangles which gives an optimal packing of the target. We have found in our related work on packing problems that cutting and packing (CP) problems have numerous industrial applications, spanning from the direct use of CP models (production, loading cargo into ships, vehicles, containers, wood) to their abstract adaptation to more complex problems (scheduling problems, budgeting, etc). A significant improvement of the proposed algorithm results, over those obtained by either the Hopper results and pure genetic algorithm (GA), has been achieved.

2.1.1 *Genetic Algorithm (GA)*

A genetic algorithm (GA) is a procedure used to find approximate solutions to search problems through application of the principles of evolutionary biology. Genetic algorithms use biologically inspired techniques such as genetic inheritance, natural selection, mutation, and sexual reproduction (recombination, or crossover). For this problem, members of a space of candidate solutions, called individuals, are represented using abstract representations called chromosomes. The GA consists of an iterative process that evolves a working set of individuals called a population toward an objective function, or fitness function. The evolutionary process of a GA is a highly simplified and stylized simulation of the biological version. It starts from a population of individuals randomly generated according to some probability distribution, usually uniform and updates this population in steps called generations. Each generation, multiple individuals are randomly selected from the current population based upon some application of fitness, bred using crossover, and modified through mutation to form a new population [1].

2.1.2 *Tabu Search (TS) Algorithm*

Tabu search is a metaheuristic that guides local heuristic search procedures to explore the solution space beyond local optimality. It was introduced by Glover [2, 3] specifically for combinatorial problems. Since then, tabu search has successfully been applied to a wide range of problems. For example, tabu search has been applied to flow shop scheduling [4, 5], architectural design [6], time tabling problem [7], among others. The tabu search starts at some initial point and then moves successively among neighboring points. At each iteration, a move is made to the best point in the neighborhood of the current point which may not be an improving solution. The method forbids (makes tabu) points with certain attributes with the goals of preventing cycling and guiding the search towards unexplored regions of the solution space. This is done using an important feature of the tabu search method called tabu list. A tabu list consists of the latest moves made so that recently visited points are not generated again. The size of the tabu list can be either fixed or variable.

2.2 Literature Review

Further background Packing problems in general are important in manufacturing settings; for example, one might need n specific rectangular pieces of glass to put together a certain piece of furniture, and the goal is to cut those pieces from the minimum height fixed-width piece of glass. The more general version of the problem allows for irregular shapes, which is required for certain manufacturing problems such as clothing production. However, the rectangular case has many industrial applications [8]. The 2D rectangular strip packing problem has been the subject of a great deal of research, both by the theory community and the operations-research community [9,10]. One focus has been on heuristics that

lead to good solutions in practice. One line in this area considers simple heuristics for greedily placing an ordered list of rectangles, the most widely used and well-studied of which is the Bottom-Left heuristic.

3 RESOLUTION METHODS

In this section, we explain the techniques used to resolve cutting and packing (CP) problems. Because the Bottom-Left heuristic is a foundation for our work, we describe it in some details.

3.1 *Bottom Left (BL) Algorithm*

The Bottom-Left (BL) heuristic was introduced in [11]. The placement Bottom Left (BL) algorithm using the permutation is executed to place rectangular pieces into the main sheet (object). We can describe the BL algorithm process using the following defenition: This Bottom-Left placement algorithm takes a sheet size and an input sequence of pieces and their allowable rotations (rotation criteria). The algorithm progresses packing by placing the first piece in the lower left corner of the sheet in its most efficient orientation (the orientation that yields the smallest bounding rectangle height within the set of rotation criteria). With subsequent pieces, a valid location for placement is found by testing for intersections and containment. If the piece is not intersecting by (or containing) other already placed pieces, then the location of the piece is valid and therefore can be assigned to the sheet. When a piece is in a position that intersects with already assigned pieces, we use the rotation technique. The process continues as before with overlap/intersection tests and resolution until the piece does not intersect and can be placed. Packing is completed when all pieces have been assigned to the sheet.

3.2 *Guillotine Cutting Problems*

The two-dimensional guillotine-cutting problem has been widely studied in the operational research literature. The unrestricted problem is known to be NP-hard. When cutting specific materials like glass it may be required that the rectangles can be cut out of the bin by a number of guillotine cuts which can be thought of as edge-to-edge cuts. The most common constraint requires guillotine cutting patterns; i.e., patterns where pieces can be obtained using a series of horizontal and vertical cuts. This constraint demands that all placed pieces are reproducible through a series of guillotine cuts. As is known, a guillotine cut through a rectangle runs from one edge to the opposite edge and parallel to the other two edges of the rectangle in a straight line. That is, there should exist a series of face parallel straight cuts that can cut the bin into pieces so that each piece contains a box and no box has been intersected by a cut. In our example, the first packing is guillotine cuttable and the solution to obtain a guillotine cut is feasible (Figure 1),

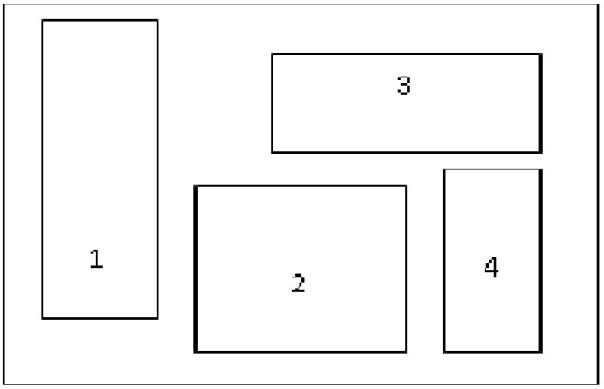


Figure 1: Guillotine Configuration

while the second packing is not and the solution to obtain a guillotine cut is unfeasible (Figure 2).

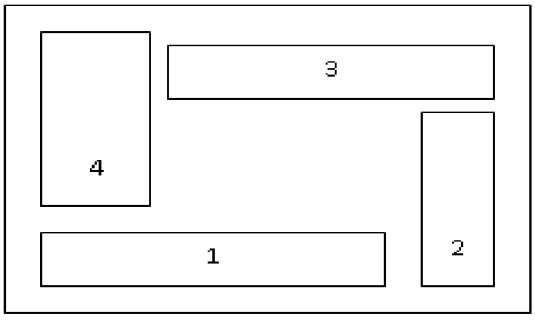


Figure 2: Non-Guillotine Configuration

Notation and example:
Alternatively, a packing pattern (sequence) can be represented by a permutation π [12].
i is the index of the rectangle (r_i).
$\pi = (i_1,...,i_n)$ is a permutation.
The permutation represents the sequence in which the pieces (rectangles) are packed. The advantage of this data structure is the facile creation of new permutations by changing the sequence. A consequence of the variable data structure is the fact that every permutation has to be assigned to a unique packing pattern.
Allocation of some of pieces with BL is illustrated in Figure 3. As can be seen in Figure 3,

In $\pi_1 = (-1, 3, -2, 4)$

piece 1 is placed first to the bottom and then to the left as far as possible with a rotation of 90^o, (the signe - represents a rotation of the piece by 90^o). Pieces 3 and 2 are placed in the same manner with a rotation of the piece 2. Then, piece 4 is placed in its optimal location.

In $\pi_2 = (-1, 2, 3, 4)$

piece 1 is placed first to the bottom and then to the left as far as possible with a rotation. Piece 2 is then placed in its optimal location and pieces 3 and 4 are placed after that.

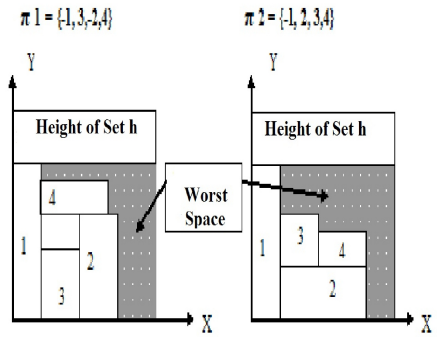


Figure 3: 2 possible permutations of pieces $(1, 2, 3, 4)$.

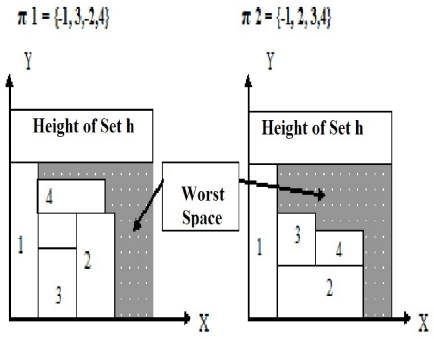
In addition, the cost of the BL-algorithm is Θ(n2). This based on the fact, that each rectangle (piece) r_i can be shifted a maximum of i times, because each shift is limited by one of the i - 1 placed rectangles (pieces) or by the corners of the board. Hence, the cost of placing rectangle r_i is Θ(i) and the whole cost amounts to Θ(n2).

4 HYBRID APPROACH DESCRIPTION

A genetic search algorithm is a heuristic search process that resembles natural selection. There are many variations and refinements, but any genetic algorithm has the features of reproduction, crossover and mutation. Initially a population is selected, and by means of crossovers among members of the population or mutation of members, the better of the population will remain. In the case of our hybrid approach the mutation is replaced by the Tabu Search (TS) Algorithm. For this reason we call this genetic method as an hybrid genetic approach.

4.1 Fitness-Function

For the genetic algorithm, the evaluation of a model set is obligatory, this is represented by a Fitness-Function f: $\pi \rightarrow R_+$ with the propriety f(π_i) $\rangle$ f(π_j) if π_i is better than π_j.
The Fitness-Function value is inversely proportional to the height of a model set: f(π) = 1/h(π); Where h(π) is the model set height follow to the permutation π created by the BL algorithm.
π represents one permutation (arrangement) of rectangular pieces. The following picture (figure 3) is an example of placement(arrangement) of 4 pieces (1, 2, 3, 4).



To reducing the internal waste we have developed two variants of the placement (arrangement) as schown in the figure 3 with the same height of the rectangle (sheet) but different in quality, so by logic they have not the same Fitness Function. With the last definition of the Fitness Function ($f(\pi) = 1/h(\pi)$) if two set placement have the same height, their Fitness Function are equal even if one is better than the other. For this reason, it's necessary to define another Fitness Function more better. Following the last picture (figure 3), it's clear that π_2 is better than π_1 because the remaining space in the result configuration is better with π_2. With this comparaison, and if we consider only one objective which is the wastage minimization. Our Fitness Function is given by the following formula:

F(π) = H(π) + Area (worst remaining space)/h(π)* width

Where:

• H(π) = H - h(π), where H is a maximum given height that guarantee H(π) be a positive value.
• h(π) is the model set height follow to the permutation π.
• Area (worst remaining space) is the area of the remaining space in the placement (arrangement) model following to the permutation π.
• width is a target (rectangle) width dimension.
• h(π)* width is the rectangle area occupied by the placed rectangles (items).

4.2 Initialisation

it's evident that genetic algorithms uses m objects. Here the m objects are m permutations: π_1, π_2,.., π_m. We assign for each arrangement it's Fitness Function value: f_i = f(π_i); i = 1, 2, ...,m.
We consider the permutation π_i and it's Fitness Function f_i together like one individu A_i: A_i = (π_i, f_i).

4.3 Recombination

The crossover operator is special in this case in order to respect the constraint that every piece which is copied didn't repeated another time in order to respect the pieces unicity and creates two solutions or childs (permutations) $\pi_i new$ and $\pi_j new$ by combining two parents π_i and π_j. For each pair of parents (permutations), two crossover integers P_1 and P_2 are randomly chosen (generated) with the condition $1 \leq P_1, P_2 \leq n$. In the random position P_1, the crossover operator copy P_2 elements from π_i for the begining of $\pi_i new$ and copy P_2 elements from π_j for the begining of $\pi_j new$. Then, $\pi_i new$ is completed by the remaining elements of π_j with respect to the same order of appearance and $\pi_j new$ is completed also by the remaining elements of π_i with respect also to the same order of appearance.
Example
$\pi_i = (1, -2, 3, 4, -5, 6)$ and $\pi_j = (-6, 4, 2, 5, -3, 1)$

if ($P_1 = 2$ and $P_2 = 3$) then
$\pi_i new(1) = \pi_i(P_1) = \pi_i(2)$ = -2;
$\pi_i new(2) = \pi_i(P_{1+1}) = \pi_i(3) = 3$;
$\pi_i new(3) = \pi_i(P_{1+2}) = 4$;
$\pi_i new(4) = \pi_j(1)$ = -6;
$\pi_i new(5) = \pi_j(4) = 5$;
$\pi_i new(6) = \pi_j(6) = 1$;
Then, π_i new = (-2, 3, 4, -6, 5, 1). With the same method, π_j new = $(4, 2, 5, 1, 3, 6)$. So, with the crossover processus we get m new permutations.

4.4 Tabu Search

This algorithm was proposed to the cutting problem including the guillotine constraint and replaced the mutation operator and was evaluated in a series of numerical experiments that are run on problem instances taken from the literature, as well as on randomly generated instances which prooved our approach. For this case, we consider the model set X composed by n permutations (arrangements) which are represented by a rectangular items (pieces). We adopt the change of one piece by another of their neighbors as an elementary transformation and we evaluate the Fitness-Function f for each arrangement set.

– The neighborhood function
Let $T_{i,j}$,$(when(i,j) \in [1..n] * [1..n]; i \neq j)$; the transformation that consists of the changement

of the piece i by the piece j. The neighborhood of the current solution π_c, that corresponds to all possible transformations is $N(\pi_c)$. for each taboo iteration, the current solution π_c must be updated by changing it by the best solution of it's neighborhood that minimizes the Fitness-Function f. This piece of the best solution was added to the taboo List.

4.5 The Tournament Selection

The principle of selection by tournament increases the chances for poor individuals (arrangements) to participate in the improvement of the population. The principle is very quick to implement. A tournament consists of a meeting between two arrangements (parents) at random in the population. The tournament winner is the arrangement better P_1 (parent P_1). We repeat this method to select the second best arrangement P_2 (parent P_2). The principe of the arrangement (parent) P selection is the following:
1. (P_1,P_2) = random(|n|).
2. P = random(P_1, P_2).

5 EXPERIMENTATION AND ANALYSIS OF RESULTS

The algorithm was tested against the Hopper results and pure genetic algorithm (classic) GA. The results indicate that performance of the combined (hybrid) algorithm is better than that of either Hopper results or pure (classic) GA. We now present experimental results demonstrating the effectiveness of our method for finding perfect packing set. We use the benchmarks developed by Hopper. Those benchmarks contain collections with size ranging from 17 to 199 rectangles (items), all the instances are tested into a single rectangle of width (W = 300) and minimum height (H = 250). We evaluate our hybrid algorithm on the guillotinable instances from this set by an interesting comparison between the running results which are summarized in Table 1. This table reports, for each collection of instances (files) the number of items or placed rectangles (n), and the average of occupied space of both the three methods (pure genetic algorithms (Classic (%)), Hopper results (Hopper (%)) and our hybrid genetic approach (Hybrid (%)).

Algorithm		**Classic (%)**	**Hybrid (%)**			**Hopper (%)**
Data[16]	N	✓	Low	Average	Better	✓
t1c.xls	**17**	88.22	88.60	91.62	92.67	83.47
t2a.xls	**25**	91.98	93.07	93.73	93.95	95.73
t2b.xls	**25**	91.78	92.16	92.61	93.10	88.16
t2c.xls	**25**	88.33	92.17	92.17	92.17	83.68
t3a.xls	**29**	88.11	91.32	93.46	93.90	90.91
t3b.xls	**29**	86.58	91.74	92.94	93.46	89.68
t3c.xls	**29**	90.50	93.02	93.15	93.46	88.46
t4a.xls	**49**	93.46	93.89	94.38	95.24	89.68
t4b.xls	**49**	91.74	94.78	95.10	95.24	89.68
t4c.xls	**49**	91.11	96.15	96.15	96.15	88.49
t5a.xls	**73**	93.46	95.78	95.14	95.24	93.89
t5b.xls	**73**	93.88	95.69	95.92	96.15	89.28
t5c.xls	**73**	92.17	94.33	95.14	95.24	88.10
t6a.xls	**97**	94.48	96.75	96.75	96.75	94.48
t6b.xls	**97**	94.79	95.23	95.64	96.15	93.89
t6c.xls	**97**	94.34	95.24	95.92	96.15	95.69
t7a.xls	**199**	95.56	97.09	97.09	97.09	95.23
t7b.xls	**199**	95.69	95.69	96.36	96.93	96.15
t7c.xls	**199**	95.71	96.16	96.69	97.14	93.90

Table 1: Comparison results applied to the Hopper Benchmarks

By Examining Table 1 and considering only those instances, we proove the performance of our hybrid algorithm and we observe that:
1. The Hybrid Genetic Algorithm (Hybrid (%)) performs better than pure genetic algorithms (Classic (%));
2. The Hybrid Genetic Algorithm (Hybrid (%)) performs better than Hopper results (Hopper (%)), and offer more gains in terms of occupied space, and minimum waste of sheet.

6 CONCLUSION AND FUTURE TRENDS

In this paper we have proposed an hybrid approach for solving the constrained two-dimensional cutting (2DC) problem. Starting from a packing with height H, this approach tries to solve the (2DC) problem with decreasing values of H while avoiding solutions with

tall and thin wasted spaces. A specific fitness function f is designed to guide the search. Computational results show that our approach is very promising. The approach was tested and the results indicate that performance of the combined algorithm is better than others. Since we were unable to find a counterexample for which the approach fails, we conjecture that it always finds an optimal constrained guillotine cutting. Numerical examples also showed the superiority of the proposed approach compared with the classical methods in the literature.
There are some practical issues that need to be addressed. However, this hybrid approach optimises the solution to minimize the chutes and the waste. This should be extended to include other parameters such as considering three-dimensional cutting (3DC) problem. Further research in this area will include studying the impact of both rectangular and irregular forms of pieces on guillotine cutting problems.

REFERENCES

[1] J. E. Beasley: Algorithms for unconstrained two-dimensional guillotine cutting. Journal of the Operational Research Society, 1985, 36, 297-306

[2] Glover, F., 1989, "Tabu Search-Part I", ORSA Journal on Computing, 1, pp. 190-206.

[3] Glover, F., 1990, "Tabu Search-Part II", ORSA Journal on Computing, 2, pp. 4-32.

[4] Widmer, M., and A. Hertz, 1989, "A new heuristic method for the ow shop sequencing problem", European Journal of Operational Research, 41, pp. 186-193.

[5] Taillard, E., 1990, "Some efficient heuristic methods for the ow shop sequencing problem", European Journal of Operational Research, 47, pp. 65-74.

[6] Bland, J.A., G.P. Dawson, 1991, "Tabu search and design optimization", Computer-aided Design, 23, 3, pp. 195-201.

[7] Hertz, A., 1991, "Tabu search for large scale time tabling problems", European Journal of Operational Research, 51, pp. 39-47.

[8] Hifi- Mhand. The DH/KD algorithm: a hybrid approach for unconstrained two-dimensional cutting problems. European Journal of Operational Research (1997) 97, 41-52.

[9] Hifi- Mhand, Zissimopoulos V. A recursive exact algorithm for weighted two-dimensional cutting. European Journal of Operational Research (1996) 91, 55-64.

[10] Morabito R, Arenales M. An and/or graph approach to the container loading problem. International Transactions in Operational Research (1994) 58, 63-71.

[11] Gilmore PC, Gomory RE. The theory and computation of knapsack functions. Operations Research (1966), 1045-1074.

[12] S. Jakobs: On genetic algorithms for packing polygons, Euro. J. of Op. Res., 88 (1996) 165-181.

[13] H.F. Teng, D. Liu, An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles, European Journal of Operational Research 112 (1999) 413-420.

[14] H.F. Teng, Z.C. Yu, X. Gao, Layout optimization for the dishes installed on a rotating table, Science in China (series A) (1994) 1272-1280.

[15] K.A. Dowsland, W.B. Dowsland, Packing problems, European Journal of Operational Research 56 (1992) 2-14.

[16] Hopper E & Turton BCH, (2002), Problem Generators for Rectangular Packing Problems, Studia Informatica Universalis, 2(1), 123-136.

Optimal calibration of Parameter of a Conceptual Rainfall-Runoff Model Using Genetic Algorithm

A. Luis A. Alfaro Casas[1], **B. José Herrera Quispe**[1], **C. Juan Carlos Gutiérrez**[1], **D. Jorge L. Suaña Ch.**[1] **and E. Henry Gallegos Velgara**[1]
[1]National University of San Agustín – Catedra Concytec, Arequipa, Perú

Abstract - *The development an efficient and robust method to the parameter estimation of hydrological models is a critical issue to basin. The genetic algorithms have been demonstrated to be highly effective optimization methods. Accordingly, in the present study a Genetic Algorithm (GA) has been used to optimize the parameter values where calibration the rainfall-runoff model GR4J (Ge´nie Rural a` 4 parame`tres Journalier) on two subbasin of Chili River Basin. The data input of daily rainfall and potential evapotranspiration are used, and the result will calibrated with the observed data of daily discharge. The fitness function used, is to minimize the error deviation between the results generated and observed data. There are two stages in this modeling. Firstly is model calibration and secondly is model validation. Model calibration uses the first of five years of data input and model validation uses the following one year. The Nash-Sucliffe Coefficient (NS) and The Relative Volume Error (RVE) methods are used to obtain the error deviation. This study is intended to optimize four free parameters of the model, there is the maximum capacity of production store (X1), the groundwater exchange coefficient (X2), the maximum capacity of routing store (X3), and the time base of unit hydrograph (X4). The constraints considered for this optimization are the bounds for decision variables. The result of this study shows that the first five years data has parameters quite similar with the second one year data. Hence based on the present case study it can be concluded that GA model has the capability to perform efficiently, if applied in calibration the rainfall-runoff model.*

Keywords: Rainfall-runoff, GR4J method, flood study, unit hydrograph, genetic algorithm, optimization

1 Introduction

The rainfall-runoff modeling, usually involves a statistical analysis, to extend and predict future data. There are problems to build and calibrate a rainfall-runoff model, because it depends on the characteristics of the river basin. One of daily rainfall-runoff model (used for this study) is GR4J (Genie Rural a 4 parametres Journalier) which was developed by Perrin [9] and was proven having strong basic and efficient in a modeling. GR4J is developed from earlier model that is GR3J which is originally proposed by Edijatno and Mitchel [3] and then successfully improved by Nascimento [8] and Edijatno [4].

The need for an efficient optimization method to find the best values of the parameters of the model is very necessary. Many traditional methods such as Linear Programming (LP) and dynamic programming (DP) have been used to solve problems.

Gas, are based on Darwinian natural selection, which combines the concept of survival of the strongest natural genetic operators [5]. The way they work and their implementation are well documented by Goldberg [6] and Michaelwiez [7]. One of the advantages of GAs is identifying alternative near-optimal solution. In the field of hydrology, Gas, are optimization methods shown as powerful. Franchini and Galeati [2], make a comparative study of various schemes of genetic algorithms for the calibration of conceptual rainfall-runoff models, and concluded that GAs are a very robust method, proving to be a valid instrument for optimization of complex functions objective. In a study Esat and Hall (1994), apply a GA for a problem with four reservoirs. They concluded that Gas, have potential in optimization of water resources and significant savings could be achieved in memory space and execution time. Sharif and Wardlaw [8] uses GA in the development of water resources and compared with dynamic programming, which concluded that both results were comparable.

GAs can be configured in many ways, but so far there is little literature on the type most suitable for the calibration of a rainfall-runoff model. This paper intends to address this gap by applying GA to calibrate the rainfall-runoff model GR4J. The aim has been to present that GAs can easily cope with these problems and present as robust optimization tools. Several experiments were designed to choose the right components for the GA, including selection by roulette, tournament, Stochastic and Scattered, Single point, Two point, Intermediate. The evolution cycle is repeated until a termination criterion. This approach may be a number of evolution cycles (computational runs), the variation of individuals between different generations or a predefined value of fitness.

2 Study area

Chili River Basin is located in Southern Peru, mainly located in the department of Arequipa, with small areas of the departments of Cusco, Puno and Moquegua, as shown in Figure 1. [11].

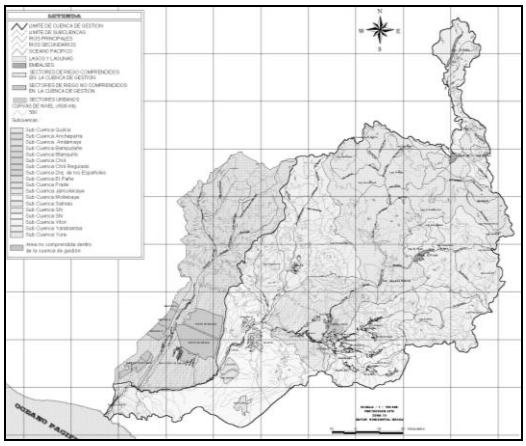


Figure. 1. Scope of the Chili River Basin [11]

The following describes the sub-basins chosen for the study:

2.1 Pañe Subbasin

Located at the north end of the study area is about 4 585 m characterized as having a humid (tropical). It has an area of 198 km^2, average daily rainfall of 2.21 mm/d, average evapotranspiration is 4 mm / d and the average daily flow is 2.66 m^3/s [11]. Weather Station has a staff gauge and a station, both called The Pane (located in the same reservoir).

2.2 Fraile Subbasin

Fraile Subbasin comprises the birth of the rivers Yamanayo, Collpamayo, Paltimayo, Cancusane, Pasto Grande (among other minor rivers) to the Rio Blanco (born of the confluence of the rivers already mentioned) by submitting a drainage area 1041 Km^2 and ends at El Fraile reservoir located on the White River at an average altitude of 4000 meters, regulating water resources. Weather Station has a staff gauge and a station, both called The Frayle (located in the same reservoir). Taking an average annual rainfall of 386 mm, a mean annual flow of 3.32 m^3/s.

3 Description of model

GR4J model is rainfall runoff modeling which was based on four free parameters from daily rainfall data. The GR4J model is the last modified version of the GR3J model originally proposed by Edijatno and Michel [3] and then successively improved by Nascimento [8] and Edijatno [4].

GR4J optimize four free parameters, they are:

- X_1: Maximum capacity of production store (mm).
- X_2: Groundwater exchange coefficient (mm).
- X_3: Maximum capacity of routing store (mm).
- X_4: Time peak ordinate of hydrograph unit UH1 (day).

From earlier study which developed by Perrin et al [9], GR4J give better result than other rainfall runoff modeling. From that study, Perrin et al [9] used 429 river basins in which have different climates.

The description of physical GR4J modeling from rainfall process to runoff at river is gives as following below. Production Store (X1) is storage in the surface of soil which can store rainfall. There are evapotranspiration and percolation in this storage. The capacity of this storage depends on the types of soil in that river basin. Few porosity of soil can make production store bigger. Groundwater exchange coefficient (X2) is a function of groundwater exchange which influence routing store. When it has a negative value, then water enter to depth aquifer, when it has a positive value, then water exit from aquifer to storage (routing storage). Routing storage (X3) is amount of water which that can be storaged in soil porous. The value of this routing store depends to the type and the humidity of soil. Time Peak (X4) is the time when the ordinate peak of flood hydrograph is created on GR4J modeling. The ordinate of this hydrograph is created from runoff, where 90 % of flow is slow flow that infiltrates into the ground and 10 % of flow is fast flow that flows on the soil surface (View Figure 2).

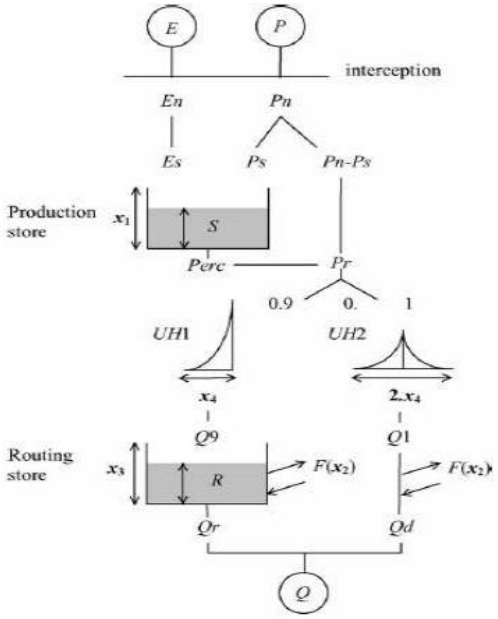


Figure 2. Model Diagram of Rainfall-Runoff GR4J [9]

4 Model development

4.1 Input data

To calibrate the model, we used historical daily bellonging: precipitation, evaporation and flow observed in 5 years. In Figure 3 shows the rainfall used in the study.

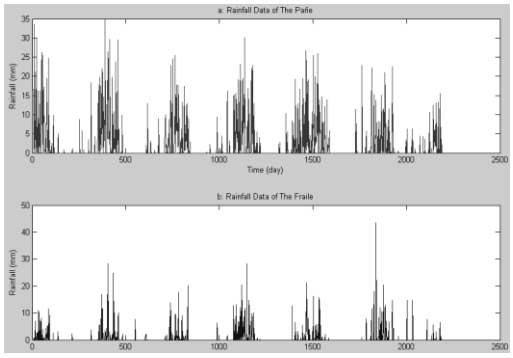


Figure 3. Rainfall Data. a) Rainfall Data, Pañe subbasin b) Rainfall Data, Fraile subbasin.

4.2 Objective Function

The main consideration of the objective function optimization model is to maximize the method of Nash-Sutcliffe Coefficient (NS), which is used for computing the change between square summation of observation data to modeling result data. General Equation of Nash-Sutcliffe Coefficient is given as:

$$E(Q) = 100 * \left(1 - \frac{\sum \left(Q_i - \hat{Q}_i \right)^2}{\sum \left(\overline{Q_i} - Q_i \right)^2} \right) \qquad (1)$$

Where:

Q_i : is the flow observed on day *i*.

$\hat{Q}_i$: is the simulated flow for the day *i*.

$\overline{Q_i}$: is the average flow of the observed.

The above fitness function of GA model is subjected to the following constraints and bounds:

$$1 < x_1 < 2000,\ x_1 \in R$$

$$-50 < x_2 < 50,\ x_2 \in R$$

$$1 < x_3 < 400,\ x_3 \in R$$

$$0 < x_4 < 99,\ x_4 \in R$$

Additionally, the method Relative Volume Error (RVE) is used for computing the volume of observation data with modeling result data [12]. The general equation of Relative Volume Error as:

$$RVE = \frac{\sum \left(Q_i - \hat{Q}_i \right)}{\sum \left(Q_i \right)} \qquad (2)$$

5 Model Application and discussion

To apply the model developed GA has been used daily bellonging: precipitation, evaporation and observed flow rates of 5 years, from 2003 to 2007. Then, after obtaining the values of the parameters, the model is validated with 2008 data. In this model, calibration value from Nash-Sutcliffe coefficient (NS) should be one hundred and calibration value from Relative Volume Error (RVE) should be zero.

In a GA, one of the important parameters is the size of the population; to obtain an optimum population is very important. In applications in water resources, this value is between 64 and 300, and even 1000 [13]. A large population helps to maintain greater diversity, but involves a considerable computational cost. To find the optimal size of the population in this study were considered different population sizes.

The initial search was conducted with a crossover probability of 0.80 and a population size of 50 individuals, bringing the population to 15 in each event, reaching a population of 305. The result of the system initially was 79,269 %, being increased to 83,436 % for the Pañe subbasin and 42.436 % to 55.9 % for the Fraile subbasin. Figure 4 shows how the fitness function is reduced, resulting in improved system performance. System performance improves significantly when the size of the population increases to a certain number. With increasing population, the system produces better results but there is no significant improvement. The significant point this work is produced with 200 individuals, then the performance does not improve significantly.

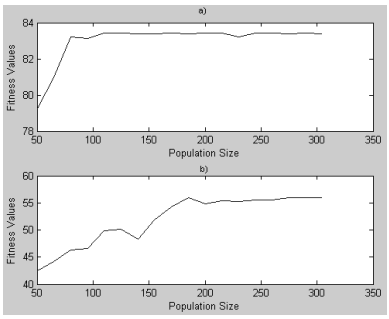


Figure 4. Fitness value for different Population size. a) Pañe subbasin b) Fraile subbasin.

The second important parameter influencing the performance of GA is the crossover probability, the method of crossing and crossing selection form. Its effect on system performance is studied by varying the crossover probability of 0.6 to 0.9, with an increase of 0.02, crossing a selection as roulette, tournament, stochastic uniform and crossover methods as: Scattered, Single point, Two point, having an optimum population of 200 individuals. The results are shown in Figure 5, comparing crossover probability vs system performance, you can see that system performance improves with increased probability of crossing up to 0.75. After this value, the system performance decreases with increasing the probability of crossing.

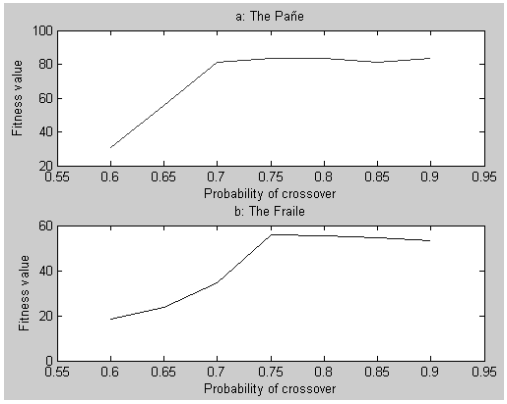


Figure 5. Fitness value for different crossover. a) Pañe subbasin b) Fraile subbasin.

In the Figure 6 shows the results obtained by different methods of crossing.

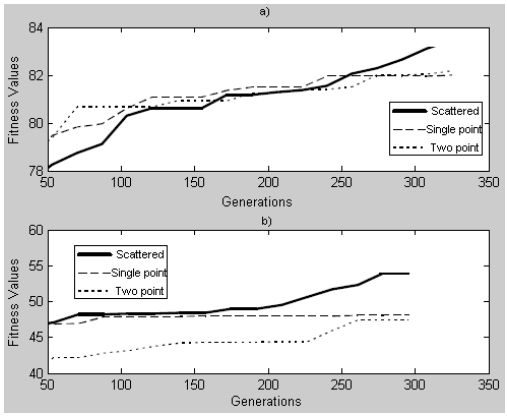


Figure 6. Fitness value for different crossover methods. a) Pañe subbasin b) Fraile subbasin.

In Figure 7 and Figure 8 show the comparation of observed discharge data to discharge of modeling result computed using parameters values that give the smallest deviation.

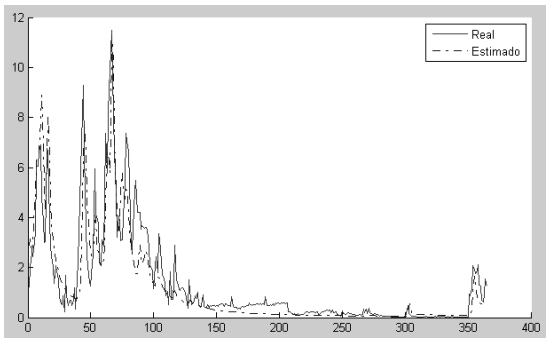


Fig. 7. Series of simulated and observed flows for Pañe subbasin.

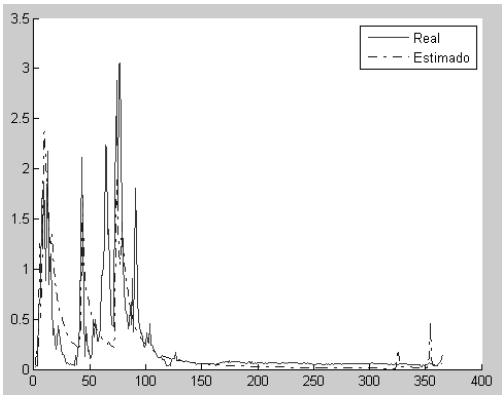


Fig. 8. Series of simulated and observed flows for Fraile subbasin.

The Table 1 shows the deviation using the Nash-Sutcliffe coefficient for Pañe subbasin, gives a value of 83.43%. This shows that the calibration and validation data have the same patterns. Nash-Sutcliffe described the similar value of modeling results compared with observed discharge. If the value is close to one, then download the results of modeling has a similar pattern with the performance observed.

Parameters	Pañe Subbasin	Fraile Subbasin
Nash(Q)	83,436	55.900
RVE(Q)	0.6760	10.20

Table 1. The fitness function through GA

The Table 2 shows the deviation using the Nash-Sutcliffe coefficient for the Frayle subbasin, gives a value of 55.90%. Nash-Sutcliffe does not describe the similar value of modeling results compared with observed discharge. If the value exceeds 50, then download the results of modeling has a higher than average pattern of flow observed.

Parameters	Pañe Subbasin	Fraile Subbasin
X_1 (mm)	2.77	40.27
X_2 (mm)	0.00	0.00
X_3 (mm)	4.67	5.15
X_4 (día)	0.00	0.00

Table 2. The optimal parameters through GA

6 Conclusions

The application of genetic algorithms is satisfactory for the adjustment of parameters in the model GR4J overall, a significant representation of the series of daily flows, especially the tendency of these and minimum flows. The criteria of efficiency are between 50 and 100%, this shows the potential of the model for use in the generation of operating rules.

In this study, very close to the optimal solution is achieved within 100 generations, with a population of 200 individuals. A string representation of real-value, the incorporation of Stochastic uniform selection method and the crossing Scattered along with a crossover probability of 0.75, the best results.

GR4J model is easy to implement, in comparison with robust models because it is difficult to have total control over all variables and parameters. The use of 4 parameters facilitates optimization of parameters using genetic algorithms. It is expected that a hybrid combination of different methods of calibration and optimization based on metaheuristics and numerical methods can give better results.

7 References

[1] Goldberg David E., Genetic Algorithms in Search, Optimization, and Machine Learning, Massachusetts, 1989.

[2] Franchini, Marco and Galeati, Giorgio, Comparing several genetic algorithm schemes for the calibration of conceptual rainfall-runoff models, Hydrological Sciences Journal, 42: 3, 357 — 379, 1997

[3] Edijatno, C. Michel, "Un Modele Pluie-Debit Journalier a Trois Parameters", La Houille Blanche (2), 1989, 113 – 121.

[4] Edijatno," GR3J: a daily watershed model with three free parameters", Journal of Hydrology, 1999.

[5] Holland, J. H., "Adaptation in natural and artificial systems", University of Michiyan Press annarbov, Cambridge Mass, 1975.

[6] Michalewicz, Z), "Genetic algorithm+data structures = evolution programmes", Springer, Berlin, 1996.

[7] Sharif and Wardlaw, "Multireservoir system optimization using genetic algorithms-Case study", J. computation in civil engineering 14(4).pp-255-263, 2000.

[8] N. O. Nascimento," Appreciation a L'aide D'un Modele Emirique Des Effets D'action Anthropiques Sur La Relation Pluie-Debit a L'echelle Du Bassin Versant", PhD Thesis, CERGRENE/ENPC, Paris, France, 1995, 550 pp.

[9] M. Perrin, Andre´assian," Improvement of a parsimonious model for streamflow simulation", Journal of Hydrology, 2003.

[10] Esat, V., and Hall, M. J., Water resources system optimization using genetic algorithms, in Proc. of the 1st Int. Conf., On Hydroinformatics, Balkema, Rotterdam, The Netherlands, 225-231, 1994.

[11] AUTODEMA, Diagnostico de Gestión de la Oferta de Agua de la Cuenca Quilca-Chili. Arequipa – Perú. , 2002.

[12] M. J. Booij," Impact of climate change on river flooding assessed with different spatial model resolutions", Journal of Hydrology, 2005.

[13] Rao S. Govindraju and Ramchandrarao A., "Water science and technology library –Artificial neural network in hydrology" www.books.google.com/books pp142, 2000.

Hybrid GEMs for Multi-Biometric Recognition via X-TOOLSS

Aniesha Alford[1], Khary Popplewell[2], Gerry Dozier[2], Kelvin Bryant[2], John Kelly[1], Josh Adams[2], Tamirat Abegaz[3], Joseph Shelton[2], Damon L. Woodard[4], and Karl Ricanek[5]

Center for Advanced Studies in Identity Sciences (CASIS)
[1]Electrical and Computer Engineering, NC A&T State University
1601 East Market St. Greensboro, NC, 27411, USA
[2] Computer Science, NC A&T State University
1601 East Market St. Greensboro, NC, 27411, USA
[3] Computational Science and Engineering, NC A&T State University
1601 East Market St. Greensboro, NC, 27411, USA
[4] School of Computing, Clemson University
314 McAdams Hall, Clemson, S.C. 29634-0974, USA
[5] Computer Science, University of North Carolina at Wilmington
CIS Building, Room 2010, 601 South College Road, Wilmington NC, 28403, USA

Abstract - *In [1], Alford et al. compared the performances of two Genetic and Evolutionary Methods (GEMs) for multi-biometric feature selection and weighting. In this paper, we present two hybrid feature weighting/selection GEMs. Our results show that the hybrid GEMs outperform the GEMs presented in [1], using significantly fewer features while achieving practically the same recognition accuracy.*

Keywords: Eigenface, Feature Selection, Feature Weighting, Genetic and Evolutionary Computation, Local Binary Pattern, Multi-biometrics

1 Introduction

Genetic and Evolutionary Methods (GEMs) are population-based problem solvers based on simulated evolution [22]. GEMs seek optimal or near optimal solutions to problems, and have been successfully applied to solve problems in a variety of areas including robotics [25], machine learning [23] and biometrics [1, 6, 7, 19, 20]. GEMs work as follows. Typically, an initial population of candidate solutions is randomly generated. Each candidate solution of the initial population is assigned a fitness based on a user-defined evaluation function. Parents are then selected from the population and reproduce creating offspring. The offspring are assigned a fitness and usually replace the members of the population with the worst fitness. This evolutionary process continues until the population converges, a user-specified threshold is reached, or after a user-specified number of function evaluations.

The two types of GEMs used in this research are a Steady-State Genetic Algorithm (SSGA) [5, 22] and an Estimation of Distribution Algorithm (EDA) [4, 16]. SSGAs apply crossover and mutation to parents chosen from the population in an effort to create offspring. EDAs do not use crossover or mutation operators to create offspring; instead, after creating and evaluating the population, a percentage of the top performing candidate solutions are used to form a probability density/distribution function (PDF). Offspring are created by sampling the PDF. A new population is then created using a percentage of the best performing members of the population, known as the elites, and a percentage of the offspring.

A number of studies have been published comparing the performances of these two GEMs on different problems [4, 10, 16]. The results of these studies have shown that neither GEM outperforms the other for every problem tested. Alford et al. [1] compared the performances of SSGA and EDA based feature selection (GEFeS) and weighting (GEFeW) for multi-biometric recognition. The modalities tested were face and periocular biometrics. Their results showed that SSGA-based feature weighting ($GEFeW_{SSGA}$) produced the highest average recognition accuracy for the multi-biometric experiment, while EDA-based feature selection ($GEFeS_{EDA}$) used the fewest average percentage of features. In this paper, we extend the research of [1], comparing the performances of two additional GEMs that are hybrid feature weighting/selection methods. These new methods are instances of what we will refer to as Genetic and Evolutionary Feature Weighting and Selection (GEFeWS) [19].

The goal of our research is to develop short length biometric templates that can be used in a two-stage hierarchical system as proposed by Gentile et al. [9] for iris recognition. Conventional biometric identification systems compare a probe to every individual within the biometric database. As described by Alford et al. [19], the number of feature checks performed by a conventional biometric system, f_c, is equivalent to the number of individuals in the database, n, multiplied by the number of features extracted to represent the individuals, m, as shown in Equation 1:

$$f_c = nm \quad (1)$$

A two-stage hierarchical system, as proposed by Gentile et al. [9], reduces the number of feature checks performed. In the first stage, the short length biometric templates consisting of k features are used to select a subset of the r closest matches to a probe, p. In the second stage, the subset is compared to p using the full length biometric templates which consist of m features. The number of feature checks performed by a hierarchical system, f_h, is [19]:

$$f_h = nk + rm \quad (2)$$

which is the sum of the feature checks performed in the two stages. By reducing the number of features compared in the first stage and the number of individuals compared in the second stage, the hierarchical system results in a savings of [19]:

$$f_s = \frac{f_h}{f_c} = \frac{nk + rm}{nm} = \frac{k}{m} + \frac{r}{n} \quad (3)$$

The remainder of this paper is as follows. In Section 2, a brief overview of the feature extraction techniques used in the research presented in this paper, Eigenface and the Local Binary Patterns (LBP) method, are discussed. In Section 3, GEFeS, GEFeW, and GEFeWS are presented. Section 4 presents our experiments, and Section 5 presents our results. Finally, in Section 6, the conclusion and future work are presented.

2 Feature Extraction Overview

Eigenface is a feature extraction method that uses Principal Component Analysis (PCA) [21] to reduce the dimensionality of an image space into a smaller dimensionality face space. Only the most discriminating eigenvectors (i.e. those with the largest eigenvalues) produced by PCA are used to form the face space. The feature templates are then created by projecting the face images onto the face space. For a more detailed description see [13, 18, 20].

The Local Binary Patterns (LBP) method extracts features from an image by first segmenting the image into a grid of evenly sized patches and then analyzing the pixel-to-pixel intensity changes within each patch. The intensity change pattern of each pixel (when compared to its neighbors) is encoded into a numerical value that is used to build a histogram for that patch based on the frequency of each numerical value (intensity pattern). The histograms from each patch are concatenated together to form a single feature template. For a more detailed description see [1, 7, 14, 19, 24].

3 GEFeS, GEFeW, and GEFeWS

The GEMs used in this paper are instances of the eXploratory Toolset for the Optimization of Launch and Space Systems (X-TOOLSS) SSGA and EDA [12]. As described in [1], GEFeS and GEFeW evolve a population of real-coded feature masks composed of values within the interval [0..1]. For GEFeS, a masking threshold of 0.5 is used to determine if a feature should be used. If the feature mask value is less than the masking threshold, the feature is not used during matching. Otherwise, the feature will be used during matching. For GEFeW, the real-coded feature mask is multiplied by the feature template, resulting in a weighted feature template.

GEFeWS [19] is a hybrid of GEFeW and GEFeS. Like GEFeS and GEFeW, GEFeWS evolves a population of real-coded feature masks composed of values within the interval [0..1]. However, if the value is less than the masking threshold, 0.5, then the feature is not used during matching as in GEFeS. Otherwise, the feature is weighted as done by GEFeW.

The evaluation function used to assign a fitness to each candidate solution is:

$$fit = 10\varepsilon + \frac{k}{m} \quad (4)$$

where ε is the number of recognition errors associated with the evolved feature mask, k is the number of features within the reduced length feature template, and m is the original number of features.

4 Experiment

As in [1], our experiments were performed on a subset of the first 105 subjects taken from the Face Recognition Grand Challenge (FRGC) dataset [15]. The probe set consisted of 105 images, one image per subject. The gallery set consisted of 210 images, two additional images per subject. The images used were frontal views of the subjects with neutral facial expressions.

The tested biometric modalities for the experiments were face, periocular, and face plus periocular. The Eigenface method was used to extract 210 facial features from each image. The LBP method was used to extract 2832 periocular features (1416 features from each of the left and right periocular regions). GEFeS, GEFeW, and GEFeWS were performed on each of the biometric modalities. In addition, the modalities were tested using all of the originally extracted features, which served as the baseline for our experiments. For the multi-biometric experiment, the face and periocular features were evenly fused.

Recognition was performed by computing the Manhattan distance between the probe template and the templates in the gallery. The subject of the template within the gallery with the smallest Manhattan distance was considered the match to the probe.

5 Results

For our experiments, the SSGA instances of GEFeS, GEFeW, and GEFeWS, had a population size of 20, a crossover and mutation rate of 1.0, and a mutation range of 0.2. The EDA instances had a population size of 20 and always retained the 5 best individuals, known as the elites. Each of the SSGA and EDA instances were run 30 times, with a maximum of 1000 function evaluations per run.

In Table I, the average performance of the methods for each of the tested modalities is shown. The first column represents the tested biometric modalities, the second column represents the methods used, the third column represents the average percentage of features used, and the last column represents the average accuracy of the 30 runs. The best results are highlighted in bold. The ANOVA and t-test were used to separate the methods into equivalence classes.

5.1 Face-Only

For the Face-Only experiment, all of the GEMs had significantly better recognition accuracies when compared to the baseline Eigenface method. When comparing the GEMs, in terms of accuracy, $GEFeW_{EDA}$ performed the best, having an average accuracy of 87.81%. $GEFeW_{SSGA}$, $GEFeW_{EDA}$, and $GEFeWS_{EDA}$ were all in the first equivalence class. The remaining GEMs were in the second equivalence class.

In terms of the percentage of features used, $GEFeS_{EDA}$ and $GEFeWS_{EDA}$ were in the first equivalence class. $GEFeS_{SSGA}$ was in the second equivalence class and $GEFeWS_{SSGA}$ was in the third equivalence class. $GEFeW_{SSGA}$ was in the fourth equivalence class, and $GEFeW_{EDA}$ was in the fifth equivalence class. The GEFeW instances used the highest percentage of features.

5.2 Periocular-Only

For the Periocular-Only experiment, in terms of average recognition accuracy, $GEFeWS_{SSGA}$ performed the best. $GEFeS_{EDA}$ and $GEFeWS_{EDA}$ were in the second equivalence class, $GEFeS_{SSGA}$ and $GEFeW_{SSGA}$ were in the third equivalence class, and $GEFeW_{EDA}$ was in the fourth equivalence class.

In terms of the percentage of features used, $GEFeS_{EDA}$ and $GEFeWS_{EDA}$ were in the first equivalence class, using only 41% of the features. $GEFeWS_{SSGA}$ was in the second equivalence class and $GEFeS_{SSGA}$ was in the third equivalence class. $GEFeW_{SSGA}$ and $GEFeW_{EDA}$ were in the fourth and fifth equivalence classes respectively. As with the Face-Only experiment, the GEFeW instances used the highest percentage of features.

TABLE I
COMPARISON BETWEEN SSGA AND EDA BASED GEFES, GEFEW, AND GEFEWS

Modalities Tested	Methods	Average Percentage of Features Used	Average Recognition Accuracy
Face Only	Eigenface	100.00%	64.76%
	Eigenface + $GEFeS_{SSGA}$	50.30%	86.13%
	Eigenface + $GEFeS_{EDA}$	**42.86%**	85.59%
	Eigenface + $GEFeW_{SSGA}$	87.16%	87.59%
	Eigenface + $GEFeW_{EDA}$	96.53%	**87.81%**
	Eigenface + $GEFeWS_{SSGA}$	51.71%	86.38%
	Eigenface + $GEFeWS_{EDA}$	43.35%	87.02%
Periocular Only	LBP	100.00%	94.29%
	LBP + $GEFeS_{SSGA}$	48.03%	95.14%
	LBP + $GEFeS_{EDA}$	41.03%	95.87%
	LBP + $GEFeW_{SSGA}$	86.22%	95.46%
	LBP + $GEFeW_{EDA}$	95.78%	94.67%
	LBP + $GEFeWS_{SSGA}$	45.39%	**96.16%**
	LBP + $GEFeWS_{EDA}$	**41.01%**	95.75%
Face + Periocular	Eigenface + LBP [evenly fused]	100.00%	90.77%
	Eigenface + LBP + $GEFeS_{SSGA}$	48.18%	97.40%
	Eigenface + LBP + $GEFeS_{EDA}$	45.24%	96.70%
	Eigenface + LBP + $GEFeW_{SSGA}$	87.59%	**98.98%**
	Eigenface + LBP + $GEFeW_{EDA}$	97.40%	96.64%
	Eigenface + LBP + $GEFeWS_{SSGA}$	46.24%	98.48%
	Eigenface + LBP + $GEFeWS_{EDA}$	**41.72%**	98.10%

5.3 Face + Periocular

For the Face + Periocular experiment, $GEFeW_{SSGA}$ performed the best in terms of accuracy, while $GEFeWS_{SSGA}$ and $GEFeWS_{EDA}$ were in the second and third equivalence classes respectively. Although the GEFeWS instances were statistically different from $GEFeW_{SSGA}$, the average recognition accuracy achieved by these methods are of no practical significance.

In terms of the percentage of features used, $GEFeWS_{EDA}$ was in the first equivalence class, using only 41.72% of the features to achieve an average accuracy of 98.10%. $GEFeS_{EDA}$ and $GEFeWS_{SSGA}$ were in the second equivalence class. $GEFeS_{SSGA}$ was in the third equivalence class. As before, $GEFeW_{SSGA}$ and $GEFeW_{EDA}$ were in the fourth and fifth equivalence classes respectively, using the highest percentage of features.

6 Conclusions

Our results show that $GEFeW_{SSGA}$ produces the best recognition accuracy for multi-biometric recognition. Conversely, $GEFeWS_{EDA}$ performed the best at reducing the number of necessary features. Although there was a statistical significant difference in the average accuracy achieved by $GEFeWS_{EDA}$ when compared to $GEFeW_{SSGA}$, there is no practical significance. $GEFeWS_{EDA}$ was able to achieve an average recognition accuracy only 0.88% lower than GEFeW while using almost 50% fewer features. Therefore, we consider $GEFeWS_{EDA}$ to be the best method.

Our future work will focus on the development of more efficient methods for solving the multi-biometric recognition problem. In addition, we plan to investigate how well the evolved feature masks generalize on a larger dataset of unseen subjects.

7 Acknowledgment

This research was funded by the Office of the Director of National Intelligence (ODNI), Center for Academic Excellence (CAE) for the multi-university Center for Advanced Studies in Identity Sciences (CASIS) and by the National Science Foundation (NSF) Science & Technology Center: Bio/computational Evolution in Action CONsortium (BEACON). The authors would like to thank the ODNI and the NSF for their support of this research.

8 References

[1] A. Alford, C. Hansen, G. Dozier, K. Bryant, J. Kelly, J. Adams, T. Abegaz, K. Ricanek, and D.L. Woodard, "A Comparison of GEC-Based Feature Selection and Weighting for Multimodal Biometric Recognition", *accepted to IEEE Congress on Evolutionary Computation* (CEC), 2011.

[2] A. Jain, K. Nandakumar, and A. Ross, "Score Normalization in Multimodal Biometric Systems", *Pattern Recognition*, Vol. 38, No. 12, pp. 2270–2285. December 2005.

[3] A. Ross, "An Introduction to Multibiometrics", In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*, Poznan, Poland, September 2007.

[4] E. Cantú-Paz, "Feature Subset Selection by Estimation of Distribution Algorithms", In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO*), 2002.

[5] F. Hussein, N. Kharma, and R. Ward, "Genetic Algorithms for Feature Selection and Weighting, a Review and Study", In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, Seattle, WA, 2001.

[6] G. Dozier, K. Frederiksen, R. Meeks, M. Savvides, K. Bryant, D. Hopes, and T. Munemoto, "Minimizing the Number of Bits Needed for Iris Recognition via Bit Inconsistency and GRIT," In *Proceedings of IEEE Workshop on Computational Intelligence in Biometrics: Theory, Algorithms, and Applications, 2009. CIB 2009*, pp.30-37, March 30 2009-April 2 2009.

[7] J. Adams, D.L. Woodard, G. Dozier, P. Miller, K. Bryant, and G. Glenn "Genetic-Based Type II Feature Extraction for Periocular Biometric Recognition: Less is More," In *Proceedings of the Int. Conf. on Pattern Recognition*, 2010.

[8] J.E Gentile, N. Ratha, and J. Connell, "SLIC: Short-Length Iris Codes," In *Proceedings of the 3rd IEEE International Conference on Biometrics: Theory, Applications, and Systems, 2009. BTAS '09*, pp.1-5, 28-30 Sept. 2009.

[9] J.E. Gentile, N. Ratha, and J. Connell, "An Efficient, Two-Stage Iris Recognition System", In *Proceedings of the 3rd IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS)*, 2009.

[10] J. M. Peña, V. Robles, P. Larrañaga, V. Herves, F. Rosales and M. S. Pérez. "GA-EDA: Hybrid Evolutionary Algorithm Using Genetic and Estimation of Distribution Algorithms", In *Proceedings of the 17th International Conference on Innovations in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence*, Vol. 3029, Springer, Berlin, pp. 361–371, 2004.

[11] M. L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain, "Dimensionality Reduction Using Genetic Algorithm", In *Proceedings of the IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 2, pp. 164-171, July 2000.

[12] M. L. Tinker, G. Dozier, and A. Garrett, "The Exploratory Toolset for the Optimization Of Launch and Space Systems (X-TOOLSS)," http://xtoolss.msfc.nasa.gov/, 2010.

[13] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3 no. 1, pp. 76-81, Winter 1991.

[14] T. Ojala, M. Pietikäinen, and T. Mäenpää. "Multiresolution Gray-scale and Rotation Invariant Texture Classification with Local Binary Patterns". *In Proceedings of IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(7): 971-987

[15] P. J. Phillips, P.J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of Face Recognition Grand Challenge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[16] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A new tool for evolutionary computation*. Springer, 2002.

[17] R. Giot, M. El-Abed, and C. Rosenberger, "Fast Learning for Multibiometrics Systems Using Genetic Algorithms", In *Proceedings of the IEEE International Conference on High Performance Computing and Simulation (HPCS)*, Caen, France, pp. 266 – 273, 2010.

[18] Y.V. Lata, C. K. B. Tungathurthi, H. R. M. Rao, A. Govardhan, and L. P. Reddy, "Facial Recognition using Eigenfaces by PCA", In *International Journal of Recent Trends in Engineering*, Vol. 1, No. 1, pp. 587-590, May 2009.

[19] A. Alford, K. Popplewell, G. Dozier, K. Bryant, J. Kelly, J. Adams, T. Abegaz, and J. Shelton, "GEFeWS: A Hybrid Genetic-Based Feature Weighting and Selection Algorithm for Multi-Biometric Recognition", in *Proceedings of The 22nd Midwest Artificial Intelligence and Cognitive Science Conference (MAICS)*, 2011.

[20] T. Abegaz, G. Dozier, K. Bryant, J. Adams, K. Popplewell, J. Shelton, K. Ricanek, and D. L. Woodard, "Hybrid GAs for Eigen-Based Facial Recognition", *IEEE Symposium Series in Computational Intelligence 2011* (SSCI 2011).

[21] I. Jolliffe, "Principal Component Analysis", *Encyclopedia of Statistics in Behavioral Science*, 2005.

[22] G. Dozier, A. Homaifar, E. Tunstel, and D. Battle. "An Introduction to Evolutionary Computation" (Chapter 17), *Intelligent Control Systems Using Soft Computing Methodologies*, A. Zilouchian & M. Jamshidi (Eds.), pp. 365-380, CRC press, 2001.

[23] K. DeJong, "Learning with genetic algorithms: an overview" *Machine Learning*, Vol. 3, No. 3, pp. 121–138, 1988.

[24] P.E. Miller, A.W. Rawls, S.J. Pundlik, and D.L. Woodard, "Personal Identification Using Periocular Skin Texture". In *Proc. 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2010.

[25] J.M. Ahuactzin, E. Talbi, P. Bessière and E. Mazer, "Using genetic algorithms for robot motion planning", *Lecture Notes in Computer Science*, Volume 708, pp. 84-93, 1993.

Genetic Algorithms for Group Decision Problems Using Ordinal Interval Numbers

Tatiana Tambouratzis

Abstract—**Group decision making is investigated using ordinal interval number ranking. The closed interval preference is given independently by each decision maker and for every course of action. Genetic algorithms are employed for establishing the mutually preferred course(s) of action. The combination of crossover and adjusted mutation is found efficient in terms of problem representation and genetic-algorithm operation/convergence to (one of) the preferred course(s) of action.**

I. Introduction

Group decision making is performed by an ensemble of decision makers who collectively decide upon the preferred course of action (solution) to a problem, as selected out of a number of alternative courses of action which are available at the time of decision making. Although more time consuming than individual decision making, a group decision allows the combination of the strengths and the expertise of each decision maker in the group in order to maximize the joint agreement upon the selected course of action.

A variety of group decision making methods exist [1-2] for reaching the final decision, with each method having advantages and disadvantages in terms of speed and decision maker satisfaction/agreement with the mutually preferred course of action. Group decision making methods include – among others - decision by authority, decision by majority, decision by negative minority, decision by unanimity, decision by consensus and decision via preference ranking.

In this piece of research, group decision making is investigated using ordinal interval number preference ranking. The closed intervals of preference are given independently by each decision maker, who provides an interval for every available course of action. Genetic algorithms [3] are employed for establishing the preferred course(s) of action. The combination of crossover and adjusted mutation is proved efficient in terms of problem representation and genetic-algorithm operation/convergence to a preferred course of action.

This paper is organized as follows: Section II introduces the group decision making and ranking methods while also detailing its operation under the ordinal interval number representation of the decision makers' preferences; Section III describes the inspiration, structure and mode of operation of genetic algorithms; Section IV demonstrates the application of genetic algorithms to ordinal interval number preference ranking and group decision making while also evaluating their performance on a number of problems from the existing literature; finally, Section V concludes the paper.

Manuscript received March 10, 2011.

T. Tambouratzis is with the Department of Industrial Management & Technology, University of Piraeus, 107 Deligiorgi St, Piraeus 185 34, Greece, and the Department of Nuclear Engineering, Chalmers University of Technology, SE-412 Göteborg, Sweden (phone: +30-210-4142423; fax: +30-210-4142392; e-mail: tatianatambouratzis@gmail.com).

II. Group Decision Making and Ranking

Ranking aggregates the preferred courses of action of the different decision makers in order to select the mutually preferred course of action. Ranking has a number of advantages: it is robust to the composition and size of the group of decision makers as well as to the familiarity of each decision maker with the problem at hand; different evaluation criteria that are related to each decision maker's background and outlook towards the problem can be simultaneously accommodated; finally, the assignment of relative importance values to the preferences of the various decision makers is allowed.

A number of formats are available for ranking. These formats include nominal [4], ordinal [5-18] and ordinal interval [19-20] numbers; the choice of format depends (a) on the way in which the constraints between alternative courses of action are expressed and (b) on the means of aggregating and, subsequently, evaluating the alternative courses of action.

In the following it is assumed that n possible courses of action exist and that m decision makers are involved in the group decision making process.

When performing ranking using ordinal interval numbers, the cth course of action ($1 \le c \le n$) is attributed an interval $[x_{cd} \;\; y_{cd}]$ by the dth decision maker ($1 \le x_{cd}, y_{cd} \le n$, $x_{cd} \le y_{cd}$ and $1 \le d \le m$) which expresses the preference that the dth decision maker assigns to the cth course of action when compared to the entire set of courses of action. Since interval creation is performed independently by each decision maker and for every course of action, nm intervals are needed in order to cover all courses of action as well as all decision makers. These intervals state the constraints/preferences between courses of action in a purely implicit manner, thus providing (i) a straightforward and uncomplicated means of expressing the preference(s) for each course of action, and (ii) an efficient way of performing course of action comparison, aggregation and ranking. To date, group decision making via

ordinal interval number ranking as well as the identification of the preferred course(s) of action has been implemented using interval goal programming [19] and ranking degrees of possibility [20] on uniformly distributed intervals. An example of ordinal interval number representation (quoted from [20], p. 1413), involving a single decision maker, helps to illustrate these points:

> *"A consumer wants to buy a car among the four color cars (black, white, blue and yellow). His/her preferences are the following: The black one is ranked top 2, the white one is top 3, the blue one is second or third, and the yellow one is bottom."*

Clearly, the available courses of action are the four car colours and the decision maker's constraints/preferences are expressed by the ranking intervals [1 2], [1 3], [2 3] and [4] for the black, white, blue and yellow car colours, respectively. As will be shown in Section III, integration of the four intervals for the single decision maker results into three optimal preference rankings, one of them being {black, white, blue, yellow}, i.e. selecting the black coloured car. Although the aforementioned example is useful for understanding the principles of the ordinal interval number decision making method as well as of the genetic algorithm representation and mode of operation (Section IV.*A.*), it should be stressed that - as demonstrated in Section IV.*B.* - the power of the proposed representation and course-of-action selection, especially during the aggregation stage, can only be fully understood for groups involving more than one decision maker.

III. Genetic Algorithms

Dovetailing the principles of natural evolution within stationary or changing environments, genetic algorithms operate by creating and manipulating populations of candidate solutions to a problem. These solutions are encoded as chromosomes that evolve via the operators of crossover, mutation and selection. Each gene of a chromosome represents an elementary component of the problem that is salient in determining the make-up as well as the quality of the encoded solution. Additionally, a fitness function is used for explicitly grading each chromosome in terms of its goodness in constituting a solution to the problem.

Genetic algorithms begin their operation with a randomly created initial population of candidate solutions to a given problem. Offspring are created by combining homologous parts of the corresponding chromosomes of the population (crossover) while also randomly changing genes of the offspring chromosomes with a small probability (mutation). Crossover and mutation are complemented by selection, according to which the offspring with higher fitness values have higher probabilities of being inserted into the new population, while members of the population with lower fitness values have higher probabilities of being excluded from it. The inclusion of both old and new chromosomes in the evolving population is possible, with their proportion in the population being problem- as well as solution-specific. The repeated application of the aforementioned three genetic operators drives the population towards an increasing value of average fitness such that, after an adequate number of generations, a (near-) optimal solution to the problem at hand appears in the population.

IV. Genetic Algorithms for Ordinal Interval Number Preference Ranking Group Decision Making

A. Genetic Algorithm Set-up

An alternative to the aforementioned group decision making and ranking approaches using ordinal interval numbers [19-20] is put forward. The proposed methodology is significantly simpler in expressing/aggregating[1] the various ordinal interval numbers; with the course of action ranking being implemented via genetic algorithms. In the next paragraphs, chromosome and fitness function construction are detailed, followed by the application of the crossover, adjusted mutation and selection operators.

Each chromosome i ($1 \le i \le H$, where H is the total number of chromosomes in the population) comprises n genes, i.e. as many as there are alternative courses of action; the jth gene (j=1, 2, …, n) of chromosome i takes on a distinct integer value g_{ij} between 1 and n representing the rank corresponding to the jth course of action. Clearly, for the chromosome ($1 \le i \le H$) to be valid, the set of the values $\{g_{i1}, g_{i2}, \ldots, g_{in}\}$ of its genes must constitute a permutation of the numbers 1, 2,…, n.

The fitness function expresses the collective compatibility of the ranking represented by all the genes of chromosome i ($1 \le i \le H$) in terms of the intervals proposed by all the decision makers. As shown in Equation (1), the fitness fit_i of the ith chromosome is calculated by incrementing its value every time the ranking proposed by a gene of the chromosome lies within the interval proposed by a decision maker for the particular course of action and decremented every time it lies outside that interval. The evaluation of the fitness value of chromosome i is performed for all combinations of the n courses of action and the m decision makers via

$$fit_i = \frac{1}{n} \cdot \sum_{c=1}^{m} \sum_{d=1}^{n} s_{cd}, i = 1,2,...,H \qquad (1)$$

with

$$s_{cd} = \begin{cases} +imp_d & if \quad x_{cd} \le g_{ic} \le y_{cd} \\ -imp_d & otherwise \end{cases} \qquad (2)$$

where s_{cd} is the increment/decrement to fit_i for each combination of course of action (index c) and decision maker (index d), depending on whether the rank g_{ic} of the cth

[1] For one or more decision makers, respectively.

course of action lies within the interval [x_{cd} y_{cd}] proposed by the *d*th decision maker. The increment/decrement in Equation (2) is given by the relative importance imp_d assigned to the *d*th decision maker ($1 \le d \le m$), where it is necessary that

$$\sum_{d=1}^{m} imp_d = 1 \qquad (3)$$

Finally, the factor $1/n$ in Equation (1) acts as a normalization term, ensuring that fit_i ($1 \le i \le H$) lies within the interval [-1 1], with fit_i=-1 if the *i*th chromosome does not satisfy any interval proposed by any decision maker and fit_i=1 if the *i*th chromosome satisfies all the intervals proposed by all the decision makers.

The shape of the fitness function landscape pertaining to the example introduced in Section II is portrayed in Table I. The exhaustive presentation of the 24 possible chromosomes is tabulated, together with the contribution of each gene to the total fitness of the chromosome; the symbols *R* and *P* represent rewards and penalties, respectively. The chromosomes of maximum fitness, which in this example satisfy all of the decision maker's preferences, are highlighted. Since, in this case *m*=1, the importance imp_1 assigned to the sole decision maker equals 1.

TABLE I
GENETIC ALGORITHM FITNESS FUNCTION LANDSCAPE ([20], p. 1413)

Chromosome	1st course of action [1 2]	2nd course of action [1 3]	3rd course of action [2 3]	4th course of action [4]	Fitness function
1 2 3 4	*R*	*R*	*R*	*R*	1
1 2 4 3	*R*	*R*	*P*	*P*	0
1 3 2 4	*R*	*R*	*R*	*R*	1
1 3 4 2	*R*	*R*	*P*	*P*	0
1 4 2 3	*R*	*P*	*R*	*P*	0
1 4 3 2	*R*	*P*	*R*	*P*	0
2 1 3 4	*R*	*R*	*R*	*R*	1
2 1 4 3	*R*	*R*	*P*	*P*	0
2 3 1 4	*R*	*R*	*P*	*R*	0.5
2 3 4 1	*R*	*R*	*P*	*P*	0
2 4 1 3	*R*	*P*	*P*	*P*	-0.5
2 4 3 1	*R*	*P*	*R*	*P*	0
3 1 2 4	*P*	*R*	*R*	*R*	0.5
3 1 4 2	*P*	*R*	*P*	*P*	-0.5
3 2 1 4	*P*	*R*	*P*	*R*	0
3 2 4 1	*P*	*R*	*P*	*P*	-0.5
3 4 1 2	*P*	*P*	*P*	*P*	-1
3 4 2 1	*P*	*P*	*R*	*P*	-0.5
4 1 2 3	*P*	*R*	*R*	*P*	0
4 1 3 2	*P*	*R*	*R*	*P*	0
4 2 1 3	*P*	*R*	*P*	*P*	-0.5
4 2 3 1	*P*	*R*	*R*	*P*	0
4 3 1 2	*P*	*R*	*P*	*P*	-0.5
4 3 2 1	*P*	*R*	*R*	*P*	0

The genetic algorithm is implemented by submitting pairs of chromosomes of the current population to crossover; these chromosomes are selected via roulette-wheel according to their fitness values. Subsequently, random-point crossover is implemented on every selected pair. The aforementioned combination of selection and crossover implies that here may be duplicated genes in the resulting new chromosomes. Each new chromosome is checked for consistency, i.e. whether its genes constitute a permutation of values 1,2,…, *n* and, if the chromosome is not consistent, adjustment mutation is put into action by randomly choosing one gene of each pair of duplicated genes and changing its value so as to restore validity; mutation is not performed if the chromosome is consistent following crossover. All the (consistent) chromosomes are added to the current population.

Subsequently, roulette-wheel selection is employed for determining the chromosomes to be included in the new population. With the dual aim of promoting diversity as well as of preventing the premature convergence of the genetic algorithm to a sub-optimal solution, the inclusion of duplicate chromosomes from the current population is prohibited. If the distinct chromosomes that are candidates for inclusion in the next population are not adequate, new - and different from the existing - chromosomes are created in a random fashion.

The size of the population *H* as well as the number of iterations until the termination of the genetic algorithm are determined by gene length, i.e. by the total number *n* of courses of action, but are independent of the number *m* of decision makers.

Premature termination of operation is effectuated as soon as the fitness of a chromosome *i* ($1 \le i \le C$) of the population reaches a perfect value of

$$fit_i = 1 \qquad (4)$$

i.e. the chromosome satisfies all the intervals proposed by all the decision makers for all the courses of action. Clearly, this becomes increasingly rare as the total number of courses of action and/or of decision makers rises.

TABLE II
EASTFONE GDECISION MAKER ([20], p. 1420)

Decision maker	1st course of action	2nd course of action	3rd course of action	4th course of action
1	[2 3]	[1 1]	[2 4]	[3 4]
2	[3 4]	[1 2]	[1 2]	[3 4]
3	[4 4]	[2 3]	[2 3]	[1 2]
4	[1 1]	[2 3]	[4 4]	[2 3]
5	[2 3]	[1 3]	[1 2]	[4 4]

B. Demonstration

The set-up and mode of operation of the genetic algorithm are demonstrated on a problem discussed in [20] (p. 1420). This problem involves four alternative courses of action (*n*=4) and five decision makers (*m*=5). It describes the group decision making process of EASTFONE, one of the top software companies in China, as to which of four transatlantic companies (HP, PHILIPS, EMC and SAP) it

should establish a strategic alliance with. The five decision makers that are consulted come from a different background each (management, engineering, finance, human resources and business process outsourcing). The preferences of the five decision makers are deemed as equally important (imp_d=0.2, $1 \le d \le 5$), whereby the relative importance vector of their decisions equals [0.2, 0.2, 0.2, 0.2, 0.2]. The preference intervals of this problem are depicted in Table II, organised horizontally per decision maker and vertically per alternative course of action.

As for the problem described in Table I, the fitness function landscape comprises 24 points. Table III enumerates the 24 chromosomes that comprise the problem space, accompanied by their fitness values; additionally, the blue bars of Fig. 1 depict the distribution of the fitness values over the 24 chromosomes. A single global maximum fitness exists for chromosome {3 1 2 4} which takes on the fitness value of 0.3; there are also two local maxima for chromosomes {3 2 1 4} and {4 1 2 3}, which take on the next best fitness value of 0.2. These chromosomes are direct neighbours to chromosome {3 1 2 4} as they each have two genes swapped compared the chromosome of maximum fitness.

TABLE III
GENETIC ALGORITHM FITNESS FUNCTION LANDSCAPE ([20], p. 1420)

Chromosome	Relative Importance Vector					
	[0.2 0.2 0.2 0.2, 0.2]	[0.4 0.15 0.15 0.15 0.15]	[0.15 0.4 0.15 0.15 0.15]	[0.15 0.15 0.4 0.15 0.15]	[0.15 0.15 0.15 0.4 0.15]	[0.15 0.15 0.15 0.15 0.4]
1 2 3 4	0	0	0	0	0	0
1 2 4 3	0	0	0	-.125	**.25**	-.125
1 3 2 4	.1	.075	.075	.075	.075	.2
1 3 4 2	-.2	-.275	-.4	-.15	.1	-.275
1 4 2 3	-.2	-.15	-.15	-.275	-.15	-.275
1 4 3 2	-.5	-.5	-.625	-.375	-.375	-.625
2 1 3 4	0	.25	0	-.125	-.25	.125
2 1 4 3	0	.25	0	-.25	0	0
2 3 1 4	**0**	**0**	**0**	**-.125**	**-.125**	**.25**
2 3 4 1	-.2	-.15	-.4	-.15	-.15	-.15
2 4 1 3	-.3	-.225	-.225	-.475	-.35	-.25
2 4 3 1	-.5	-.375	-.625	-.375	-.625	-.5
3 1 2 4	**.3**	**.475**	**.475**	.1	-.025	**.475**
3 1 4 2	0	.125	0	-.125	0	0
3 2 1 4	.2	.15	.4	**.025**	.025	.4
3 2 4 1	0	0	0	0	0	0
3 4 1 2	-.3	-.35	-.225	-.35	-.35	-.225
3 4 2 1	-.2	-.15	-.15	-.15	-.4	-.15
4 1 2 3	.2	.275	.4	.15	.025	.15
4 1 3 2	-.1	-.075	-.075	.05	-.2	-.2
4 2 1 3	.1	-.05	.325	.075	.075	.075
4 2 3 1	-.1	-.2	-.075	.175	-.2	-.2
4 3 1 2	-.1	-.325	-.075	.05	-.075	-.075
4 3 2 1	0	-.125	0	**.25**	-.125	0

During the operation of the genetic algorithm, the population is composed of four chromosomes and is initialized to four randomly created, valid and distinct-from-each-other chromosomes. Roulette-wheel selection is applied to two pairs of chromosomes of the original population, whereby it is possible for one or even two chromosome to be selected in both pairs. Each pair of chromosomes is submitted to random-point crossover. If necessary, one or more of the four new chromosomes are submitted to adjusted mutation in the manner described in Section IV.*A*. in order to be transformed into consistent candidate solutions. Subsequently, roulette-wheel selection is applied to the set comprising both the chromosomes of the current population and the four newly created chromosomes and four chromosomes are selected. If duplication occurs in the new population, adjusted mutation attempts to change the duplicate chromosomes; if that fails, a completely new chromosome that is different from the chromosomes already inserted in the new population is created.

It has been found that, out of a total of 250 trials, the appearance of optimal solution (chromosome {3 1 2 4}) in the original population occurs in 19% of the trials. Overall, an average of 2.3 iterations are needed until successful convergence upon the optimal solution is accomplished. This means that only about 9.2 out of the 24 chromosomes must be created until the preferred course of action is converged upon.

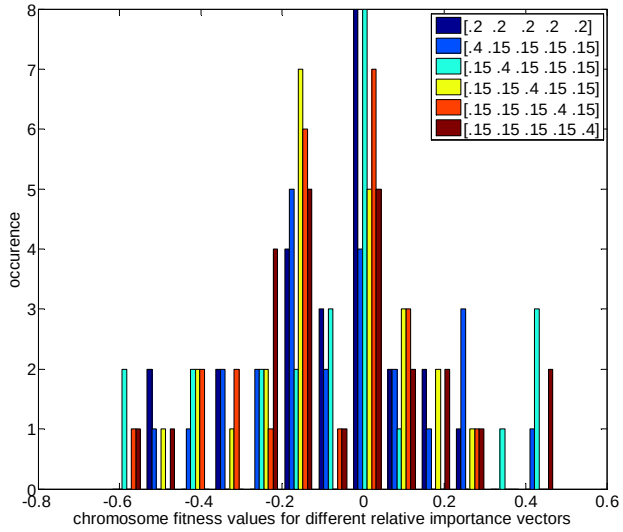


Fig. 1. Occurrence of fitness values over the entire population of solutions (24 chromosomes) for the different relative importance vectors.

C. Sensitivity Evaluation

Finally, the sensitivity of the proposed approach is investigated when the relevant importance values assigned to the five decision makers are perturbed. The rightmost five columns of Table III show the fitness function landscape for the five cases where one decision maker is assigned a relative importance value of 0.4 and the remaining four decision makers are assigned relative importance values of 0.15.

In three out of five cases, chromosome {3 1 2 4}

constitutes the global maximum, exactly as it does in the case of equal relative importance values being assigned to all decision makers. It is interesting that, for the remaining two cases, chromosome {3 1 2 4} is ranked fourth and eleventh, respectively. For both of these cases, this is due to the significantly greater importance assigned to a decision maker whose preferred interval for at least one candidate course of action differs considerably from that of the other decision makers, namely

(a) the third decision maker's interval of [1 2] for the fourth course of action diverges from the intervals proposed by the remaining four decision makers for the same course of action;
(b) the fourth decision maker's interval of [1 1] for the first course of action diverges from the intervals proposed by the remaining four decision makers for the same course of action.

TABLE IV
EFFECT OF THE RELATIVE IMPORTANCE VECTOR ON THE FITNESS FUNCTION LANDSCAPE ([20], p. 1420)

perturbed relative importance vectors similarity criteria	[0.4 0.15 0.15 0.15 0.15]	[0.15 0.4 0.15 0.15 0.15]	[0.15 0.15 0.4 0.15 0.15]	[0.15 0.15 0.15 0.4 0.15]	[0.15 0.15 0.15 0.15 0.4]
mean of absolute differences	.0854	.075	.1146	.1104	.074
standard deviation of differences	.1157	.1109	.1408	.143	.1001

These findings are further confirmed in Table IV, where the difference in fitness values of homologous chromosomes between the relative importance vector of [0.2 0.2 0.2 .02 0.2] and each of the five perturbed relative importance vectors is tabulated in terms of two similarity criteria, namely the mean of the absolute differences in fitness values and the standard deviation of the differences in fitness values. The fifth, second and first of the perturbed relative importance vectors appear closest to the original relative importance vector in terms of both similarity criteria, while the remaining two vectors show significantly higher differences in terms of the two criteria. The medium and light blue as well as the yellow, orange and brown bars of Fig. 1 further illustrate the differences in the distribution of the fitness values over the 24 chromosomes when compared with the blue bars created for the [0.2 0.2 0.2 0.2 0.2] relative importance vector.

D. Comparison with Existing Techniques

It is worth comparing the results obtained by the proposed approach with those reported in [20]. The preferred ranking in [20] is represented by chromosome {2 3 1 4} (rendered bold and highlighted in Table III), which only ranks sixth best by the proposed approach. It is necessary to try and understand the differences between the two approaches that lead to this discrepancy (in terms of goal, methodology and operation), a task that constitutes the aim of future research.

V. CONCLUSIONS - FUTURE DIRECTIONS

Group decision making has been investigated using ordinal interval number preference ranking. Genetic algorithms have been employed for encoding the different rankings of the alternative courses of action, as put forward by each decision maker, and for establishing the mutually preferred course(s) of action. In terms of representation, genes and chromosomes are simple to create, transparent as well as easy to manipulate; additionally, the relative importance values assigned to the decision makers can be directly incorporated in the fitness function. Finally, in terms of operation, the combination of crossover and adjusted mutation has been found efficient for convergence upon a preferred course of action.

REFERENCES

[1] C. M. Moore "Group Techniques for Idea Building (Applied Social Research Methods)," Sage Publications Inc., Thousand Oaks, CA, U.S.A., 1987
[2] C. L. Hwang and M. J. Lin "Group Decision Making under Multiple Criteria: Methods and Applications," Springer-Verlag, Berlin, Germany, 1987
[3] D.E. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning," Addison Wesley, 1989
[4] J. C. Borda "Memoire sur les Elections an Scrutin," Histoire de l' Academie Royale de Science, Paris, 1784
[5] J. G. Kemeny & L. J. Snell "Preference Ranking: an Axiomatic Approach," Mathematical Models in the Social Sciences, Ginn, Boston, pp. 9-32, 1962
[6] M. Kendall "Rank Correction Methods (third edition)," Hafners, New York, 1962
[7] J. M. Blin "Assessment Formulation of the Multiattribute Decision Problem," Review Automatique, Informatique et Recherche Operationelle, vol. 10, pp. 21-23, 1976
[8] R. D. Armstrong, W. D. Cook and L. Seiford "Priority Ranking and Consensus Formation: The Case of Ties," Management Science, vol. 28, pp. 638-645, 1982
[9] K. J. Arrow "Social choice and Individual Values," Wiley, New York, 1951
[10] K. Inada "The Simple Majority Rule," Econometrica, vol. 37, pp. 490-506, 1969
[11] W. D. Cook and M. Kress "Ordinal Ranking and Preference Strength," Mathematical Social Sciences, vol. 11, pp. 295-306, 1986
[12] Y. M. Wang, J. B. Yang and D.L. Xu "A Preference Aggregation Method Through the Estimation of Utility Intervals," Computers & Operations Research, vol. 32, pp. 2027-2049, 2005
[13] Z.- P. Fan, J. Ma, Y.- H. Sun and L. Ma "A Goal Programming Approach to Group Decision Making Based on Multiplicative Preference Relations and Fuzzy Preference Relations," European Journal of Operational Research, vol. 174,pp. 311-321, 2006
[14] W. D. Cook and L. M. Seiford "On the Borda-Kendall Consensus Method for Priority Ranking Problem," Management Science, vol. 28, pp. 621-637, 1982
[15] W. D. Cook "Distance-Based and ad hoc Consensus Models in Ordinal Preference Ranking," European Journal of Operational Research, vol. 172, pp. 369-385, 2006

[16] E. Herrera-Viemakera, F. Herrera, F. Chiclana and M. Luque ”Some Issues on Consistency of Fuzzy Preference Relations,” European Journal of Operational Research, vol. 154, pp. 98-109, 2002

[17] Z. Xu “Uncertain Linguistic Aggregation Operators Based Approach to Multiple Attribute Group Decision Making under Uncertain Linguistic Environment,” Information Sciences, vol. 168, pp. 171-184, 2004

[18] Y.- P. Jiang, Z.- P. Fan and J. Ma “A Method fir Group Decision Making in Multi-Granularity Linguistic Assessment Information,” Information Sciences, vol. 178, pp. 1098-1109, 2007

[19] J. Gonzalez-Pachon, C. Romero “Aggregation of Partial Ordering Rankings: An Interval Goal Programming Approach,” Computers and Operations Research, vol. 28, pp. 827-834, 2001

[20] Z.- P. Fan, Y, Liu “An Approach to Solve Group-Decision-Making Problems with Ordinal Interval Numbers,” IEEE Transactions on Systems, Man, and Cybernetics,” vol. 40, pp. 1413-1423, 2010

Genetic and Evolutionary Feature Extraction via X-TOOLSS

Joseph Shelton[1], Gerry Dozier[1], Kelvin Bryant[1], Lasanio Small[1] Joshua Adams[1], Khary Popplewell[1], Tamirat Abegaz[2], Aniesha Alford[3], Damon L. Woodard[4], Karl Ricanek[5]
Center for Advanced Studies in Identity Sciences (CASIS@A&T)
[1] Computer Science, North Carolina Agricultural and Technical State University
1601 East Market St. Greensboro, NC, 27411, United States of America
[2] Computational Science and Engineering, North Carolina Agricultural and Technical State University
1601 East Market St. Greensboro, NC, 27411, United States of America
[3] Electrical Engineering, North Carolina Agricultural and Technical State University
1601 East Market St. Greensboro, NC, 27411, United States of America
[4] Computer Science, 314 McAdams Hall, Clemson University, Clemson, S.C. 29634-0974, United States of America
[5] Computer Science, CIS Building, Room 2010, 601 South College Road, Wilmington NC, 28403

Abstract - *In [14], Shelton et al. presented two Genetic and Evolutionary Methods (GEMs) that evolved feature extractors (FE) for facial recognition. One of the methods presented evolved FEs that consisted of non-uniform, overlapping "patches" that did not cover the entire image. The other was similar with the exception that it evolved FEs that consisted of patches that were of uniform size. The two GEMs, referred to as Genetic & Evolutionary Feature Extractors (GEFE), were instances of a Steady-State Genetic Algorithm (SSGA). In [14], the instance of GEFE that evolved FEs consisting of uniform sized patches outperformed its counterpart that evolved FEs consisting of non-uniformed sized patches. This paper compares SSGA instances of GEFE, as reported in [14], with two additional Estimation of Distribution Algorithm (EDA) instances of GEFE. Our results show that the EDA instance of GEFE that evolved FEs consisting of uniform sized patches had the best overall performance.*

Keywords: Local Binary Patterns Method (LBPM), Estimation of Distribution Algorithm (EDA), Steady-State Genetic Algorithm (SSGA), Feature Extraction, Feature Reduction.

1 Introduction

Biometric recognition is the science of identifying an individual or group of individuals based on physical or behavioral characteristics or traits [2,18]. One of the most popular biometric modalities is the face [3,10,15]. One of the more widely used techniques for extracting features from facial images for the purpose of biometric recognition is the Local Binary Patterns method (LBPM) [1].

In [14], Shelton et al. introduces two genetic and evolutionary methods (GEMs) [11] 10for evolving feature extractors (FEs) for facial biometric recognition. These GEMs were referred to as Genetic & Evolutionary Feature Extractors (GEFE) and were instances of a Steady-State Genetic Algorithm (SSGA) [23]. The two GEFEs differed in that one evolved FEs that consisted of uniformed sized patches (referred to as $GEFE_u$) while the other evolved FEs consisting of non-uniform sized patches (referred to as $GEFE_n$). It was shown in [14] that $GEFE_u$ outperformed $GEFE_n$. This paper presents and compares the performance of two additional GEFE that are instances of an Estimation of Distribution Algorithm (EDA) [16].

The motivation behind this research presented in this paper is a two-tier hierarchical system proposed in [19] to reduce feature checks by using Short-Length Iris Codes [5]. In a conventional biometric system, a total of *nm* feature checks are used to identify an individual where *n* represents the number of subjects and *m* represents the number of features.

In the proposed two-tier hierarchical system, the first stage uses a reduced set of *k* features is used to identify a subset of *r* individuals that are closest to a probe, *p*. In the second stage, the system uses the full feature set consisting of m features to compare *p* to each of the *r* subjects.

Equation 1 shows the relative savings (in terms of feature checks) of the proposed two stage system where f_s represents to savings as a ratio of number of feature check made by the proposed system divided by the number of feature checks made by conventional system.

$$f_s = \frac{k}{m} + \frac{r}{n} \quad (1)$$

The remainder of this paper is as follows. In Section 2, we will introduce the LBP method (LBPM) and the concept of GEFE. In Section 3, we present our experiment, and in Section 4, we present our results. In Section 5, we present our conclusions and future work.

2 LBPM and GEFE

2.1 The LBPM for Feature Extraction

The LBPM [1] is a texture descriptor that can be used to extract features from any textured image. Within this research, LBP is applied to facial recognition, specifically images of the face. As presented in [10], the traditional LBPM uses patches that cover an entire image, are uniform, and non-overlapping.

A LBPM feature extractor (FE) can be described as follows. Initially, an entire image is partitioned in a user-specified number of uniform patches as shown in Figure 1.

Figure 1: Standard LBP

For each patch on an image, the center pixels must be sought out. The LBPM compares the center pixel to its surrounding neighboring pixels to create a LBP value. Equation 2 shows the function that computes the LBP value. *N* represents a set of neighboring pixel values, *C* is the center pixel value, and *k* is the neighborhood size, where $k = 8$ in this research.

$$LBP(N,C) = \sum_{k=0}^{7} P(N_k - C)2^k \qquad (2)$$

The function LBP(N,C) uses $P(N_k - C)$, which shown in Equation 3 returns a 1 or a 0 depending on the difference between the center pixel and the neighbor.

$$P(y) = \begin{cases} 1, y >= 0 \\ 0, y < 0 \end{cases} \qquad (3)$$

Figure 2 provides an example of a neighborhood of pixels shown with intensity values displayed in the Pixel Values matrix. The center intensity value of 150 is the center pixel, because it meets the requirements of being surrounded by 8 neighboring pixels. The differences are calculated and shown in the Differences matrix.

The transition from the Differences matrix and the Bit String Values matrix is possible by Equation 2. The Bit String Values matrix shows a series of zeros and ones. This matrix represents Equation 3, in which the summation of 2^k represents the value of a bit string. A starting position must be arbitrarily chosen to calculate the LBP value, by unwrapping the bit string and decoding it. If the starting position was assumed to be the top, left corner of the final matrix, the sequence would be 11011000, which decodes to 216 as the LBP value.

200	190	5 0
120	150	150
225	4 0	150

Pixel Values

5 0	4 0	-100
-30	N/A	0
-75	-110	0

Differences

1	1	0
0		1
0	0	1

Bit String Values

Figure 2: LBP Example

In an effort to create a feature vector of an image to be used in a distance measure, the number of all bit string patterns within a patch must be counted. There are a total of 256 possible bit string patterns for an 8 bit string. In an effort to reduce this number, a binary pattern is classified as either uniform or non-uniform. A uniform pattern is a bit string whose bits, when compared in sequential order, considering circular comparisons, shifts value 2 or less times. A non-uniform pattern is a bit string that has more than two bit shifts when comparing circularly, and sequentially.

If one considers the string 00011100 as an example, we see only 2 shifts occur. One is between the third and fourth position, and one between the sixth and seventh position. This string meets the requirements to be declared uniform. The string 01101111, has shifts that occur between the first and second position, the third and fourth position, the fourth and fifth position, and the eighth and first position; because the amount of shifts exceeds 2, this pattern is non-uniform. Out of the total 256 possible patterns, 58 of those patterns are uniform.

A histogram is created for every patch of an image, and is composed of 59 bins. The 58 uniform patterns are assigned to 58 bins in the histogram, with each bin storing the frequencies of the patterns. The 59^{th} is a bin that holds the count of all non-uniform patterns found in the patch. The work of Ojala and Pietikainen, in [1], suggests that the most discriminating features of a facial image contain predominantly uniform patterns. The histogram vectors associated with each patch are then concatenated to form a histogram representing the features extracted by the LBP method [14].

2.2 GEFE

The GEFEs developed in [14], evolved candidate FEs consisting of uniform and non-uniform sized patches. A candidate FE, fe_i, is a 6-tuple ($<X_i,Y_i,W_i,H_i,M_i,f_i>$) consisting of either uniform/non-uniform patches [14] and can be defined as follows: $X_i = \{x_{i,0}, x_{i,1},\ldots, x_{i,n-1}\}$ and $Y_i = \{y_{i,0}, y_{i,1}, \ldots, y_{i,n-1}\}$ describes the x and y coordinates of the center pixel of the *n* possible patches. $W_i = \{w_{i,0}, w_{i,1}, \ldots, w_{i,n-1}\}$ and $H_i = \{h_{i,0}, h_{i,1},\ldots, h_{i,n-1}\}$ describes the dimensions of the *n* possible patches, the widths and heights respectively. $M_i = \{m_{i,0}, m_{i,1},\ldots, m_{i,n-1}\}$ describes the mask for each patch (1 means to extract features from the corresponding patch, 0 means do not extract features from the corresponding patch), and f_i

represents the fitness of fe_i . The purpose of the mask value is to reduce the total amount of features that need to be compared in the feature vector. The fitness is the number of errors made when comparing probe sets and gallery sets multiplied by 10 plus the fraction of the n patches from which features were extracted (see Equation 4).

$$f_i = (NumErr*10) + ((\sum_{k=0}^{n-1} m_{i,k}) / n + 1) \quad (4)$$

Candidate FEs consisting of patches with uniform patch size [14] are similar with the exception that for any FE, fe_k , $W_k = \{w_{k,0}, w_{k,1}, \ldots , w_{k,n-1}\}$ where, $w_{k,0} = w_{k,1},\ldots, \ w_{k,n-2} = w_{k,n-1}$, meaning that the widths of every patch is the same. Similarly, $H_k = \{h_{k,0}, h_{k,1}, \ldots , h_{k,n-1}\}$ where, $h_{k,0} = h_{k,1},\ldots, \ h_{k,n-2} = h_{k,n-1}$, meaning that the height of every patch is the same.

The SSGA instances of GEFE worked as follows. Initially, population of candidate FEs was created. The candidate FEs are evaluated and assigned a fitness. Two parents are selected via tournament selection [7,8,9,12,21]. The parent FEs are used to create an offspring FE via uniform crossover and Gaussian mutation [22]. The offspring is then evaluated, assigned a fitness, and replaces the worst fit candidate FE in the population. This process of selecting parents, allowing them to create an offspring, evaluating the offspring, and replacing the worst fit individual of the population with the offspring, is repeated for a user-specified number of function evaluations. Figure 3 provides pseudo code version for an SSGA [14].

The EDA instances of GEFE are similar to the SSGA instances. However, in the EDA instances, a initial population of candidate FEs is created and evaluated. Next, a user specified number of FEs are then selected to build a probability density function (PDF) (for our experiment we used the top 50% of the population). Offspring are then created by sampling the PDF. The offspring then replace the previous population except for a user specified number of best individuals of the previous population (referred to as 'elites'). Figure 3 provides pseudo code for an EDA [14].

3 Experiment

The Experiment was based on a subset of 105 subjects taken from the Facial Recognition Grand Challenge (FRGC) dataset [13]. Each subject in the FRGC dataset has three slightly different images associated with it (an example subject is shown in Figure 4). Our dataset of 105 subjects consisted of a probe set (one image per subject), and a gallery set (two images per subject). As shown in Figure 4, the images were pre-processed so that there was no great variance in alignment. Since our dataset contained 105 subjects, a total of 105 images were in the probe set and 210 images were in the gallery set. The dimensions of the images were 100 by 127 pixels.

Probe Image Gallery Image Gallery Image

Figure 4: Subject 27's Snapshots

```
compute SSGA{
t = 0;
initialize pop(t)
evaluate pop(t)
While(Not done){
   Parent1 = Select_From_Pop(t)
   Parent2 = Select_From_Pop(t)
   Child = Procreate(Parent1, Parent2)
   Evaluate(Child)
   Replace(Worst(Pop(t+1), Child)
   t = t+1;
   }
}
```

```
compute EDA{
t = 0;
initialize pop(t)
evaluate pop(t)
While (Not done) {
   S(t) = selected subpopulation the
          best individuals;
   Build a probabilistic model, PDF(t),
     of S(t);
   Sample PDF(t) to generate O(t);
   Replace P(t)- Elites with O(t);
   t = t+1;
  }
}
```

Figure 3: Pseudo-code for the SSGA and EDA

For this experiment, we compared the Standard LBP method (SLBPM), GEFE with non-uniform sized patches using SSGA (SSGA-$GEFE_n$), GEFE with uniform sized patches using SSGA (SSGA-$GEFE_u$), GEFE with non-uniform sized patches using EDA (EDA-$GEFE_n$), and GEFE with uniform sized patches using EDA (EDA-$GEFE_u$).

4 Results

For our results, the SSGA and EDA instances of GEFE evolved a population of 20 candidate FEs. The SSGA used Gaussian mutation range of 0.1. The EDA constructed Gaussian based probability density functions and always kept the best FE of the previous generation (Elites = 1). The GEFE were run a total of 30 times with a maximum of 1000 function evaluations allowed for each run. The GEFE instances (SSGAs and EDAs) were taken from the eXploratory Toolset for the Optimization of Launch and Space Systems (X-TOOLSS) application [17].

In Table I, the average performance of the three methods is shown. The standard LBP method (SLBPM) needed to be run only once since the patch characteristics were deterministic. SSGA-GEFE$_n$ used an average of 38.65% of patches, with an average accuracy of 99.77% while SSGA-GEFE$_u$ used an average of 35.82% of patches, with an average accuracy of 100%. EDA-GEFE$_n$ used an average of 34.74% of patches, with an average accuracy of 99.81% while EDA-GEFE$_u$ used an average of 29.90% of patches, with an average accuracy of 100%. All of the GEFEs outperformed SLBPM and the EDAs outperformed their SSGA counterparts. The precent of patches activated was counted as a degree of goodness for a method because fewer patches activated means fewer features to be considered in subject matching.

Statistical analysis was performed on the different method, and the results confirmed that EDA-GEFE$_u$ had a statistically significant better performance (in terms of accuracy and features) than SSGA-GEFE$_n$, SSGA-GEFE$_u$, and EDA-GEFE$_n$.

In [15], it is noted that there are certain areas of a face that are discriminating enough to effectively distinguish between different persons. Figure 5 shows an approximate positioning of patches for the best feature extractors created using the GEFE$_n$ and the GEFE$_u^i$.

As also seen in [14], it is interesting to see that the majority of patches are around the ocular region. Because the GEFE$_n$ and the GEFE$_u$ chose this region to focus on, this suggests that this area holds textures that are unique enough to differentiate individuals from one another. This result is consistent with conclusions presented in [4,6].

TABLE I
EXPERIMENTAL RESULTS FOR LBP (EVEN DISTRIBUTION), SSGA AND EDA METHODS (UNIFORM AND NON-UNIFORM)

Methods	Patches Used	Avg. Accuracy	Best Accuracy
SLBPM	100.0%	99.04%	99.04%
SSGA-GEFE$_n$	38.65%	99.77%	100%
SSGA-GEFE$_u$	35.82%	100%	100%
EDA-GEFE$_n$	34.74%	99.81%	100%
EDA-GEFE$_u$	29.90%	100%	100%

5 Conclusion and Future Work

In this paper, four forms of GEFE were compared (along with SLBPM). Both methods GEFE$_u$ and GEFE$_n$ had better performance than the SLBPM. EDA-GEFE$_u$ had a better performance than EDA-GEFE$_n$, SSGA-GEFE$_u$ and SSGA-GEFE$_n$ in terms of features and accuracy. Future work will be devoted toward the investigation and comparison of GEFE using a variety of other forms of genetic and evolutionary computations found in X-TOOLSS.

6 Acknowledgement

This research was funded by the office of the Director of National Intelligence (ODNI), Center for Academic Excellence (CAE) for the multi-university, Center for Advanced Studies in Identity Sciences (CASIS) and by the National Science Foundation (NSF), Science & Technology Center: Bio/Computational Evolution in Action Consortium (BEACON). The authors would like to thank the ODNI and the NSF for their support of this research.

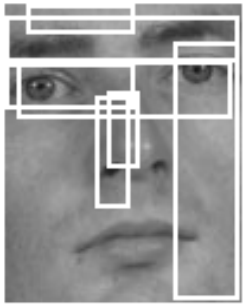


Figure 5a: SSGA Non Uniform

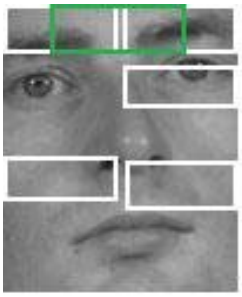


Figure 5b: SSGA Uniform

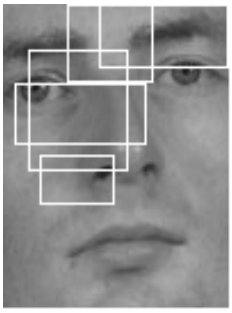


Figure 5c: EDA Non Uniform

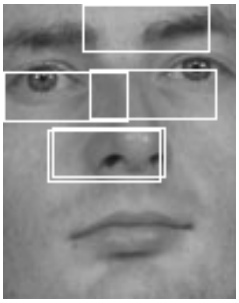


Figure 5d: EDA Uniform

Figure 5: Best Individuals

7 References

[1] Timo Ojala, Matti Pietikainen, *Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns*, IEEE Trans. Pattern Analysis and Machine Intelligence; 971-987; 2002

[2] Arun Ross, *An Introduction To Multibiometrics,* Appeared in Proc. of the 15th European Signal Processing Conference (EUSIPCO), (Poznanm Poland), September 2007

[3] S. Z. Li and A. K. Jain, Eds., *Handbook of Face Recognition*. Springer, 2005.

[4] Damon L. Woodard Shrinivas J. Pundlik Jamie R. Lyle Philip E. Miller, *Periocular Region Appearance Cues for Biometric Identification* In *CVPR Workshop on Biometrics, 2010*

[5] James E. Gentile, Nalini Ratha, Jonathan Connell *SLIC: Short-Length Iris Codes*, in Submission to: Biometrics: Theory, Applications and Systems (BTAS '09), 2010

[6] P. Miller, A. Rawls, S. Pundlik, and D. Woodard, "Personal identification using periocular skin texture," in *SAC '10: Proceedings of the 2010 ACM symposium on Applied Computing* . New York, NY, USA: ACM, 2010.

[7] L. Davis, *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.

[8] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, 2000.

[9] Kennedy and R. Eberhart, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[10] Timo Ahonen, Abdenour Hadid, Matti Pietikinen, "*Face Description with Local Binary Patterns: Application to Face Recognition*," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037-2041, Dec. 2006, doi:10.1109/TPAMI.2006.244.

[11] Jiming Liu, Y. Y Tang, Y. C. Cao, *An Evolutionary Autonomous Agents Approach to Image Feature Extraction IEEE Trans. Evolutionary Comput.* **1** 2 (1997), pp. 141–158.

[12] Ajith Abraham, Nadia Nedjah, Luiza de Macedo Mourelle *Evolutionary Computation: from Genetic Algorithms to Genetic Programming* "Studies in Computational Intelligence" Springer (2006), 1-20.

[13] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoff, J. Marques, J. Min, and W. Worek, "Overview of face recognition grand challenge," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[14] Joseph Shelton, Gerry Dozier, Kelvin Bryant, Lasanio Smalls, Joshua Adams, Khary Popplewell, Tamirat Abegaz, Damon L. Woodard, Karl Ricanek, *Comparision of Genetic-based Feature Extraction Methods for Facial Recognition* in *The 22nd Midwest Artificial Intelligence and Cognitive Science Conference (MAICS)*, 2011.

[15] J. Matas, P. B´ılek, M. Hamouz, and J. Kittler, *Discriminative Regions for Human Face Detection* in The 5th Asian Conference on Computer Vision, 23–25 January 2002, Melbourne, Australia

[16] P. Larranaga, and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2002.

[17] M. L. Tinker, G. Dozier, and A. Garrett, "The exploratory toolset for the optimization of launch and space systems (x-toolss)," http://nxt.ncat.edu/, 2010.

[18] Anil K. Jain, Patrick Flynn, and Arun A. Ross. 2007. *Handbook of Biometrics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[19] J.E. Gentile, N. Ratha, and J. Connell, "An efficient, two-stage iris recognition system", In *Proc. 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS)*, 2009.

[20] A. Alford, K. Popplewell, G. Dozier, K. Bryant, J. Kelly, J. Adams, T. Abegaz, and J. Shelton, "GEFeWS: A Hybrid Genetic-Based Feature Weighting and Selection Algorithm for Multi-Biometric Recognition", in *Proceedings of The 22nd Midwest Artificial Intelligence and Cognitive Science Conference (MAICS)*, 2011.

[21] G. Dozier, A. Homaifar, E. Tunstel, and D. Battle. "An Introduction to Evolutionary Computation" (Chapter 17), Intelligent Control Systems Using Soft Computing Methodologies, A. Zilouchian & M. Jamshidi (Eds.), pp. 365-380, CRC press, 2001.

[22] Gilbert Syswerda. 1989. Uniform Crossover in Genetic Algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, J. David Schaffer (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2-9.

[23] Smith, J.; Fogarty, T.C.; , "Self adaptation of mutation rates in a steady state genetic algorithm," *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on* , vol., no., pp.318-323, 20-22 May 1996 doi: 10.1109/ICEC.1996.542382

[i] To avoid confusion, one of the patches was colored differently.

Genetic Algorithm Finding the Shortest Path in Networks

Bilal Gonen[1]
[1]Department of Computer Science and Engineering, University of Nevada, Reno, Reno, Nevada, U.S.A.

Abstract— *With the growth of the Internet, Internet Service Providers (ISPs) try to meet the increasing traffic demand with new technology and improved utilization of existing resources. Routing of data packets can affect network utilization. Packets are sent along network paths from source to destination following a protocol. Open Shortest Path First (OSPF) is the most commonly used intra-domain Internet routing protocol (IRP). Traffic flow is routed along shortest paths, splitting flow at nodes with several outgoing links on a shortest path to the destination IP address. Link weights are assigned by the network operator. A path length is the sum of the weights of the links in the path. In this paper, I study the problem of optimizing OSPF weights, given a set of projected demands, with the objective of minimizing network congestion. The weight assignment problem is NP-hard. I developed a genetic algorithm (GA) to solve this problem.*

Keywords: genetic algorithm, shortest path, computer networks

1. Introduction

Routing is a fundamental engineering task on the Internet. It consists in finding a path from a source to a destination host. Routing is complex in large networks because of the many potential intermediate destinations a packet might traverse before reaching its destination [2]. The link weights are assigned by the network operator. The lower the weight, the greater the chance that traffic will get routed on that link [3]. When one sends or receives data over the Internet, the information is divided into small chunks called packets or datagrams. A header, containing the necessary transmission information, such as the destination Internet Protocol (IP) address, is attached to each packet. The data packets are sent along links between routers on Internet. When a data packet reaches a router, the incoming datagrams are stored in a queue to await processing. The router reads the datagram header, takes the IP destination address and determines the best way to forward this packet for it to reach its final destination [3].

The configuration of network protocols is widely considered a black art and is normally performed based on network administrators' experience, trial and error, etc... These manual methods are often error-prone and not scalable to large complex networks. The emphasis of the search algorithm should be on finding a better operating point within the limited time frame instead of seeking the strictly global optimum. Network conditions vary with time and the search algorithm should quickly find better network parameters before significant changes in the network occur. Another feature of these problems; for example, AT&T's network has 1000s of routers and links. If all OSPF link weights of this network are to be configured, there will be thousands of parameters present in the optimization [4].

2. The Shortest Path Problem

The shortest path problem is defined as that of finding a minimum-length (cost) path between a given pair of nodes [5]. Shortest path problem is a classical research topic. It was proposed by Dijkstra in 1959 and has been widely researched. The Dijkstra algorithm is considered as the most efficient method. It is based on the Bellman optimization theory. But when the network is very big, then it becomes inefficient since a lot of computations need to be repeated. Also it can not be implemented in the permitted time [9].

3. Genetic Algorithm

As a special kind of stochastic search algorithms, genetic algorithm is a problem solving method which is based on the concept of natural selection and genetics [6].

In the 1970s, Holland first introduced genetic algorithms to explain the adaptive processes of natural systems and to design an artificial system, which retains the robust mechanism of natural systems [5].

The steps of a GA are shown in Algorithm 1.

Algorithm 1 $GA\ Steps$[7]

1: Choose initial population
2: Evaluate the fitness of each individual in the population
3: **while** <terminating condition> **do**
4: Select best-ranking individuals to reproduce
5: Breed new generation through crossover and mutation (genetic operations) and give birth to offspring
6: Evaluate the individual fitnesses of the offspring
7: Replace worst ranked part of population with offspring
8: **end while**

4. The GA algorithm that I implemented

The steps of my GA algorithm are explained in this section.

4.1 Choose Initial Population

When initializing the population, my algorithm starts from the SOURCE. SOURCE is a constant in the program, so the user may want to pick another node as the starting point. The algorithm selects one of the neighbors provided that it has not been picked before. It keeps doing this operation until it reaches to DESTINATION. Like SOURCE, DESTINATION is also a constant that user may change as they wish.

4.2 Evaluate the fitness of each individual in the population

The evaluation function takes a path in the population. It gets the distance between each node pair in the path, by calling a function to read from the distance array. Adds them together and returns the sum as the cost of the path.

4.3 Select best-ranking individuals to reproduce

My algorithm selects two individuals from the population with the lowest costs.

4.4 Crossover and Mutation

With some probability, the program mates the two individuals. The crossover function takes two parents to mate. It looks for the common points in the parents. The common nodes are where these two paths intersect. Among the common points, the program selects one of them randomly. It makes the crossover from that point. The crossover operation is illustrated in Figure 1.

4.5 Evaluate the individual fitnesses of the offspring

I send these offspring to the evaluation function to get their fitnesses. If the offsprings' fitnesses are less than the nodes with maximum fitnesses in the population, I replace them with the nodes with the maximum fitnesses.

4.6 Terminating Condition

My terminating condition is a predefined number of iterations. Because, in the network topology, the goal is not to find the global optimum, but to find a path with a reasonable cost in a limited time.

5. Experiment Results

I generated a network topology with 20 nodes and 62 links to test my Genetic Algorithm. Each link has a cost associated with them. I set two nodes as source and destination. The goal of my GA application is to find a path between source and destination with the lowest cost.

In Figure 3, the cells with 10,000 in them represent that there is no direct link between those nodes. Because, 10,000 is too big compared to other small costs, therefore my implementation ignore those big numbers, and pick the links

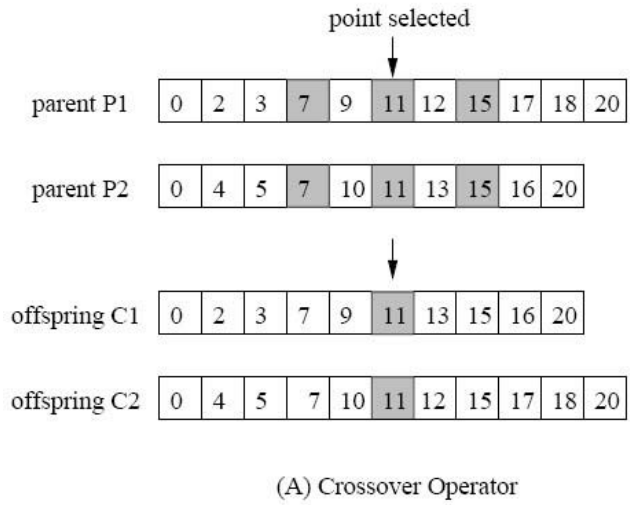


Fig. 1: Crossover Operator [7]

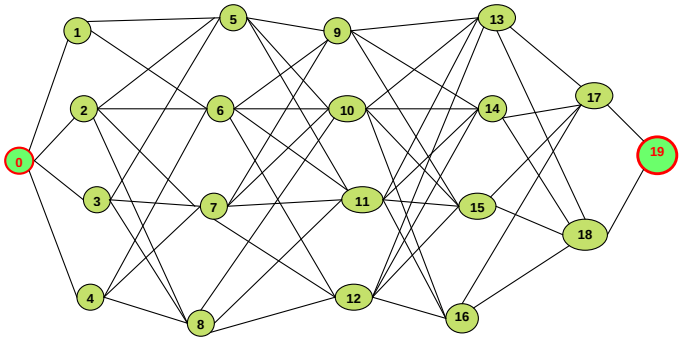


Fig. 2: Network topology used

with small costs, instead. Figure 4 shows a sample of initial population, and their fitnesses.

I set several parameters for the experiment. They are as follows; Population size=50, Number of runs=30, Number of generations=50, Crossover probability = 0.99, Mutation probability=0.1 I run the steps selection, crossover, and replace part 50 times (number of generations). Figure 5 and Figure 6 shows the average of maximum numbers of 30 runs, the average of minimum numbers of 30 runs, and the average of average numbers of 30 runs.

6. Analysis of Results

The results show that GA gets close to optimum very quickly. This is a promising result for my research. When using this GA algorithm besides other search algorithms in the USF [8], such as, multi-start hill-climbing, simulated annealing, Controlled Random Search and RRS (Recursive Random Search), I can start searching the space with GA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	10000	52	61	8	16	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
1	52	10000	10000	10000	10000	78	41	6	92	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
2	61	10000	10000	10000	10000	84	63	2	99	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
3	8	10000	10000	10000	10000	71	48	223	73	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
4	16	10000	10000	10000	10000	63	55	44	88	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
5	10000	78	84	71	63	10000	10000	10000	10000	11	22	33	44	10000	10000	10000	10000	10000	10000	10000
6	10000	41	63	48	55	10000	10000	10000	10000	21	32	43	54	10000	10000	10000	10000	10000	10000	10000
7	10000	6	2	223	44	10000	10000	10000	10000	74	85	96	14	10000	10000	10000	10000	10000	10000	10000
8	10000	92	99	73	88	10000	10000	10000	10000	46	64	75	35	10000	10000	10000	10000	10000	10000	10000
9	10000	10000	10000	10000	10000	11	21	74	46	10000	10000	10000	10000	66	55	44	11	10000	10000	10000
10	10000	10000	10000	10000	10000	22	32	85	64	10000	10000	10000	10000	91	97	73	19	10000	10000	10000
11	10000	10000	10000	10000	10000	33	43	96	75	10000	10000	10000	10000	45	65	25	85	10000	10000	10000
12	10000	10000	10000	10000	10000	44	54	14	35	10000	10000	10000	10000	73	37	87	16	10000	10000	10000
13	10000	10000	10000	10000	10000	10000	10000	10000	10000	66	91	45	73	10000	10000	10000	10000	86	84	10000
14	10000	10000	10000	10000	10000	10000	10000	10000	10000	55	97	65	37	10000	10000	10000	10000	74	76	10000
15	10000	10000	10000	10000	10000	10000	10000	10000	10000	44	73	25	87	10000	10000	10000	10000	2	6	10000
16	10000	10000	10000	10000	10000	10000	10000	10000	10000	11	19	85	16	10000	10000	10000	10000	7	9	10000
17	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	86	74	2	7	10000	10000	52
18	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	84	76	6	9	10000	10000	25
19	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	52	25	10000

Fig. 3: The costs on the links

```
Individual-0   Fitness: 222  path: 0 - 1 - 6 - 12 - 16 - 17 - 19 -
Individual-1   Fitness: 1025 path: 0 - 4 - 5 - 9 - 15 - 17 - 16 - 12 - 14 - 11 - 8 - 1 - 6 - 3 - 7 - 10 - 13 - 18 - 19
Individual-2   Fitness: 231  path: 0 - 3 - 5 - 11 - 16 - 18 - 19 -
Individual-3   Fitness: 879  path: 0 - 1 - 8 - 12 - 14 - 17 - 13 - 11 - 6 - 2 - 7 - 3 - 5 - 9 - 16 - 18 - 19 -
Individual-4   Fitness: 728  path: 0 - 3 - 7 - 11 - 13 - 10 - 5 - 2 - 6 - 9 - 15 - 18 - 19 -
Individual-5   Fitness: 574  path: 0 - 2 - 7 - 9 - 13 - 10 - 8 - 12 - 14 - 18 - 16 - 17 - 19 -
Individual-6   Fitness: 390  path: 0 - 3 - 5 - 1 - 6 - 12 - 14 - 18 - 19 -
Individual-7   Fitness: 901  path: 0 - 4 - 6 - 1 - 5 - 12 - 16 - 10 - 15 - 17 - 14 - 11 - 7 - 2 - 8 - 9 - 13 - 18 - 19
Individual-8   Fitness: 755  path: 0 - 3 - 6 - 1 - 5 - 9 - 13 - 12 - 7 - 4 - 8 - 11 - 14 - 18 - 16 - 17 - 19 -
Individual-9   Fitness: 466  path: 0 - 2 - 5 - 9 - 8 - 10 - 13 - 18 - 19 -
Individual-10  Fitness: 364  path: 0 - 4 - 8 - 12 - 13 - 18 - 16 - 17 - 19 -
Individual-11  Fitness: 210  path: 0 - 1 - 7 - 12 - 14 - 18 - 19 -
Individual-12  Fitness: 380  path: 0 - 3 - 7 - 9 - 15 - 18 - 19 -
Individual-13  Fitness: 281  path: 0 - 2 - 8 - 9 - 15 - 18 - 19 -
Individual-14  Fitness: 792  path: 0 - 2 - 8 - 11 - 15 - 18 - 14 - 10 - 6 - 1 - 5 - 12 - 7 - 9 - 16 - 17 - 19 -
Individual-15  Fitness: 531  path: 0 - 4 - 7 - 1 - 8 - 11 - 6 - 10 - 14 - 17 - 19 -
Individual-16  Fitness: 954  path: 0 - 2 - 8 - 4 - 5 - 12 - 7 - 11 - 13 - 18 - 16 - 9 - 15 - 10 - 14 - 17 - 19 -
Individual-17  Fitness: 700  path: 0 - 3 - 7 - 12 - 6 - 11 - 13 - 18 - 14 - 9 - 15 - 17 - 19 -
Individual-18  Fitness: 243  path: 0 - 3 - 6 - 11 - 16 - 17 - 19 -
Individual-20  Fitness: 219  path: 0 - 1 - 5 - 11 - 15 - 18 - 19 -
Individual-21  Fitness: 566  path: 0 - 3 - 7 - 10 - 15 - 17 - 14 - 18 - 19 -
Individual-22  Fitness: 476  path: 0 - 1 - 8 - 2 - 5 - 10 - 15 - 17 - 19 -
Individual-23  Fitness: 683  path: 0 - 2 - 7 - 3 - 6 - 11 - 13 - 9 - 15 - 10 - 16 - 17 - 19 -
Individual-24  Fitness: 292  path: 0 - 3 - 6 - 12 - 13 - 18 - 19 -
Individual-25  Fitness: 329  path: 0 - 1 - 6 - 12 - 13 - 18 - 19 -
Individual-26  Fitness: 147  path: 0 - 1 - 7 - 12 - 16 - 17 - 19 -
Individual-27  Fitness: 765  path: 0 - 2 - 7 - 9 - 14 - 11 - 8 - 3 - 5 - 12 - 16 - 10 - 13 - 17 - 15 - 18 - 19 -
Individual-28  Fitness: 188  path: 0 - 4 - 5 - 9 - 15 - 17 - 19 -
Individual-29  Fitness: 712  path: 0 - 4 - 5 - 11 - 13 - 9 - 8 - 12 - 6 - 2 - 7 - 10 - 14 - 17 - 15 - 18 - 19 -
Individual-30  Fitness: 290  path: 0 - 3 - 6 - 11 - 14 - 17 - 19 -
Individual-31  Fitness: 271  path: 0 - 2 - 5 - 10 - 15 - 18 - 19 -
Individual-32  Fitness: 407  path: 0 - 1 - 6 - 9 - 16 - 12 - 5 - 10 - 13 - 18 - 19 -
Individual-33  Fitness: 565  path: 0 - 4 - 7 - 11 - 15 - 9 - 16 - 18 - 13 - 12 - 14 - 17 - 19 -
Individual-34  Fitness: 288  path: 0 - 3 - 5 - 11 - 15 - 10 - 16 - 17 - 19 -
Individual-35  Fitness: 471  path: 0 - 3 - 8 - 9 - 16 - 18 - 14 - 12 - 13 - 17 - 19 -
Individual-36  Fitness: 391  path: 0 - 4 - 8 - 10 - 14 - 17 - 19 -
Individual-37  Fitness: 1081 path: 0 - 4 - 6 - 12 - 5 - 2 - 8 - 1 - 7 - 10 - 15 - 11 - 16 - 9 - 13 - 18 - 14 - 17 - 19
Individual-38  Fitness: 653  path: 0 - 2 - 6 - 10 - 15 - 12 - 14 - 18 - 16 - 9 - 13 - 17 - 19 -
Individual-39  Fitness: 190  path: 0 - 4 - 6 - 9 - 15 - 17 - 19 -
Individual-40  Fitness: 477  path: 0 - 2 - 6 - 9 - 13 - 11 - 5 - 12 - 14 - 17 - 15 - 18 - 19 -
Individual-41  Fitness: 354  path: 0 - 2 - 8 - 11 - 16 - 18 - 19 -
Individual-42  Fitness: 308  path: 0 - 3 - 8 - 9 - 14 - 17 - 19 -
Individual-43  Fitness: 1134 path: 0 - 2 - 6 - 11 - 7 - 3 - 8 - 4 - 5 - 10 - 15 - 18 - 16 - 9 - 14 - 12 - 13 - 17 - 19
Individual-44  Fitness: 349  path: 0 - 1 - 6 - 9 - 15 - 11 - 14 - 18 - 19 -
Individual-46  Fitness: 808  path: 0 - 3 - 5 - 4 - 8 - 1 - 7 - 11 - 15 - 9 - 16 - 17 - 13 - 12 - 14 - 18 - 19 -
Individual-47  Fitness: 373  path: 0 - 1 - 7 - 12 - 6 - 2 - 5 - 10 - 16 - 17 - 19 -
Individual-48  Fitness: 543  path: 0 - 1 - 7 - 12 - 14 - 10 - 5 - 4 - 6 - 11 - 13 - 18 - 19 -
Individual-49  Fitness: 929  path: 0 - 3 - 7 - 2 - 8 - 1 - 5 - 12 - 14 - 11 - 15 - 17 - 16 - 10 - 6 - 9 - 13 - 18 - 19
```

Fig. 4: A sample of initial population, and their fitnesses

Averages of 30 runs			
gens	Avg of Max #s	Avg of Avg #s	Avg of Min #s
1	1105	736.287	252
2	1105	736.287	252
3	1061	701.767	136
4	992	667.847	136
5	956	634.327	136
6	892	603.207	136
7	881	573.607	136
8	851	544.407	136
9	832	516.427	136
10	826	489.107	136
11	817	462.027	136
12	798	435.587	136
13	787	409.767	136
14	776	384.227	136
15	769	359.167	136
16	756	334.527	136
17	711	311.047	136
18	678	288.627	136
19	631	267.667	136
20	621	248.407	136
21	588	229.507	136
22	583	211.927	136
23	495.6	195.087	136
24	482.8	181.396	136
25	434	168.12	136
26	309	157.22	136
27	252	151.22	136
28	159	147.5	136
29	159	147.04	136
30	159	146.58	136
31	159	146.12	136
32	159	145.66	136
33	159	145.2	136
34	159	144.74	136
35	159	144.28	136
36	159	143.82	136
37	159	143.36	136
38	159	142.9	136
39	159	142.44	136
40	159	141.98	136
41	159	141.52	136
42	159	141.06	136
43	159	140.6	136
44	159	140.14	136
45	159	139.68	136
46	159	139.22	136
47	159	138.76	136
48	159	138.3	136
49	159	137.84	136
50	159	137.38	136

Fig. 5: Average values of runs

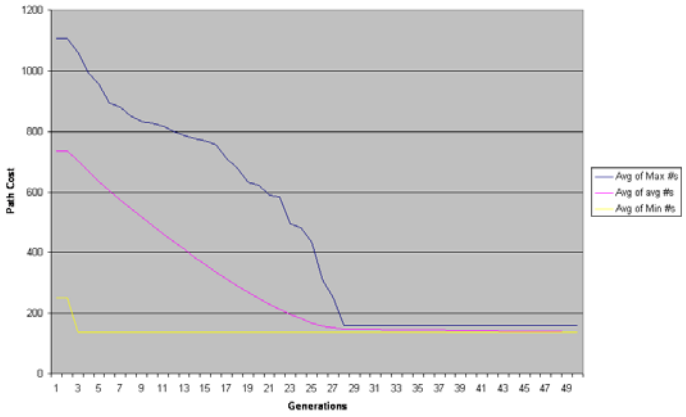


Fig. 6: Average of 30 runs for 50 generations

first, and then after GA gets close to optimum, then I can switch to other search techniques.

7. Conclusions and Future Research

In this project, I developed a genetic algorithm that finds a shortest path in a limited time. This algorithm is meant to be used in OSPF routing, which is the most commonly used intra-domain Internet routing protocol (IRP). As the future research, I would like to test my GA algorithm on some real network topologies containing much more nodes and links.

References

[1] A Genetic Algorithm for the Weight Setting Problem in OSPF Routing, M. Ericsson, M.G.C. Resende, and P.M. Pardalos

[2] T.M. Thomas II. OSPF Network Design Solutions. Cisco Press, 1998

[3] U. Black. IP Routing Protocols, RIP, OSPF, BGP, PNNI & Cisco routing protocols. Prentice Hall, 2000.

[4] A Recursive Random Search Algorithm for Network Parameter Optimization, Tao Ye , Shivkumar Kalyanaraman

[5] Genetic Algorithms for Solving Disjoint Path Problem with Proportional Path-Costs, Burcu Ozcam, North Carolina State University

[6] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, 1989.

[7] http://en.wikipedia.org/wiki/Genetic_algorithm

[8] A Unified Search Framework for Large-scale Blackbox Optimization, Tao Ye, Shivkumar Kalyanaraman, Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute

[9] Faster Genetic Algorithm for Network Paths, Yinzhen Li, Ruichun He, Yaohuang Guo.

[10] Minimizing Packet Loss by Optimizing OSPF Weights Using Online Simulation, Hema Tahilramani Kaur, Tao Ye, Shivkumar Kalyanaraman, Kenneth S. Vastola, Electrical Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY-12180

SESSION

SIMULATED ANNEALING + ANT COLONY / SWARM OPTIMIZATION

Chair(s)

TBA

Adding an ACO Operator to a Genetic Algorithm

David Hibler
Physics Computer Science and Engineering Department,
Christopher Newport University, Newport News, Virginia USA

Abstract *- The purpose of this paper is to discuss the addition of a new operator, called an ACO operator, to a genetic algorithm. The operator is based on an analogy with Ant Colony Optimization. We use the ACO operator in an application of genetic algorithms to engineering design of conduit systems. The conduit optimization problem involves optimizing both the location of components of conduit systems and the routing of conduits between those components. Our Conduit Routing Optimization Tool, COT, uses a genetic algorithm with an ACO operator to solve this problem. The genetic algorithm provides the basic means to search for an optimal solution to the problem. Pheromone trails, a method from Ant Colony Optimization, are used to influence the genetic algorithm. We discuss our methods and the Conduit Optimization Tool. We also discuss when an ACO operator might be useful for other types of problems.*

Keywords: *Genetic Algorithms, Design, Ant Colony Optimization*

1 Introduction

This paper discusses the use of an augmented genetic algorithm for engineering design. The augmentation is based on ideas from Ant Colony Optimization. Our previous work has dealt mostly with optimization of engineering design decompositions [1, 2]. In the present paper we describe a tool for aiding a different part of the design process. The use of pheromones is the major innovation. We have demonstrated the usefulness of this tool for optimization of the location of equipment and the routing of conduits for ships. We expect, however, that this tool will be an aid in other types of engineering design. It also illustrates the use of ideas from Ant Colony Optimization to guide a genetic algorithm.

1.1 The problem

The first stage of design of a complex system such as a process plant or a ship often consists of a basic functional design. This design shows each element of the system and its intended function, but does not show much additional information. In our case, the functional design is the output of a CAD tool. Additional stages of design can be thought of as adding missing information to the functional design.

One such stage involves conduit systems. On a ship, conduits are used to convey a service over some distance. Examples of conduits include wire, pipes, and ducting. These conduits are parts of systems such as the fresh water cooling system, the seawater cooling system, the electrical system, and the fuel oil system. These systems involve other components than just conduits. Some of these components are static. Static components have a location that is fixed by other factors in the design. Many components, however, are non-static. An important phase of the design is the optimization of the location of these components and the routing of conduits between them.

1.2 Solution

We originally attempted to solve the conduit optimization problem using a pure genetic algorithm.

Genetic algorithms are a well-known technique which can be applied to solve optimization problems. They have been applied to engineering design [3,4,5] They are robust, can be applied to situations where the quality of a solution changes discontinuously as parameters change, are good for finding global as opposed to local optimizations, and can provide multiple, qualitatively different results that are close to optimal.

Genetic algorithms map solutions to a problem to representations called chromosomes. Often these chromosomes are bit strings. First, a population of chromosomes is randomly initialized. Then members of the population are selected for reproduction based on a fitness function. Operations of mutation and crossover are applied during reproduction to produce members of a new generation. Thus, the population evolves by a process similar to natural evolution.

We discovered that a pure genetic algorithm functions very poorly for our conduit optimization problem. The fundamental reason seems to be that the search space is just too large and most solutions in this search space are undesirable. For example, conduits can twist and turn repeatedly in three dimensions. Most such twists and turns are undesirable.

In order to deal with this large search space problem, we introduce an idea from Ant Colony Optimization for improving the process.

1.3 Contents of this paper

In the second section we describe The ACO operator. As a simple example, we discuss a map coloring problem. The third section of this paper briefly describes the basic graph representation for conduit problems. More details are given elsewhere [1]. In the fourth section we describe results on conduit system problems using the ACO operator. Finally we evaluate the system and the usefulness of the ACO operator as an aid to genetic algorithms in general.

2 The ACO operator

2.1 ACO steps

Ant colony optimization is a well known metaheuristic [6, 7] for solving problems which can be represented by finding good paths through graphs. It is based on an analogy with the behavior of ants and the use of chemical markers called pheromones to determine best or at least good paths.

For details on this method the reader is referred to [8]. There are many possible variations. A simple model of this method has the following steps:

1. Initialize pheromone values to small random values.
2. For each ant, construct a path from start to destination by choosing the next node based on pheromone strength.
3. Evaporate pheromone.
4. Update pheromone by depositing an amount of pheromone on each ant's path that is proportional to the quality of the path.
5. Return the path with the largest amount of pheromone.

2.2 An example problem

As a simple example of our procedure, we consider a problem not involving design. The problem is to color a map of the United States so that no two states have adjacent colors. This can be solved using four colors. We formulate this for a genetic algorithm by using bit strings to represent possible colorings. We allocate two bits per state to indicate color. The result is a genome with 100 bits. The fitness is a constant minus the number of states that are miscolored.

2.3 Pheromone graph

Unlike a genetic algorithm, the solution to an Ant Colony Optimization problem is a path through a graph. We can call this a pheromone graph since pheromones values will be deposited on vertices of this graph. This set of vertices specifies the solution to the problem. Therefore, our first step in developing the ACO operator is to design a pheromone graph. We will refer to it as a P graph for short.

A genetic algorithm represents possible solutions to a problem, not by a graph, but by a genome which we will consider to be specified by bit strings of some fixed length. If the vector has length n, the genome space, G, has n dimensions and there will be 2^n different points in this space. Each point represents a possible chromosome and therefore each corresponds to a solution to the problem.

In order to construct the P graph, we partition the bit vectors into specific sets of bits. Each set of bits specifies a subspace of G that we will refer to as a section. The bits of any given chromosome specify a point in each section. To construct the P graph we arrange these sections in a specific order and add two ideal points. These represent the start and finish points. The P graph has directed edges from the start point to each of the points in the first section. There are edges from each section to each point in the next section until the final section. Each point in the final section has an edge from that point to the finish point. This is the P graph.

By construction, each chromosome is equivalent to a path through the P graph from the start point to the finish point. This means that the chromosome is analogous to an ant trail, not an ant. The trail consists of one point for each section plus the start and finish points. Another difference between the ACO operator and most Ant Colony Optimization methods is that we will mark nodes with pheromone, not edges.

In the map coloring example, a set of colorings can be represented by a bit string with 100 bits. Possible sections could be specified by states in the far west, southwest, west central, southeast, and northeast in that order. Each of the 5 sections would correspond to 10 states and therefore 20 bits. This means each section would consist of 2^{20} points. The P graph would have a start node which would have edges to each of the points in the far west section; each of these points would have edges to all the points in the southwest section and so forth. All points in the northeast section have edges to the final point.

The P graph would be too large in most cases to specify explicitly; fortunately, this is not necessary. Since the only purpose of the two ideal points in the P graph is to provide specific start and finish points, we consider them as implicit and do not represent them. Furthermore, since the existence of an edge is specified by a rule, the edges can be kept implicit also. Finally, most points are implicit. The only points that need to be explicitly represented are those that are occupied by a chromosome (ant path). These points are already represented by the section bits of the respective chromosomes.

Since most of the P graph is implicit, the representation of the explicit part is easy. All that is needed is an ordered list of sections. Each section can be represented by a mask and a pheromone array. The mask determines which bits on a chromosome specify the point in the section that is on the chromosome's "path" through the section. The array holds pheromone for each chromosome's point in the section. Since we mark nodes with pheromone instead of edges the array's size is just the number of chromosomes.

2.4 Section fitness and pheromone values

The pheromone value is partially based on the section fitness. The section fitness is calculated by the fitness function and represents a guess at the fitness contribution from this section of the chromosome. The expectation is that each section represents partial solutions that are useful building blocks for the total solution. In the map coloring example section fitness is evaluated by counting the number of miscolorings in the states in the section. This is why a contiguous collection of states was chosen for each section, rather than (for example) states that are close alphabetically.

Evaluating section fitness in the general case is usually not difficult for design problems. The evaluation function which evaluates the global fitness can usually evaluate the section fitness easily. If not, then a heuristic evaluation function can be used. In the worst case the section fitness could be made a constant.

It should be pointed out that having good section fitness may be useful for finding a solution, but there is no guarantee. In the map problem, states on either side of adjoining sections must have appropriate colors. There is no guarantee that this can be done, even if the sections themselves are colored appropriately. Similar difficulties can easily occur in other problems. This means that we do not want to use just section fitness for our pheromone value.

For the pheromone value we choose to balance the advantages of local and global improvement. Therefore, the pheromone value for each point of a section that is occupied by a chromosome is taken to be a sum of the weighted total fitness and the section fitness. The total fitness is weighted by a fraction which is roughly the contribution of the section to the total fitness. This fraction is the ratio of the number of section bits divided by the total number of bits. In the map coloring problem the weighting would be 20/100 = 1/5.

Unlike ACO, pheromone values will not slowly evaporate, but will totally vanish from one generation to another. This avoids the difficulty of keeping track of points from previous generations that are not currently occupied by chromosomes.

2.5 The ACO operator

In Ant Colony Optimization step 2 was: **"**For each ant, construct a path from start to destination by choosing the next node based on pheromone strength".

In a GA problem we don't construct the chromosome through sequential choices. This means that we must represent step 2 as a deformation of the existing path toward a nearby path based on pheromone strength. The ACO operator performs these deformations.

There are two parameters associated with the ACO operator. One parameter governs the frequency of deformations. The other gives the strength of deformations. The frequency parameter specifies the average number of deformations to apply per chromosome of the population in a single generation. The strength parameter specifies the fraction of the genes to be modified per deformation.

To prepare for a deformation we find two chromosomes with points that are close in some section, but not the same. In order to do this, a section and a chromosome are chosen randomly. A second chromosome is found by locating the chromosome which occupies the closest point in the same section. (Distance ties are broken arbitrarily). The distance measure we use is the Hamming distance for the bits defining the section. The closest chromosome is therefore the one that has the most section bits the same, but is not identical.

The deformation changes bits in the chromosome with the lower pheromone value to match those in the chromosome with higher value. The bits to change are chosen randomly. The number of bits to change is determined by the deformation strength parameter.

If the chromosomes have equal pheromone values then the deformation is aborted and tried with a different pair of chromosomes. If there are fewer bits to change than would be indicated by the strength parameter then all are changed.

In our genetic algorithm all candidates for the next generation are selected (perhaps redundantly) using tournament selection. The ACO operator, mutation, and crossover are performed on these chromosomes based on probability/strength parameters. The operators are performed in the order given.

Earlier we gave five general ACO steps. Step one, initialize pheromone values, is taken care of by the initialization of the chromosome population and the first fitness evaluation. Step two, constructing a path is replaced by using the ACO operator to deform existing paths. Step three, evaporating the pheromone, is done by not using any pheromone from previous generations. Step four, updating the pheromone is done by the fitness function each generation. Step five, returning the path with the best pheromone value is replaced by choosing the fittest chromosome.

3 Conduit problem representation

3.1 Representing design graphs

Based on experience using genetic algorithms for design decomposition problems, we choose to represent an engineering design as a graph. (From now on we consider the term graph to mean undirected graph.) The basic functional design of the system is the source of this graph. Each vertex of the graph represents a component of the design. Each edge represents a connection. Further information is not known at this stage.

Additional stages of design add more detailed information. This can usually be thought of as attaching labels to vertices or to edges or to both. To optimize some aspect of design, we merely optimize the labels. An important assumption in our method is that the graph does not change. All chromosomes in a population apply to the same graph.

Chromosomes, in this method, are simply collections of labels and can be bit strings as in ordinary genetic algorithms. The difference is that the bit strings must be interpreted with the aid of the underlying graph.

3.2 Conduit System Representation

We need to apply our representation to a conduit system. We said that a conduit system is a system that supplies a service to a collection of other objects. An example is a fresh water cooling system on a ship. This is a collection of pipes, valves, filters, pumps, and other items that deliver fresh water to equipment in different locations, which need the water for cooling.

We distinguish between static and dynamic components of such a system. Static components have their location determined by outside factors. These locations cannot be optimized, at least not by the current tool. Dynamic components can have their position changed for optimization purposes.

The simplest way to represent this type of system is to consider each component other than the conduits as a vertex in our graph. In order to simplify the structure, we add virtual joint components at regular distances along each conduit. These virtual joints represent locations at which the conduit can change direction. There may or may not be an actual joint at this location. We have a set number of these joints so that the design graph is fixed and does not change its structure during optimization.

Using this representation, our optimization problem is purely a vertex labeling problem. The labels represent the locations of components. Edges represent conduits that travel in a straight line between the components they connect. There are actually some additional constraints. Some components have fixed ports. The ports represent attachment points for conduits. Conduits can only leave the port in a straight line in the direction specified by the port. This means that in our graph, the first virtual joint must lie along the straight line specified by the port. More details on this representation are given elsewhere. [1]

4 Results

4.1 Test Problems

The map coloring example was implemented in order to test ideas. Map coloring took less than half as many generations as a simple genetic algorithm with states in alphabetical order.

Based on the success with the simple example, the ACO operator was added to a genetic algorithm for conduit optimization. Some simple routing problems were constructed. Regions through which conduits were to pass were divided into three dimensional cells. These cells correspond to the sections of the P graph. This was much like the sets of states employed for the map problem. The size of the cells varied depending on the size of ship compartments, but usually corresponded to a single ship compartment. The number of bits per section varied based on the design, but in this simple example usually consisted of 100 - 180. The number of sections was varied from 10 to 15

4.2 Test parameters

The genetic algorithm was run in four different modes. In the first mode, called the S (simple) mode, the strength of the ACO operator was zero. This was just a simple genetic algorithm. The second mode was the L (local) mode. In this mode the pheromone value was just based on local (section) fitness. The third mode was the G (global) mode. The pheromone in this case was based on the global fitness with no section fitness. The fourth and final mode was the F (full) mode using both section and overall fitness in equal amounts.

The genetic algorithm used tournament selection. Various parameters were tried until reasonable values were found to be: mutation probability = .001, crossover probability = .7, ACO frequency =.20, ACO strength = 0.15.

If there was no trend of improving costs for 100 generations the algorithm was terminated. Success in finding a solution was defined as finding a solution before termination with a cost within a factor of 10 of what a human might produce. Since hard constraints were enforced by high costs, there was always technically a solution.

The data on success and failure was highly dependent the sample problems so we only indicate rough values.

In tests, the S mode solved the sample problems 17% of the time. The L mode succeeded 63% of the time. The G mode succeeded 47% of the time and the F mode succeeded 71% of the time.

In order to compare quality, we first chose only problems where all 4 methods solved the problem. We rated the quality of the solutions with the S method (basic genetic algorithm) having quality one.

4.3 Test results

The table below shows results from some typical runs.

Table 1

Run	L Quality	G Quality	F Quality
1	1.25	1.33	1.55
2	1.99	1.15	1.72
3	1.56	0.55	1.72
4	2.04	0.95	1.74
5	1.51	1.28	1.82

The average quality of L/S was 1.67. The average quality of G/S was 1.052. The average quality of F/S was 1.71.

Since the L mode and the F mode solved the most problems. We present 5 typical runs where both solved the problems but the other two did not. We show F mode quality relative to L mode (L mode quality normalized to 1).

Table 2

Run	F Quality
1	0.82
2	1.28
3	1.40
4	1.04
5	1.67

The average value of F/L was 1.24.

5 Conclusions

The crucial aspect of this study was to improve the ability to find any solution. The simple (S) genetic algorithm failed very frequently. We attribute this failure to the fact that the space of reasonable solutions was very sparse. Mutation was needed to explore this space; however, mutation would also immediately break down partial solutions. It should be noted in this connection that although the ACO operator changed bits, it changed them in a conservative fashion. That is to say, it makes an inferior partial solution more similar to an existing one that was better.

A simplification to the ACO operator is obtained by using it in the G mode. This means the pheromone values are simply the fitness. In some ways this is more in accord with usual Ant Colony Optimization. Our experience is that the local (section) fitness helps the algorithm. The full method was also better than just using the local fitness for pheromone values. It should be noted that when the local fitnesses of the chromosomes in a section become roughly similar, differences in the actual (total) fitness can become the determining factor. This can push the sections of a chromosome to change to accommodate global optimization.

Obviously the use of the ACO operator makes the algorithm much slower and is also more complicated. Of course, it need not be used in a given situation. In which case, there is no penalty. It can be thought of as an additional weapon to be used on difficult problems.

The next question is: When would the ACO operator be useful? It would be expected to work well on problems which were similar to those tested. There are two key features that are likely to be important for similar problems. The first is that the problem difficulty be high due a very large problem space in which good solutions are sparse. The second is that the problem should be decomposable into subproblems that aid in finding a global solution even if the global solution cannot be obtained by simply solving the subproblems.

6 References

[1] Hibler, D. L., "A Hybrid Genetic Algorithm Application", Proceedings of the International Conference on Artificial Intelligence, June 2004, vol. II, pp 596 – 603, (published by CSREA press).

[2] Hibler, D. L. (2010). In Hamid R. Arabnia, Ray R. Hashemi, Ashu M. G. Solo (Ed.), An Active Region Approach to Genetic Algorithms (pp. 23-28). WorldComp'10 - GEM 2010 track / CSREA Press. www.world-academy-of-science.org/worldcomp10

[3] D.E. Grierson, and P. Hajela, eds., Emergent Computing Methods in Engineering Design: Applications of Genetic Algorithms and Neural Networks, New York; Springer-Verlag, 1996.

[4] K. Rasheed, H. Hirsh, and A. Gelsey, "A Genetic Algorithm for Continuous Design Space Search", Artificial Intelligence in Engineering, 11(3) 1997.

[5] L.J. Murphy, A.R. Simpson, and G.C. Dandy, "Design of a pipe network using genetic algorithms". Water, August, 1996, pp 40-42.

[6] Dorigo M. Optimization, Learning and Natural Algorithms. PhD thesis [Book]. - Politecnico di Milano : [s.n.], 1992.

[7] J. Zhang, H. Chung, W. L. Lo, and T. Huang, "Extended Ant Colony Optimization Algorithm for Power Electronic Circuit Design", IEEE Transactions on Power Electronic. Vol.24,No.1, pp.147-162, Jan 2009.

[8] A.P. Engelbrecht, Computational Intelligence 2nd ed., New Jersey, John Wiley and Sons, 2007.

Hybrid Constraint-Handling Mechanism for Particle Swarm Optimization with Applications in Power Systems

Caisheng Wang[1,2], M. Hashem Nehrir[3], Le Yi Wang[1], Feng Lin[1] and Chris M. Colson[3]
[1] Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI, USA
[2] Division of Engineering Technology, Wayne State University, Detroit, MI, USA
[3] Department of Electrical and Computer Engineering, Montana State University, Bozeman, MT, USA

***Abstract*—This paper presents a hybrid mechanism of handling constraints for particle swarm optimization (PSO) algorithms. A Newton-like method is embedded into the algorithm to help handle equality constraints while penalty/fitness functions are used to deal with inequality constraints. The proposed technique has been tested for benchmark optimization problems reported in the literature. The experimental results show that the proposed hybrid method successfully and efficiently handles the equality and inequality constraints for PSO algorithms. An economic power dispatch problem is given as an example of the applications of the proposed method in power systems.**

***Key Words*— Constraints handling, economic power dispatch, hybrid method, Newton-like method, optimization, PSO.**

I. Introduction

INSPIRED by the social swarming behavior such as bird flocking or fish schooling, Kennedy and Eberhart first introduced the particle swarm optimization (PSO) algorithm to solve nonlinear optimization problems [1]. PSO is one of many heuristic algorithms; it shares many similarities with evolutionary computation techniques such as genetic algorithms (GA). However, in some cases it is easier and faster to implement PSO than GA due to inherent difficulties in data representation that evolutionary operators such as crossover and mutation utility. Since its introduction in 1995, PSO has been used in many areas including nonlinear optimization, control, and artificial intelligence [2].

PSO was originally developed for solving unconstrained nonlinear optimization problems [1]. Since real optimization problems are normally constrained, several variants of PSO capable of handling constraints have been proposed [2]-[12]. Nevertheless, how to handle different types of constraints that could exist in real world problems is still one of the most difficult challenges for using PSO. Contrary to its fast spread of applications in different areas, research on finding a general method of handling constraints for PSO has moved relatively slowly.

This work was supported by National Science Foundation Grant ECS-0823865.

The most common way of handling constraints for PSO is to convert the constrained problem into a unconstrained problem by assigning certain penalty factors into the original objective function [10]. It is very important to choose appropriate penalty functions to maintain the diversity of the population in the evolutionary algorithms like PSO [3], [10]. This method has been proven efficient in handling inequality constraints [3], [5], [10], [13]. However, for the situations where the search spaces are highly constrained, the optimums might be achieved with active constraints, or the equality constraints must be satisfied at high accuracy, the method of penalty functions may fail to find the optimal solutions even with dynamic penalty factors [3], [13].

Ranking schemes were proposed in [3] and [11] to handle constraints so that leaders with better performance were chosen to set the directions for the rest in the swarm. A Pareto ranking method was used in [11] while a compete-and-win scheme based on closeness to the feasible region was proposed in [3]. However, ranking schemes may reduce the exploratory capabilities of PSO. To help prevent this, a turbulence factor was also introduced in [3]. Nevertheless, as noted in [3], the ranking schemes may not work properly for some ill-defined constraints.

Xu and Eberhart presented a feasibility-checking method to handle constraints for PSO [4], [5]. The proposed scheme randomly initializes particle positions until all particles are in the feasible region. During the updating process, only feasible solutions are saved and used for updating. Though this approach is simple and straightforward, it has difficulties with a prohibitive computational burden when solving some real problems. For problems with wide search spaces and stringent constraints, the algorithm can stall in its initialization stage or at an intermediate stage of computation when attempting to randomly place all particles in the feasible region [3].

Recently, Luo et al. employed a method of reduced space transformation to transform the problem into one without equality constraints [12]. Liang et al. proposed a dynamic multi-swarm approach to handle constraints in each subpopulation [9]. These methods have shown to be effective for certain application areas/problems. However, they still have difficulties for some large scale nonlinear optimization problems [9], [12]

In this paper, a hybrid mechanism combining penalty/fitness functions and a Newton-like method is proposed to handle constraints for PSO. The proposed mechanism offers the freedom for particles to search spaces and also gives them directions to update their positions. This new method is verified by solving benchmark problems reported in previous work [3], [13].

The paper is organized as follows. The background of PSO algorithm and Newton's method are briefly discussed in section II. The constraint-handling mechanism using a Newton-like method and penalty functions is addressed in section III. Section IV shows and discusses the results of applying the proposed method for solving benchmark problems reported in literature. Section V concludes the paper.

II. PSO Algorithm and Newton's Method

A. Optimization Problems and the Original PSO Algorithm

There are numerous types of optimization problems such as minimizing cost or maximizing benefit with various types of constraints in the real world. In general, a constrained optimization problem can be defined as:

$$\textbf{Min}\ f(X),\ X=[x_1,\quad x_2,\quad \cdots\quad x_n]^T, \text{ and } X\in R^n \tag{1}$$

Subject to:

$$G(X)\le 0,\ G=[g_1,\quad g_2,\quad \cdots\quad g_p]^T, \tag{2}$$

$$H(X)=0,\ H=[h_1,\quad h_2,\quad \cdots\quad h_q]^T, \text{ and} \tag{3}$$

$$x_i^{(l)}\le x_i\le x_i^{(u)},\ i=1,2,\cdots,n \tag{4}$$

where f is the objective function, X is the optimization variable vector, n is the variable vector dimension, p is the number of inequality constraints, q is the number of equality constraint equations, and $x_i^{(l)}$ and $x_i^{(u)}$ are the lower and upper bounds of the variable x_i.

PSO is a population based stochastic optimization technique, which has been widely explored to find the solution to (1). Each particle in the population represents a candidate solution to the problem. All the particles start with randomly initialized positions and then "fly" throughout the search space to find the best possible solution to the problem. During the process, the particles communicate with each other and promulgate the best local solutions/positions that each of them has achieved. Then, based on the global and local information obtained, each particle updates its position towards a desired global optimum.

The basic elements of a global version of PSO algorithm are summarized below.

- *Particle, $X_j(t)$:* Each particle is a candidate solution vector containing n optimization variables. $X_j(t)$ is the jth particle at time t, and it can be described as:
$$X_j(t)=[x_{j,1}(t),\quad x_{j,2}(t),\quad \cdots\quad x_{j,n}(t)]^T \tag{5}$$
This particle vector is said to describe the particle's "position" within the solution space.
- *Population, Popu(t):* It is a set consisting of m particles at time t, i.e., $Popu(t)=[X_1(t),\quad X_2(t),\quad \cdots\quad X_m(t)]^T$.
- *Particle Velocity, $V_j(t)$*: The velocity of the jth particle at the time t in the n-dimensional search space can be represented as:
$$V_j(t)=[v_{j,1}(t),\quad v_{j,2}(t),\quad \cdots\quad v_{j,n}(t)]^T \tag{6}$$
The velocity of the particle indicates its relative change within the solution space with respect to its current position vector. For each time increment, a particle's velocity demonstrates the time rate of change to the particle's solution vector.
- *Individual best, $X_{j,\,pbest}(t)$*: It is the best position that the jth particle has achieved so far at time t. Each particle saves its best position throughout the whole searching procedure.
- *Global best, $X_{gbest}(t)$:* It is the best position (or solution) that has been achieved so far among all the particles. Based on the definition, it is clear that
$$X_{gbest}(t)\le X_{j,pbest}(t),\ j=1,2,\cdots,m \tag{7}$$
The information of global best is known to each and every particle in the population through communication among particles.

The original PSO algorithm is implemented as follows:

- *Initialization:* At the starting point t = 0, all particles are initialized with a randomly assigned position and velocity value. For example, the ith dimensional position ($x_{j,i}$) of the jth particle X_j is initialized with a uniform random value between $x_i^{(l)}$ and $x_i^{(u)}$, i = 1, 2, …, n. The ith dimensional velocity of the particle is initialized with a uniform random value between $-v_i^{\max}$ and $v_i^{\max}$. $v_i^{\max}$ can be defined as:
$$v_i^{\max}=\left(x_i^{(u)}-x_i^{(l)}\right)/N_i^m \tag{8}$$
where N_i^m is the minimum number of steps that change a particle position from its lower bound to its upper bound of the ith dimension. The value of N_i^m is chosen by users.
- *Velocity updating:* During each iteration cycle, the particle velocity is updated based on the following formula:
$$\begin{aligned}V_j(t)=&\,w(t)V_j(t-1)+c_1r_1[X_{j,pbest}(t-1)-X_j(t-1)]\\&+c_2r_2[X_{gbest}(t-1)-X_j(t-1)]\end{aligned} \tag{9}$$
where $w(t)$ is the inertia weighting factor, c_1 and c_2 are two positive constants, and r_1 and r_2 are uniform random numbers in [0, 1]. The first term in the equation stands for the particle inertia property, which tends to keep the velocity of a particle constant. The second term shows how a particle changes its speed based on its own knowledge (individual best position). The third term represents the social information exchange behavior of particles that each particle also adjusts its speed based on the global best position.
- *Position updates:* After updating its velocity, each particle changes its position (or solution) according to the following simple formula:
$$X_j(t)=X_j(t-1)+V_j(t) \tag{10}$$
- *Process termination*

A PSO program will stop only when a certain stopping criterion is met. For real applications, a PSO program can have a set of stopping criteria set by users. Some typical criteria used

are: (1) the number of iterations since the last update of the best solution exceeds a pre-set value, N_L; and (2) the total number of iterations reaches a pre-specified maximum value, N_{tot}.

B. Newton's Method for Solving Nonlinear Equations

Newton's method, also called Newton-Raphson's method, is one of the most commonly used techniques for solving nonlinear equations [14]. By linearizing the nonlinear function to its first order Taylor series, Newton's algorithm approximates the solution by solving a linear equation at each step. The method starts with an initial estimate of the solution and then calculates inductively as

$$x(t+1) = x(t) - \frac{f(x(t))}{f'(x(t))} \tag{11}$$

where f is the nonlinear function to solve, $x(t)$ is the estimate to the solution at the current (tth) step, and $x(t+1)$ is the next [$(t+1)$th] step estimate.

Newton's method for solving a single nonlinear function can be easily extended to solve a set of nonlinear equations as follows:

$$X(t+1) = X(t) - J^{-1}(t)F(X(t)) \tag{12}$$

where $X(t)$ is the N-dimensional variable vector at the tth step, F is the N-dimensional function vector, and J is the Jacobin matrix. Variable and function vectors are defined as:

$$X(t) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_t \text{ and } F(X(t)) = \begin{bmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_N(X) \end{bmatrix}_{X=X(t)} \tag{13}$$

Jacobin matrix is obtained as:

$$J(t) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{bmatrix}_{X=X_t} \tag{14}$$

For a set of underdetermined nonlinear equations (i.e., the number of variables is larger than the number of consistent equations), a Newton-like method can be used to find a solution with minimum norm by using pseudoinverse [14]. A similar iterative formula to (12) can be used as follows:

$$X(t+1) = X(t) - pinv(J(t))F(X(t)) \tag{15}$$

where *pinv* represents pseudoinverse operation. In this case, $J(t)$ is an $M \times N$ matrix, i.e. not a square matrix, i.e. $M \neq N$. M is the number of equations, N is the number of variables, and $M<N$.

Newton's method is fast and it can achieve quadratic convergence. It is a common application to combine Newton's method with other methods in a hybrid format to achieve numerical global convergence [14].

III. Hybrid Mechanism For Handling Constraints for PSO Algorithm

When attempting to solve a general optimization problem, such as that described in (1) – (4), a challenge for using PSO becomes how to effectively handle the constraints listed in (2) – (4). It is especially true for equality constraints and active equality constraints. Inspired by Newton's method discussed in section II.B, a hybrid mechanism of handling constraints is introduced in this section. Newton-like method is used to handle equality constraints while penalty factors incorporated into a fitness function is applied to handle inequality constraints. The upper and lower limits in (4) are normally easy to handle by forcing possible violations within the limits as follows:

$$x_i = \begin{cases} x_i = x_i^{(l)} \; if \;\; x_i < x_i^{(l)} \\ x_i = x_i^{(u)} \; if \;\; x_i > x_i^{(u)} \end{cases} \tag{16}$$

The focus of the remainder of this section is devoted to handling equality constraints with Newton-like method and inequality constraints using penalty fitness functions.

A. Handling Equality Constraints Using Newton-Like Method

In general, the group of equality constraints in (3) is a set of underdetermined nonlinear equations, which should have a set of solutions. Otherwise, there is either no solution to the optimization problem or there is no need to carry out the optimization study if only the equality constraint set itself can determine the solution of the problem.

PSO algorithm starts with a randomly initialized point, which may or may not satisfy all the constraints. Each particle in the population will then "fly" in the search space, update its speed and position based on the experience of its own and the whole community. In the original PSO algorithm summarized in (9) and (10), there is no direct mechanism to tell particles how to adjust themselves to meet the equality constraints. This could cause the procedure fail to find a feasible solution, take a longer time to find a solution, or come with a larger error to the equality constraints if the value of the error is still acceptable.

The idea of using Newton-like method to solve a set of underdetermined nonlinear equations can be applied to guide particles in a population to change their velocities and positions to meet the equality constraints. The new PSO incorporated with Newton-like method is implemented as follows:

……

Conventional PSO Velocity Update using (9)

Conventional PSO Position Update using (10)

$$dX_j(t) = -pinv\left(J_j(t)\right)H\left(X_j(t)\right) \tag{17}$$

$$X_j(t+1) = X_j(t) + dX_j(t) \tag{18}$$

……

In (17), $dX_j(t)$ is the incremental change of particle X_j at time t, H is the equality constraint vector given in (3), and J_j is the Jacobin matrix of particle X_j, defined as:

$$J_j(t) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial h_q}{\partial x_1} & \frac{\partial h_q}{\partial x_2} & \cdots & \frac{\partial h_q}{\partial x_n} \end{bmatrix}_{X=X_j(t)} \tag{19}$$

B. Handling Constraints with Penalty Functions

For PSO algorithms, it is normally easier to handle inequality constraints than equality ones [3]. Incorporation with penalty functions has been proven to be an effective way to deal with inequality constraints in (2). In general, a specific problem may need a specialized penalty function; nevertheless, the set of equations given below can be used as candidate penalty functions for generic purposes.

$$J_{ineq}(x) = \begin{cases} 0 & if\ x \le 0 \\ \alpha_1 \left[e^{\alpha_2 \sinh(\alpha_3 x)} - 1 \right] & otherwise \end{cases} \quad (20)$$

where α_1, α_2 and α_3 are all positive parameters.

For the inequality constraints given in (2), the overall penalty function can be:

$$J_{ineq} = \sum_{k=1}^{p} J_{ineq}(g_k) \quad (21)$$

The larger the error (away from the constraints), the greater will be the penalty function values. By changing the values of α_1, α_2 and α_3, a smoother or steeper penalty function curve can be obtained.

In addition to the Newton-like method discussed in the previous sub-section, the penalty function method can also be incorporated for handling equality constraints, though it is not always effective. As a secondary method, the penalty function can help meet the equality constraints. The following set of equations can be used as penalty functions for handling equality constraints.

$$J_{eq}(x) = \beta_1 \left[\cosh(\beta_2 x) - 1 \right] \quad (22)$$

where β_1 and β_2 are positive valued parameters.

The overall penalty function all the equality constraints will be:

$$J_{eq} = \sum_{k=1}^{q} J_{eq}(h_k) \quad (23)$$

The penalty function is zero when the equality constraint is satisfied. Similarly, it shows that the larger the error away (both positively and negatively) from the constraints, the greater will be the penalty function values. By changing the values of β_1 and β_2, a smoother or steeper penalty function curve can be obtained as desired.

C. Hybrid Mechanism of Handling Constraints

A hybrid mechanism combining the two methods discussed above (Newton-like method and penalty function method) is proposed in this section to handle constraints for PSO algorithms. The Newton-like method gives directions to particles toward equality constraint surface in each iteration loop. Penalty function method has shown its effectiveness in handling inequality constraints. The penalty function method is also a commonly used alternative for handling equality constraints, especially when the derivatives of the equality constraints can not be calculated or estimated where the proposed Newton-like method will have difficulties.

Fig. 1 shows the overall flow chart of the PSO algorithm with the proposed hybrid mechanism for handling constraints. Some prior knowledge can be utilized to give a better initial particle positions and velocities [2], [15]. This also gives a better position for the Newton-like method to converge to the equality constraint surfaces. Following this, a global objective/fitness function is formulated/augmented by combing the penalty functions for the inequality constraints and/or equality constraints. In addition to the same position and velocity update procedure as a typical PSO algorithm, given in (9) and (10), the position is also updated using the Newton-like method defined by (17) and (18). The stop criteria are checked in each iteration cycle; and if satisfied, the program will stop.

IV. Experimental Results

The proposed hybrid mechanism for handling constraints has been tested on the benchmark problems reported in the literature [3], [13]. Three benchmark problems with equality constraints and/or inequality constraints are chosen for performance verification purposes. Table I shows the original sources and numbers of the testing problems used in the paper, the details of which are given in Appendix A. A simple application example of the algorithm in economic power dispatch is also given to show its potential in power system optimization.

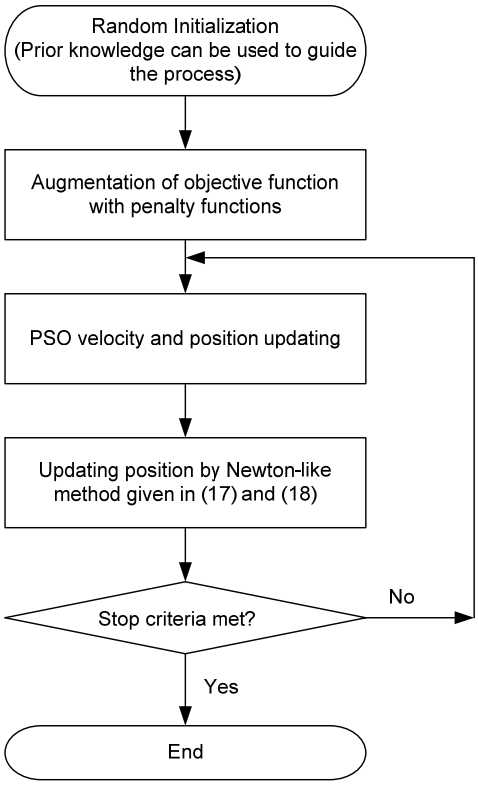


Fig. 1. Flow chart of a PSO algorithm with the proposed hybrid mechanism for handling constraints.

Table I. Testing Problems [3], [13]

Problem #	*g01*	*g02*	*g03*
Problem # in the original sources	*g05*	*g11*	*g13*

A. Experimental Setup

The experiments were carried on a PC with 2.8 GHz CPU and 1.0 GB memory, running on Microsoft Windows XP

Service Pack (SP) 2. All the testing programs were coded using MATLAB 7.0.4.365 (R14) SP 2.

The parameter settings of the PSO programs are listed in Table II. The fixed value parameters are the ones whose values are fixed in the tests in this paper. These can, of course, be changed as needed for other applications. The inertia weighting factor, $w(t)$ in (9), is chosen as:

$$w(t) = \alpha_w w(t\text{-}1) \tag{24}$$

where α_w is the annealing factor.

B. Experimental Results

For each of the ten benchmark problems, the corresponding PSO program was run for thirty times. The experimental results are summarized in Tables III and IV, which include the best, mean, median and worst results of the thirty runs. The standard deviation (STD), feasible rate (FR) and success rate (SR) are also given in the tables. Feasible rate is defined as the number of feasible runs (where a feasible solution is found) over the total number of runs. Similarly, success rate is defined as the ratio of the number of successful runs over the total number of runs [13]. A successful run is a feasible run that also satisfies the following condition:

$$\left|\frac{f(X) - f(X^*)}{f(X^*)}\right| < 0.1\% \tag{25}$$

where $f(X)$ is the objective function value obtained and $f(X^*)$ is the best known value reported in [13].

Table II Parameter Settings for the PSO Algorithms

	Parameter	Memo
Fixed-Value Parameters	$\alpha_w = 0.98$	See (24)
	$c_1 = 2$	See (9)
	$c_2 = 2$	See (9)
	$m = 100$	Total number of particles
	$N_{tot} = 40$	Maximum number of total iterations
	$N_L = 1000$	Maximum number of iterations since the last update of the best solution
Varying-Value Parameters	α_1	See (20)
	α_2	
	α_3	
	β_1	See (22)
	β_2	

The experimental results given in Table III and IV show that the PSO algorithm with the proposed technique for handling constraints found the global optimal solutions with 100% feasible rate and very high success rate for all the test problems.

Table III. Experimental results of problems *g01* to *g03*

Problem	*g01*	*g02*	*g03*
Best	5126.49811	0.75000000000	0.053949847770
Worst	5126.49823	0.75000000000	0.053960390302
Mean	5126.49812	0.75000000000	0.053953099787
Median	5126.49811	0.75000000000	0.053952004747
STD	2.50917E-5	1.21391464E-015	3.0779265536E-6
FR	100%	100%	100%
SR	100%	100%	100%

Table IV. Optimal solutions found for the benchmark problems in the thirty runs.

Problem	Optimal solution, X^*
g01	[679.9453276, 1026.067124, 0.118876359, -0.396233556]
g02	[0.707106785603411, 0.500000006246389]
g03	[-1.71714357, 1.59570969, 1.827245754, -0.763643082, 0.763643075]

C. Application Example in Economic Power Dispatch

A simple example of using the proposed PSO algorithm in economic power dispatch is given to show its effectiveness and potentials in power system optimization. This optimal power dispatch problem is taken from Example 7.7 in [16]. The problem is stated as:

Problem g04

The fuel cost (in $/h) of three thermal power plants in a power system are known as:

$$\begin{aligned} C_1 &= 200 + 7.0P_1 + 0.008P_1^2 \\ C_2 &= 180 + 6.3P_2 + 0.009P_2^2 \\ C_3 &= 140 + 6.8P_3 + 0.007P_3^2 \end{aligned} \tag{26}$$

where C_i ($i = 1, 2, 3$) are the fuel costs, and P_i ($i = 1, 2, 3$) are the plant power outputs in MW. The generation limits for the three plants are:

$$\begin{aligned} &10 \le P_1 \le 85 \\ &10 \le P_2 \le 80 \ \text{(MW)} \\ &10 \le P_3 \le 70 \end{aligned} \tag{27}$$

The total system power loss can be estimated by the following equation:

$$P_{LOSS} = 0.000218P_1^2 + 0.000228P_2^2 + 0.000179P_3^2 \tag{28}$$

The problem is to determine the economic power dispatch when the total load demand is 150 MW. This problem can be re-stated mathematically as:

$$\textbf{\textit{Min}} \ f(X) = \sum_{i=1}^{3} C_i \tag{29}$$

$$s.t. \ dP = \sum_{i=1}^{3} P_i - (150 + P_{Loss}) = 0, \text{ and} \tag{30}$$

$$P_{i,\min} \le P_i \le P_{i,\max}, \quad i = 1, 2, 3$$

where $X = [P_1, P_2, P_3]$, and the generation limits of $P_{i,min}$ and $P_{i,max}$ are given in (27).

Parameters β_1 and β_2 are set to 10000 and 4, respectively, for this problem. Other fixed-value parameters are the same as listed in Table II. The program has also been run for thirty times, and the optimal solution has been achieved for each run. The optimal solution obtained is X^* = [35.09067643433635, 64.13175182066293, 52.47667471130085] with $f(X^*)$ = 1592.649548173904, which is exactly the same as the analytical result given in [16].

If only the penalty function method is used for the PSO, the results are not as close as they should be. Table V shows the comparison between the proposed hybrid method and the penalty-function-only method.

Another type of penalty function (like the quadratic function used in [17] and [18]) was also tested and compared in table V. The exact expressions used for penalty functions are also given

in the table. A feasible solution is considered when the generation limits in (27) are satisfied and $|dP| < 1\times10^{-4}$, see (30). A comparison of results shows the superior performance of the proposed hybrid mechanism, handling the constraints successfully and efficiently.

Table V. Comparison results of different PSO algorithms

Method	FR	SR
Proposed hybrid mechanism	100%	100%
Penalty function only $10000[\cosh(4\times dP)-1]$	30%	26.67%
Quadratic penalty function only ($100000\times dP^2$)	30%	13.33%

V. Conclusion

In this paper, a hybrid mechanism consisting of a Newton-like method and fitness/penalty function method was presented for handling constraints for PSO algorithms. The proposed technique has been verified by solving ten benchmark optimization problems reported in the literature. The experimental results show the effectiveness of the proposed method in handling constraints, especially the equality constraints that have been considered tougher to deal with than the inequality ones. An economic power dispatch problem was also given and solved by the proposed hybrid method to show its great application potential in power system optimization.

Appendix A Testing Problems

Problem g01 [3], [13]

$$\textbf{Min}\ f(X) = 3x_1 + 1.0\times10^{-6}x_1^3 + 2x_2 + (2.0\times10^{-6}/3)x_2^3 \quad (31)$$

s.t.

$$g_1(X) = -x_4 + x_3 - 0.55 \le 0$$
$$g_2(X) = -x_3 + x_4 - 0.55 \le 0$$
$$h_1(X) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_2(X) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_3(X) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0 \quad (32)$$

where $0 \le x_1, x_2 \le 1200$, and $-0.55 \le x_3, x_4 \le 0.55$. The best known solution is X^* = (679.945148297028709; 1026.06697600004691; 0.118876369094410433; -0.39623348521517826) with $f(X^*)$ = 5126:4967140071.

Problem g02 [13]

$$\textbf{Min}\ f(X) = x_1^2 + (x_2 - 1)^2 \quad (33)$$

$$s.t.\ h_1(X) = x_2 - x_1^2 = 0 \quad (34)$$

where $-1 \le x_1, x_2 \le 1$. The optimum solution is X^* = ($\pm\sqrt{2}/2$, 0.5) with $f(X^*)$ = 0.75.

Problem g03 [3], [13]

$$\textbf{Min}\ f(X) = e^{x_1x_2x_3x_4x_5} \quad (35)$$

s.t.

$$h_1(X) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$h_2(X) = x_2x_3 - 5x_4x_5 = 0 \quad (36)$$
$$h_3(X) = x_1^3 + x_2^3 + 1 = 0$$

where $-2.3 \le x_1, x_2 \le 2.3$ and $-3.2 \le x_3, x_4, x_5 \le 3.2$. The optimum solution is X^* = (-1.71714224003; 1.59572124049468; 1.8272502406271; -0.763659881912867; -0.76365986736498) with $f(X^*)$ = 0.053941514041898.

References

[1] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," *Proc*., The 1995 IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, 1995.

[2] X. Hu, R. Eberhart, and Y. Shi, "Recent advances in particle swarm," *Proceedings of IEEE Congress on Evolutionary Computation 2004*, pp. 90-97, Portland, Oregon, USA

[3] Gregorio Toscano Pulido and Carlos A. Coello, "A Constraint-Handling Mechanism for Particle Swarm Optimization," *Proceedings of the 2004 Congress on Evolutionary Computation*, pp. 1396-1403, 2004.

[4] X. Hu and R. C. Eberhart "Solving constrained nonlinear optimization problems with particle swarm optimization", *Proc*., The Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002), vol. 5, 2002, Orlando, USA.

[5] Xiaohui Hu, R.C. Eberhart and Yuhui Shi, "Engineering optimization with particle swarm," *Proc*., The 2003 IEEE Swarm Intelligence Symposium (SIS'03), pp. 53-57, 2003.

[6] Xiao-Feng Xie, Wen-Jun Zhang, De-Chun Bi, "Handling equality constraints by adaptive relaxing rule for swarm algorithms", *Proc*., The 2004 IEEE Congress on Evolutionary Computation, pp. 2012-2016, 2004.

[7] C.K. Monson, K.D. Seppi, "Linear equality constraints and homomorphous mappings in PSO," *Proc*., The 2005 IEEE Congress on Evolutionary Computation, pp. 73-80, vol. 1, 2005, Edinburgh, Scotland.

[8] Mihaela Breaban, Madalina Ionita and Cornelius Croitoru, "A new PSO approach to constraint satisfaction," *Proc*., The 2007 IEEE Congress on Evolutionary Computation, pp. 1948-1954, vol. 1, 2007.

[9] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," *Proc.*, The 2006 IEEE Congress on Evolutionary Computation, pp. 424-445, July, 2006, Vancouver, BC, Canada.

[10] K. E. Parsopoulos and M. N. Vrahatis, "Particle swam optimization method for constrained optimization problems," *Proceedings of the Euro-International Symposium on Computational Intelligence 2002*.

[11] T. Ray and K. M. Liew, "A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem," *Proceedings of IEEE Congress on Evolutionary Computation* (CEC 2001), pp. 75-80, 2001, Seoul, Korea.

[12] Y. Luo, X. Yuan and Y. Liu, "An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints," *Computers and Chemical Engineering*, vol. 31, pp.153–162, 2007.

[13] J. J. Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, P. N. Suganthan, Carlos A. Coello and K. Deb, "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization," *Technical Report*, Nanyang Technological University, Singapore, Dec 2005.

[14] D. R. Kincaid and E. W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, 3rd Edition. Brooks/Cole, Pacific Grove, CA, 2002.

[15] S. L. Ho, Shiyou Yang, and Guangzheng Ni, "Incorporating A Priori Preferences in a Vector PSO Algorithm to Find Arbitrary Fractions of the Pareto Front of Multiobjective Design Problems," *IEEE Transactions on Magnetics*, vol. 44, no. 6, pp. 1038 – 1041, June 2008.

[16] Hadi Saadat, *Power System Analysis*, 2nd Edition, McGraw Hill, 2002.

[17] M.A. Abido, "Optimal power flow using particle swarm optimization," *International Journal of Electrical Power and Energy Systems*, v 24, n 7, October, 2002, p 563-571.

[18] Leandro dos Santos Coelho and Chu-Sheng Lee, "Solving economic load dispatch problems in power systems using chaotic and Gaussian particle swarm optimization approaches," *Electrical Power and Energy Systems*, vol. 30, pp. 297–307, 2008.

Evolutionary Local Search Algorithm for Portfolio Selection Problem: Spin Glass Based Approach

Majid Vafaei Jahan[1], **Mohammad-R. Akbarzadeh-T.***[2]
[1] Department of Software Engineering, Mashhad Branch - Islamic Azad University, Mashhad, Iran
[2] Departments of Electrical Engineering and Computer Engineering, Ferdowsi University of Mashhad, Iran

Abstract

Nowadays, various imitations of natural processes are used to solve challenging optimization problems faster and more accurately. Spin glass based optimization, specifically, has shown strong local search capability and parallel processing. However, generally, spin glasses have a low rate of convergence, since they use Monte Carlo simulation techniques such as simulated annealing (SA). Here, we investigate a new hybrid local search method based on spin glass for using adaptive distributed system capability, extremal optimization (EO) for using evolutionary locally search algorithm and SA for escaping from local optimum states. As shown in this paper, this strategy can lead to faster rate of convergence and improved performance than conventional SA and EO algorithm. The resulting are then used to solve the portfolio selection problem that is a non-deterministic polynomial complete (NPC) problem. This is confirmed by test results of five of the world's major stock markets, reliability test and phase transition diagram.

Keywords: *Spin glass model, portfolio selection, simulated annealing, extremal optimization, phase transition*.

1. Introduction

Similar to artificial neural networks, genetic algorithms, and ant colony systems, spin glass is a paradigm that is inspired from the governing laws of nature. However, as compared to many of its older counterparts, the main distinguishing feature of spin glasses is their unique distributed parameter optimization by emphasizing strong parameter interaction. More specifically, spin glass model is a system of spins interacting with each other due to the existence of magnetic property among them. These spins change their quantity frequently to reach a lower energy level. When the system is at its minimum energy (or minimum temperature) state, there is no longer a visible change in spins' states and the system is said to have reached its ground state [1]. In contrast to most other optimization algorithms such as GA (Genetic Algorithm) where each chromosome represents a complete solution, every spin is only a *part* of an entire solution. The complete solution is found by the interaction of the many spins in the glass. The spin glass paradigm is therefore, a promising paradigm of adaptive distributed systems. In addition, the spin glass model enjoys lots of properties, including limited interaction of each spin with neighboring spins [2], non-exponential growth of optimized (ground) states with the increase in spin glasses' number of bonds [3], the effectiveness of environmental factors such as temperature on system behavior, and a continuing movement towards optimized states at different temperatures [4].

Considering these capabilities, many optimization problems can be solved using such distributed facility [2]. However, like many other heuristic methods, the rate of convergence of finding ground states is low when the problem dimension grows [5]. More specifically, this is reported to be a challenging aspect of the more conventional approaches such as the simulated annealing (SA) as reported earlier in [5, 6].

To speed up the spin glass's rate of convergence, it would be desirable to choose the "right" spin that promises the most improvements in terms of convergence rate and accuracy. This would be in contrast to the standard approach where spins are chosen arbitrarily. In this paper, we address this problem by combining SA with local search strategies, specifically, Extremal Optimization (EO) [8]. In EO, on the other hand, the spin with lowest energy is chosen to change its state with a higher probability. This scheme works since changing each spin influences its other neighboring spins, and so they also change. If the total changes lead to a reduction in glass energy, the overall state of the glass improves and the correlation between spins increases. Hence, any change in each spin's state would lead to rearrange major parts of the glass. In 2001, Boettcher and Percus likened this property to an avalanche that can lead to a faster survey of different spin glasses' states and an increased rate of convergence [8]. There is no need to tune control parameters with precise values; this is the great advantage of EO [9]; but this advantage is equal to deficiency: that is a trap in local optimum. Therefore, EO is fast but non-accurate. Instead, SA is slow but accurate.

Here, we investigate a new hybrid local search method based on *spin glass* for using adaptive distributed system capability, *EO* for using evolutionary locally search algorithm and *SA* for escaping from local optimum states. This algorithm that is named (EO-SA) needs a tune parameter such as temperature (from SA), spin selection method such as aside from ranking (from EO) and locally interaction such as neighborhood spin interaction (from spin glass).

Section 2 reviews the various applications of spin glasses in solving optimization problems. Section 3 provides a mathematical description of spin glasses. The portfolio selection problem is discussed in Section 4. This section also explains how this problem can be mapped onto a spin glass. The algorithm, EO-SA, is then presented in Section 5. In Section 6, the experimental results from applying the above algorithms to five of the world's

reputable stock markets are provided. The reliability test of algorithms and their performance validity is studied in Section 7. In Section 8, the resultant Pareto frontier is compared against the benchmark Pareto frontier; and finally in Section 9, phase transition analysis of the algorithms EO-SA and SA are presented.

2. Literature Review

There is a wealth of existing literature on spin glasses in various domains in general, and physics, in particular. For the sake of brevity as well as the better focus, we are concerned here with that research related to engineering, and especially optimization, in which literature is relatively scarce. Minimum cost flow and matching problem are two examples of this kind [10]. In minimum cost flow problem, the ground state configuration of an Ising Spin glass in a random environment, in which all energies are non-negative, can be obtained with Dijkstra's algorithm to find the shortest path in the directed network with non-negative cost on the edges. In the Matching Problem, the ground state of a two dimensional spin glass model on a square lattice with nearest neighbor interaction with free boundaries can be mapped onto a matching problem of a general graph [2, 10, 11].

In 1999, Gabor and Kondor [12] used spin glasses for the first time in solving the portfolio selection problem with regard to its constraints. In their paper, they used a similar energy function to that of a Hopfield neural network [13]. In 2001, Nishimori [14, 15] considered the application of spin glasses in transferring information in noisy channels. In 2004, Horiguchi et al [16] proposed a spin glass-based routing algorithm for adaptive computer networks. Later in 2009, Vafaei and Akbarzadeh in [6] introduced migration and elitism operators to find ground state of spin glasses with only a limited number of bonds, i.e. *short range* spin glasses. There [6], authors exploited *local* interaction among spins. In contrast, we consider here the *short range* effect of spin interaction by investigating the use of extremal optimization (EO).

The EO heuristic was first motivated by the Bak-Sneppen model of biological evolution [9] in 1993 intended for a lattice (glass) of cooperating species (spins). Some applications of this method were analyzed in solving optimization problems, including, solving the problem of the travel salesman problem [11], graph partitioning [17, 18], graph coloring [19, 20], social modeling [21], complex network analysis [22], and molecular dynamics' simulation [23].

More specifically, EO is inspired by self-organized criticality (SOC), which is a statistic physics concept to describe a class of systems that have a critical point as an attractor [24]. In SOC, there is no need to tune control parameters with precise values. Just inspired by this principle, EO drives the system far from equilibrium: aside from ranking, there exists no adjustable parameter, and new solutions are accepted indiscriminately [24].

3. Spin Glass Model

Spin glass is a model which can be used to investigate the collective properties of physical systems made from a large number of simple elements. The important feature in this paradigm is that the interactions among these elementary components yield a collective phenomenon, such as stable magnetization orientation and the crystalline state of metal or alloy. In the Ising spin glass model [1, 25], an Ising spin on a lattice point takes on one of two possible values (directions) (i.e., ± 1 or up and down). By generalizing the Ising spin glass model to a XY spin glass model (hereafter referred to as Spin Glass model for short) [2, 11], each spin can point to any direction in a plane instead of just two possible directions.

A suitable theoretical model describing spin glasses consists of N spins placed on the regular sites of a d-dimensional lattice with linear extension L, e.g., quadratic ($N = L^2$) or cubic ($N = L^3$). The spins interact ferromagnetically or antiferromagnetically with their neighbors. The energy of such a network comes from two contributions [4, 25] and can be written as below:

$$E(\{x_i\}) = \left[-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{m} x_i J_{ij} x_j \right] + \left[-\sum_{i=1}^{N} h_i x_i \right] \qquad (1)$$

Where $E(\{x_i\})$ is the energy of all spins; the sum i, j runs over all pairs of nearest neighbors; m is the number of nearest neighbors of each spin i (that can be $m = 4$ in Van Neumann cellular automata (CA), or $m = 8$ in Moore CA [26], or $m = N$ for full connection); and J_{ij} denotes the strength of the bond connecting spins i and j. $J_{ij} > 0$ describes a ferromagnetic interaction, while $J_{ij} < 0$ describes an antiferromagnetic interaction. The quantity h_i is the external field acting on spin i and describes the energy due to the spin's orientation. Furthermore, the factor $\frac{1}{2}$ corrects for double counting of the interaction between every two neighboring spins. Here the task is to find a spin configuration x_i that minimizes the energy of the spin glass, given $\{J_{ij}\}$ and $\{h_i\}$.

4. Portfolio Selection Problem

Let us consider the Markowitz mean-variance model [27] for the portfolio selection problem as stated below,

$$\text{Min} \sum_{i=1}^{N} \sum_{j=1}^{N} x_i \sigma_{ij} x_j \qquad (2)$$

$$\text{Max} \sum_{i=1}^{N} \mu_i x_i \qquad (3)$$

$$\text{Subject to} \sum_{i=1}^{N} x_i = 1 \qquad (4)$$

$$0 \le x_i \le 1, \quad i = 1, \ldots, N \qquad (5)$$

Where N is the number of different assets, μ_i is the mean return of asset i, and σ_{ij} is the covariance between returns of assets i and j. The decision variable x_i represents the fraction of capital to be invested in asset i.

Eqs. (1)-(2) are two cost functions that should be solved with constraints (3) and (4). μ_i is the mean return of asset i in n intervals of time, i.e. $\mu_i = \sum_{t=1}^{n} \frac{W_{ei}(t) - W_{bi}(t)}{W_{bi}(t)}$, where W_{bi} is the i^{th} asset value at the beginning, and W_{ei} is the i^{th} asset value at the end of each interval.

A feasible solution of the portfolio selection problem is an optimal solution if there is no other feasible solution improving one objective without deteriorating the other. Usually, multiobjective optimization problems such as those in [28] have multiple non-dominant optimal solutions. This set of solutions form an *efficient frontier*. For the problem defined in Eqs. (1)–(4), the efficient frontier is an increasing curve that gives the best tradeoff between mean return and variance (risk).

In this paper, we change the multi-objective problem into a multimodal problem with a single objective function as follows,

Minimize

$$\lambda.\left[\sum_{i=1}^{N}\sum_{j=1}^{N} x_i \sigma_{ij} x_j\right] + (1-\lambda).\left[-\sum_{i=1}^{N} \mu_i x_i\right] \quad (6)$$

Subject to $\sum_{i=1}^{N} x_i = 1$ (7)

$0 \le x_i \le 1, i = 1,..., N$ and $0 \le \lambda \le 1$ (8)

Here, $\lambda = 0.5$, for the equal effect of risk and return.

5. Solving Portfolio Selection Problem Using Short Range Spin Glass

To solve the portfolio selection problem, as studied in [6], each asset is supposed to be a spin having a value between 0 and 1. A glass (network) of such spins has an energy function as indicated in Eq. (1). To solve the portfolio selection problem in Eq. (6), the following relationship is observed (for short range spin glass, i.e. $\boldsymbol{m < N}$)

$$J_{ij} = -2\lambda\sigma_{ij} \quad (9)$$

$$h_i = (1-\lambda)\mu_i \quad (10)$$

Eqs. (9) and (10) refer to the *interaction* among spins and the *external field* energy of each spin, respectively. Search for optimal solutions begins with an initial assignment of each spin to $\frac{1}{N}$ (or randomly assigned). Then, any of the various search strategies can be used, in order to put the system in its minimum energy. At any moment (spin flip or spin change), a spin is randomly selected, and ε is added to the spin's value ($\varepsilon = 0.05$, is a small fixed value). Then the values of neighboring spins change in such a way that they always meet constraints (7) and (8).

5.1. A Hybrid Local Search Algorithm (EO-SA)

In contrast to the above SA method that selects spins randomly in each flip, EO-SA gives the highest selection probability to a spin that has lowest local energy (from Eq. (14)) hence avoiding locally optimal solutions. Here, spins are ordered based on their local energy. At each step, a 'superior' spin is selected based on its given probability in Eq. (16), with higher probability given to lower energy spins. After several iterations, because the glass moves toward lower energy and each spin affects its neighboring spins, many spins have lower energy than their initial values, i.e. the given value in Eq. (15) reduces. Hence, the system's correlation increases, and the change in each spin leads to a change in many other spins, which leads to SOC [7, 9]. In this state, any small change leads to major changes in the system, so it is expected that most possible states are accessible. Therefore, one can easily escape local optimal solutions and survey most possible states for the system.

Algorithm (3) describes how the EO-SA method can be applied to spin glasses. Temperature and cooling schedule plays a central role in SA strategy [29]. The system's temperature is usually initialized to a high value to allow all possible states to be the initially producible, i.e. more global exploration. The system is then gradually cooled to allow better local search. To do so, the temperature of the glass is considered to be initially set to $T_0 = 1$ (at high temperatures all states can occur). Each time the changes are applied, the temperature is decreased until it reaches near zero. Temperature variation is calculated as follows,

$$T(n) = \frac{T_0}{n^2}, \qquad n \ge 1 \quad (11)$$

In this algorithm, λ_i is local energy of each spin in Eq. (14). Spin glasses' total energy can then be obtained from a sum of λ_i's in Eq. (15).

$$\lambda_i = x_i\left(\frac{1}{2}\sum_{j=1}^{m} J_{ij}.x_j + h_i\right) \quad (14)$$

$$E(\{x_i\}) = -\sum_{i=1}^{n} \lambda_i \quad (15)$$

All λ_i's are computed and ordered in rising order at each step and selected based on the power law distribution in (Eq. (16)). The selected spin's value is then changed. If this change leads to lower (better) glass energy, it is accepted; otherwise, it is accepted with a probability of $e^{-\frac{\Delta E}{T}}$.

Algorithm 3: EO-SA Spin Glass
Begin

1 *Initialize spin glass and set amount of all spins to* $\frac{1}{N}$.

2 *Calculate* λ_i *for each spin and sort them with a decreasing order.*

3 *Selected spins with power law distribution in Eq. (16) based on calculated* λ_i.

4 *Change the state of the selected spin* i *by* ε *(very*

small change) and change all the neighboring spins to satisfy Eqs. (7) and (8).

5` *Calculate the energy of the changed spin and its neighboring spins* ($E_{new} = \sum_{i=1}^{m} E_i$)

6 $\Delta E = E_{new} - E_{old}$

7 *If* $\Delta E < 0$ *then accept this change, else*

8 *If* $\Delta E > 0$ *then accept this change with probability* $e^{-\frac{\Delta E}{T}}$

9 *Continue this process with decreasing temperature till either* ΔE *remains near 0 for several iterations (i.e., the system has reached steady state, or* T *has reached near 0 (system has cooled)*

End

In the above algorithm, E_{old} and E_{new} are glass's energy before and after applying a change, and T is the system's temperature at the time of applying the change. This SA-based algorithm ensures convergence to global solutions if T is reduced sufficiently slowly. However, this also means a slow rate of convergence. In the below two algorithms, we investigate two alternative heuristics that *choose* the next spin based on a given criterion, hence aiming for faster convergence.

Selecting every spin at each step depends on the following Equation [2]:

$$\boldsymbol{k} = (1 + (\boldsymbol{n}^{1-\tau} - 1).\alpha)^{\frac{1}{1-\tau}} \qquad (16)$$

Where k refers to the selected spin's number, whose set is ordered from low energy to high energy spins, and $0 \le \alpha \le 1$ is a random number. When $\tau = 0$, the algorithm acts like SA and when $\tau \to \infty$, the algorithm always selects the spin with minimum energy. Therefore, it can be expected that the above algorithm has the power law distribution equaling $P_k \propto k^{-\tau}$ in which $1 \le k \le n$ [8].

5.2 Problem Constraints

The two constraints in Eqs. (7) and (8) in portfolio selection problem must be considered in the algorithm. To maintain the first constraint, whenever ε is added to each spin's value, ($x_i := x_i + \varepsilon$), the value $\frac{\varepsilon}{m}$ is reduced from each of the spin's m neighbors, ($x_j := x_j - \frac{\varepsilon}{m}$). This ensures that the sum of all spin values remain at 1. If $x_i \ge 1$, then $x_i := 1$ and its extra value is reduced from ε. Furthermore, if for each neighbor $x_j - \frac{\varepsilon}{m} \le 0$, then $x_j := 0$ and the difference is added to x_i. Considering the last two cases, the second constraint (Eq. (8)) is also maintained.

6. Mathematical and Experimental Results

In order to verify the effectiveness of the above algorithms, the benchmarked "standard efficient frontier" (Pareto Front) is compared with the efficient frontier resulting from the proposed methods.

Experiments on the benchmark data were originally performed in [30]. These data are obtained from five major stock exchange markets, during the time period extending from March 1992 to September 1997. These five stock exchange markets include Hang Seng in Hong Kong (31 assets), Deutscher Aktien Index (DAX100) in Germany (85 assets), Financial Times London Stock Exchange (FTSE100) in Britain (89 assets), Standard & Poor's (S&P100) in USA (98 assets), and Nikkei in Japan (225 assets). The efficient frontier for each of these five stock markets in the available time period is characterized by mean return as in Eq. (2) and variance of return as in Eq. (1). Fig. (4) illustrates this efficient frontier for the benchmark data.

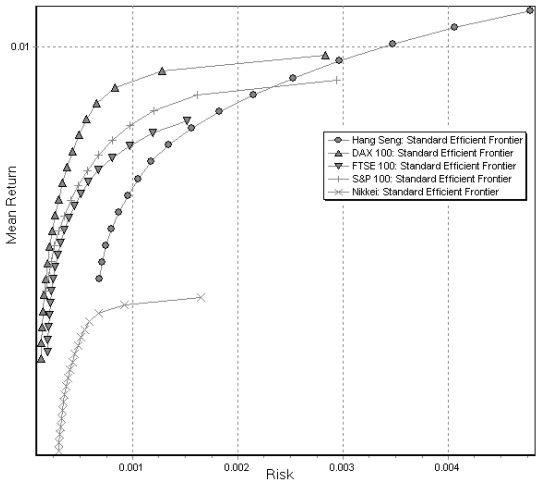


Figure. 4: Efficient frontier for benchmark data from five major stock markets as reported in [30]

Three sets of tests are performed to analyze the spin glass behavior as follows. Firstly, spin glass's accuracy and rate of convergence are compared for the proposed two approaches, i.e. EO-SA, as well as the more conventional SA approach. Secondly, the resulting efficient frontier is compared with the benchmark's efficient frontier. Thirdly, the reliability of presented algorithms and phase transition analysis are tested and compared with those of SA. All the experiments were performed using Borland Delphi 6.0 running on a Pentium 2.0 GHz PC, under Windows XP operating system. It should be mentioned that each epoch equals 50 spin flips.

6.1 Comparing SA, EO, EO-SA

As seen in Fig. (5), all three spin glasses begin under similar random initial states and reach same final states using the two SA, EO-SA algorithms for the S&P stock market. However, they have significantly different rates of convergence. Based on the results seen in all studied stock markets, EO-SA method quickly approaches the final ground states, but fluctuates around the final states for much iteration before reaching it. Because, in each iteration the number of qualified spins increases, this cause increasing the correlation between spins; therefore, changing in each spin leads to change in many other spins

and cause fluctuates around the response range. In contrast, the SA method has a simple random behavior and slowly moves to the ground state. As illustrate in Fig. (5), Both SA and EO-SA algorithm reach near to the ground state. But, EO algorithm drops in the local optimum and fluctuates far from the response range.

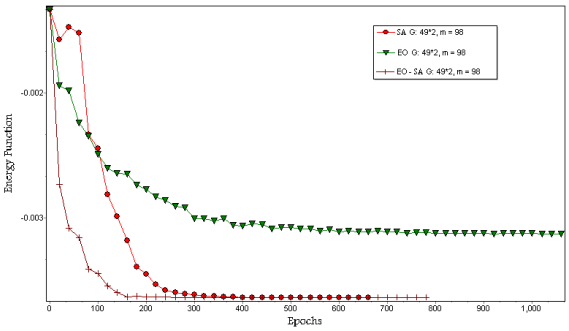


Figure 5: Comparing SA, EO and EO-SA algorithms' rate of convergence for S&P stock market.

Fig. (6) plots the final spin selection probability vs. spins that are ordered in descending selection probability levels. This conclusion is illustrated in an experiment carried out for S&P stock market, with $\tau = 0.9$ for EO-SA and $T_0 = 1$ for SA. This state occurs when the glass has passed its transient state.

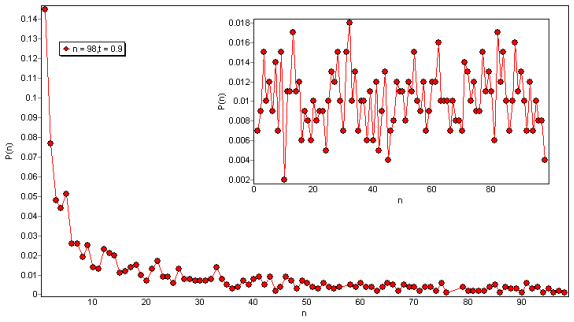


Figure 6: Comparing the distribution of spins' selection resulting from EO-SA (external chart) and SA (internal chart), after reaching final state for S&P stock market. n is the size of the portfolio.

Table (1) shows a comparison between the computation time and the accuracy of reaching ground states in the three mentioned methods. While both SA and EO-SA method reach the ground state and have a generally comparable accuracy, EO-SA method has been more reliable than the other two methods in our experiments; however, the convergence time is more than EO.

Table 1: Comparing the three algorithms (SA, EO and EO-SA) for accuracy and rate of convergence over 100 independent runs for each of the five World's stock markets. Each approach is evaluated in terms of convergence time (shown in milliseconds), average minimum energy during the runs, and accuracy defined by the ratio of obtained minimum energy over actual minimal energy.

Stock market	**SA**			**EO**			**EO-SA**		
	Convergence time (ms)	Average Minimum Energy	Accuracy	Convergence time (ms)	Average Minimum Energy	Accuracy	Convergence time (ms)	Average Minimum Energy	Accuracy
HangSeng (31 Assets)	2140	-0.00336	99.20%	833	-0.0031	92.26%	1253	-0.00337	99.40%
DAX (85 Assets)	19111	-0.00412	99.10%	6064	-0.0038	92.23%	4091	-0.00412	99.18%
FSTE (89 Assets)	25040	-0.00335	99.20%	7741	-0.0032	95.52%	6169	-0.00335	99.24%
S&P (98 Assets)	22828	-0.00363	99.36%	2597	-0.0029	79.94%	9048	-0.00363	99.3%
Nikkei (225Assets)	214045	-0.00142	98.91%	89761	-0.00135	95.07%	98103	-0.00142	98.89%

7. Reliability Test

Test of reliability is performed by running the algorithm n times independently with the same data [10]. To pass the test, the test runs are expected to yield similar results with small variance. To do so, the reliability test of the three algorithms is carried out for the five benchmarks. For brevity, the analysis of S&P stock market is shown here. Results are shown in the form of the frequency chart in Fig. (7, 8). It is done in such a way that spin glass's minimum energy (E_{gs}) in the ground state is counted and the probability to reach that state is also shown. The variance between the final energy states is given in Table (1). Experimental results from 100 trials indicate that the algorithm's final value has a small variance. In other words, final spin glass's energy at each trial is in the range of best responses. Even though the movement towards this final response is random in the above algorithms, they consistently reach the ground state.

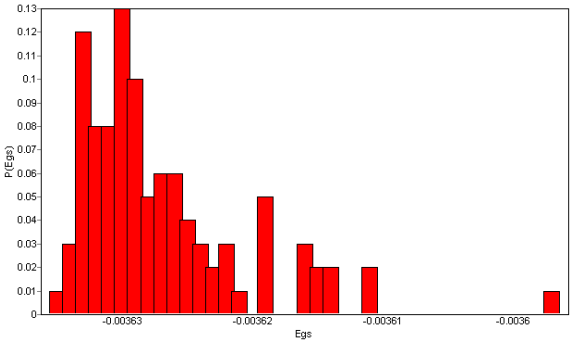


Figure 7. The SA Reliability Test of S&P 100 stock market (where Egs is glass energy at ground state).

8. Optimization Frontier

The final Pareto front from EO-SA algorithm can be

seen in Fig. (9) (the results are similar to SA). It shows the validity of energy reduction and avoiding local optimums for the five mentioned stock markets. The standard frontier for different λ's is also drawn. For having a Pareto frontier, λ is considered in the range of 0.05 to 0.95 with 0.05 differences. For any λ, timization state was found and its risk found,apital return values were defined with points. The validity of the presented algorithm in finding optimization response with different λ was seen through comparing the resulting and standard (benchmark) Pareto fronts. Since the surface of whole optimization frontier is covered, one can draw that the presented method gives response for any**Error! Bookmark not defined.**·

9- Phase Transition Analysis

The temperature at phase transition is defined as the temperature at which the likelihood of reaching the glass's actual minimum state suddenly decreases [11]. Fig. (10) illustrates the spin glass temperature at phase transition. As it can be observed, before phase transition the probability of reaching the minimum state ($\frac{E_{gs}}{E_{\min}}$) nears 1 (E_{gs} is the ground state energy of glass, and $E_{\min}$ is the actual minimum of cost function).

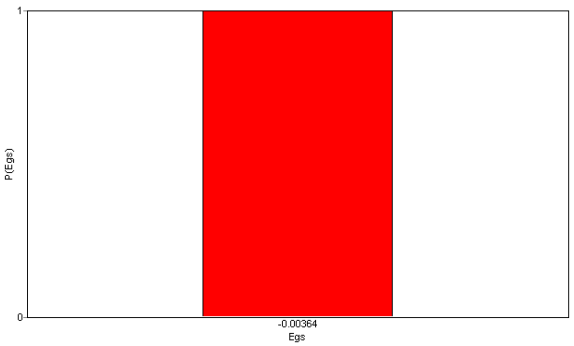


Figure 8. The EO-SA Reliability Test with $\tau = 0.9$ of S&P 100 stock market (where Egs is glass energy at ground state).

As the temperature increases, this probability is expected to gradually decrease, but this decrease does not occur until reaching near the temperature at phase transition, at which time, there is a sudden change in glass behavior. Here, phase transition is defined to occur where the likelihood ratio $\frac{E_{gs}}{E_{\min}}$ is decreased by 1%. As illustrated in Fig. (10), most of the benchmarks reach phase transition at 1.12×10^{-6} temperature (as indicated by a vertical line) for the SA algorithm. As Fig (11) suggests, most of the benchmarks with EO-SA find their minimum states even at higher temperatures ($7.9*10^{-6}$) as compared with SA, prompting EO-SA as the algorithm that converges soonest. The above phase transition analysis also confirms the conclusions of Table (1). Specifically, this experiment shows that, in EO-SA, the temperature of phase transition is higher than the SA, and accordingly better rate of convergence.

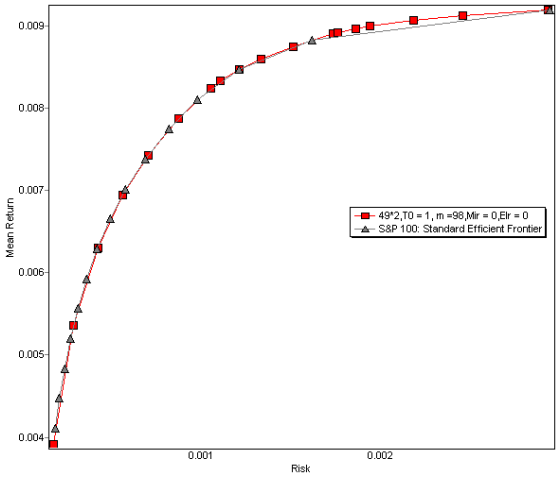


Figure 9: Efficient frontier obtained from EO-SA algorithm compared to standard efficient frontier from benchmark data [22].

10- Conclusion

In this paper, a hybrid approach are proposed for finding short range spin glass's ground state based on extremal optimization (EO-SA). The EO-SA method selects spins with minimum energy with a higher probability. A flip (change) in any spin leads to changes in its neighboring spins. If all of these changes reduce spin glass's energy, more and more spins will be better qualified and the correlation between spins increases. A process of self organizing criticality then occurs where the change in each spin leads to changes in many spins, allowing the glass to escape local optimums more easily. As the experiments on phase transitions illustrate, the temperature at phase transition is elevated, hence the rate of convergence is improved.

A comparison of experiments shows the superiority of EO-SA to conventional EO and SA. EO only has a faster rate of convergence with not reliable accuracy. This is also confirmed by phase transition analysis and reliability test.

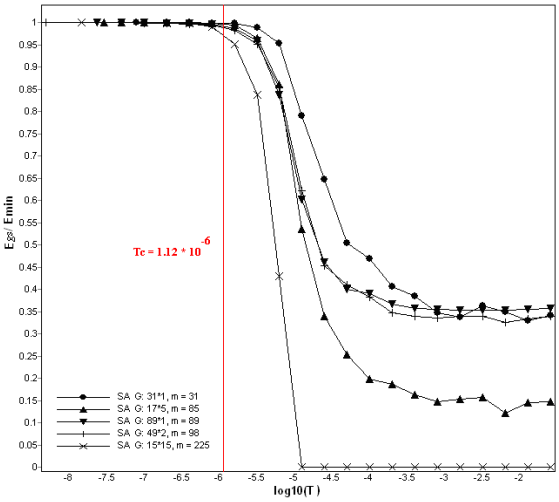


Figure 10: Portfolio selection phase transition phenomena based on SA for the five benchmark stock market data;

transition temperature is approximately 1.12×10^{-6}

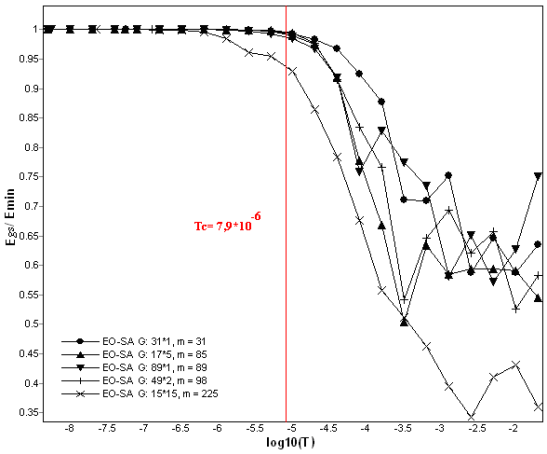


Figure 11: Portfolio selection phase transition phenomena based on EO-SA for the five benchmark stock market data; transition temperature is approximately $7.9*10^{-6}$

References

[1] E.Bolthausen, A.Bovier, "Spin Glasses," *Springer-Verlag Berlin Heidelberg,* (2007).

[2] A. K. Hartmann, H. Rieger, "Optimization Algorithms in Physics," *Wiley-VCH Verlag Co,* (2002).

[3] S. Galluccio, J.P. Bouchaud, and M. Potters, "Rational Decisions, Random Matrices and Spin Glasses," *Journal of Physica A* 259, pp: 449-456, (1998).

[4] S. Kirkpatrick, R.H. Swendsen, "Statistical Mechanics and Disordered Systems," *Journal of Communications of the ACM*, Vol. 28, No 4, April (1985).

[5] Y. Crama, M. Schyns, "Simulated Annealing for Complex Portfolio Selection Problems," *European Journal of Operational Research 150*, pp: 546-571, (2003).

[6] M. Vafaei Jahan, M.R Akbarzadeh-T, "From Local Search to Global Conclusions: Migrating Spin Glass-based Distributed Portfolio Selection," Accepted for publication, 2009, *IEEE Transactions on Evolutionary Computations*.

[7] K.S. Narendra and M.A.L Thathachar, "Learning Automata: An Introduction," *Prentice Hall,* (1989).

[8] S.Boettcher, A.G. Percus, "Extremal optimization: An Evolutionary Local search algorithm," *Neural, Parallel & Scientific Computations,* Vol. 10, Issue 2, pp: 249 - 258 (2002).

[9] P. Bak and K. Sneppen, "Punctuated Equilibrium and Criticality in a simple Model of Evolution," *Phys. Rev. Lett.* 71, 4083-4086 (1993).

[10] A.K. Hartmann and H. Rieger, "New Optimization Algorithms in Physics," *Wiley-VCH Verlag Co,* (2004).

[11] A. K. Hartmann, M. Weigt, "Phase Transitions in Combinatorial Optimization Problems, Basics, Algorithms and Statistical Mechanics," *Wiley-VCH Verlag Co,* (2005).

[12] A. Gabor, I. Kondor, "Portfolio with Nonlinear Constraints and Spin Glasses," *Physica A 274, 222-228,* (1999).

[13] A. Fernandez, S. Gomez, "Portfolio Selection Using Neural Networks," *Computers & Operations Research 34,* pp: 1177-1191 (2007).

[14] H. Nishimori, "Statistical Physics of Spin Glasses and Information Processing: An Introduction," *Clarendon press oxford,* (2001).

[15] H. Nishimori, *"*Spin glasses and information*", Physica A 384, 94-99* (2007).

[16] T. Horiguchi, H. Takahashi, K. Hayashi, C. Yamaguchi, "Ising Model for Packet Routing Control," *Journal of Physics Letters A 330*, pp:192-197, (2004).

[17] S. Boettcher, AG. Percus, "Extremal optimization for graph partitioning," *Phys. Rev. E Stat. Nonlin. Soft Matter Phys*, 64(2 Pt 2):026114. Epub (2001).

[18] S. Boettcher, "Extremal Optimization of graph partitioning at the percolation threshold," *J. Phys A.: Math.Gen* **32**, pp: 5201–5211, (1999).

[19] S. Boettcher, "Extremal Optimization at the Phase Transition of the 3-Coloring Problem," *Phys. Rev. E Stat Nonlin Soft Matter Phys*, 69(6 Pt 2):066703. Epub (2004).

[20] S. Boettcher, AG. Percus, "Optimization with extremal dynamics," *Phys. Rev. Lett*. Jun 4, 86(23); pp: 5211-4, (2001).

[21] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, 72(2), pp.1-4, 2005.

[22] D. Garlaschelli, A. Capocci and G. Caldarelli, "Self-organized network evolution coupled to extremal dynamics," *Nature Physics*, Vol. 3, pp.813-817, 2007.

[23] T. Zhou, W. J. Bai and L.J. Cheng, "Continuous extremal optimization for Lennard-Jones clusters," *Physical Review E*, 72(1), pp.1-5, 2005.

[24] G.Q. Zeng and Y.Z. Lu, "Survey on Computational Complexity with Phase Transitions and Extremal Optimization," *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, Shanghai, P.R. China, December 16-18, 2009.

[25] Bar-Yam, Yaneer, "Dynamics of Complex Systems," *Addison Wesley Longman, Inc,* (1997).

[26] P. Sarkar, "A Brief History of Cellular Automata," *ACM Computing Surveys, Vol. 32, No. 1, March* (2000).

[27] H. Markowitz, "Portfolio selection," *Journal of Finance No 7:77-91,* (1952).

[28] C.A. Coello Coello, "An Updated Survey of GA-Based Multiobjective Optimization Techniques," *ACM Computing surveys,* Vol. 32, No. 2, June (2000).

[29] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical Computer Modeling,* Vol. 18, No. 11, Dec (1993).

[30] Portfolio selection benchmark data at "http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html "

SESSION

APPLICATIONS

Chair(s)

TBA

Comparative results of DE variants and a SQP algorithm to maximize the dexterity of an omnidirectional wheeled mobile robot

Miguel G. Villarreal-Cervantes[1*], Carlos A. Cruz-Villar[2], Jaime Álvarez-Gallegos[2] and Edgar A. Portilla-Flores[1]
[1]**Mechatronic Section, Postgraduate Department, CIDETEC-IPN, Mexico, D. F., Mexico.**
[2]**Electrical Engineering Department, CINVESTAV-IPN, Mexico, D.F., Mexico.**

Abstract— *In this paper the formulation of the optimal design for wheel location of an omnidirectional mobile robot is proposed as an optimization problem. The empirical comparison of the SQP algorithm and eight different DE variants for this particular optimization problems is presented. The importance of using heuristic approaches in real world optimization problem is analyzed via empirical results.*

Keywords: Differential evolution, heuristic algorithms, omnidirectional mobile robot.

1. Introduction

An omnidirectional wheeled mobile robot (OWMR) is a wheeled vehicle with full mobility in the plane which means that they can move at each time instant in any direction without any reorientation [1]. Several researches in the control of this kind of robot have been developed in the last decades [2], [3]. Nevertheless the control performance of the mobile robot for trajectory tracking can be affected by the placement of the robot wheels. So, the dexterity of the robot is one important factor to have an adequate control performance.

The dexterity can be defined as the ability of the robot to move and apply forces and torques in arbitrary directions with equal ease [4]. The location of the OWMR wheels is an important factor in the robot dexterity. A poor dexterity of the OWMR means that small changes in the linear velocity of the wheels results in large changes in the linear and angular velocity of the OWMR. A good dexterity of the OWMR means that small changes in the linear velocity of the wheels results in small changes in the linear and angular velocity of the OWMR. In this paper the optimum wheel location for the omnidirectional wheeled mobile robot with three omnidirectional wheels [5] is stated as an optimization problem.

On the other hand, a real world decision problem can be stated as an optimization problem [6]. Nevertheless, the algorithm used to solve the optimization problem depends on the problem at hand [7]. There are two main approaches to be considered in the selection of the algorithm and they depend on how the search direction is computed: 1) The gradient based algorithms [8] and 2) the heuristic based algorithms [9].

The gradient based algorithms such as sequential quadratic programming (SQP) algorithm, depend on the initial parameters and the kind of optimization problem (linear, nonlinear, etc.) [10]. The convergence to local solutions near the initial condition is the first drawback in the solution of nonlinear optimization problems. The second drawback is that it can not be used when the optimization problem is discontinuous. Hence, gradient based algorithms do not ensure global optimum and they have limited application [10].

Heuristic based algorithms (HBA) have been widely used to solve real world optimization problems [10], [11], [12], [13]. Differential evolution (DE) [14] is an HBA which can be used to solve continuous, discontinuous, linear, nonlinear, dynamic and static optimization problems. DE performs mutation based on the distribution of the solutions in the current population such that, the search direction depends on the location and selection of individuals. There are several DE variants but the most popular is called "DE/rand/1/bin" where just one difference (of two randomly chosen individuals added to another solution) is calculated. Besides the selection of the population size and maximum generation number, an important factor when using DE is the selection of the variant. In those DE variants the mutation and recombination operator is changed and the performance of the DE variants depends on the problem as it is establish in [7].

In this paper the performance of a set of eight DE variants and the performance of the SQP algorithm are studied to identify which algorithm is more suitable to solve a particular real world optimization problem. The main motivations of this work are 1) the proposal of a new optimization problem based on a real word problem which can be considered as a multimodal-nonseparable problem, 2) the performance evaluation of the chosen algorithms when solving this particular real world optimization problem and 3) the optimal solution for the particular optimization problem, i. e. the optimum location of the robot wheels.

2. Optimization problem statement

Let $p = [\delta_1, \delta_2, \delta_3, L_1, L_2, L_3]^T \in R^6$ be the design parameters of the wheel locations of a mobile robot where δ_i and $L_i \; \forall \; i = 1, 2, 3$, are the $i-th$ angle and the $i-th$

distance of the robot wheel as it is shown in Fig. 1. The mass center of the omnidirectional mobile robot is considered to be on the origin of the coordinate system "m" (robot center).

The transformation between the linear velocity of the wheel $v = [v_1, v_2, v_3]^T$ and the linear and angular velocity of the mobile $V_L = [V_x, V_y, w]^T$ is stated in (1).

$$V_L = A\ v \tag{1}$$

where $A \in R^{3\times 3}$ is the Jacobian matrix.

$$A = \begin{bmatrix} -\sin(\delta_1) & \cos(\delta_1) & L_1 \\ -\sin(\delta_2) & -\cos(\delta_2) & L_2 \\ \cos(\delta_3) & \sin(\delta_3) & L_3 \end{bmatrix} \tag{2}$$

The optimization problem consists in finding the optimal design parameter vector $p^* \in R^6$ such that a small change in the linear velocity of the wheel v or in the Jacobian matrix A, results in a small change in the linear and the angular velocity of the mobile V_L. The condition number of A measures how much the solution of (1) can change in proportion to small changes in the matrix A or the vector v. A low condition number means that the matrix A is well-conditioned, while high condition number means that the matrix A is ill-conditioned. Hence the optimization problem formulation (3-4) is, to find the optimal design parameter vector $p^* \in R^6$ which minimizes the condition number of the Jacobian (3), subject to the limits in the design parameter vector (4), i.e.,

$$\min_{p^*}\ J = \min_{p^*}\ \|A\|_F \left\|A^{-1}\right\|_F \tag{3}$$

subject to:

$$p_{\min} \le p \le p_{\max} \tag{4}$$

The term $\|*\|_F$ is the Frobenius norm of $*$. The performance index $J = \|A\|_F \left\|A^{-1}\right\|_F$ can be stated as in (5).

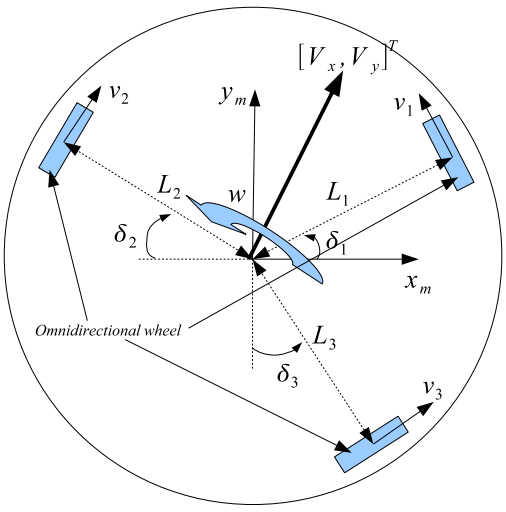


Fig. 1: Omnidirectional mobile robot.

It is observed that the problem in (3-4) is a nonseparable optimization problem.

$$J = \sqrt{\frac{B}{C}}\sqrt{(L_1^2 + L_2^2 + L_3^2 + 3)} \tag{5}$$

where $B = (4L_1^2 + 4cos(\delta_1 + \delta_2)L_1L_2 + 4sin(\delta_1 - \delta_3)L_1L_3 + 4L_2^2 + 4sin(\delta_2 + \delta_3)L_2L_3 + 4L_3^2 - cos(2\delta_1 + 2\delta_2) + cos(2\delta_1 - 2\delta_3) + cos(2\delta_2 + 2\delta_3) + 3)$ and $C = L_1^2cos(2\delta_2 + 2\delta_3) + L_2^2cos(2\delta_1 - 2\delta_3) - L_3^2cos(2\delta_1 + 2\delta_2) + L_1^2 + L_2^2 + L_3^2 + 2L_1L_3sin(\delta_1 + 2\delta_2 + \delta_3) + 2L_1L_2cos(\delta_1 + \delta_2) + 2L_2L_3sin(\delta_2 + \delta_3) + 2L_1L_2cos(\delta_1 - \delta_2 - 2\delta_3) + 2L_2L_3sin(2\delta_1 + \delta_2 - \delta_3) + 2L_1L_3sin(\delta_1 - \delta_3)$.

3. Optimization algorithms

The optimization problem (3-4) is solved by using a Sequential Quadratic Programming (SQP) algorithm [8] and by using eight variants of the differential evolution (DE) algorithm [14], [15].

SQP algorithm represents the state of the art in nonlinear programming techniques. The SQP algorithm allows to closely mimic Newton's method for constrained optimization just as it is done for unconstrained optimization. At each major iteration, an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton updating method. This is then used to generate a QP subproblem whose solution is used to form a search direction for a line search procedure.

The main idea is the formulation of a quadratic programming (QP) subproblem based on a quadratic approximation of the Lagrangian function (6) where $J(p)$ is the performance function, $g_i(p)$ and $h_i(p)$ are the $i-th$ inequality and equality constraints.

$$L(p, \lambda) = J(p) + \sum_{j=1}^{n_g} \lambda_j \cdot g_j(p) + \sum_{k=1}^{n_h} \lambda_k \cdot h_k(p) \tag{6}$$

Therefore, the QP subproblem (7-8) is obtained by linearizing the nonlinear constraints, where the matrix $H_i = \nabla^2 L$ is a positive definite approximation of the Hessian matrix of the Lagrangian function (6). H_i is updated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. Hence, if the search direction d_i solves the subproblem given in (7-8) and $d_i = 0$, then the parameter vector p is an optimal solution of the original problem. Otherwise, we set $p^{i+1} = p^i + \alpha_i d_i$ and with this new vector the process is repeated again. The step length parameter α_i is determined by an appropriate line search procedure so that a sufficient decrease in the performance index is obtained.

$$\underset{p\in n_p}{Min}\ \frac{1}{2} d^T H_i d + \nabla J(p^i)^T d \tag{7}$$

subject to:

$$\begin{array}{c} \nabla g_j(p^i)^T d + g_j(p^i) \le 0,\ j = 1, \ldots, n_g \\ \nabla h_k(p^i)^T d + h_k(p^i) \le 0,\ k = 1, \ldots, n_h \end{array} \tag{8}$$

1 **BEGIN**
2 $G = 0$
3 Create a random population $\vec{x}_G^i \ \forall i = 1, ..., NP$
6 Evaluate $J(\vec{x}_G^i)$, $g(\vec{x}_G^i)$, $h(\vec{x}_G^i) \ \forall i = 1, ..., NP$
7 **Do**
8 **For** $i = 1$ to NP **Do**
9 $\Longrightarrow$ Select randomly $\{r_1 \neq r_2 \neq r_3\} \in \vec{x}_G$.
10 $\Longrightarrow$ j_{rand} =randint$(1, D)$
11 $\Longrightarrow$ **For** $j = 1$ to D **Do**
12 $\Longrightarrow$ **If** (rand$_j[0, 1) < CR$ or $j = j_{rand}$) **Then**
13 $\Longrightarrow$ $u_{j,G+1}^i = x_{j,G}^{r_1} + F(x_{j,G}^{r_2} - x_{j,G}^{r_3})$
14 $\Longrightarrow$ **Else**
15 $\Longrightarrow$ $u_{j,G+1}^i = x_{j,G}^i$
16 $\Longrightarrow$ **End If**
17 $\Longrightarrow$ **End For**
18 Evaluate $f(\vec{u}_{G+1}^i)$, $g(\vec{u}_{G+1}^i)$, $h(\vec{u}_{G+1}^i)$
19 **If** $(\vec{u}_{G+1}^i < \vec{x}_G^i)$ **Then**
20 $\vec{x}_{G+1}^i = \vec{u}_{G+1}^i$
22 **Else**
23 $\vec{x}_{G+1}^i = \vec{x}_G^i$
24 **End If**
25 **End For**
29 $G = G + 1$
36 **While** $(G \leq G_{Max})$

Fig. 2: DE/rand/1/bin algorithm.

The DE/rand/1/bin, DE/rand/1/exp, DE/best/1/bin, DE/best/1/exp, DE/current-to-rand/1, DE/current-to-best/1, DE/current-to-rand/1/bin and DE/rand/2/dir are the proposed DE variants. The DE/rand/1/bin algorithm is shown in Fig. 2. The general convention used is DE/x/y/z. DE means Differential Evolution, x represents the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exponential or binomial).

The main differences among the DE variants are in the recombination operator (rows 10-17 in Fig. 2) and in the way of selecting the individuals in the mutation vector (row 9 in Fig. 2). The DE variants implementation details are summarized in Fig. 3.

4. Results and discussion

In this work, the SQP algorithm and DE variants are programmed in Matlab Release 7.9 on a Windows platform. Computational experiments were performed on a PC with a 1.83 GHz Core 2 Duo processor and 2 GB of RAM. One hundred independent runs were carried out for the algorithms.

The initial condition for the SQP algorithm is randomly chosen and the stop criterion is when the change in the performance index value in two consecutive iterations is smaller than 1×10^{-10}.

Four parameters must be proposed in order to tune the DE algorithm. In this case, the population size NP consists of 100 individuals. The scaling factor F and the crossover constant CR are randomly generated in the interval $0.3 \leq F \leq 0.9$ at each generation, and in the interval $0.8 \leq CR < 1$ at each optimization process. The K value is randomly generated in the interval $0.3 \leq F \leq 0.9$ for the current-to-rand/1, current-to-best/1 and current-to-rand/1/bin DE variants. The stop criterion is when the number of generations is fulfilled $G_{Max} = 300$ or when the error between the mean objective function and the global optimum is smaller than 1×10^{-10} .

4.1 Performance of the Algorithms

In Table 1 the performance for 100 independent runs of the SQP algorithms are shown. We realized that the optimization problem in (3-4) presents several local solutions. So, this optimization problem is a multimodal-nonseparable problem.

The best objective function value found is $J^* = 7.0380710741$. Hence we considered it as the global optimal solution of the problem. It is important to comment that the best objective function value is found once (see column 3 in Table 1). Hence, the mean performance of the SQP algorithm is not adequate because this algorithm presents high sensitivity to the initial condition since the optimization problem is a nonlinear one. In Table 2 the performance of the all DE variants for 100 independent runs are presented. It is observed in Table 1 and Table 2 that the SQP algorithm requieres less computational time than the DE variants.

The results in Table 2 show that the best results were provided by "rand/2/dir", "best/1/bin" and "best/1/exp". Those variants find the optimal objective function value in all runs (see column 4 in Table 2). Nevertheless, "rand/2/dir" is the best of them (better mean objective function value). "best/1/bin" presents smaller generation number than "rand/2/dir" and "best/1/exp", such that the computational time is reduced.

On the other hand, "rand/2/dir", "best/1/bin" and "best/1/exp" present smaller standard deviation of the objective function value of the population than the others variants (see column 3 in Table 2). This deviation indicates that all individuals of the population converge to a solution and hence a rapid convergence towards a solution is done.

The "current-to-best/1" variant can find the optimal solution in 99% of the runs (see column 4 in Table 2). Nevertheless the high standard deviation indicates that the individuals of the population are far apart, such that 1% of the runs can not find the optimal solution.

A high standard deviation means that the algorithm requires more generations for converging to the objective function values of the individuals of the population to a solution (local or global solution). Hence, slow convergence towards a solution is observed and it means that the algorithm can not rapidly find good solutions.

The worst DE variants are "current-to-rand/1", "current-to-rand/1/bin", "rand/1/exp" and "rand/1/bin". In those variants the optimal solution is not found in all runs (see column

Nomenclature	Variant
rand/1/bin	$u_j^i = \begin{cases} x_j^{r_3} + F(x_j^{r_1} - x_j^{r_2}) & \text{if rand}_j(0,1) < CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$
rand/1/exp	$u_j^i = \begin{cases} x_j^{r_3} + F(x_j^{r_1} - x_j^{r_2}) & \text{from rand}_j(0,1) < CR \text{ or } j = j_{rand} \\ x_j^i & \text{otherwise} \end{cases}$
best/1/bin	$u_j^i = \begin{cases} x_j^{best} + F(x_j^{r_1} - x_j^{r_2}) & \text{if rand}_j(0,1) < CR \text{ or } j = j_{rand} \\ x_j^i & \text{otherwise} \end{cases}$
best/1/exp	$u_j^i = \begin{cases} x_j^{best} + F(x_j^{r_1} - x_j^{r_2}) & \text{from rand}_j(0,1) < CR \text{ or } j = j_{rand} \\ x_j^i & \text{otherwise} \end{cases}$
current-to-rand/1	$\vec{u}^i = \vec{x}^i + K(\vec{x}^{r_3} - \vec{x}^i) + F(\vec{x}^{r_1} - \vec{x}^{r_2})$
current-to-best/1	$\vec{u}^i = \vec{x}^i + K(\vec{x}^{best} - \vec{x}^i) + F(\vec{x}^{r_1} - \vec{x}^{r_2})$
current-to-rand/1/bin	$u_j^i = \begin{cases} x_j^i + K(x_j^{r_3} - x_j^i) + F(x_j^{r_1} - x_j^{r_2}) & \text{if rand}_j(0,1) < CR \text{ or } j = j_{rand} \\ x_j^i & \text{otherwise} \end{cases}$
rand/2/dir	$\vec{v}^i = \vec{v}^1 + \frac{F}{2}(\vec{v}^1 - \vec{v}^2 + \vec{v}^3 - \vec{v}^4)$ where $f(\vec{v}^1) < f(\vec{v}^2)$ and $f(\vec{v}^3) < f(\vec{v}^4)$

Fig. 3: DE variants.

4 of Table 2). Nevertheless, "current-to-rand/1" is the worst DE variant among them because it presents the worst mean objective function value and a high standard deviation.

The variant "rand/1/bin" is the most useful variant in several real word optimization problem [10], [11]. Nevertheless in this particular optimization problem, this algorithm can not find the global solution in 300 generations. The result of the standard deviation shows that the "rand/1/bin" presents a slow convergence to a solution. So, the converge of the "rand/1/bin" to a solution depends on the selected population number and if the generation number is not appropriately selected, the "rand/1/bin" does not reach the best solution.

The main highlight of the results is summarized below:

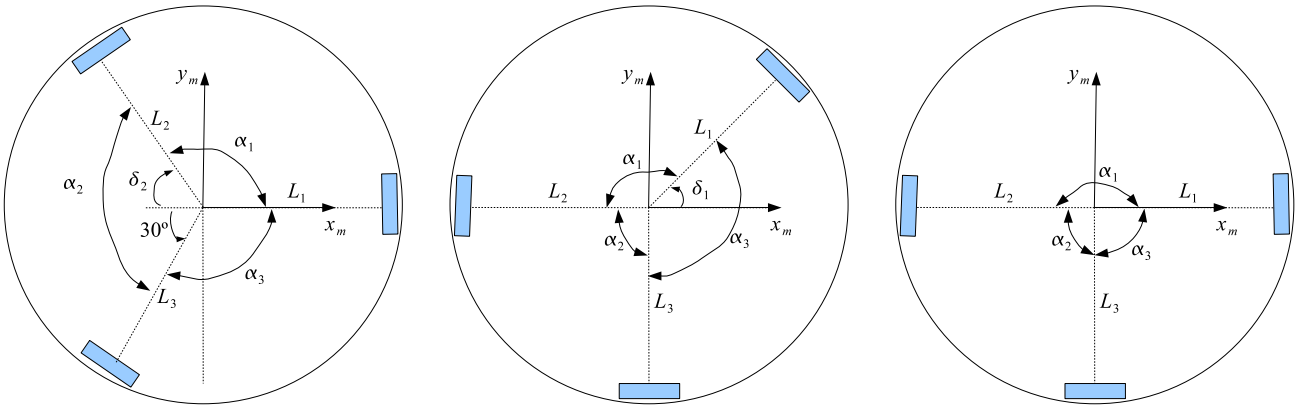
- The optimization problem (3-4) presents several local solutions and it is considered as multimodal-nonseparable problem.
- The results of the comparison of the DE variants show that the "rand/2/dir", "best/1/bin" and "best/1/exp" are the most competitive DE variants for solving the optimization problem (3-4) and they present a rapid convergence towards a solution. Nevertheless, "rand/2/dir" is the best of them (better mean objective function value). The "DE/best/1/bin" presents rapid convergence towards a solution.
- The rand/1/bin could converge to suboptimal solutions if the generation number is not appropriately selected.
- The SQP algorithm requieres less computational time than the DE variants. Nevertheless, the main drawbacks of using SQP algorithm are the high sensitivity to the initial condition when a nonlinear optimization problem is solved. Besides the optimization problem must be twice continuously differentiable in order to compute both the gradient vector and the Hessian matrix. In addition a constraint handling mechanism must be included when constrained optimization problems are solved.
- The DE variants are easier to implement than the SQP algorithm. In addition, a special constraint handling mechanism for the upper and lower constraint in the design variables is not required because it is included in the original DE algorithm [14].
- In real world complex optimization problem the use of a hybrid optimization algorithm based on heuristic and gradient approach must be considered [16].

4.2 Optimal design

In Table 3 the performance index and the global and two local solutions (location of the robot wheel) using the SQP algorithm are shown. The optimal designs of the location of the robot wheel for those solutions are presented in Fig. 4. As it is stated before, several local objective functions exist in the optimization problem (3-4). Nevertheless in some local solutions the angles δ_1, δ_2, δ_3, were different in spite of having the same performance index. Analyzing the results, we realize that there are several angle combinations that result in the local and global performance function. Two particular local solutions ($J = 8.4990685764$ and $J = 7.1713738830$) and the global solution ($J^* = 7.0380710741$) are analyzed. The main relation among the designs with the particular performance function of 8.4990685764 is that the angle α_i, $\forall i = 1, ..3$ (see Fig. 4c) must have whatever combination of the following angles: π, $\frac{\pi}{2}$, and $\frac{\pi}{2}$ radians. The designs with the performance function of 7.1713738830 requires that the angles α_i $\forall i = 1, ...3$ (see Fig. 4b) must have whatever combination of the following angles: $\frac{3\pi}{4}$, $\frac{\pi}{2}$, and $\frac{3\pi}{4}$ radians. Finally, the designs with the global performance function of 7.0380710741 requires that the angles α_i $\forall i = 1, ...3$ (see Fig. 4a) must have $\frac{2\pi}{3}$ radians.

5. Conclusion

In this paper the formulation of the optimal design for wheel location of an omnidirectional mobile robot is proposed as an optimization problem. The empirical comparison of the SQP algorithm and eight different DE variants for this particular optimization problems is presented.



a) $J = 7.0380710741$ Global solution

b) $J = 7.1713738830$ Local solution

c) $J = 8.4990685764$ Local solution

Fig. 4: Global and local solutions for the optimal location of the omnidirectional robot wheel.

The obtained results shown that the most competitive algorithm is the "rand/2/dir", "best/1/bin" and "best/1/exp". The "DE/best/1/bin" found optimal solutions with smaller number of generations, i.e., rapid convergence towards a solution. Nevertheless "rand/2/dir" finds a better mean objective function value.

The "DE/rand/1/bin" requires more generations to find the optimal solution. So, when using the "DE/rand/1/bin" in real world optimization problem, the generation number must be carefully chosen.

The SQP algorithm presents a high sensitivity to the initial condition. So the performance of the DE variants was more consistent in the searching of the optimal solution and they are not sensitive to the initial set of solutions. However the computational time is higher than those required by the SQP algorithm.

From an engineering design point of view, a heuristic approach must be considered first and then a gradient approach in order to finely search in the complete space and hence finding the best design.

Acknowledgment

The first and fourth author acknowledges support from COFAA and SIP of the National Polytechnic Institute through project no. SIP-20110165. The second author acknowledges support from the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through project no. 084060.

References

[1] W. Dixon, D. Dawson, E. Zergeroglu, and A. Behal, "Nonlinear control of wheeled mobile robots," *Springer, Lecture Notes in Control and Information Sciences*, vol. 262, 2001.

[2] Y. Liua, J. J. Zhu, R. L. W. II, and J. Wu, "Omni-directional mobile robot controller based on trajectory linearization," *Robotics and Autonomous Systems*, vol. 56, 2008.

[3] D. Xu, D. Zhao, J. Yi, and X. Tan, "Trajectory tracking control of omnidirectional wheeled mobile manipulators: Robust neural network-based sliding mode approach," *IEEE Transactions on systems, man, and cybernetics Part B: Cybernetics*, vol. 39, no. 3, pp. 788–799, 2009.

[4] T. Yoshikawa, *Foundations of robotics*. The MIT Press, June 1990.

[5] K. Watanabe, "Control of an mnidirectional mobile robot," *International Conference on Knowledge-Based Intelligent Electronic Systems*, 1998.

[6] G. Zhao, B. Chen, and Y. Gu, "Control-structural design optimization for vibration of piezoelectric intelligent truss structures," *Struct Multidisc Optim*, vol. 37, 2009.

[7] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," *Genetic And Evolutionary Computation Conference*, pp. 485–492, 2006.

[8] S. S. Rao, *Engineering Optimization*. John Wiley & Sons, 1996.

[9] C. A. Coello-Coello, G. B. Lamont, and D. A. Van-Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.

[10] E. A. Portilla-Flores, E. Mezura, J. Alvarez-Gallegos, C. A. Coello-Coello, and C. A. Cruz-Villar, "Integration of structure and control using an evolutionary approach: An application to the optimal concurrent design of a cvt," *International Journal for Numerical Methods in Engineering*, vol. 71, no. 8, pp. 883–890, 2007.

[11] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, J. Alvarez-Gallegos, and E. A. Portilla-Flores, "Differential evolution techniques for the structure-control design of a five-bar parallel robot," *Engineering Optimization*, vol. 42, no. 6, pp. 535–565, 2010.

[12] J. Cabrera, F. Nadal, J. Muñoz, and A. Simon, "Multiobjective constrained optimal synthesis of planar mechanisms using a new evolutionary algorithm," *Mechanism and Machine Theory*, vol. 42, 2007.

[13] H.-S. Yan and G. jhih Yan, "Integrated control and mechanism design for the variable input-speed servo four-bar linkages," *Mechatronics*, vol. 19, no. 2, pp. 274–285, March 2009.

[14] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: A practical approach to global optimization*. Springer, December 2005.

[15] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," *Proceedings of the 18th International parallel and distribuited processing symposium*, 2004.

[16] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, and J. Alvarez-Gallegos, "Structure-control mechatronic design of the planar 5r 2dof parallel robot," *IEEE International Conference on Mechatronics*, 2009.

Table 1: Performance of the SQP algorithm for 100 independent runs with random initial condition. J_{mean}: Mean objective function value of individuals of 100 independent runs. Percentage J^*: Percentage of runs where the optimal solution is found (at least in one individual of the population). $MaxIter_{mean}$: the mean maximum number of iterations.

	J_{mean}	Percentage J^*	Convergence Time [s]	$MaxIter_{mean}$
SQP algorithm	7.6750633252	1%	2.83×10^{-5}	10.3

Table 2: Performance of the DE variants. J_{mean}: Mean objective function value of individuals of 100 independent runs. $\sigma(J)_{mean}$: Mean standard deviation ($J^* - J$) of individuals of 100 independent runs. Percentage J^*: Percentage of runs where the optimal solution is found. $MaxGen_{mean}$: mean maximum generation number of the 100 independent runs.

Algorithm	J_{mean}	$\sigma(J)_{mean}$	PercentageJ^* %	Convergence time [s]	$MaxGen_{mean}$
Rand/1/bin	7.0383599933	9.57×10^{-5}	0%	0.003321	300
Rand/1/exp	7.0382518471	6.66×10^{-5}	0%	0.003998	300
Best/1/bin	7.0380710746	6.54×10^{-10}	100%	0.001664	153.48
Best/1/exp	7.0380710746	6.71×10^{-10}	100%	0.002368	177.93
Current-to-rand/1	7.0403045990	0.011766	0%	0.002999	300
Current-to-best/1	7.0397307031	0.010710	99%	0.001364	135.69
Current-to-rand/1/Bin	7.0398328304	0.007471	0%	0.003274	300
Rand/2/dir	7.0380710745	3.65×10^{-10}	100%	0.002323	232.72

Table 3: Global solution and two particular local solutions found by the SQP algorithm. Units in degree and meters.

Solution	p_1	p_2	p_3	p_4	p_5	p_6	J
Global	0	60	330	0.15	0.15	0.15	7.0380710741
Local	45	360	360	0.15	0.15	0.15	7.1713738830
Local	360	360	0	0.15	0.15	0.15	8.4990685764

A Stochastic Optimization Approach for Unsupervised Kernel Regression

Oliver Kramer
Institute of Structural Mechanics
Bauhaus-University Weimar
oliver.kramer@uni-weimar.de

Fabian Gieseke
Institute of Structural Mechanics
Bauhaus-University Weimar
fabian.gieseke@uni-weimar.de

***Abstract*—Unsupervised kernel regression (UKR), the unsupervised counterpart of the Nadaraya-Watson estimator, is a dimension reduction technique for learning of low-dimensional manifolds. It is based on optimizing representative low-dimensional latent variables with regard to the data space reconstruction error. The problem of scaling initial local linear embedding solutions, and optimization in latent space is a continuous multi-modal optimization problem. In this paper we employ evolutionary approaches for the optimization of the UKR model. Based on local linear embedding solutions, the stochastic search techniques are used to optimize the UKR model. An experimental study compares covariance matrix adaptation evolution strategies to an iterated local search evolution strategy.**

I. Introduction

Unsupervised kernel regression (UKR) is the unsupervised counterpart of the Nadaraya-Watson estimator. It is based on optimizing latent variables w.r.t. to the data space reconstruction error. The evolved latent variables define a UKR solution. Projection back into data space yields the UKR manifold. But the optimization problem has multiple local optima. In the following, we employ evolutionary optimization strategies, i.e., the CMA-ES and Powell evolution strategies (Powell-ES) [11], to optimize UKR manifolds. Furthermore, we analyze the influence of local linear embedding (LLE) [18] as initialization procedure.

This paper is organized as follows. In Section II we give a brief overview of manifold learning techniques. In Section III we introduce the UKR problem by Meinicke *et al.* [13], [12]. In this section we also shortly present the standard UKR optimization approach that we are going to replace by an evolutionary framework. Section IV introduces the components of the evolutionary UKR approach. An experimental study in Section V will give insights into the evolutionary optimization process. Important results are summarized in Section VI.

II. Related Work

High-dimensional data is usually difficult to interpret and visualize. But many datasets show correlations between their variables. For high-dimensional data, a low-dimensional simplified manifold can often be found that represents characteristics of the original data. The assumption is that the structurally relevant data lies on an embedded manifold of lower dimension. In the following, we will overview famous manifold learning techniques pointing out that the overview can only be subjective depiction of a wide field of methods.

One of the most famous and widely used dimension reduction methods is principal component analysis (PCA) that assumes linearity of the manifold. Pearson [16] provided early work in this field. He fitted lines and planes to a given set of points. The standard PCA as most frequently known today can be found in the depiction of Jolliffe [8]. The PCA computes the eigenvectors, i.e., the largest eigenvalues of the covariance matrix of the data samples. An approach for learning of non-linear manifolds is kernel PCA [19] that projects the data into a Hilbert space similarly to the SVM and SVR principle. Hastie and Stuetzle [6] introduced principal curves that are self-consistent smooth curves that pass through the middle of data clouds. Self-consistency means that the principal curve is the average of the data projected on it. Bad initializations can lead to bad local optima in the previous approaches. A solution to this problem is k-segments by Verbeek, Vlassis, and Kröse [21] that alternates fitting of unconnected local principal axes and connecting the segments to form a polygonal line.

Self-organizing feature maps [10] proposed by Kohonen learn a topological mapping from data space to a map of neurons, i.e., they perform a mapping to discrete values based on neural (codebook) vectors in data space. During the training phase the neural vectors are pulled into the direction of the training data. Generative topographic mapping (GTM) by Bishop, Svensén, and Williams [3], [2] is similar to self-organizing feature maps, but assumes that the observed data has been generated by a parametric model, e.g., a Gaussian mixture model. It can be seen as constrained mixture of Gaussian, while the SOM can be viewed as constrained vector quantizer.

Multi-dimensional scaling is a further class of dimension reduction methods, and is based on the pointwise embedding of the dataset, i.e., for each high-dimensional point $\mathbf{y}_i$, a low dimensional point $\mathbf{x}_i$ is found, and for which similarity or dissimilarity conditions hold. For example, the pairwise (Euclidean) distances of two low-dimensional points shall be consistent with the high-dimensional counterparts. Another famous dimension reduction method based on multi-dimensional scaling is Isomap introduced by Tenenbaum, Silva, and Langford [20]. It is based on three steps: first, a neighborhood graph of $\mathbf{Y}$ is computed, second, the shortest distances between its

nodes $\mathbf{y}_i$ and $\mathbf{y}_j$ are computed (e.g. with Dijkstra), third, multi-dimensional scaling is applied based on the distances along the neighborhood graph that more represent curvilinear distances than Euclidean distances. The optimal embedding can be computed by the solution of an eigendecomposition.

III. Unsupervised Kernel Regression

In this section we introduce the UKR approach, regularization techniques and the Klanke-Ritter optimization approach [9].

A. Kernel Functions

UKR is based on kernel density functions $K : \mathbb{R}^d \to \mathbb{R}$. A typical kernel function is the Gaussian (multivariate) kernel:

$$K_G(\mathbf{z}) = \frac{1}{(2\pi)^{q/2}\det(\mathbf{H})} \exp\left(-\frac{1}{2}\left|\mathbf{H}^{-1}\mathbf{z}\right|^2\right), \quad (1)$$

with bandwidth matrix $\mathbf{H} = \mathrm{diag}(h_1, h_2, \ldots, h_d)$. Figure 1 illustrates that the UKR result significantly depends on the choice of an appropriate bandwidth. For a random data cloud the kernel density estimate is visualized. A too small bandwidth results in an overfitted model (left).

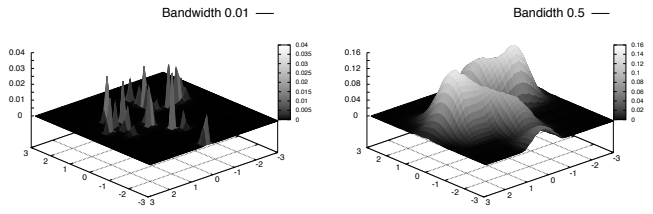


Fig. 1. Comparison of kernel density estimates with two bandwidths of a data cloud.

B. UKR Formulation

UKR has been introduced by Meinicke *et al.* [13], [12] within a general regression framework for the reversed problem of learning manifolds. The task of UKR is to find the input parameters of the regression function that are the latent variables of the low-dimensional manifold. UKR reverses the Nadaraya-Watson estimator [14], [22], i.e., the latent variables $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_N) \in \mathbb{R}^{q\times N}$ become parameters of the system, in particular $\mathbf{X}$ becomes the lower dimensional representation of the observed data $\mathbf{Y} = (\mathbf{y}_1, \ldots \mathbf{y}_N) \in \mathbb{R}^{d\times N}$. The UKR regression function can be written as follows:

$$\mathbf{f}(\mathbf{x}; \mathbf{X}) = \sum_{i=1}^{N} \mathbf{y}_i \frac{K(\mathbf{x} - \mathbf{x}_i)}{\sum_{j=1}^{N} K(\mathbf{x} - \mathbf{x}_j)}, \quad (2)$$

which is the revised Nadaraya-Watson estimator. The free parameters $\mathbf{X}$ define the manifold, i.e., the low-dimensional representation of the data $\mathbf{Y}$. Parameter $\mathbf{x}$ is the location where the function is evaluated, and is based on the entries of $\mathbf{X}$.

For convenience, Klanke *et al.* [9] introduced a vector $\mathbf{b}(\cdot) \in \mathbb{R}^N$ of basis functions that define the ratios of the kernels:

$$b_i(\mathbf{x}, \mathbf{X}) = \frac{K(\mathbf{x} - \mathbf{x}_i)}{\sum_{j=1}^{N} K(\mathbf{x} - \mathbf{x}_j)}. \quad (3)$$

Each component i of the vector $\mathbf{b}(\cdot)$ contains the relative kernel density of point $\mathbf{x}$ w.r.t. the i-th point of matrix $\mathbf{X}$. Equation (2) can also be written in terms of these basis functions:

$$\sum_{i=1}^{N} \mathbf{y}_i b_i(\mathbf{x}; \mathbf{X}) = \mathbf{Y}\mathbf{b}(\mathbf{x}; \mathbf{X}). \quad (4)$$

The matrix $\mathbf{Y}$ of observed data is fixed, and the basis functions are tuned during the learning process. The basis functions b_i sum to one as they are normalized by the denominator. The quality of the principal manifold learning is evaluated with the data space reconstruction error, i.e., the Euclidean distance between training data and its reconstruction.

$$R(\mathbf{X}) = \frac{1}{N}\|\mathbf{Y} - \mathbf{Y}\mathbf{B}(\mathbf{X})\|_F^2, \quad (5)$$

using the Frobenius norm, and with the matrix of basis functions. The Frobenius norm of a matrix $\mathbf{A}$ is defined as follows:

$$\|\mathbf{A}\|_F^2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}. \quad (6)$$

To summarize, $b_i \in \mathbb{R}$ is a relative kernel density, $\mathbf{b} \in \mathbb{R}^N$ is a vector of basis functions, and $\mathbf{B} \in \mathbb{R}^{N\times N}$ is a matrix, whose columns consists of the vector of basis functions. Hence, the product of $\mathbf{Y} \in \mathbb{R}^{d\times N}$ and $\mathbf{B} \in \mathbb{R}^{N\times N}$ (which is the Nadaraya-Watson estimator) results in a $d \times N$-matrix.

$$\mathbf{B}(\mathbf{X}) = (\mathbf{b}(\mathbf{x}_1; \mathbf{X}), \ldots, \mathbf{b}(\mathbf{x}_N; \mathbf{X})). \quad (7)$$

Leave-one-out cross-validation (LOO-CV) can easily be employed by setting the diagonal entries of $\mathbf{B}$ to zero, normalizing the columns, and then applying Equation 5.

Instead of applying LOO-CV an UKR model can be regularized via penalizing extension in latent space (corresponding to penalizing small bandwidths in kernel regression) [9]:

$$R(\mathbf{X}) = \frac{1}{N}\|\mathbf{Y} - \mathbf{Y}\mathbf{B}(\mathbf{X})\|_F^2 + \lambda\|\mathbf{X}\|_F^2. \quad (8)$$

The regularized approach will be used in the comparison between the CMA-ES and the Powell-ES in Section V-C.

C. Klanke-Ritter Optimization Scheme

Klanke and Ritter [9] have introduced an optimization scheme consisting of various steps. It uses PCA [8] and multiple LLE [18] solutions for initialization, see Section III-D. In particular, the optimization scheme consists of the following steps:

1) Initialization of $n+1$ candidate solutions are, n solution from LLE, one solution from PCA,
2) selection of the best initial solution w.r.t. CV-error,
3) search for optimal scale factors that scale the best LLE solution to an UKR solution w.r.t. CV-error,
4) selection of the most promising solution w.r.t. CV-error,
5) CV-error minimization:
   - if the best solution stems from PCA: search for optimal regularization parameters η with the homotopy method,

- if the best solution steps form LLE: CV-error minimization with the homotopy method / resiliant backpropagation (RPROP) [17],

6) final density threshold selection.

For a detailed and formal description of the steps, we refer to the depiction of Klanke and Ritter [9]. They discuss that spectral methods do not always yield an initial set of latent variables that is close to a sufficient deep local minimum. We will employ evolution strategies to solve the global optimization problem, but also use LLE for initial solutions.

D. Local Linear Embedding

For non-linear manifolds LLE by Roweis and Saul [18] is often employed. LLE assumes local linearity of manifolds. It works as follows for mapping high-dimensional data points $\mathbf{y} \in \mathbf{Y}$ to low-dimensional embedding vectors $\mathbf{x} \in \mathbf{X}$. LLE computes a neighborhood graph like for the other nonlinear spectral embedding methods, see Section II. Then it computes the weights $\mathbf{w}_{ij}$ that best reconstruct each data point $\mathbf{y}_i$ from its neighbors, minimizing the cost function:

$$R(\mathbf{w}) = \sum_{i=1}^{N} \|\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j\|^2. \tag{9}$$

The resulting weights capture the geometric structure of the data as they are invariant under rotation, scaling and translation of the data vectors. Then, LLE computes the vectors $\mathbf{y}_i$ best reconstructed by the weights $\mathbf{w}_{ij}$ minimizing

$$R(\mathbf{w}) = \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j=1}^{N} w_{ij}\mathbf{x}_j\|^2 \tag{10}$$

For a detailed introduction to LLE we refer to [18], and Chang and Yeung [4] for a variant robust against outliers. A free parameter of LLE is the number of local models, which can reach from 1 to N. LLE is employed as initialization routine. The best LLE solution (w.r.t. the CV-error of the UKR manifold) is used as basis for the subsequent stochastic CV-error minimization.

IV. Evolutionary Unsupervised Kernel Regression

A. Evolutionary Optimization Scheme

We employ the CMA-ES, and the Powell-ES to solve two steps of the UKR optimization framework. Our aim is to replace the rather complicated optimization scheme we briefly summarized in Section III-C. It consists of the following steps:

1) Initialization of n candidate LLE solutions,
2) selection of the best initial solution $\hat{\mathbf{X}}^*_{init}$ w.r.t. CV-error,
3) search for optimal scale factors

$$\mathbf{s}^* = \arg\min_s R_{CV}(\mathrm{diag}(\mathbf{s})\hat{\mathbf{X}}^*_{init}) \tag{11}$$

   with CMA-ES/Powell-ES, and
4) CV-error minimization with CMA-ES/Powell-ES.

We employ Huber's loss function [7], see Section IV-B, for the following experiments. The subsequent sections describe an experimental analysis of the evolutionary UKR approach. A side effect of the use of an evolutionary scheme is that arbitrary, also non-differentiable kernel functions can be employed. LOO-CV can easily be implemented by setting the diagonal entries of $\mathbf{X}$ to zero, and normalizing the columns, and then applying Equation (5). In the following, we compare two optimization approaches for optimizing the UKR model: (1) Powell's conjugate gradient ES [11], and (2) the CMA-ES [15].

B. Huber's Loss

In regression typically different loss function are used that weight the residuals. In the best case the loss function is chosen according to the needs of the underlying data mining model. With the design of a loss function, the emphasis of outliers can be controlled. Let $L : \mathbb{R}^q \times \mathbb{R}^d \to \mathbb{R}$ be the loss function. In the univariate case $d = 1$ a loss function is defined as $L = \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i))$. The L_1 loss is defined as

$$L_1 = \sum_{i=1}^{N} \|y_i - f(\mathbf{x}_i)\|, \tag{12}$$

and L_2 is defined as

$$L_2 = \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2. \tag{13}$$

Huber's loss [7] is a differential alternative to the L_1 loss, and makes use of a trade-off point δ between the L_1 and the L_2 characteristic, i.e.:

$$L_H = \sum_{i=1}^{N} L_h(y_i - f(\mathbf{x}_i)), \tag{14}$$

and:

$$L_h(r) = \begin{cases} \frac{1}{2\cdot\delta} r^2 & |r| < \delta \\ |r| - \frac{1}{2}\delta & |r| \geq \delta \end{cases} \tag{15}$$

Parameter δ allows a problem specific adjustment to certain problem characteristics. In the experimental part we use the setting $\delta = 0.01$.

V. Experimental Study

A. Datasets and Fitness Measure

The experimental analysis is based on the following datasets:

- *2-D-S*: Noisy "S": 100 data points, $d = 2$, noise magnitude $\sigma = 0.1$, and 1,000 test points without noise,
- *3-D-S*: Noisy "S": 100 data points, $d = 3$, noise magnitude $\sigma = 0.1$ and 1,000 test points without noise,
- *digits 7*: 100 samples with $d = 256$ ($16 \cdot 16$ greyscale values) of figure 7 from the *digits* dataset [5], and 250 test samples.

The test error is computed by the projection of 1000 test points uniformly generated in latent space, and mapped to data space. The test error is the sum of distances between each test point

$\mathbf{x}_t \in \mathcal{T}$, and closest projection and the sum distances between each projected point $\mathbf{x}_p \in \mathcal{P}$ and the closest test point:

$$R_t = \sum_{\mathbf{x}_t \in \mathcal{T}} \min_{\mathbf{x}_p \in \mathcal{P}} \|\mathbf{x}_p - \mathbf{x}_t\| + \sum_{\mathbf{x}_p \in \mathcal{P}} \min_{\mathbf{x}_t \in \mathcal{T}} \|\mathbf{x}_p - \mathbf{x}_t\| \quad (16)$$

For training (validation) and test measurement of residuals Huber's loss function is employed, see Section IV-B.

B. LLE Initialization and Scaling Balance

Initialization with LLE is part of the optimization scheme introduced by Klanke and Ritter [9]. The question arises, how much optimization effort should be invested into the scaling in comparison to the CV-error minimization process. In the following, we analyze the balance of optimization effort for both optimization steps with a budget of 4,000 optimization steps. Table I shows the UKR CV-errors (1) of the best LLE model, (2) after scaling optimization with the CMA-ES, (3) after the final CV-error minimization, and (4) the error on the test set. We test five optimization balances, the first number indicates the number of steps for the LLE scaling optimization, the second number states the number of steps for the latent variable-based optimization. We test the following combinations: (0/4) meaning no scaling, 4,000 generations of final CV minimization, and the balances (1/3) meaning 1,000 scaling and 3,000 CV, (2/2) meaning 2,000 scaling and 2,000 CV, (3/2) meaning 3,000 scaling and 1,000 CV, and finally (4/0) meaning 4000 scaling and no final CV optimization. The values shown in Table I present the best test error of 100 runs. The results show that the (2/2) variant achieves the lowest rest error, while the (0/4) variant achieves the lowest training error.

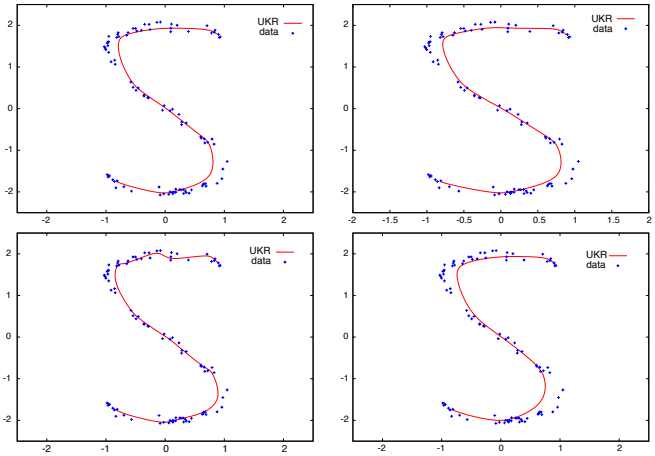


Fig. 2. Evolutionary UKR on noisy $\mathcal{S}$ dataset. The figures show the results of various optimization efforts spent on scaling and final CV-error minimization, i.e., 0/4, 2/2, 3/1, and a regularized approach with $\lambda = 0.25$, see Equation (8).

The projection is used for visualization. Figure 2 gives a visual impression of the evolved UKR manifolds. The original data is shown by (blue) spots, the manifold is drawn as a (red) line. With the help of the test set, we compute a test error, see Equation (16).

TABLE I
EXPERIMENTAL ANALYSIS OF OPTIMIZATION STEPS INVESTED INTO SCALING OF THE LLE SOLUTIONS, AND CV-ERROR MINIMIZATION (MINIMAL VALUES).

balance	0/4	1/3	2/2	3/1	4/0
LLE	7.23733	7.23733	7.23733	7.23733	7.23733
scaling	7.23733	3.02818	2.19235	2.10442	1.89369
CV	1.80625	1.95179	2.04789	2.07737	1.89369
test	0.00276	0.00225	**0.00195**	0.00313	0.00329

C. CMA-ES and Powell-ES

In the following, we compare two optimization algorithms, i.e., the CMA-ES and the Powell-ES (also known as Powell-ILS, see [11]) as optimization approaches for the UKR learning problem. It is based on Powell's fast local search. To overcome local optima the Powell-ES makes use of a $(\mu+\lambda)$-ES [11]. Each objective variable $\mathbf{x} \in \mathbb{R}$ is mutated using Gaussian mutation [1] with a step-size (variance σ), and a Powell-search is conducted until a local optimum is found. The step-sizes of the ES are mutated as follows: if in successive iterations the same local optimum is found, the step-sizes are increased to overcome local optima. In turn, if different local optima are found, the step-sizes are decreased.

Table II compares the two optimizers on three artificial datasets, using the penalized UKR variant, see Equation (8) with $\lambda = 0.1$. The values show the test error, i.e., the distance between the original data to the projections of 1000^d samples after 4000 fitness function evaluations, i.e., 2000 steps of scale optimization, and 2000 steps of CV error minimization. The experiments show that the Powell-ES achieves a lower training error in each of the experiments. But only on the problems *2D-S* this is reflected in a lower test error. This means that on *3D-S* and *digits* overfitting effect occurred that could not been prevented with the penalty regularization approach. A deeper analysis of the balancing parameter λ will be necessary.

TABLE II
EXPERIMENTAL COMPARISON OF CMA-ES AND POWELL-ES ON THREE DATASETS *2D-S*, 3D-2 AND *digits*.

data	**CMA-ES**			**Powell-ES**		
	train	test	max	train	test	max
2D-S	9.4408	0.0303	0.3845	8.7854	0.0205	0.2241
3D-S	17.7041	0.1351	1.0351	15.8019	0.2296	1.4088
digits	51.9710	0.0147	0.3991	48.3865	0.0168	0.5242

VI. CONCLUSIONS

The multi-modal optimization problem of UKR can be solved with the CMA-ES or the Powell-ES leading to an easier optimization framework that is also capable of handling non-differentiable loss and kernel functions. However, initial LLE solutions still improve the optimization process. Overfitting effects might occur that have to be avoided by improved regularization approaches. For this sake further search has to be invested into parameter λ, e.g., employing grid-search.

In the future we plan to balance regularization with multi-objective optimization techniques.

REFERENCES

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[2] C. M. Bishop, M. Svensén, and C. K. I. Williams. Developments of the generative topographic mapping. *Neurocomputing*, 21(1-3):203–224, 1998.

[3] C. M. Bishop, M. Svensén, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[4] H. Chang and D. Yeung. Robust locally linear embedding. *Pattern Recognition*, 39:1053–1065, 2006.

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, Berlin, 2009.

[6] Y. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 85(406):502–516, 1989.

[7] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[8] I. Jolliffe. *Principal component analysis*. Springer series in statistics. Springer, New York u.a., 1986.

[9] S. Klanke and H. Ritter. Variants of unsupervised kernel regression: General cost functions. *Neurocomputing*, 70(7-9):1289–1303, 2007.

[10] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[11] O. Kramer. Fast blackbox optimization: Iterated local search and the strategy of powell. In *Proceedings of the 2009 International Conference on Genetic and Evolutionary Methods*, 2009.

[12] P. Meinicke. *Unsupervised Learning in a Generalized Regression Framework*. PhD thesis, University of Bielefeld, 2000.

[13] P. Meinicke, S. Klanke, R. Memisevic, and H. Ritter. Principal surfaces from unsupervised kernel regression. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1379–1391, 2005.

[14] E. Nadaraya. On estimating regression. *Theory of Probability and Its Application*, 10:186–190, 1964.

[15] A. Ostermeier, A. Gawelczyk, and N. Hansen. A derandomized approach to self adaptation of evolution strategies. *Evolutionary Computation*, 2(4):369–380, 1994.

[16] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[17] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *In Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[18] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[19] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[20] J. B. Tenenbaum, V. D. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[21] J. Verbeek, N. Vlassis, and B. Kröse. A k-segments algorithm for finding principal curves. *Pattern Recognition*, 23:1009–1017, 2002.

[22] G. Watson. Smooth regression analysis. *Sankhya Series A*, 26:359–372, 1964.

A methodical study for the Extraction of Landscape Traits using Membrane Computing Technique

Daya Gupta[1], Bidisha Das[1], V.K.Panchal[2]
[1]Department of Computer Engineering, Delhi Technological University, Delhi, India
[2]Defence Terrain Research Laboratory, Defence Research & Development Oranization, Delhi, India

Abstract - *The aim of this paper is to propose a systematic overview of membrane computing model in land cover feature extraction. Membrane computing is a new branch of natural computation which has a great deal of distribution and handles maximal parallelism. The bio-inspired technique is used for image classification and these images are remote sensing satellite image. The terrain features like water, barren, rocky, vegetation and urban are needed to be classified as their information provides immense support during natural disaster, climatic behavioral changes and in other areas of environmental changes. Though membrane computing has been applied to wide field of biology, computer science but Land cover Classification is never taken into consideration. We have defined parameters of membrane computing in land cover feature extraction terms thus showing that the idiom of membrane computing is useful for image classification and then we have proposed our algorithm of P system. The proposed algorithm is applied to Alwar region of Rajasthan of 472X576 dimension which contains 7 Band Indian Resourcesat Satellite Digital Numbers. The algorithm has captured almost all the terrain features of these region. It shows almost 99% efficiency on water and vegetation region. The KHAT statistics shows that the proposed algorithm has an overall efficiency of 0.68812.*

Keywords: Membrane Computing, Natural Computation, Remote Sensing, Feature Extraction, Image Classification, Maximal Parallelism.

1 Introduction

Image classification plays an important role in remotely sensed data. Often image classification is applied to remotely sensed landuse data to generate a map for image analysis. Thus, it is used as an examination tool for the digital data. Image classification is one of the important approaches for recognizing different terrain features. The analysts determines which classification method meets his specific task. At present, numerous techniques like Evolutionary algorithms, Artificial Neural Network, Ant Colony Optimization, Particle Swarm Optimization and Biogeography Based Optimization are being applied to image classification. These natural computing technique can even be termed as nature inspired algorithm. Therefore each analysts decides which classifier is appropriate for the task in hand.

Remote sensing [2] refers to the technology of acquiring information about the earth's surface features (land and water) and atmosphere using space-borne platforms. Satellite remote sensing has been recognized as a valuable tool for viewing, analyzing, characterizing, and making decisions about our environment [3]. Multi-spectral images capture different terrain features like water, urban, rocky, vegetation, barren which are being classified for further requirement for image analysis. Thus image classification and remote sensing are inter-twined to each other. Though many natural Computation techniques have already been introduced as a classifier, membrane Computing which also comes under natural Computation very recently introduced in this category.

Membrane Computing [4] deals with computing models abstracted from the structure and the functioning of living cells, as well as it is the study of organization of cells in tissues or higher order structure. Each living cells is being compartmentalized by membranes. Each cell is encapsulating the features within itself. A membrane consists of hierarchical structure of cell-like compartments or regions. These bio-inspired mechanism proceeds in a maximally parallel manner by the non-deterministic choice of application of the transformation rules, as well as of the objects to which they are to be applied. Membrane Computing often generically termed as P system. It has been applied to wide range of biological field, computer science areas like software requirement [5] and even to linguistics.

With the help of satellite image various features of our nature can be identified. The general features on which scientists generally work are water, barren, urban and vegetation areas. We have worked on Alwar region of Rajasthan as shown in figure 1. Indeed other features are being recognized but these are commonly available in any kind of land areas. Thus land covers all these common type of features. During feature extraction we attempt to identify all those features that land is consisting of. As our dataset is Alwar region of Rajasthan thus we have depicted its image to understand land cover feature.

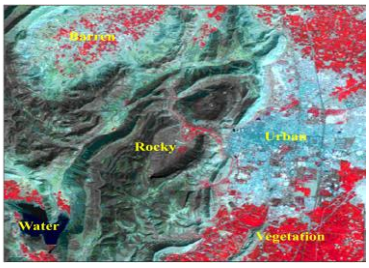


Figure 1. Different features of Alwar region.

The paper tried to focus on extracting the natural terrain features like urban, vegetation, rocky, barren and water from satellite image using membrane computing. Though water has been already extracted through membrane computing [6] but these paper classified the image into other different terrain features also and shows how membrane computing classifies the homogeneous, heterogeneous and sparse regions of the specified areas.

The organization of the paper is as follows: The paper is divided into VII sections. Section II presents a brief introduction to membrane computing. Section III presents land cover feature extraction interpreted in membrane computing terms. Section IV presents the proposed scaffold for the membrane computing technique for land cover feature extraction – the dataset, characteristics, the proposed architecture, the algorithm. Section V presents the accuracy assessment of the proposed technique by analyzing the KHAT statistics. Section VI presents the classification results of the Alwar region in Rajasthan using Membrane computation. Section VII presents conclusion and future scope of the proposed work.

2 A Glimpse on Membrane Computing

Membrane was first introduced by Gheoghe Păun [7][8] in the year 1998. Membrane computing came into existence when structure of living cells are considered as "protected reactors". The compartments have multisets of chemicals (called objects) which follow some evolution rules. Each cell has multisets of objects which follow some evolution rule to decide the next state. These rules either follow some rewriting rules or inspired from biological processes, such as objects can pass through membrane depending on symport or antiport condition or based on communication rules.

Membrane is considered as a three- dimensional vesicle which is covering a Eucledian space. Membrane separates the "inside" space from the "outside" space. It can be represented graphically as two- dimensional plane using Euler-Venn Diagram [7]. Figure 2 is a structure of membrane computing. Region 5 is elementary membrane and region 3 is the lower neighbor consists both region 4 and 5 which are therefore siblings. The space outside the skin membrane is known as environment.

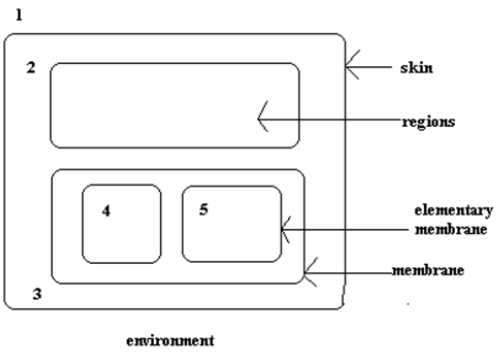


Figure 2. Membrane structure

Parenthesis expression for membrane computing is another standard form in these literature. Thus for example the membrane structure S from figure 2 can be represented as parenthesis form. Thus figure 3 depicts it.

$$[\,[\,]_2[\,[\,]_4[\,]_5\,]_3\,]_1$$

Figure 3. Parenthesis representation of membrane computing

When it is represented in tree form then figure 4 illustrates it,

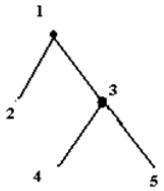


Figure 4. Tree representation

In the cell certain biochemical reactions often take place and those reactions take the form as u→v where u and v are strings(representing multiset of objects from a given set O). The rule u→v, has [u] as the weight of the rule, a rule of weight atleast two is cooperative whereas a rule of weight 1 is called non-cooperative. Catalytic rules are cooperative rules of the form ca→cv. The rule of the form u→v, where u is a string over O, and v is a string over Ox{here,in,out}. Each element of v is of the form (a,tar) where tar is target indication.

i) If tar = here, then a does not move to any neighbouring region and it resides along with the region where the rule reside.

ii) If tar = out, then a is sent to the upper neighbour of the region where the rule resides.

iii) If tar = in, then a is sent non-deterministically to one of the lower neighbours.

Thus a P system [7] (with multisets rewriting rules) of degree m>=1 is a construct

$\Pi = (O, H, \mu, w_1,\ldots\ldots,w_m,R_1,\ldots\ldots,R_m,i_0)$.

where :

1. O is the alphabet of objects;
2. H is the alphabet of membrane labels;
3. μ is a membrane structure of degree m;
4. $w_1,\ldots..,w_m \in O^*$ are the multisets of objects associated with the m regions of μ;
5. R_i, $1 \leq i \leq m$, are there finite sets of multiset rewriting rules associated with the m regions of μ;
6. $i_0 \in H \cup \{e\}$ specifies the input/output region of Π, where e is a reserved symbol not in H.

3 Land Cover Feature Extraction Interpreted in Membrane Computing terms

The aim of this section is to provide an idea how components of membrane computing has been easily interpreted in terms of land cover features. Though it tries to give some light in these areas but a more systematic and detailed study remains to be elaborated. The

membrane computing parameters [9] for land cover feature extraction algorithm are defined as follows :-

- Object : The multi-spectral band [10] of image consists huge number of pixels. Once a rule is applied to object it moves between adjacent/neighbouring cells. Thus pixel focuses on objects of membrane computing.Thus objects $\in$ C is an integer and C $\in$ [0,255136].
- Membrane : Each pixel can move from one cell to another which are being compartmentalized through membrane. Stated otherwise, membrane delimited inside from outside and this separation means that each membrane represents 5 specific features i.e. rocky, barren, water, urban, vegetation that are to be extracted from the image. Each membrane has its own object processing rule.
- Membrane structure : The framework of hierarchical structure where Region 1 is has 5 sub-membranes. It contains region 2,3,4,5 and 6 which represents each identified features of the Alwar region, i.e. region 2 is for urban, 3 for water, 4 for vegetation, 5 for barren and 6 represents rocky features. Each of the sub-membranes are elementary membrane. Figure 5 shows depicted the hierarchical structure.

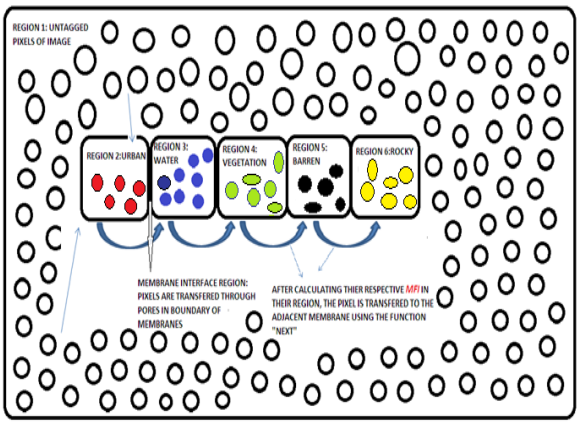


Figure 5. Membranous structure of Alwar Region [6]

- Environment : The outside of the skin membrane is considered as empty. Thus only objects which are unclassified in nature are thrown outside. The pixels which cannot be classified are thrown in the environment.
- Multiset Rewriting Rules : Depending on the evolution rule which are cooperative in nature, pixels can be absorbed by different membrane. The Region 1 of figure 5 consists of multisets of objects as $a^{112496}b^{83239}c^{49049}d^{10395}e^{2540}$ where a represents barren pixels, b represents rocky, c represents vegetation, d represents urban and e represents water pixels. Thus a^{112496} means the image has 112496 pixels which are barren in nature and so forth. All these values are computed after classification. Since each feature membrane i.e region 2, 3, 4, 5, 6 consists homogeneous pixels thus multiset of each feature membrane can be represented as :

i) Water^{548} (Region 3)
ii) Barren^{3157} (Region 5)
iii) Vegetation^{3049} (Region 4)
iv) Urban^{2609} (Region 2)
v) Rocky^{5164} (Region 6)

- Symport rule: They can represent input into a membrane or ejection out of the membrane of pixels. If the rule condition is satisfied it takes the form (pixel, in$_k$) where k is the region (water, rocky, barren, vegetation, urban) otherwise it is of the form (pixel, out).
- Promoter/inhibitor : The rules can only be applied under certain conditions. Depicting the barren condition for promoter case which can be applied to other feature membrane also can be written as:-

$$(\text{barren})^p \rightarrow (\text{barren})^{p+1} \mid \text{Max(corr (barren, pixel\_value))}$$

While the inhibitor case is as follows:

$$(\text{barren})^p \rightarrow (\text{barren})^{p+1} \mid \text{Min(corr (barren, pixel\_value))}$$

In either case we take the maximum or minimum value of correlation between barren value and each pixel value. Here p represents the number of pixels which are classified as barren till now and if the promoter condition is met then the pixel is absorbed in barren region.

- Maximal Parallelism : It means the rule should be applied in parallel to the maximum degree. The multiset of objects present in Region 1 as in figure 5 is w= $a^{112496}b^{83239}c^{49049}d^{10395}e^{2540}$ and the set of rules R residing in the same region is R = {r_1 , r_2, r_3, r_4, r_5}. The definition of these rules are given at Proposed Architecture.

Thus land cover feature extraction P system transition system can be defined as follows:

$\Pi = (O, H, \mu, w_1, r_1, r_2, r_3, r_4, r_5, i_0)$,

where:

1. O ={a,b,c,d,e}.
2. H ={1,2,3,4,5,6}
3. μ is membrane structure represented in parenthesis form as:
   [[ ]$_2$ [ ]$_3$ [ ]$_4$ [ ]$_5$ [ ]$_6$]$_1$
4. $w_1 = a^{112496}b^{83239}c^{49049}d^{10395}e^{2540}$
5. r_i, i=1 to 5, already defined while discussing maximal parallelism.
6. i_0 = {2, 3, 4, 5, 6}.

4 Proposed Scaffold For Membrane Computing Technique for Land Cover Feature Extraction

In this section 4 components have been described. The data on which we had our experiment, description of Alwar area, the architecture and algorithm which we have proposed.

4.1 Dataset

To fulfil our objective of image classification using membrane computing we have used multi-spectral, multi-resolution and multi-sensor image which has dimension of 472 X 576. The image is of Alwar Region of Rajasthan. With the use of remote sensing the expert have captured satellite image of 7 different bands. The Bands are as : Red, Green, Radarsat -1(RS1), Radarsat -2 (RS2), Near Infra-Red(NIR), Middle Infra-Red(MIR) and Digital Elevation Model(DEM). LISS(Linear Imaging Self Scanning Sensor) –III and sensor of resourcesat an Indian remote sensing satellite give us Red, Green, NIR and MIR band images. RS1 and RS2 are from Canadian satellite Radarsat whereas Digital elevation model is derived by using images of RS1 and RS2. All the seven bands of Alwar region is shown in figure 6.

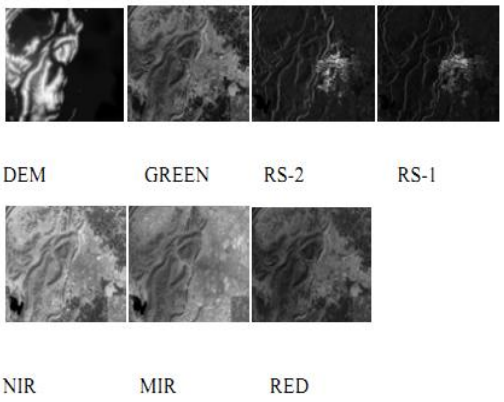


Figure 6. Images of 7 band of Alwar Region

4.2 Characteristics

Alwar region is geographically situated at Latitude 27˚ 34' North and Longitude 76˚ 35' East at an elevation of 270 meters above sea level. Most part of Alwar region is covered with Rocky and Vegetation areas whereas a low proportion of water and urban area respectively

4.3 Proposed Architecture

Before applying Membrane Computing in image classification the following computations along with some definitions are to be performed:

1.Create a excel sheet which contains the x-coordinate ,y-coordinate and digital numbers of all the bands.

2.For each pixels/objects i.e. [0,255136] we have pixel_value=[P_1 P_2 P_3 P_4 P_5 P_6 P_7] where P_1 is the value of digital numbers of red band, P_2 is the value of digital numbers of green band, P_3 is the value of digital numbers of NIR band, P_4 is the value of digital numbers of MIR band, P_5 is the value of digital numbers of RS1 band, P_6 is the value of digital numbers of RS2 band, P_7 is the value of digital numbers of DEM band.

3.The expert has given us the training set (i.e. water, barren, urban, rocky vegetation). These features are being observed by our expert during their experiments in the Alwar region. From the training set we compute the mean of each attribute of each land cover feature. For example we get the mean of water(i.e. a land cover feature) as water_mean =[M_1,M_2 ,M_3 ,M_4 ,M_5 ,M_6 ,M_7] where M_1 is the mean of red band values, M_2 mean of NIR band value, M_3 mean of MIR band value, M_4 mean of RS1 band value, M_5 mean of RS2 band value, M_6 mean of green band value, M_7 mean of DEM band value

4.Depending on the application of image classification the following evolution rules are defined :

$r_1 : a^5 b^3 d \rightarrow v_1$

where v_1:

$\{(d, in_k) , (a^5 b^3, here)\} \mid k=\{ \forall\ max\{C(i,PV)\}\ \&\ i=1\ \ k=urban$

$r_2 : a^4 e c^4 \rightarrow v_2$

where v_2:

$\{(e, in_k) , (a^4 c^4, here)\} \mid k=\{ \forall\ max\{C(i,PV)\}\ \&\ i=1\ \ k=water$

$r_3 : c d^5 \rightarrow v_3$

where v_3:

$\{(c , in_k) , (d^5, here)\} \mid k=\{ \forall\ max\{C(i,PV)\}\ \&\ i=1\ \ k=vegetation$

$r_4 : a d^9 e^2 \rightarrow v_4$

where v_4:

$\{(a, in_k) , (d^9 e^2, here)\} \mid k=\{ \forall\ max\{C(i,PV)\}\ \&\ i=1\ \ k=barren$

$r_5 : b c^7 \rightarrow v_5$

where v_5:

$\{(b, in_k) , (c^7, here)\} \mid k=\{ \forall\ max\{C(i,PV)\}\ \&\ i=1\ \ k=rocky$

where :

{ PV =pixel_value of a, b, c, d, e, C=correlation function, i= all the features values i.e. water_mean, barren_mean ,vegetatuin_mean, rocky_mean, urban_mean of Alwar region}

5.Then use all the evolution rules i.e r_1 ,r_2, r_3, r_4, r_5 to the pixels residing in Region 1 of figure 5 to decide which land cover poperty each pixel belongs to. This is done in Matlab [11].

- Any number of evolution rules are to be applied synchronously.
  Max{correlation(feature_value,pixel_value)}
  Therefore the evolution rule returns the feature which correlates with the pixel the most and

correspondingly the pixel is absorbed in that feature/region.

Here Pearson's correlation coefficient[12] has been used. The formula has been given in equation 1.

$$\rho(x,y) = \frac{\mathrm{E[xy]}}{\sigma(x)\sigma(y)} \qquad (1)$$

where x and y are two zero-mean real-valued random variable, E[xy] is the cross-correlation between x and y, and $\sigma_x^2=E[x^2]$ and $\sigma_y^2=E[y^2]$ are the variance of the two variables x, y respectively.

In our approach x represents the value of the attributes of the pixel and y represents the mean value of the attribute of the training set.

4.4 Algorithm

1.Place all the unclassified pixels/objects in one class (Reg 1).

2. Create sub-membranes for each land cover features (for water, barren, vegetation, rocky,urban).

3. Find out the mean of each sub-membrane/training set i.e. Barren_Mean, Urban_Mean, Water_Mean, Vegetation_Mean, Rocky_Mean.

4. while(no_of_unclassified_objects ≠ NULL)
{

i) Take any combinations of evolution rules (e.g. r_1r_4, r_3r_5, $r_2r_5r_1$) [To achieve maximal parallelism no. of evolution rules taken must be greater than or equal to 2]

ii) Find out object(pixel_value) depending on considered evolution rules. [If r_1 and r_3 are to be applied parallel then find out the value of any randomly taken 15 pixels. r_1 has a^5b^3d and r_3 has cd^5 multisets of objects. Thus r_1 contributes 9 pixels and r_3 6 pixels. Total of 15 (9+6) pixels]

iii) Apply all the considered evolution rules parallel on randomly taken objects/pixels.

iv) If (evolution rule is satisified)
Then absorb the pixel in the specified feature
Else
Leave the pixels in the same region i.e. Region 1.

v) No_of_unclassified_objects - -

}

5. End of program.

The flowchart of the working of membrane computing is depicted in figure 7. All the steps clearly describe the model of membrane computing which we have adopted in our image classification mechanism.

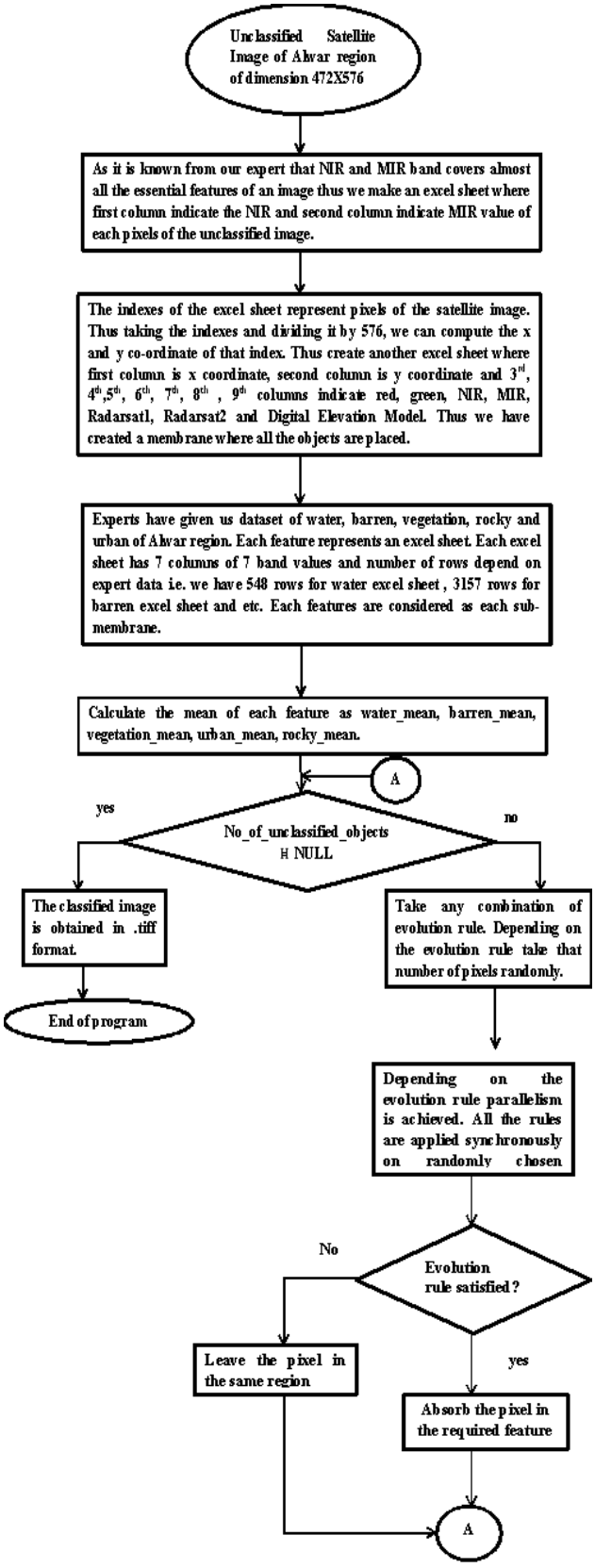


Figure 7 Flowchart of our proposed algorithm

5 Accuracy Assessment of the Proposed Technique

The aim of this step is to determine quantitatively that how pixels are correctly absorbed in their required features.

To determine our accuracy assessment we need to find the error matrix. Error matrices compare, on category-by category basis, the relationship between known reference data(ground truth) and the corresponding results of an automated classification [10]. From our experiments we have came with following number of pixels in each region i.e.

i) 150 vegetation pixels.
ii) 190 urban pixels.
iii) 200 rocky pixels
iv) 70 water pixels,
v) 170 barren pixels

The bar graph in figure 8 shows that the region is highly concentrated with rocky region, whereas density decreases incase of vegetation, urban and for barren region.Water is sparsely populated in Alwar region.

From the training set and the error matrix which is obtained after implementing membrane computing is shown in Table I.The error matrix's interpretation along column suggests how many pixels are classified correctly by our algorithm. The diagonal elements means no. of properly classified pixels in that category. The error matrix which we have obtained shows high values for vegetation and rocky but urban values need to be explored. Water is almost perfectly classified which can be inferred from our experiment.

Table I. Error Matrix when membrane computing is applied.

	Vegetation	Urban	Rocky	Water	Barren	total
Vegetation	146	24	1	0	10	181
Urban	0	24	0	0	2	26
Rocky	0	0	120	0	33	153
Water	0	0	0	69	0	69
Barren	0	55	19	0	89	163
Total	146	103	140	69	134	592

The table shows it has classified the Water pixels with almost 99% efficiency(with minimal omission errors), in case of vegetation we have achieved with minimal omission error(non-ideal classification) thus we can say membrane computing is a good classifier for water and vegetation pixels. For water a zero commission error is achieved whereas vegetation has got commission error of 31 in 181.

The Kappa coefficient [2] is usually considered for evaluating the classifier's accuracy in remote sensing domain. The Kappa(K) coefficient of the Alwar image is 0.68812.

The bar graph in figure 8 shows that our classified pixels has high concentration of vegetation and rocky area as it has been portrayed in our original image. The urban features need to be explored.

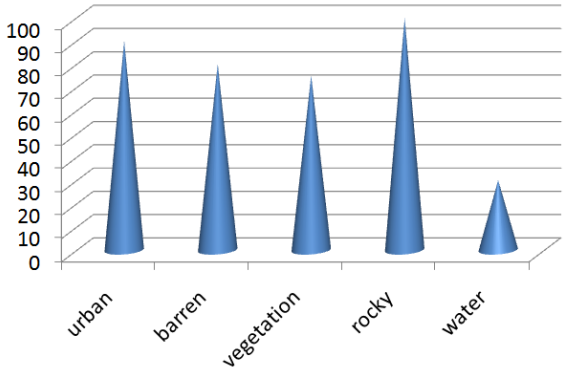


Figure 8. Density of each feature in Alwar region

6 Result & Discussion

We have observed that almost 99% efficiency is achieved for extracting water and vegetation using membrane computing technique. After applying our natural computation technique i.e. membrane computing we have got the classified image shown in figure 9. From the figure it is clearly shown that almost all regions are correctly classified. The yellow, black, blue, green, red colour represents rocky, barren, water, vegetation and urban region respectively.

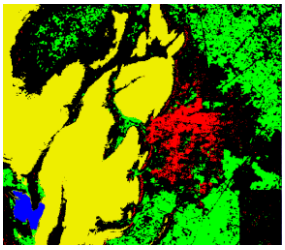


Figure 9. Classified Alwar Image

Membrane computing has almost perfectly classified all the 5 land cover features of Alwar region. The graph in figure 10 is plotted which represents the percentage on the number of pixels of the image which are classified as water, barren, vegetation, rocky and urban respectively.

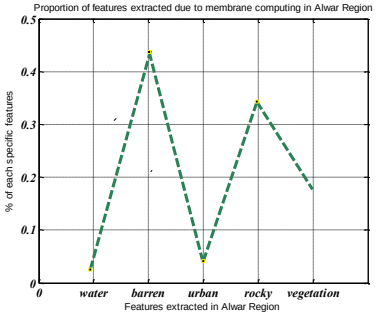


Figure 10. Plot of terrain features of Alwar region

7 Conclusion & Future Work

A large number of soft computing techniques have been already applied to remote sensing satellite image. All these techniques have efficiently classified the geospatial features of the terrain despite its discrepant uncertainties. Membrane computing is an emerging area which had hardly have any impact on image classification. Thus terrain features classification is taken as a case study. It is perceived that the Kappa coefficient can be considered as a well founded metric for assessing accuracy of classification in remote sensing.

A novel approach for feature extraction from high resolution multi- spectral satellite image is presented in this paper. From the Kappa coefficient it has been observed that vegetation region has been classified with almost zero omission error (columnwise) which lacks by only 4 pixels. Membrane computing has achieved almost 99% omission error in water region and it has been proved that it is equally at par with other soft computing technique. As membrane computing follows maximal parallelism, even the time complexity is reduced than other soft computing algorithm.

Though the algorithm has proved efficiently for water and vegetation region but it lacks in urban areas. It has got huge omission errors. This can be taken as a future work so that the algorithm can classify all regions perfectly and even the efficiency may be improved. It has perfectly classified homogeneous region but heterogeneous and sparse region is hardly classified. As it work on pixel by pixel rather than as a cluster it can be proved as a better classifier than other when a region has lot of mixed pixels. In future it may be explored for heterogeneous areas as it work on a single unit of object.

8 References

[1] V.K.Panchal, Samiksha Goel, Mitul Bhatnagar, "Biogeography Based Land Cover Feature Extraction", VIII International Conference on Computer Information Systems and Industrial Management (CISIM), Coimbatore, December, 2009.

[2] Ralph W. Kiefer, Thomas M. Lillesand, "Remote Sensing and Image Interpretation", John Wiley, New York, 2000.

[3] Zhou, Wei, "Survey and Analysis of Land Satellite Remote Sensing applied in Highway Transportation Infrastructure and System Engineering", IEEE International Geoscience and Remote Sensing Symposium, 2008.

[4] Lila Kari, Grzegorz Rozenberg, "The Many Facets of Natural Computing", Communication of the ACM, vol. 51, no. 10, October 2008.

[5] Velayutham Pavanasam, Chandrasekaran Subramaniam, Thulukkanam Srinivasan, Jitendra Kumar Jain D., "Membrane Computing Model for Software Requirement Engineering", IEEE, 2nd International Conference on Computer and Network Technology, 2010.

[6] Bidisha Das, Nitish Gupta, V.K.Panchal, Vartika Singh, "Membrane computing for water body area extraction in Satellite Image", 5th International MultiConference on Intelligent Systems & Nanotechnology (IISN), February, 2011.

[7] Gheorghe Păun, " An Introduction to and an overview of Membrane Computing", Chapter 1, 1998.

[8] Gheorghe Păun, Mario J. Pérez-Jiménez, "Membrane Computing: Brief Introduction, recent results and applications", Sciencedirect, Biosystems, vol. 85, Issues 1, pg. 11-22, July 2006.

[9] Gheorghe Păun, Radu A. Păun, "Membrane Computing as a Framework for Modeling Economic Processes", IEEE Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing(SYNASC), 2005.

[10] Lavika Goel, V.K.Panchal, Daya Gupta, "Embedding Expert Knowledge to Hybrid Bip-Inspired Techniques- An Adaptive Strategy towards Focussed Land Cover Feature Extraction", International Journal of Computer Science and Information Security(IJCSIS), vol. 8, no. 2, 2010.

[11] The Matlab Version 7,The MathWorks, Inc.

[12] Jacob Benesty, Jingdong Chen, Yiteng(Arden) Huang, "On the Importance of the Pearson Correlation Coefficient in Noise Reduction", IEEE Transactions on Audio, Speeach And Language Processing , vol. 16, no. 4, May 2008.

Quantifiable Metrics for Complex Emergence in Spatial Agent-Based Models

K.A. Hawick and C.J. Scogings
Computer Science, Institute for Information and Mathematical Sciences,
Massey University, Albany, P.Bag 102-904 NSMC, Auckland, New Zealand
{k.a.hawick, c.scogings}@massey.ac.nz
Tel: +64 9 414 0800 Fax: +64 9 441 8181

Abstract

Spatial animat agents can be used to construct sophisticated spatially-rich macroscopic models to study complex and emergent phenomena based only on localised microscopic control parameters. We have developed a predator-prey based animat model that we have successfully used to explore various collective behaviours. In studying collective phenomena using computer models it is important to develop quantifiable metrics and measurement apparatus in tandem with the model itself. We discuss some of the macroscopic metrics and statistical measurement approaches we have used to relate localised animat parameters to the emergent patterns of behaviour identified in our system.

Keywords: spatial animat; complexity; emergence; quantifiable metric.

1 Introduction

Complexity and emergence [1] are two deep concepts relevant to understanding the behaviour of multi-agent systems. One approach to exploring these concepts quantitatively is to model a particular system and establish specific metrics based upon the macroscopic system properties and to try to relate these back to the model control parameters.

Spatial agents provide a means of exploring many complex and emergent behaviours of multi-agent systems that would otherwise be inaccessible to quantitative experiments. A number of multi-agent models have been developed for studying artificial life phenomena [2] including [3–6]. These projects place an emphasis on the evolution of "digital organisms" and the corresponding emergent macro behaviours.

One of the main problems in modelling and exploring evolutionary behaviour through simulations [5] is the sheer size of the parameter space or fitness landscape that is usually encountered. It is extraordinarily difficult to apply brute-force search methods to the phase spaces of many biologically-inspired computing models. The notion of genetic algorithms makes use of biologically-inspired mechanisms to combine and adapt existing "solutions" to find even better ones. This approach can still take very large computational resources to run simulated evolutionary models for long enough to see statistically significant changes or dramatically new solutions. It also presupposes that the starting conditions are sensible solutions and that in effect the experimenters are looking in roughly the right place - or at least a plausible position in solution space. While real biological systems proceed using thermodynamically scaled system sizes and very long time scales, even for simple computational models, it is not sufficient to start somewhere completely random and "hope" to evolve a solution somewhere in phase space.

The very nature of emergent behaviour implies a result that is unexpected. Published results (e.g. [7]) naturally focus on interesting emergent behaviour. However such results are often obtained after many CPU-hours of simulation time. In addition, there must be many long running experiments that fail and are abandoned simply because no interesting behaviours "emerge". In these cases many hours worth of simulation are often abandoned without any clear reason of where the problem lies – due to the large number of control variables required and the complexity of the system.

Popular science accounts of genetics [8] often fail to draw attention to the large number of components and time-scales involved in the evolution of real genetic systems [9]. A useful starting point for us was to identify a fundamental behaviour such as predation and build up a microscopic model around it. Our predator prey model [10] has been refined over a period of several years. Instead of noting evolution-

ary behaviour (which is often difficult to measure) we have concentrated on building up a suite of quantitative metrics to produce a range of statistical data as an experiment progresses. In this way we can rapidly analyse the current state of the model and use this knowledge to explore other avenues which may otherwise have remained unnoticed.

The details of the model system are described in section 2. The main focus of the paper appears in section 3 in which we describe various metrics and present examples using our model. It is also useful to compare agent-based models with theoretical models and this is presented in section 4. Finally, we offer some concluding remarks in section 5.

2 The Animat Model

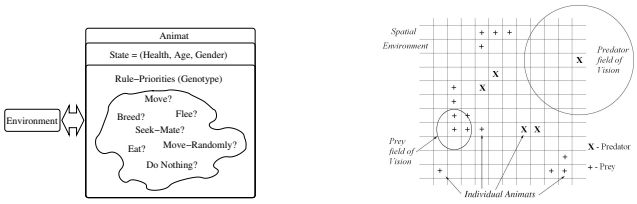


Figure 1: The internal workings of the spatial animat agents and the model world in which predator and prey species interact.

Our multi-agent system is based on the notion of multiple spatial agents or "animats" that coexist on a discrete square mesh. Each agent thus has: (x, y) coordinates; a health state; an age; a gender; and a set of microscopic rules that govern its behaviour. The particular set of rules that an animat follows determines its species and we have constructed the model using the primary species split in terms of predators and prey. This notion allows us to set up systems with appropriate competing forces that establish a dynamic equilibrium and which display a very rich set of emergent spatial patterns and macroscopic behaviours. A unique feature of this system is the relatively large number of microscopic animats that we are able to simulate. We routinely model quite large-scale experiments on systems of up to around one million animats, evolving for time-scales of a few thousand timesteps. Each experiment is repeated 10 times with different random number streams to make ensemble averages for a particular set of control parameters.

Figure 1 shows the outline of the model with individual animats containing an internal state and a genotypical rule embodying their microscopic behaviour preferences located on a spatial world map at particular (x, y) coordinates and interacting with their environment as embodied by their animat peers. The model contains two types of "animats" – predators that need to eat prey to survive and prey that need to eat "grass". Grass is placed only in certain areas of the map, thus effectively containing the animat populations. Previous work [11] has shown that animat behaviour is not affected by the edges of the grassed region. At every timestep, each animat executes a rule. Rules for predators are:

1. breed (females only) if health $> 50\%$ and mate adjacent
2. eat prey if health $< 50\%$ and prey adjacent
3. seek mate if health $> 50\%$
4. seek prey if health $< 50\%$
5. move randomly (50% chance of succeeding)

and the rules for prey are:

1. breed (females only) if health $> 50\%$ and mate adjacent
2. eat grass if health $< 50\%$
3. move randomly (50% chance of succeeding)
4. seek mate if health $> 50\%$
5. move away from adjacent predator

Each animat always executes the first rule in its list for which the conditions are satisfied. We have experimented with changing the order of priority of the rules [12] and thus produced different sub-groups of animats where each sub-group has the same set of rules but with a different priority order. The interaction of the animats as they follow their rules has produced fascinating emergent features in the form of macro-clusters often containing many hundreds of animats. We have analysed and documented these emergent clusters in [13]. The most interesting cluster is a spiral and several spirals and other patterns are visible in the figures in this article.

The simulation model was designed from scratch and is implemented as a highly optimised C++ program. A number of look-up tables are employed for computationally expensive operations such as computing square roots for distance sorting and for determining which animats are within a particular animat's field of vision and interaction. The spatial area of the model is organised into a coarse-grained grid and this allows us to optimise animat neighbour lookups. We employ a multi-phase update cycle that avoids bias amongst animat updates.

3 Metrics

This section is the main focus of the paper. We have developed several quantitative metrics that can be used to explore model behaviour in experiments involving the frequencies with which animats execute their possible rules; the role of global tuning parameters such as breeding and animat reproduction success rates; and the age distribution of animats when they "die." All of these support understanding of the life-cycle of individual animats and how their microscopic behaviours can be related to the observed macroscopic patterns observed in the whole multi-agent model system.

Each animat has a priority list of rules and depending upon its circumstances it will succeed or fail to execute particular rules. A statistical analysis of the rule execution frequencies gives some insights into the life cycle details of the animats. Table 1 shows a number of interesting effects over the typical lifetime of the multi-agent system. The values shown are the normalised frequencies of occurrence amongst predators at time epochs around: 40, 90, 140, 250, 1000, 2000 and 3000 timesteps. "Do nothing" indicates that the animat failed to execute any of the rules – recall that a rule will only be executed if conditions are met. Averages are over ten independent model configuration runs and appear to be reasonable estimates to within two significant figures.

There is a temporary crash in the animat populations as the model equilibrates after initialisation – see figure 2. Note how this affects the rule execution frequencies for timesteps 90 and 140 when less time is spent on breeding (females only) and more time is spent on seeking (temporarily scarce) prey. Timestep 250 reflects the shift in rule use as the population starts to recover. We also note that all animats spend a surprising amount of time doing nothing or moving randomly on average. Seeking prey and eating form the next most likely activities for predators.

The statistical results can be compared with the visual results of typical animat system configurations shown in figure 2. Individual animats are illustrated as white (prey) and black (predators). The initial population is started randomly, then rapidly grows too large for the land capacity of the system and therefore crashes. The system is then seen to make a gradual recovery to a relatively stable dynamic equilibrium. The long-term dynamic equilibrium state of the model is one of cluster formation, cluster erosion, breakaway groups, and the formation and dissipation of a variety of battlefront formations including spiral patterns.

The length scales present in the long term model appear to be largely independent of the detailed starting configuration. Complex patterns including spiral clusters and other formations occur regularly. We believe specific lengths such as the size of the spirals typically formed are implicitly controlled by model parameters such as the vision range of individual animats (nearest neighbour distances for prey and 50 spatial units or pixels for predators). All animats in the model can only move at most one spatial unit per timestep.

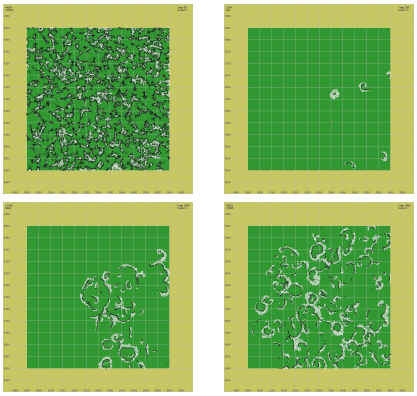


Figure 2: Timesteps 90, 250, 1000, 3000 showing the initial population crash followed by a recovery and long term dynamic equilibrium involving the formation and dissipation of battlefront patterns. Animats are shown as white (prey) and black (predators). The spatial grid is a guide to the length scales of the model system.

Another key observable for the model system is the total population of animat species as it varies with time starting at model initialisation. Figure 3 shows a logarithmic plot of the populations of the two interacting species in the multi-agent spatial environment. The plot shows the periodic oscillations superposed on a more slowly changing time envelope function. We are most interested in global parameter regimes that can be tuned so that the initial population configuration will eventually find a long-term sustainable dynamic equilibrium around stable average population values.

The model contains a global probabilistic success rate of breeding for each species. Thus success is not guaranteed when a female animat executes the breed rule. Without this constraint the model system can completely crash with the whole population dying out and therefore unable to ever recover. A single value (per species) provides a useful parameter to encapsulate and abstract likely effects such as birth difficulties, providing shelter for young and other submicroscopic details which would otherwise greatly increase the list of adjustable model parameters.

The global probabilities for breeding success can be adjusted and we have experimented with a range of values. Figure 4 shows a set of typical model configuration snapshots alongside population graphs that have been aver-

		40	90	140	250	1000	2000	3000
Predator Male	Do nothing	0.369	0.262	0.257	0.319	0.311	0.299	0.308
	Breed	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Eat	0.125	0.063	0.059	0.142	0.100	0.096	0.101
	Seek mate	0.068	0.031	0.036	0.087	0.049	0.049	0.050
	Seek prey	0.057	0.386	0.417	0.124	0.239	0.240	0.228
	Random	0.380	0.258	0.232	0.328	0.301	0.316	0.313
Predator Female	Do nothing	0.339	0.246	0.223	0.344	0.283	0.289	0.292
	Breed	0.072	0.027	0.043	0.064	0.044	0.044	0.047
	Eat	0.127	0.064	0.069	0.130	0.098	0.094	0.100
	Seek mate	0.056	0.025	0.036	0.053	0.043	0.039	0.043
	Seek prey	0.061	0.392	0.443	0.134	0.245	0.243	0.222
	Random	0.344	0.245	0.186	0.274	0.287	0.291	0.296

Table 1: Normalised rule frequencies of execution (for predators only) as measured over ten independent runs of the model. Note the effects of the population crash between timesteps 90 and 140, followed by relatively stable values thereafter. Note also the surprising amount of time individual animats spend doing nothing or moving randomly.

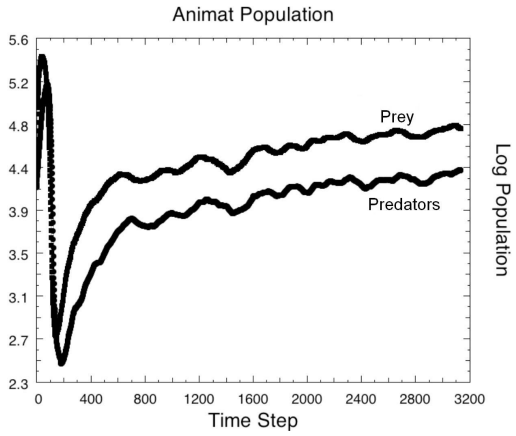


Figure 3: Populations of prey (top) and predators (bottom) shown on a logarithmic scale for a typical model run. The model is initialised randomly and displays a shock-recovery regime followed by a gradual convergence to a stable dynamic equilibrium with boom-bust periodic fluctuations superposed on a stable envelope function.

aged over ten model runs at the same time epoch of 3000 timesteps after initialisation. The probabilistic successes of breeding for predators/prey is shown for three different cases. In the top case predators have a 10% probability of successful reproduction and prey have a 40% success rate. This leads to a surfeit of prey and the model is seen to be spatially cluttered with dense clusters of prey. The population graph shows classic boom-bust periodic oscillations superposed on relatively stable average populations of around 120,000 prey and 20,000 predators.

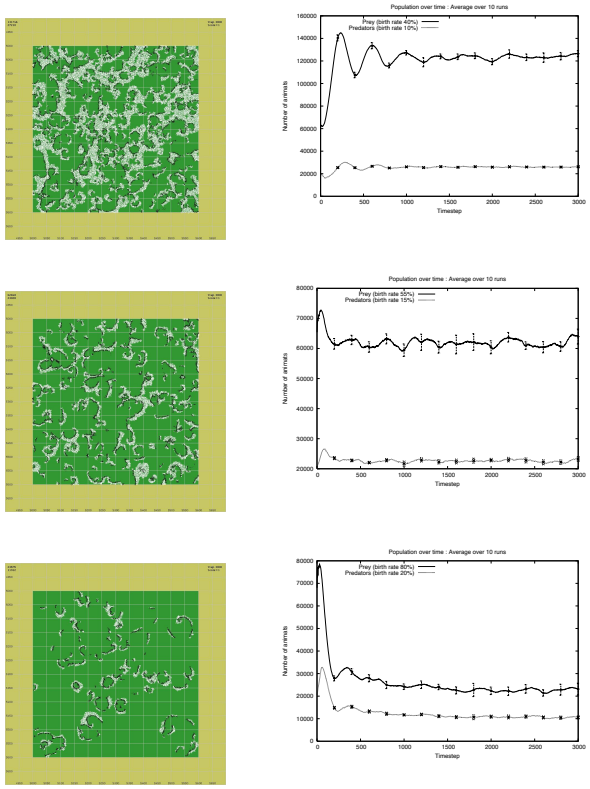


Figure 4: Snapshots after 3000 timesteps with 10%/40%; 15%/55% and 20%/80% breeding rates for predators/prey respectively. The success rate of predator breeding is seen to be critically linked to animat density and overall population.

A similar set of runs with the same spatial size of model but with 15%/55% reproduction success rate probabilities allows a predator population of around 25,000 which is enough to nearly halve the stable prey population to around 62,000. Further increases to the success rates for predator reproduction to 20% can be compensated for by increasing the prey reproduction rate to 80% and a stable predator population of around 12,000 feed off a prey population of around 25,000.

We can therefore tune the bulk average behaviour of the model in terms of numbers of animats in the average viable population. It should be noted that the spatial mixing and separation that are key components of this system give rise to spatial fluctuations across the whole system. At any given timestep one region may be "booming" while another is "busting" and it is an interesting emergent property of the model how these spatial fluctuations can combine either in phase (constructively) or out of phase (destructively).

The 10%/40% regime shown at the top of figure 4 shows a remarkable synchronisation of the boom-bust fluctuations over ten independent runs of the model. We hypothesise that this is characteristic of a dense model configuration, where information can be propagated right across the whole model system through normal animat interactions. The remaining two cases where there is a lower average spatial density also show periodic fluctuations but as can be seen they are much more smeared out by the averaging effect across the ten separate runs than for the dense case.

One approach to understanding the periodic boom-bust fluctuations is to transform the population samples shown into a frequency representation using a Fast Fourier Transform. Each of the $l = 1, 2, ..., M$ independent population curves $\mathcal{P}_{R,i}(t), \mathcal{P}_{F,i}(t)$ can be transformed to a frequency ω representation using:

$$\mathcal{P}_{l,j}(\omega) = \int_{-\infty}^{+\infty} \mathcal{P}_{l,j}(r) e^{2\pi i \omega t} dt, l = 1, ..., M; j \in R, F \tag{1}$$

The frequency histograms can be averaged for the M sample runs and the dominant frequency ω^* identified without combining and therefore destroying the associated phase information. It appears that the model always gives rise to a similar oscillatory train of boom-bust oscillations, largely independent of the starting configuration, but the exact time at which the model settles down to this behaviour does vary between runs. This offset time is what we mean by "phase" in this context. Hence the need to average, independent of phase to obtain representative period measurements. We find that the period calculated in this way is 380 ± 20 and this appears to be almost independent of the breeding rate.

Animats can "die" either of starvation or of old age. We have generally used the extreme ages for prey and predators as 20 and 50 timesteps respectively. Prey can of course also be "eaten." Figure 5 shows the histogrammed ages of animats when they died in a typical model run. There is an interesting difference between prey and predators in that the distribution is uniformly flat for predators indicating that all ages are roughly equally likely and we hypothesise that this is due to the uniformity of the prey landscape – on average. Predators can starve or die of old age and there is no particular bias in the spatial landscape to favour one age group over another. There is no particular difference between the age profile of male and female predators. However there is a significant difference in the profiles for female and male prey animats. We find that on average there is significant tendency for female prey to live less long than males. The distribution tails off with a similar shape but showing fewer long lived females.

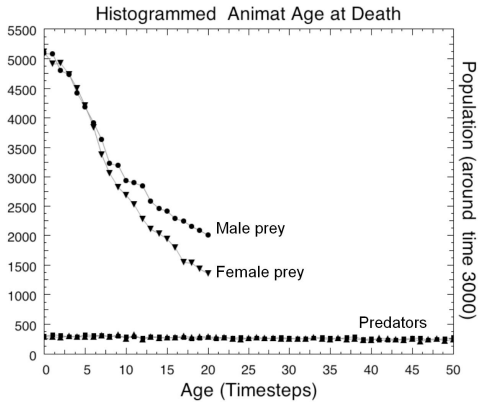


Figure 5: Ages on death around timestep 3000 with 10/40 breeding rates

We hypothesise that the shorter lifespan for female prey is due to the cost of breeding. A female animat cannot execute any other rule when the breed rule succeeds. While a male must be present for a female to successfully breed, at the same timestep the male can successfully execute some other rule. Thus on average a female has a slightly decreased likelihood of doing something useful that contributes to personal survival such as fleeing from predators.

4 Theoretical Comparisons

The model shows a number of emergent macroscopic spatial effects and we have tried to relate these to detailed observations of the microscopic behaviours. Population and ecological models are often compared to the bulk behaviour predicted by models based upon differential equations. A

noted example in the literature concerns the populations of the Canadian Lynx and its prey species [14].

Our microscopic model can usefully be compared with the Lotka-Volterra predator-prey model [15, 16]. This model is based on a system of two coupled differential equations. Let $R(t)$ be the number of prey at time t and $F(t)$ the number of predators. The uncoupled equations for predators and prey in a non-interacting world are then:

$$\frac{dR}{dt} = |a|R \quad (2)$$

$$\frac{dF}{dt} = -|g|F \quad (3)$$

so that unmolested by predators, the prey grow exponentially in number, and the predators starve through lack of prey and die off exponentially. The two controlling rate constants are positive numbers shown by the absolute value symbols in equations 2 and 3. We drop this hereafter, and assume $a >= 0, g >= 0$. It is of course interesting to consider what happens when the two populations do interact, and ignoring spatial distribution effects, we model this through a coupling term assumed to be proportional to the product RF which is related to the probability of a predator-prey encounter. We then obtain:

$$\frac{dR}{dt} = aR - bRF \quad (4)$$

$$\frac{dF}{dt} = hRF - mR \quad (5)$$

We now have some parameters to adjust. These will depend upon the frequency or probability of encounter between the species in the spatial model. It is a matter of future work to incorporate the spatial diffusion properly, but we can investigate some simple bulk behaviour for the system once it is roughly equilibrated. Solving this equation numerically, one can obtain equilibrium periodic solutions $(a = 2; b = c = g = h = 1)$ where the prey population reaches a high mean value with boom-bust periodic oscillations superposed on it. The predator population attains a lower mean oscillating value that lags behind the prey booms.

Our spatial animat multi-agent model does indeed show this **average** effect after some long term spatial fluctuations. It remains to link the equation parameters to the microscopic properties of the animat model, using diffusion constants. Some work on this has been reported using partial differential equations [17] although it remains to see how the spatial part of the equations can be factored out analytically. A numerical study of the spatial Lotka-Volterra and a careful search of its parameter space would be a valuable future area of work for making comparisons with discrete agent models such as our own.

5 Conclusion

We have described a spatial multi-agent model consisting of predator and prey animats coexisting in a spatial environment. We have given details of the model including a novel mechanism for encoding animat behaviours as rule priority lists. We have presented some visual snapshots of the model configurations and shown how these can be related to quantitative metrics such as populations; individual rule execution frequencies; animat densities; and animat age distributions.

We found that the bulk behaviour of the model is consistent with population models based on differential equations (such as the Lotka-Volterra system) but that the spatial richness and emergent structures can not be as easily explained [18]. It has been observed that the flow of energy through a non-closed system associated with corresponding decreases in entropy [19] is critically linked with the presence of life forms [20]. Energy is not conserved in our model system – it is not intended to be as it is not a closed system. This interplay of energy flow, the creation of information and the associated emergence of complex spatial structures present deeply interesting and fundamental questions about artificial life forms that it may be possible to explore further with simulated multi-agent systems.

Other models focus on emergent macro-behaviours but we have explored the life-cycle of a typical individual animat in the model. It is a surprising observation that a large fraction of animat time is spent doing nothing or moving about randomly and that the behaviours that appear more influential such as eating and breeding occupy minority fractions of animat time. This is of course an average effect. The environment for animats is defined solely in terms of the spatial substrate or background formed by other animats. Otherwise, animats perceive no spatial features on the landscape.

We might hypothesise that not all animats play important individual roles in the model as a whole. Nevertheless the silent majority are necessary to define the spatial structure of the model as a whole against which the "shakers and movers" can have a measurable effect. Future work will include an exploration of the analogies with sociological and military spatial patterns and structures and those found in our model. Although we have concentrated on statistical and general concepts, we believe the techniques we report upon may be applicable to specific and practical simulation models of physical and social phenomena.

References

[1] Cariani, P.: Emergence and Artificial Life. In C.G.Langton, C.Taylor, J., S.Rasmussen, eds.: Artificial Life II, SFI Studies in the Sciences of Complexity, Addison Wesley (1991) 775–797

[2] Bonabeau, E.W., Theraulaz, G.: Why do we need artificial life? In Langton, C.G., ed.: Artificial Life An Overview, MIT Press (1995) 303–325

[3] Adami, C.: On modeling life. In Brooks, R., Maes, P., eds.: Proc. Artificial Life IV, MIT Press (1994) 269–274

[4] Holland, J.H.: Echoing emergence: Objectives, rough definitions, and speculations for echo-class models. In Cowan, G.A., Pines, D., Meltzer, D., eds.: Complexity: Metaphors, Models and Reality. Addison-Wesley, Reading, MA (1994) 309–342

[5] Tyrrell, T., Mayhew, J.E.W.: Computer simulation of an animal environment. In Meyer, J.A., Wilson, S.W., eds.: From Animals to Animats, Proceedings of the First International Conference on Simulation of Adaptive Behavior. (1991) 263–272

[6] Ray, T.: An approach to the synthesis of life. Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity **xi** (1991) 371–408

[7] Ronald, E.M., Sipper, M., Capcarrère, M.S.: Design, Observation, Surprise! A Test of Emergence. Artificial Life **5** (1999) 225–239

[8] Levy, S.: Artificial Life The Quest for a New Creation. Penguin (1992)

[9] Turney, J.: Frankenstein's Footsteps - Science, Genetics and Popular Culture. Yale University Press (1998)

[10] Scogings, C.J., Hawick, K.A., James, H.A.: Tools and techniques for optimisation of microscopic artificial life simulation models. In Nyongesa, H., ed.: Proceedings of the Sixth IASTED International Conference on Modelling, Simulation, and Optimization, Gabarone, Botswana (2006) 90–95

[11] Scogings, C.J., Hawick, K.A.: Global constraints and diffusion in a localised animat agent model. In: Proceedings of the IASTED Int. Conf. on Applied Simulation and Modelling, Corfu, Greece (2008) 14–19

[12] Hawick, K.A., James, H.A., Scogings, C.J.: Roles of rule-priority evolution in animat models. In: Proc. Second Australian Conference on Artificial Life (ACAL 2005), Sydney, Australia (2005) 99–116

[13] Hawick, K.A., Scogings, C.J., James, H.A.: Defensive spiral emergence in a predator-prey model. Complexity International (2008) 1–10

[14] Stenseth, N.C., Falck, W., Chan, K.S., Bjornstad, O.N., O'Donoghue, M., Tong, H., Boonstra, R., Boutin, S., Krebs, C.J., Yoccoz, N.G.: From patterns to processes: Phase and density dependencies in the canadian lynx cycle. Proc. Natl. Acad. Sci. USA **95** (1998) 15430–15435

[15] Lotka, A.J.: Elements of Physical Biology. Williams & Williams, Baltimore (1925)

[16] Volterra, V.: Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. Mem. R. Accad. Naz. dei Lincei, Ser VI **2** (1926)

[17] Gallego, S.: Modelling Population Dynamics of Elephants. PhD thesis, Life and Environmental Sciences, University of Natal, Durban, South Africa. (2003)

[18] Thompson, D.W.: On Growth and Form. Cambridge University Press (1942)

[19] Ray, T.S.: Evolution, complexity, entropy, and artificial reality. Physica D **75** (1994) 239–263

[20] Adami, C.: Introduction to Artificial Life. Springer-Verlag (1998)

Simulated Docking of Oseltamivir with the 1918 Pandemic Strain Influenza A/H1N1 "Zanamivir-Conformed" Neuraminidase Active Site

Jack K. Horner
P.O. Box 266
Los Alamos NM 87544 USA

Abstract

Neuraminidases are glycoproteins that facilitate the transmission of the influenza virus from cell to cell. The neuraminidase inhibitors osteltamivir and zanamivir are currently the most widely used anti-flu therapeutics. Oseltamivir was ineffective against the dominant H1N1 strains in the 2008 flu season and decreasingly effective against the dominant influenza H1N1 mutants in the US in the 2009 "Spring/Fall" pandemic. Here I provide a computational docking analysis of oseltamivir with the active site of the neuraminidase of the 1918 strain (A/Brevig Mission/1/18 H1N1), conformed as if it were bound to zanamivir. The docking uses a Lamarckian genetic algorithm. The computed inhibitor/receptor binding energy suggests that oseltamivir would not be effective against that strain.

Keywords: Influenza, H1N1, neuraminidase, oseltamivir

1.0 Introduction

Neuraminidases are glycoproteins that facilitate the transmission of the influenza virus from cell to cell. The most widely used anti-influenza therapeutic, oseltamivir (Tamiflu™, [4]), was ineffective against the dominant H1N1 mutants in the 2008 flu season and was decreasingly effective against the dominant influenza mutant (Influenza A/H1N1) in the US in the 2009 "Spring/Fall" pandemic ([7]).

In the World Health Organization serotype-based influenza taxonomy, influenza type A has nine neuraminidase-related sero-subtypes, and these subtypes correspond at least roughly to differences in the active-site structures of the flu neuraminidases. The subtypes fall into two groups ([3]): group-1 contains the subtypes N1, N4, N5 and N8; group-2 contains the subtypes N2, N3, N6, N7 and N9. Oseltamivir was designed to target the group-2 neuraminidases.

The available crystal structures of the group-1 N1, N4 and N8 neuraminidases ([1]) reveal that the active sites of these enzymes have a very different three-dimensional structure from that of group-2 enzymes. The differences lie in a loop of amino acids known as the "150-loop", which in the group-1 neuraminidases has a conformation that opens a cavity not present in the group-2 neuraminidases. The 150-loop contains an amino acid designated Asp 151; the side chain of this amino acid has a carboxylic acid that, in group-1 enzymes, points away from the active site as a result of the 'open' conformation of the 150-loop. The side chain of another active-site amino acid, Glu 119, also has a different conformation in group-1 enzymes compared with the group-2 neuraminidases (8]).

The Asp 151 and Glu 119 amino-acid side chains form critical interactions with neuraminidase inhibitors. For neuraminidase subtypes with the “open conformation” 150-loop, the side chains of these amino acids might not have the precise alignment required to bind inhibitors tightly ([8]). The active site

of the 1918 strain has the 150-loop configuration.

The difference in the active-site conformations of the two groups of neuraminidases may also be caused by differences in amino acids that lie outside the active site. This means that an enzyme inhibitor for one target will not necessarily have the same activity against another with the same active-site amino acids and the same overall three-dimensional structure ([17]).

2.0 Method

The general objective of this study is straightforward: to computationally assess oseltamivir binding energy to the active site of crystallized 1918 pandemic strain neuraminidase active site, conformed as if it were bound to zanamivir. Unless otherwise noted, all processing described in this section was performed on a Dell Inspiron 545 with an Intel Core2 Quad CPU Q8200 (clocked @ 2.33 GHz) and 8.00 GB RAM, running under the *Windows Vista Home Premium (SP2)* operating environment.

Protein Data Bank (PDB) 3B7E is a structural description of most of the crystallized neuraminidase of Influenza A/Brevig Mission/1/18 H1N1 (the principal 1918 pandemic mutant) "folded" in a binding to *zanamivir* ([10]). This "zanamivir-folded" receptor, without the zanamivir, was chosen for analysis to determine whether it would bind more strongly with oseltamivir than the unliganded conformation (PDB 3BEQ) of the receptor.

3B7E consists of two identical chains, designated Chain A and Chain B.

3B7E was downloaded from PDB ([6]) on 31 January 2011. A PDB description of oseltamivir was extracted from PDB 2HU4 using Microsoft *Word*. The automated docking suite *AutoDock Tools* v 4.2 (ADT, [9]) was used to perform the docking of oseltamivir to the receptor. More specifically, in ADT, approximately following the rubric documented in [12]

-- Chain B, and the water in Chain A, of 3B7E were deleted

-- Chain A's active-site was extracted. (3B7E identifies the active site of Chain A as 14 amides: ARG118, GLU119, ASP151, ARG152, ARG156, TRP178, ARG224, GLU227, SER246, GLU276, GLU277, ARG292, ARG371, and TYR406.)

-- the hydrogens, charges, and torsions in the ligand and active site were adjusted using the ADT default recommendations and finally, the ligand, assumed to be flexible wherever that assumption is physically possible, was auto-docked to the active site, assumed to be rigid, using the Lamarckian genetic algorithm implemented in ADT.

The ADT parameters for the docking are shown in Figure 1. Most values are, or are a consequence of, ADT defaults.

```
autodock_parameter_version 4.2       # used by autodock to validate parameter set
outlev 1                             # diagnostic output level
intelec                              # calculate internal electrostatics
seed pid time                        # seeds for random generator
ligand_types NA C OA N               # atoms types in ligand
fld myreceptor.maps.fld              # grid_data_file
map myreceptor.NA.map                # atom-specific affinity map
map myreceptor.C.map                 # atom-specific affinity map
map myreceptor.OA.map                # atom-specific affinity map
map myreceptor.N.map                 # atom-specific affinity map
elecmap myreceptor.e.map             # electrostatics map
desolvmap myreceptor.d.map           # desolvation map
move myLigand.pdbqt                  # small molecule
about -29.8441 12.4988 -20.8904      # small molecule center
```

```
tran0 random                          # initial coordinates/A or random
axisangle0 random                     # initial orientation
dihe0 random                          # initial dihedrals (relative) or random
tstep 2.0                             # translation step/A
qstep 50.0                            # quaternion step/deg
dstep 50.0                            # torsion step/deg
torsdof 6                             # torsional degrees of freedom
rmstol 2.0                            # cluster_tolerance/A
extnrg 1000.0                         # external grid energy
e0max 0.0 10000                       # max initial energy; max number of retries
ga_pop_size 150                       # number of individuals in population
ga_num_evals 2500000                  # maximum number of energy evaluations
ga_num_generations 27000              # maximum number of generations
ga_elitism 1                          # number of top individuals to survive to next
generation
ga_mutation_rate 0.02                 # rate of gene mutation
ga_crossover_rate 0.8                 # rate of crossover
ga_window_size 10                     #
ga_cauchy_alpha 0.0                   # Alpha parameter of Cauchy distribution
ga_cauchy_beta 1.0                    # Beta parameter Cauchy distribution
set_ga                                # set the above parameters for GA or LGA
sw_max_its 300                        # iterations of Solis & Wets local search
sw_max_succ 4                         # consecutive successes before changing rho
sw_max_fail 4                         # consecutive failures before changing rho
sw_rho 1.0                            # size of local search space to sample
sw_lb_rho 0.01                        # lower bound on rho
ls_search_freq 0.06                   # probability of performing local search on
individual
set_psw1                              # set the above pseudo-Solis & Wets parameters
unbound_model bound                   # state of unbound ligand
ga_run 10                             # do this many hybrid GA-LS runs
analysis                              # perform a ranked cluster analysis
```

Figure 1. ADT parameters for the docking in this study

3.0 Results

The interactive problem setup, which assumes familiarity with the general neuraminidase "landscape", took about 20 minutes in ADT; the docking proper, about 25 minutes on the platform described in Section 2.0 The platform's performance monitor suggested that the calculation was more or less uniformly distributed across the four processors at ~25% of peak per processor (with occasional bursts to 40% of peak), and required a constant 2.9 GB of memory.

Figure 2 shows the oseltamivir/receptor energy and position summary produced by ADT. The estimated free energy of binding is ~ -8.4 kcal/mol; the estimated inhibition constant, ~646 nanoMolar at 298 K.

.

```
        LOWEST ENERGY DOCKED CONFORMATION from EACH CLUSTER
        ___________________________________________________


Keeping original residue number (specified in the input PDBQ file) for outputting.

MODEL        1
USER    Run = 1
USER    Cluster Rank = 1
```

```
USER    Number of conformations in this cluster = 10
USER
USER    RMSD from reference structure       = 0.918 A
USER
USER    Estimated Free Energy of Binding    =   -8.44 kcal/mol  [=(1)+(2)+(3)-(4)]
USER    Estimated Inhibition Constant, Ki   =  646.30 nM (nanomolar)  [Temperature =
298.15 K]
USER
USER    (1) Final Intermolecular Energy     =  -10.23 kcal/mol
USER        vdW + Hbond + desolv Energy     =   -7.34 kcal/mol
USER        Electrostatic Energy            =   -2.89 kcal/mol
USER    (2) Final Total Internal Energy     =   -1.29 kcal/mol
USER    (3) Torsional Free Energy           =   +1.79 kcal/mol
USER    (4) Unbound System's Energy  [=(2)] =   -1.29 kcal/mol
USER
USER
USER
USER    DPF = myLigand.dpf
USER    NEWDPF move    myLigand.pdbqt
USER    NEWDPF about   -29.844101 12.498800 -20.890400
USER    NEWDPF tran0   -29.752382 12.621669 -20.989646
USER    NEWDPF axisangle0     -0.545753 -0.106522 -0.831148 -6.158671
USER    NEWDPF quaternion0    -0.029317 -0.005722 -0.044648 -0.998556
USER    NEWDPF dihe0   163.29 179.37 128.27 2.12 -8.37 -36.66
USER
USER                            x       y       z    vdW   Elec        q     RMS
ATOM      1  C2  ZMR A1001     -29.802  11.354 -23.039 -0.06 +0.09    +0.156  0.918
ATOM      2  C3  ZMR A1001     -28.506  11.639 -22.803 -0.35 +0.00    +0.031  0.918
ATOM      3  C4  ZMR A1001     -28.126  12.635 -21.728 -0.25 -0.02    +0.171  0.918
ATOM      4  C5  ZMR A1001     -29.212  12.665 -20.635 -0.16 +0.03    +0.164  0.918
ATOM      5  C6  ZMR A1001     -30.596  12.845 -21.262 -0.16 +0.05    +0.179  0.918
ATOM      6  O6  ZMR A1001     -30.879  11.858 -22.246 -0.17 -0.20    -0.323  0.918
ATOM      7  C7  ZMR A1001     -31.759  12.798 -20.264 -0.10 +0.12    +0.214  0.918
ATOM      8  O7  ZMR A1001     -32.344  11.506 -20.304 -0.06 -0.16    -0.218  0.918
ATOM      9  C8  ZMR A1001     -32.842  13.843 -20.586 -0.24 +0.06    +0.218  0.918
ATOM     10  O8  ZMR A1001     -33.494  13.440 -21.802 -0.23 -0.10    -0.218  0.918
ATOM     11  C9  ZMR A1001     -32.218  15.255 -20.696 -0.29 +0.05    +0.218  0.918
ATOM     12  O9  ZMR A1001     -32.998  16.172 -19.933 -0.60 -0.16    -0.218  0.918
ATOM     13  N5  ZMR A1001     -28.990  13.757 -19.684 -0.08 -0.10    -0.235  0.918
ATOM     14  C10 ZMR A1001     -28.440  13.590 -18.467 -0.19 +0.21    +0.252  0.918
ATOM     15  O10 ZMR A1001     -28.147  12.477 -18.014 -0.71 -0.35    -0.271  0.918
ATOM     16  C11 ZMR A1001     -28.164  14.876 -17.751 -0.34 +0.09    +0.100  0.918
ATOM     17  NE  ZMR A1001     -26.860  12.328 -21.039 -0.26 +0.02    -0.078  0.918
ATOM     18  CZ  ZMR A1001     -25.937  13.198 -20.774 +0.13 -0.05    +0.783  0.918
ATOM     19  NH1 ZMR A1001     -25.957  14.473 -21.151 -0.36 -0.02    +0.063  0.918
ATOM     20  NH2 ZMR A1001     -24.890  12.794 -20.047 -0.44 +0.03    +0.063  0.918
ATOM     21  C1  ZMR A1001     -30.277  10.564 -24.182 -0.19 +0.35    +0.234  0.918
ATOM     22  O1A ZMR A1001     -29.395   9.968 -24.859 -1.06 -1.40    -0.642  0.918
ATOM     23  O1B ZMR A1001     -31.495  10.500 -24.434 -1.15 -1.45    -0.642  0.918
```

Figure 2. ADT's oseltamivir energy and position predictions.

Figure 3 is a rendering of the active-site/inhibitor configuration computed in this study.

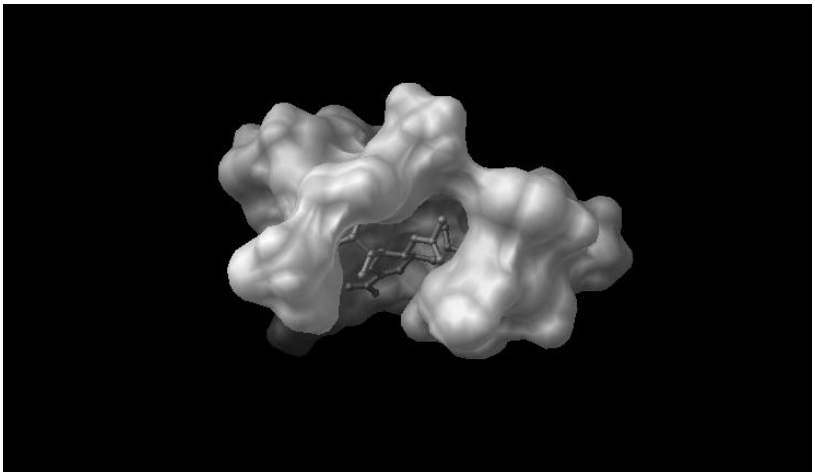


Figure 3. Rendering of oseltamivir computationally docked with the active site of PDB 3B7E. The inhibitor is shown in stick form. Only the interior, inhibitor-containing region of the molecular surface of the active site can be compared to *in situ* data: the surface distal to the interior is a computational artifact, generated by the assumption that active site is detached from the rest of the receptor.

4.0 Discussion

The method described in Section 2.0 and the results of Section 3.0 motivate several observations:

1. The inhibition constant computed in this study (~646 nanoMolar at ~298 K) is comparable to the inhibition constant of oseltamivir/neuraminidase interactions that are not clinically effective ([11], [13]). This suggests that oseltamivir would not be effective against the principal 1918 pandemic mutant, A/Brevig Mission/1/18 H1N1.

2. The docking study reported here assumes that the receptor is rigid, and as a result, calculation does not reflect any energy contributions of receptor "flexing" to the interaction of the ligand with native unliganded receptor. Future work will analyze the docking with a flexible receptor. However, based on a comparison with the binding of oseltaimivir with the unliganded ("native) conformation, it is clear that oseltamivir binds more tightly with the zanamivir-folded-receptor, than it does with the native unliganded-receptor, conformation ([15]).

3. The analysis described in Sections 2.0 and 3.0 assumes the neuraminidase is in a crystallized form (isolated at ~278 K). *In situ*, at physiologically normal temperatures (~310 K), the receptor is not in crystallized form. The ligand/receptor conformation *in situ*, therefore, may not be identical to their conformation in the crystallized form.

4. Minimum-energy search algorithms other than the Lamarckian genetic algorithm used in this work could be applied to this docking problem. Future work will use Monte Carlo/simulated annealing algorithms. In addition, a variety of torsion and charge models could be applied to this problem, and future work will do so.

5.0 Acknowledgements

This work benefited from discussions with Tony Pawlicki. For any problems that remain, I am solely responsible.

6.0 References.

[1] Russell RJ et al. The structure of H5N1 avian neuraminidase suggests new opportunities for drug design. *Nature* 443 (6 September 2006), 45-49.

[2] Johnson NP and Mueller J. Updating the accounts: global mortality of the 1918-1920 "Spanish " influenza pandemic. *Bulletin of the History of Medicine* 76 (2002), 105-115.

[3] World Health Organization. A revision of the system of nomenclature for influenza viruses: a WHO memorandum. *Bulletin of the World Health Organization* 58 (1980), 585-591.

[4] Ward P et al. Oseltamivir (Tamiflu) and its potential for use in the event of an influenza pandemic. *Journal of Antimicrobial Chemotherapy* 55, supplement 1 (2005), i5-i21.

[5] Butler D. Avian flu special: The flu pandemic: were we ready? *Nature* 435 (26 May 2005), 400-402. doi: 10.1038/435400a.

[6] Xu X, Zhu X, Dwek RA, Stevens J, and Wilson IA. Structural characterization of the 1918 Influenza virus H1N1 neuraminidase. *Journal of Virology* 82 (November 2008), 10493-10501. http://www.pdb.org/pdb/explore/explore.do?structureId=3BEQ.

[7] US Centers for Disease Control. *Summary: Interim Recommendations for the Use of Influenza Antiviral Medications in the Setting of Oseltamivir Resistance among Circulating Influenza A (H1N1) Viruses, 2008-09 Influenza Season.* 19 December 2008. URL http://www.cdc.gov/flu/professionals/antivirals/summary.htm.

[8] Luo M. Structural biology: antiviral drugs fit for a purpose. *Nature* 443 (7 September 2006), 37-38. doi:10.1038/443037a, published online 6 September 2006.

[9] Morris GM, Goodsell DS, Huey R, Lindstrom W, Hart WE, Kurowski S, Halliday S, Belew R, and Olson AJ. *AutoDock* v4.2. http://autodock.scripps.edu/. 2010.

[10] Drug Bank. *Zanamivir*. http://www.drugbank.ca/drugs/APRD00378.

[11] Govorkova EA et al. Comparison of efficacies of RWJ-270201, zanamivir, and oseltamivir against H5N1, H9N2, and other avian influenza viruses. *Antimicrobial Agents and Chemotherapy* 45 (2001), 2723-2732.

[12] Huey R and Morris GM. *Using AutoDock 4 with AutoDock Tools: A Tutorial.* 8 January 2008. http://autodock.scripps.edu/.

[13] Cheng Y and Prusoff WH. Relationship between the inhibition constant (K_i) and the concentration of inhibitor which causes 50 per cent inhibition (I_{50}) of an enzymatic reaction. *Biochemical Pharmacology* 22 (December 1973), 3099–3108. doi:10.1016/0006-2952(73)90196-2.

[14] Protein Data Bank entry 2HU4. http://www.rcsb.org/pdb/explore.do?structureId=2hu4. Russell RJ et al. The structure of H5N1 avian neuraminidase suggests new opportunities for drug design. *Nature* 443 (6 September 2006), 45-49.

[15] Horner JK. Binding energy of oseltamivir with the 1918 pandemic strain Influenza A/H1N1 "Unfolded" neuraminidase active site. *Proceedings of the 2011 International Conference on Bioinformatics and Computational Biology*. CSREA Press. 2011. Forthcoming.

Simulated Docking of Zanamivir with the 1918 Pandemic Strain Influenza A/H1N1 Neuraminidase Active Site

Jack K. Horner
P.O. Box 266
Los Alamos NM 87544 USA

Abstract

Neuraminidases are glycoproteins that facilitate the transmission of the influenza virus from cell to cell. The neuraminidase inhibitors osteltamivir and zanamivir are currently the most widely used anti-flu therapeutics. Oseltamivir was ineffective against the dominant H1N1 strains in the 2008 flu season and decreasingly effective against the dominant influenza H1N1 mutants in the US in the 2009 "Spring/Fall" pandemic. Zanamivir has proven useful against some H1N1 strains that are resistant to oseltamivir. Here I provide a computational docking analysis of zanamivir with the active site of the neuraminidase of the 1918 strain (A/Brevig Mission/1/18 H1N1). The docking uses a Lamarckian genetic algorithm. The computed inhibitor/receptor binding energy suggests that zanamivir would not be effective against that strain.

Keywords: Influenza, H1N1, neuraminidase, zanamivir

1.0 Introduction

The mortality rate in humans infected with Influenza A/H1N1 in the 1918 pandemic was ~50% ([2]). The 1918 mutant(s), unlike any genotype of H1N1 observed since, was easily transmitted among humans and killed ~10% of the world population within a single six-month period ([2]).

At present, no plausible public health regime could control an outbreak of a high-mortality-rate, highly infectious (HMR/HI) H1N1 mutant. The scale of human interaction required to sustain food and fuel distribution to large urban areas would render quarantine ineffective ([5]). Currently, the lead time for vaccine development and production is at least as long as the duration of the 1918 pandemic. A widespread administration of effective anti-influenza therapeutics is therefore the only practical defense against a 1918-scale event after the pandemic begins.

Neuraminidases are glycoproteins that facilitate the transmission of the influenza virus from cell to cell. The most widely used anti-influenza therapeutic, oseltamivir (Tamiflu™, [4]), was ineffective against the dominant H1N1 mutants in the 2008 flu season and was decreasingly effective against the dominant influenza mutant (Influenza A/H1N1) in the US in the 2009 "Spring/Fall" pandemic ([7]).

In the World Health Organization serotype-based influenza taxonomy, influenza type A has nine neuraminidase-related sero-subtypes, and these subtypes correspond at least roughly to differences in the active-site structures of the flu neuraminidases. The subtypes fall into two groups ([3]): group-1 contains the subtypes N1, N4, N5 and N8; group-2 contains the subtypes N2, N3, N6, N7 and N9. Oseltamivir was designed to target the group-2 neuraminidases.

The available crystal structures of the group-1 N1, N4 and N8 neuraminidases ([1]) reveal that the active sites of these enzymes have a very different three-

dimensional structure from that of group-2 enzymes. The differences lie in a loop of amino acids known as the "150-loop", which in the group-1 neuraminidases has a conformation that opens a cavity not present in the group-2 neuraminidases. The 150-loop contains an amino acid designated Asp 151; the side chain of this amino acid has a carboxylic acid that, in group-1 enzymes, points away from the active site as a result of the 'open' conformation of the 150-loop. The side chain of another active-site amino acid, Glu 119, also has a different conformation in group-1 enzymes compared with the group-2 neuraminidases (8]).

The Asp 151 and Glu 119 amino-acid side chains form critical interactions with neuraminidase inhibitors. For neuraminidase subtypes with the "open conformation" 150-loop, the side chains of these amino acids might not have the precise alignment required to bind inhibitors tightly ([8]). The active site of the 1918 strain has the 150-loop configuration.

The difference in the active-site conformations of the two groups of neuraminidases may also be caused by differences in amino acids that lie outside the active site. This means that an enzyme inhibitor for one target will not necessarily have the same activity against another with the same active-site amino acids and the same overall three-dimensional structure ([17]).

Zanamivir ([10]) has proven effective against some H1N1 strains that are resistant to oseltamivir.

2.0 Method

The general objective of this study is straightforward: to computationally assess the binding energy of the active site of crystallized 1918 pandemic strain neuraminidase with zanamivir. Unless otherwise noted, all processing described in this section was performed on a Dell Inspiron 545 with an Intel Core2 Quad CPU Q8200 (clocked @ 2.33 GHz) and 8.00 GB RAM, running under the *Windows Vista Home Premium (SP2)* operating environment.

Protein Data Bank (PDB) 3B7E is a structural description of most of the crystallized neuraminidase of Influenza A/Brevig Mission/1/18 H1N1 (the principal 1918 pandemic mutant), bound to zanamivir. 3B7E consists of two identical chains, designated Chain A and Chain B.

3B7E was downloaded from PDB ([6]) on 31 January 2011. The ligand description was extracted from the PDB file using Microsoft *Word*. The automated docking suite *AutoDock Tools* v 4.2 (ADT, [9]) was used to perform the docking of zanamivir to the receptor. More specifically, in ADT, approximately following the rubric documented in [12]

-- Chain B, and the water in Chain A, of 3B7E were deleted

-- Chain A's active-site was extracted. (3B7E identifies the active site of Chain A as 14 amides: ARG118, GLU119, ASP151, ARG152, ARG156, TRP178, ARG224, GLU227, SER246, GLU276, GLU277, ARG292, ARG371, and TYR406.)

-- the ligand was rotated and translated (to force ADT to seek a non-trivial docking solution)

-- the hydrogens, charges, and torsions in the ligand and active site were adjusted using the ADT recommended defaults and finally, the ligand, assumed to be flexible wherever that assumption is physically possible, was auto-docked to the active site, assumed to be rigid, using the Lamarckian genetic algorithm implemented in ADT.

The ADT parameters for the docking are shown in Figure 1. Most values are, or are a consequence of, ADT defaults.

```
autodock_parameter_version 4.2       # used by autodock to validate parameter set
outlev 1                             # diagnostic output level
intelec                              # calculate internal electrostatics
seed pid time                        # seeds for random generator
ligand_types C HD OA N               # atoms types in ligand
fld 3B7E_Receptor.maps.fld           # grid_data_file
map 3B7E_Receptor.C.map              # atom-specific affinity map
map 3B7E_Receptor.HD.map             # atom-specific affinity map
map 3B7E_Receptor.OA.map             # atom-specific affinity map
map 3B7E_Receptor.N.map              # atom-specific affinity map
elecmap 3B7E_Receptor.e.map          # electrostatics map
desolvmap 3B7E_Receptor.d.map        # desolvation map
move 3B7E_Ligand_out.pdbqt           # small molecule
about -29.5772 12.7517 -20.6465      # small molecule center
tran0 random                         # initial coordinates/A or random
axisangle0 random                    # initial orientation
dihe0 random                         # initial dihedrals (relative) or random
tstep 2.0                            # translation step/A
qstep 50.0                           # quaternion step/deg
dstep 50.0                           # torsion step/deg
torsdof 9                            # torsional degrees of freedom
rmstol 2.0                           # cluster_tolerance/A
extnrg 1000.0                        # external grid energy
e0max 0.0 10000                      # max initial energy; max number of retries
ga_pop_size 150                      # number of individuals in population
ga_num_evals 2500000                 # maximum number of energy evaluations
ga_num_generations 27000             # maximum number of generations
ga_elitism 1                         # number of top individuals to survive to next
generation
ga_mutation_rate 0.02                # rate of gene mutation
ga_crossover_rate 0.8                # rate of crossover
ga_window_size 10                    #
ga_cauchy_alpha 0.0                  # Alpha parameter of Cauchy distribution
ga_cauchy_beta 1.0                   # Beta parameter Cauchy distribution
set_ga                               # set the above parameters for GA or LGA
sw_max_its 300                       # iterations of Solis & Wets local search
sw_max_succ 4                        # consecutive successes before changing rho
sw_max_fail 4                        # consecutive failures before changing rho
sw_rho 1.0                           # size of local search space to sample
sw_lb_rho 0.01                       # lower bound on rho
ls_search_freq 0.06                  # probability of performing local search on
individual
set_psw1                             # set the above pseudo-Solis & Wets parameters
unbound_model bound                  # state of unbound ligand
ga_run 10                            # do this many hybrid GA-LS runs
analysis                             # perform a ranked cluster analysis
```

Figure 1. ADT parameters for the docking in this study

Interatomic distances between ligand and receptor in the computed form were compared to those in 3B7E.

3.0 Results

The interactive problem setup, which assumes familiarity with the general neuraminidase "landscape", took about 30 minutes in ADT; the docking proper, about 28 minutes on the platform described in Section 2.0 The platform's performance monitor suggested that the calculation was more or less uniformly distributed across the four processors at ~25% of peak per processor (with occasional bursts to 40% of peak), and required a constant 2.9 GB of memory.

Figure 2 shows the zanamivir/receptor energy and position summary produced by ADT. The estimated free energy of binding is ~ -8.9 kcal/mol; the estimated inhibition constant, ~279 nM at 298 K. All distances between receptor and ligand atoms in the computed ligand position lie within 5% of the distances of the corresponding atoms in 3B7E.

LOWEST ENERGY DOCKED CONFORMATION from EACH CLUSTER

```
Keeping original residue number (specified in the input PDBQ file) for outputting.

MODEL        4
USER    Run = 4
USER    Cluster Rank = 1
USER    Number of conformations in this cluster = 10
USER
USER    RMSD from reference structure       = 0.782 A
USER
USER    Estimated Free Energy of Binding    =   -8.94 kcal/mol  [=(1)+(2)+(3)-(4)]
USER    Estimated Inhibition Constant, Ki   =  278.50 nM (nanomolar)  [Temperature =
298.15 K]
USER
USER    (1) Final Intermolecular Energy     =  -11.63 kcal/mol
USER        vdW + Hbond + desolv Energy     =   -8.32 kcal/mol
USER        Electrostatic Energy            =   -3.31 kcal/mol
USER    (2) Final Total Internal Energy     =   -2.40 kcal/mol
USER    (3) Torsional Free Energy           =   +2.68 kcal/mol
USER    (4) Unbound System's Energy  [=(2)] =   -2.40 kcal/mol
USER
USER
USER
USER    DPF = C:\Users\Owner\3B7E.dpf
USER    NEWDPF move    3B7E_Ligand_out.pdbqt
USER    NEWDPF about   -29.577200 12.751700 -20.646500
USER    NEWDPF tran0   -29.396053 12.777381 -20.687186
USER    NEWDPF axisangle0     0.168535 -0.286819 0.943043 2.497566
USER    NEWDPF quaternion0    0.003673 -0.006251 0.020552 0.999762
USER    NEWDPF dihe0   38.67 -125.98 155.79 178.63 66.03 -1.38 -65.94 2.63 -20.03
USER
USER                              x       y       z    vdW   Elec        q     RMS
ATOM      1  C2  ZMR A1001     -29.716  11.373 -23.038 -0.06 +0.08    +0.144  0.782
ATOM      2  C3  ZMR A1001     -28.429  11.708 -22.816 -0.34 +0.00    +0.045  0.782
ATOM      3  C4  ZMR A1001     -28.071  12.663 -21.698 -0.27 -0.02    +0.150  0.782
ATOM      4  C5  ZMR A1001     -29.130  12.585 -20.582 -0.17 +0.03    +0.143  0.782
ATOM      5  C6  ZMR A1001     -30.536  12.730 -21.168 -0.15 +0.06    +0.185  0.782
ATOM      6  O6  ZMR A1001     -30.796  11.783 -22.196 -0.15 -0.20    -0.335  0.782
ATOM      7  NE  ZMR A1001     -26.775  12.384 -21.053 -0.22 +0.06    -0.217  0.782
ATOM      8  HE  ZMR A1001     -26.486  11.409 -20.974 -0.27 -0.19    +0.178  0.782
ATOM      9  CZ  ZMR A1001     -25.978  13.288 -20.580 +0.01 +0.01    +0.665  0.782
ATOM     10  NH1 ZMR A1001     -26.169  14.602 -20.674 -0.24 +0.06    -0.235  0.782
```

```
ATOM     11  NH2 ZMR A1001     -24.886  12.878 -19.926 -0.30 -0.12    -0.235  0.782
ATOM     12 2HH1 ZMR A1001     -27.002  14.915 -21.172 +0.08 -0.08    +0.174  0.782
ATOM     13 1HH1 ZMR A1001     -25.542  15.314 -20.302 -0.32 -0.09    +0.174  0.782
ATOM     14 2HH2 ZMR A1001     -24.739  11.871 -19.854 -0.44 +0.15    +0.174  0.782
ATOM     15 1HH2 ZMR A1001     -24.258  13.590 -19.554 -0.44 +0.09    +0.174  0.782
ATOM     16  N5  ZMR A1001     -28.936  13.636 -19.579 -0.02 -0.16    -0.352  0.782
ATOM     17  H5  ZMR A1001     -29.215  14.587 -19.820 +0.10 +0.05    +0.163  0.782
ATOM     18  C10 ZMR A1001     -28.412  13.419 -18.359 -0.23 +0.20    +0.214  0.782
ATOM     19  C11 ZMR A1001     -28.161  14.676 -17.582 -0.34 +0.12    +0.117  0.782
ATOM     20  O10 ZMR A1001     -28.120  12.291 -17.948 -0.70 -0.39    -0.274  0.782
ATOM     21  C1  ZMR A1001     -30.181  10.620 -24.210 -0.17 +0.34    +0.233  0.782
ATOM     22  O1A ZMR A1001     -29.288  10.116 -24.945 -1.07 -1.47    -0.642  0.782
ATOM     23  O1B ZMR A1001     -31.400  10.499 -24.430 -1.10 -1.42    -0.642  0.782
ATOM     24  C7  ZMR A1001     -31.671  12.575 -20.148 -0.09 +0.11    +0.180  0.782
ATOM     25  C8  ZMR A1001     -32.964  13.269 -20.611 -0.23 +0.07    +0.173  0.782
ATOM     26  O8  ZMR A1001     -33.366  14.183 -19.577 -0.20 -0.11    -0.391  0.782
ATOM     27  H8  ZMR A1001     -34.313  14.150 -19.512 -0.30 -0.14    +0.210  0.782
ATOM     28  C9  ZMR A1001     -34.060  12.223 -20.927 -0.15 +0.13    +0.198  0.782
ATOM     29  O9  ZMR A1001     -35.267  12.594 -20.267 -0.14 -0.09    -0.398  0.782
ATOM     30  H9  ZMR A1001     -35.492  13.486 -20.505 -0.45 -0.26    +0.209  0.782
ATOM     31  O7  ZMR A1001     -31.935  11.194 -19.964 +0.01 -0.27    -0.390  0.782
ATOM     32  H7  ZMR A1001     -31.650  10.722 -20.737 +0.07 +0.13    +0.210  0.782
```

Figure 2. ADT's zanamivir energy and position predictions.

Figure 3 is a rendering of the active-site/inhibitor configuration computed in this study.

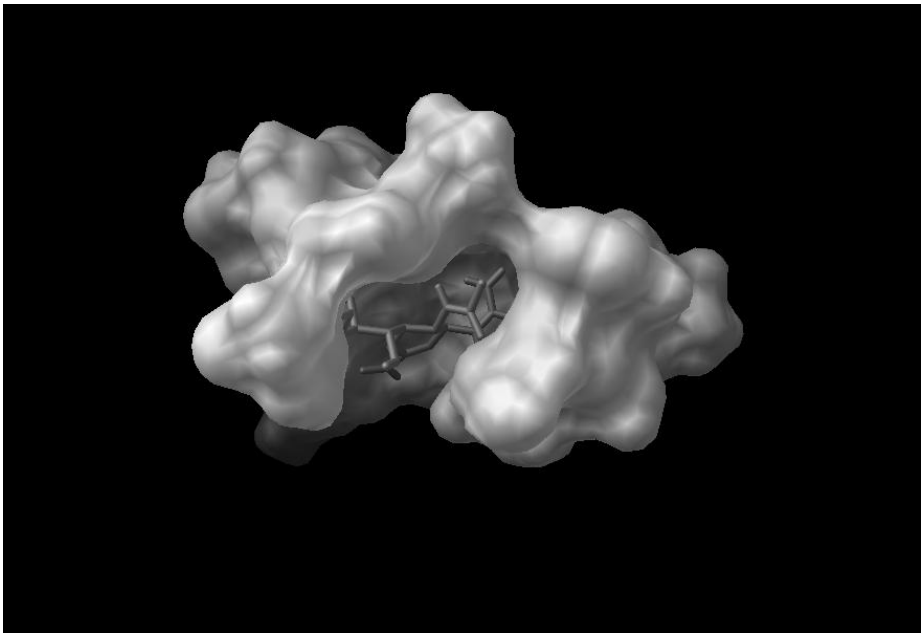


Figure 3. Rendering of zanamivir computationally docked with the active site of PDB 3B7E. The inhibitor is shown in stick form. Only the interior, inhibitor-containing region of the molecular surface of the active site can be compared to *in situ* data: the surface distal to the interior is a computational artifact, generated by the assumption that active site is detached from the rest of the receptor.

4.0 Discussion

The method described in Section 2.0 and the results of Section 3.0 motivate several observations:

1. The inhibition constant computed in this study (~279 nanoMolar at ~298 K) is comparable to the inhibition constant of zanamivir/neuraminidase interactions that are not clinically effective ([11], [13]). This suggests that zanamivir would not be effective against the principal 1918 pandemic mutant, A/Brevig Mission/1/18 H1N1.

2. All distances between receptor and ligand atoms in the computed ligand position lie within 5% of the distances of the corresponding atoms in 3B7E. (For electrostatic forces, a 5% distance difference would correspond to a (1.05^2 =) 10% difference in electrostatic force and potential energy. One could of course apply other statistics to the coordinate sets and provide a more comprehensive comparison of other forces/energies. Future work will address those issues.)

3. The docking study reported here assumes that the receptor is rigid. This assumption is appropriate for the binding energy computation for PDB 3B7E per se. However, the calculation does not reflect what receptor "flexing" could contribute to the interaction of the ligand with native unliganded receptor (e.g., PDB 3BEQ ([14])). Future work will analyze the docking of the ligand with the native form.

4. The analysis described in Sections 2.0 and 3.0 assumes the neuraminidase is in a crystallized form (isolated at ~278 K). *In situ*, at physiologically normal temperatures (~310 K), the receptor is not in crystallized form. The ligand and receptor conformations *in situ*, therefore, may not be identical to their conformations in the crystallized form.

5. Minimum-energy search algorithms other than the Lamarckian genetic algorithm used in this work could be applied to this docking problem. Future work will use Monte Carlo/simulated annealing algorithms.

6. A variety of torsion and charge models could be applied to this problem, and future work will do so.

5.0 Acknowledgements

This work benefited from discussions with Tony Pawlicki. For any problems that remain, I am solely responsible.

6.0 References.

[1] Russell RJ et al. The structure of H5N1 avian neuraminidase suggests new opportunities for drug design. *Nature* 443 (6 September 2006), 45-49.

[2] Johnson NP and Mueller J. Updating the accounts: global mortality of the 1918-1920 "Spanish " influenza pandemic. *Bulletin of the History of Medicine* 76 (2002), 105-115.

[3] World Health Organization. A revision of the system of nomenclature for influenza viruses: a WHO memorandum. *Bulletin of the World Health Organization* 58 (1980), 585-591.

[4] Ward P et al. Oseltamivir (Tamiflu) and its potential for use in the event of an influenza pandemic. *Journal of Antimicrobial Chemotherapy* 55, supplement 1 (2005), i5-i21.

[5] Butler D. Avian flu special: The flu pandemic: were we ready? *Nature* 435 (26 May 2005), 400-402. doi: 10.1038/435400a.

[6] Xu X, Zhu X, Dwek RA, Stevens J, and Wilson IA. Structural characterization of the 1918 Influenza virus H1N1 neuraminidase. *Journal of Virology* 82 (November 2008), 10493-10501.

http://www.pdb.org/pdb/explore/explore.do?structureId=3B7E.

[7] US Centers for Disease Control. *Summary: Interim Recommendations for the Use of Influenza Antiviral Medications in the Setting of Oseltamivir Resistance among Circulating Influenza A (H1N1) Viruses, 2008-09 Influenza Season.* 19 December 2008. URL http://www.cdc.gov/flu/professionals/antivirals/summary.htm.

[8] Luo M. Structural biology: antiviral drugs fit for a purpose. *Nature* 443 (7 September 2006), 37-38. doi:10.1038/443037a, published online 6 September 2006.

[9] Morris GM, Goodsell DS, Huey R, Lindstrom W, Hart WE, Kurowski S, Halliday S, Belew R, and Olson AJ. *AutoDock* v4.2. http://autodock.scripps.edu/. 2010.

[10] Drug Bank. *Zanamivir*. http://www.drugbank.ca/drugs/APRD00378.

.

[11] Govorkova EA et al. Comparison of efficacies of RWJ-270201, zanamivir, and oseltamivir against H5N1, H9N2, and other avian influenza viruses. *Antimicrobial Agents and Chemotherapy* 45 (2001), 2723-2732.

[12] Huey R and Morris GM. *Using AutoDock 4 with AutoDock Tools: A Tutorial.* 8 January 2008. http://autodock.scripps.edu/.

[13] Cheng Y and Prusoff WH. Relationship between the inhibition constant (K_i) and the concentration of inhibitor which causes 50 per cent inhibition (I_{50}) of an enzymatic reaction. *Biochemical Pharmacology* 22 (December 1973), 3099–3108. doi:10.1016/0006-2952(73)90196-2.

[14] Xu X, Zhu X, Dwek RA, Stevens J, and Wilson IA. Structural characterization of the 1918 Influenza virus H1N1 neuraminidase. *Journal of Virology* 82 (November 2008), 10493-10501. http://www.pdb.org/pdb/explore/explore.do?structureId=3BEQ

Coalescing Multiple Robots With An Evolutionary Method

Fang Tang and Johnny Yu
Computer Science Department
California State Polytechnic University, Pomona
Pomona, CA, 91768
Email: {ftang|jhyu}@csupomona.edu

Abstract— **In a heterogeneous multi-robot team, a task solution is highly dependent on the available collection of robots and their sensing and computational capabilities. We are interested in the problem of automatic synthesis of task solutions. Previous solutions to this problem include a heuristic search on the whole solution space or using an offline genetic algorithm for generating schema chunks and then search for solutions based on the chunks. In this paper, we explore the practicability of an alternative search algorithm, an evolutionary based search algorithm, to configure complete team solutions offline. Through experiments, we show that our genetic algorithm can generate complete solutions with similar qualities for a relatively small team size.**

I. Introduction and Related Work

When dealing with heterogenous multi-robot teams, one issue is particularly challenging: determining the appropriate teaming behaviors to accomplish a task when the task solution is dependent on the available collection of robots and their sensory, perceptual, and motor resources. Typical approaches to multi-robot teaming include the decomposition of task into subtasks or roles that can be accomplished by a single robot. Each robot is preprogrammed with the capability to accomplish a subtask or role. A pre-defined task decomposition tree defines the available multi-robot task solutions in advance of the mission. If the task changes or the team composition changes, then the task solutions need to be generated again to reflect the changes. Unlike the above approaches, we are particularly interested in approaches that automate the solution generation process based on the given task and the robot team composition.

Parker and Tang [4] developed an approach called ASyMTRe that automates the synthesis of multi-robot task solutions through software reconfiguration. Their approach provides the ability for heterogeneous robots to collaborate to find new solutions to tasks through various combinations of sensing, effectors, and behaviors that may be distributed across multiple robots. The ASyMTRe approach changes the abstraction that is used to represent robot competences from the typical "task" abstraction to a biologically-inspired "schema" abstraction. Each robot is preprogrammed with a set of schemas that define its capabilities. Schemas are still sensor and effector dependent, but are task independent. Thus they can be connected in different ways to accomplish different tasks. The ASyMTRe approach connects schemas at run time within or across robots to generate a solution to task. Finding a low-cost configuration of schemas for a multi-robot team is an NP-hard search problem [6]. The ASyMTRe search problem is formulated as a constraint satisfaction problem, and various heuristics are applied to direct the search toward solutions with smaller coalition sizes and higher task completion rate.

Y. Tang and Parker [7] extended the ASyMTRe architecture to SB-CoRLA that enables constructivist learning for multi-robot team tasks. They developed an Evolutionary Learning (EL) approach for the offline learning of schema chunks that could be saved and used later in an online search such as the ASyMTRe search algorithm. They found that the chunks of schemas that solve intermediate subproblems can be useful for future team tasks. In their later work, they feed the schema chunks generated through the EL approach to a centralized ASyMTRe search algorithm to find the solutions more efficiently.

In this paper, we explore the practicability of an alternative search algorithm, an evolutionary based search algorithm, to configure complete team solutions offline. A genetic algorithm is implemented to form coalitions for a team of heterogeneous robots to solve a single multi-robot task. Our approach borrows the concept of schemas from ASyMTRe as the basic building blocks to connect robots. Instead of a heuristic-based search on all different combinations of schemas on a large solution search space, we randomly connect schemas within and across robots at a certain rate and then apply the genetic algorithm. While this work is similar to the EL algorithm from SB-CoRLA, we focus more on the generation of a complete solution instead of schema chunks. Our implementation details are also different. The SB-CoRLA approach uses a graph based method to encode chromosomes, while we use value encoding, which uses a list of real numbers as representation of the chromosome which makes crossover and mutation more flexible.

The remainder of the paper is organized as follows. We define the problem in Section II and describe our genetic algorithm approach in Section III. In Section IV, experiments are performed to validate our approach and we also compare the results with the ASyMTRe approach. We finally conclude our work in Section V.

II. The Problem

The problem we address in this paper is the development of robot coalitions that solve a single multi-robot task. A multi-robot task is a task that may require multiple robots

to work closely with each to accomplish a goal. In a highly heterogeneous robot team, robots have quite different sensing and computational capabilities. Some robots with limited sensing and computational capabilities may not be able to accomplish a task even if they are duplicated multiple times. These simple robots need help from more capable robots to achieve the task. By working together, robots can share sensing and computational capabilities with each other. Coalitions are temporary organizations of robots that bring their diverse capabilities to solve the task that cannot be handled by a single robot. Our goal is to automate the coalition formation process.

Inspired by the schema theory [1], [3], we view robot capabilities as a set of environmental sensors (ES), as well as a set of perceptual schemas (PS), motor schemas (MS), and communication schemas (CS) that are preprogrammed into the robot at the design time. Unlike the above schema-based approaches, our approach autonomously connects schemas at run time instead of relying on pre-defined connections. According to the information invariants theory [2], the information needed to activate a certain schema or to accomplish a task remains the same regardless of the way that the robot may obtain or generate it. We can therefore label inputs and outputs of all schemas with a set of information types, such as *self global positioning data*. Two schemas can be connected if their input and output information labels match. Thus, schemas can be autonomously connected within or across robots based upon the flow of information required to accomplish a task. When schema connections are made across robots, we say that a coalition is formed between the robots, within which robots need to share information and tightly collaborate with each other to accomplish a task. With the run time connection capability, task solutions (schema connections) can be configured in many ways to solve the same task or can be reconfigured to solve a new task.

Given a set of n robots and a task *T*, the solution configuration problem can be represented as (R, T, U), where $R = \{R_1, R_2, \cdots, R_n\}$ is the set of n robots, $T = \{MS_1, MS_2, \cdots\}$ is the set of motor schemas that define the group-level task to be achieved, along with application-specific parameters as needed, and U provides utility information to be defined later. A robot, R_i, is represented by $R_i = (ES^i, S^i)$. ES^i is a set of environmental sensors that are installed on R_i. S^i is the set of schemas that are pre-programmed into R_i at design time. A schema can be activated if and only if its input can be obtained from the output of another schema or sensor. A set of *Connection Constraints* are used to regulate the connections between schemas. As shown in Table I, the constraints specify the restrictions on correct connections between various schemas.

Given the above information, the problem (R, T, U) has a solution if and only if for each $R_i \in R$ and for all $MS_j \in T$, the inputs of MS_j are satisfied, along with all the inputs from the schemas that feed into MS_j. A complete solution provides a way of connecting schemas for the whole team such that each robot is able to accomplish the task *T*.

TABLE I
CONNECTION CONSTRAINTS FOR SCHEMAS

Sensor/Schema	Input Sources:	Output Feeds into:
ES	Sensor Signals	PS
PS	ES, PS, or CS	PS, CS or MS
CS	PS, or CS	PS, CS, or MS
MS	PS, CS, or ES	Actuators

III. THE APPROACH

A. The Initial Population

In the initial stage, the genetic algorithm creates a population with *p* individuals. Each individual represents a potential solution, which may not be complete, to the team-level task. An intra-connection is first performed to select random robots for *j* times, and have their local schemas connected, while satisfying the connection constraints, which specify that two schemas can be connected if and only if one schema's input information type is equivalent to the output information type of another schema. A robot *r* is randomly selected from the team, and then a schema *a* is randomly selected from *r*. The intra-connection rate *j* is re-used as a loop that iterates *j* times, connecting schema *a* with another random schema *b* on *r*. By having schema *a* make *j* different tries to connect with a different schema *b*, we increase the chance of having more successful connections between the local schemas. Afterwards, an inter-connection takes place where different robots will try to have their communication schemas connected using the same strategy as mentioned above with a different connection rate *k*. Similarly, schema connections are governed by connection constraints, so after an individual is generated, it has to be validated before it is accepted into the population.

B. Encoding of An Individual

Each individual in the population is implemented as an adjacency matrix. The matrix table holds all different possible connections for all the schemas within the robot team. The row in the matrix represents the schemas that pass information, while the column represents schemas that receive information. Each cell in the matrix has a value encoding: 0, 1 and 2. The value "1" is used for a successful intra-connection which is two connected schemas on the same robot, "2" for a successful inter-connection which is two communication schemas between two robots, and "0" for no connection.

C. The Fitness Function

Once the initial population is generated, the genetic algorithm gives each individual a quality value computed by a fitness function. We apply the same fitness function as described in SB-CoRLA.

$$F = w_c \cdot (c/c_{max}) + w_x \cdot x + w_q \cdot (q/q_{max}) + w_u \cdot (u/n). \quad (1)$$

Here, c stands for cost, x for complexity, q/q_{max} is the percentage of information types required by the goal that are fulfilled, and *u/n* is the percentage of robots that can

achieve their goals. Each schema is assigned a cost. The lower the cost of the aggregated schemas, the better the solution. Additionally, an inter-connection is considered to be more complex than an intra-connection as it requires remote communication and the tight physical collaboration between robots. The complexity of the solution is determined by how many schemas are activated.

A goal is achieved when the set of motor schema(s) required to perform the task is activated on every robot. In order for a schema to be activated, it must have the necessary information inputs satisfied. There are two types of inputs to a schema: "AND" and "OR" connections. An "AND" connection is where a schema accepts multiple inputs and all of them must be satisfied. An "OR" connection is where an input must have at least one connection in order for the schema to be activated. We used a backward chaining algorithm to detect whether a goal is achieved. In this backward chaining process, the chain must reach all the way to environmental sensor(s) which do not require other types of inputs except for the environment. In the fitness function, the value q/q_{max} is to determine how close a goal schema is from activation.

A better solution usually indicates that there is higher percentage of information types required by the goal fulfilled, a higher percentage of robots that can accomplish the task, and the solution is of lower cost and higher success probability. We say that a complete solution exists when every robot in the team is able to accomplish its goal.

D. The Genetic Algorithm

In our genetic algorithm, we maintain a population and generate only *X* offsprings which replace the *X* weakest individuals in the parent population. Once each individual is assigned a fitness value, a roulette-wheel selection process will choose two chromosomes from the population for cross over and mutation. In a roulette-wheel selection, an individual's chance of being selected is proportional to the quality of its fitness. Each chromosome has a probability of *current fitness / sum of all fitnesses*. Just like spinning the roulette-wheel, a random number if generated between 0 and 1, and whichever chromosome's probability has that random value is selected for cross over and mutation.

When two individuals are selected, the algorithm performs cross over with a very high probability (ninety percent of the time). The current implementation performs a single point cross over where an exchange point is randomly generated between 0 and the length of an individual. The first chromosome will contribute the first segment starting from index 0 to the exchange point, and the second chromosome will provide the other half starting from the exchange point plus one to the end. The two segments are concatenated to become the first offspring. The second one concatenates the second segment of the first chromosome with the first segment of the second chromosome.

Another individual is selected from the roulette-wheel for mutation. The chance for mutation is much lower than cross over because we want to avoid having excessive jumping away from a potential hill, while still having a chance to jump out of a local peak when the algorithm gets stuck. When mutation is performed, the algorithm will randomly select a mutation point between 0 and the length of an individual. It will then flip into the adjacency matrix and try to connect the two schema cells. The mutated chromosome has to be validated before being accepted into the new population pool. When a mutated individual fails the validation, the algorithm will go back to the selection process to find another chromosome for mutation.

The algorithm will repeat the selection and reproduction process until the *X* weakest individuals from the population are replaced. Each offspring will have their fitness value calculated immediately. Once the new population includes *X* offspring, a new generation starts and the algorithm will rank each chromosome based on its fitness value and check if the chromosome with the best fitness is a solution. If a solution is not found, then the process will continue for the new population until one of the following conditions is met:

- The complete solution is found.
- The maximum generation has been reached.
- The fitness value has not improved within a minimum number of generations.
- The algorithm execution time exceeds a predefined deadline (in our experiment, one hour).

If a complete solution is not found, then the best individual will be displayed, which might contain some robots with solutions and others with partial solutions.

IV. Experiments

To validate the correctness and efficiency of our genetic algorithm, we performed experiments to compare the results of our genetic algorithm with the ASyMTRe approach.

A. Experiment Description

Consider the following multi-robot navigation example, where a team of heterogeneous robots need to navigate from their current positions to some goal positions. Each robot has a set of sensors and pre-programmed schemas. The environmental sensors are: `GPS`, omnidirectional camera (`camera`) and sensors for communication (`comm`), providing up to 2^3 different combinations of robot capabilities. We assume: a robot with a GPS can estimate its current global position in the environment; a robot with a camera can estimate the relative position of another robot in the environment, as long as the other robot is within its sensing range; and a robot has the computational ability to convert a relative position to a global position.

We also assume that the following schemas are pre-programmed on the robots: PS_1, which estimates its *self global position* using `GPS`; PS_2, which estimates the *relative position* of another robot using `camera`, and fiducial marker; PS_3, which estimates self global position according to another robot's global position and relative position; PS_4, which estimates the global position of another robot according to its own global position and the estimated relative position of the other robot; CS_i, which transfers information between robots;

TABLE II
INPUT AND OUTPUT INFORMATION TYPES FOR VARIOUS SCHEMAS AND THEIR CORRESPONDING SENSING COSTS AND SUCCESS PROBABILITIES

Type	Description
F_1	Self_Global_Position
F_2	Other_Global_Position
F_3	Other_Relative_Position
F_4	Motor_Commands

Schema	Input	Output	Cost	Probability
PS_1	GPS	F_1	1	95%
PS_2	Camera	F_3	2	95%
PS_3	F_2 and F_3	F_1	1	100%
PS_4	F_1 and F_3	F_2	1	100%
CS_1	F_1	F_2	4	100%
CS_2	F_2	F_1	4	100%
CS_3	F_1	F_1	4	100%
CS_4	F_2	F_2	4	100%
MS_1	F_1	F_4	1	100%

TABLE III
GENETIC ALGORITHM PARAMETERS AND DEFAULT VALUES

Description	Default
Population Size	300
Number of individuals selected for reproduction	15
Maximum number of generations	130
Maximum number of generations without improvement	20
Probability for Cross Over	90%
Probability for Mutation	90%
Intra-Robot Connection Rate	80
Inter-Robot Connection Rate	30
Weight for aggregated cost of active schemas	0.2
Weight for complexity	0
Weight for percentage of information types fulfilled	0.2
Weight for percentage of robots that achieve goals	0.6

and MS_1, which calculates motor commands that lead the robot toward the goal. We define the task, $T = \{MS_1\}$, meaning that MS_1 should be active on all robots.

In Table II, we define the set of information types *F* and label the input and output information for each schema used in this application, as well as their corresponding sensing costs and success probabilities. In our experiment, we consider four types of robots as shown in Table IV. They are highly heterogeneous based on their sensing and computational capabilities. Finally, in Table III, we list the set of parameters used in our genetic algorithm and their default values.

B. Experimental Results

We first validate our genetic algorithm by comparing it with ASyMTRe on the following three test cases.

1) Test Case 1: The robot team is composed six type II robots (with GPS and comm). Both ASyMTRe and the genetic algorithm generate the solutions as illustrated by the first schema connections shown in Figure 1. All robots can successfully complete the task by themselves since they all have GPS sensors to localize. The genetic algorithm took 562 seconds while the ASyMTRe took 1.206 seconds.

TABLE IV
FOUR DIFFERENT ROBOT TYPES USED IN OUR EXPERIMENTS

Robot	Sensors	Schemas
Type I	GPS, Camera, comm	PS_1, PS_3, PS_4, CS_i, MS_1
Type II	GPS, comm	PS_1, CS_i, MS_1
Type III	Camera, comm	PS_3, PS_4, CS_i, MS_1
Type IV	Comm	CS_i, MS_1

2) Test Case 2: The robot team is composed of two type I robots (with GPS, camera and comm) and four type IV robots (with comm). In this setup, a more capable type I robot can help a simple type IV robot to localize and thus navigate in the environment. With ASyMTRe, two coalitions of size three are formed such that each type I robot collaborate with the other two type IV robots to navigate using the second schema connections shown in Figure 1. With the genetic algorithm, two coalitions are also formed, with one type I robot helping four type IV robots and the other type I robot accomplishes the task just by itself. We can see here that the coalitions formed with the genetic algorithm are not as balanced as ones generated with ASyMTRe. This is because ASyMTRe has many other parameter settings that ensure that robots work in a non-super-additive environment [5] by limiting the number of robots that one can help and the maximum coalition size. In this case, ASyMTRe took .207 seconds while the genetic algorithm took 443 seconds to finish.

3) Test Case 3: The robot team is composed of two robots of type II, two robots of type III and two robots of type IV. In this setup, a type II robot can help a type III robot to localize, and then a type III robot can help a type IV robot to localize. The robots form a chain of help and information exchange in order to accomplish the task. The ASyMTRe approach generates a solution such that two coalitions of size three are formed, with the solution described above as the last solution in Figure 1. With the genetic algorithm, two coalitions are formed. In one coalition, a type II robot navigates by itself. In the other coalition, a type II robot helps two type III robots, while one of the type III robots helps the other type IV robots to navigate. For similar reasons as described in our test case 2, the coalitions generated by the genetic algorithm are not as balanced as ones generated in ASyMTRe. In this case, ASyMTRe took .296 seconds while the genetic algorithm took 540 seconds to finish.

We have also experimented with other robot team sizes, such as 8, 9, 10 and 12 with similar team compositions as in the above three test cases. Most of the teams are able to accomplish the tasks. Fitness values show signs of increase as the algorithm progresses through each generation. Beyond the above team size, the genetic algorithm either cannot generate a complete solution or takes too long to complete.

The genetic algorithm is sensitive to different parameter

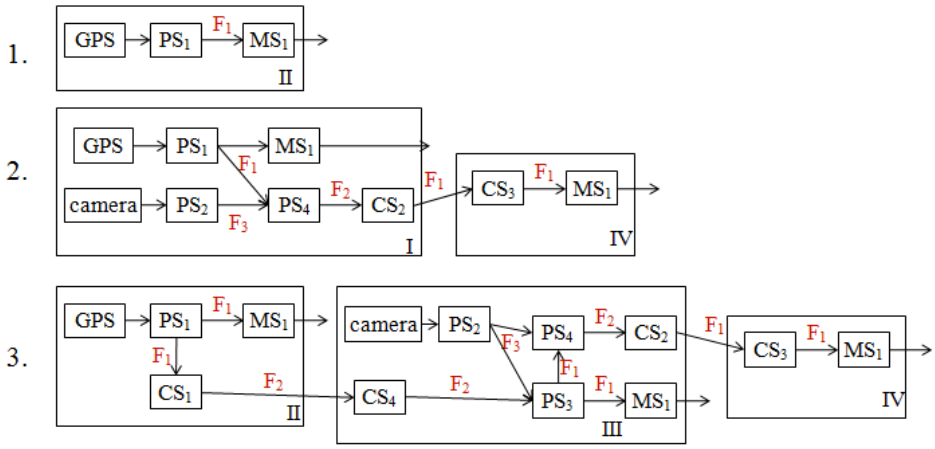


Fig. 1. Three possible ways to connect schemas within or across robots to accomplish the navigation task.

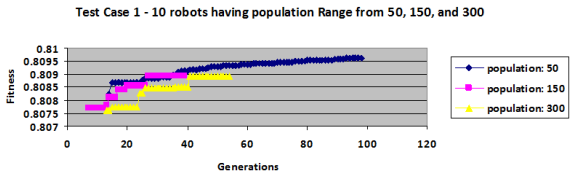


Fig. 2. Test Case 1 with various population sizes.

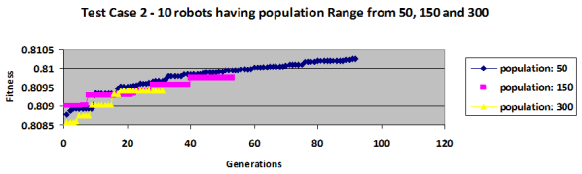


Fig. 3. Test Case 2 with various population sizes.

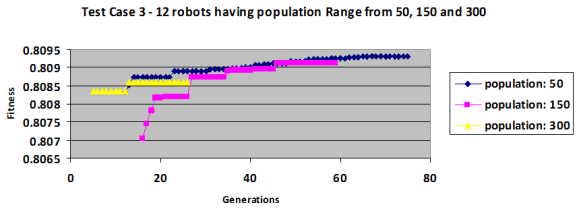


Fig. 4. Test Case 3 with various population sizes.

settings. The following experiments are performed to test its sensitivity given different population sizes. The population size is altered from 50, to 150 and 300 to see how it affects the evolutionary process (see Figures 2, 3, and 4). The results indicate that a lower population has a greater evolution compared with a larger population. The reason for this is because the small initial population does not cover as wide of the search spectrum compared with large initial population. So the individuals have more opportunities to cross over and mutate to a higher quality. On the other hand, a large initial population covers a wide range of the search spectrum and so it might not need much evolution to find the best individual. However, if a large population is specified, it is very similar to brute forcing all the possibilities in the search space. For larger populations, the fitness remains stable and it rapidly jumps and evolves sharply between two generations. Additionally, the figures show that a lower population can evolve and achieve a better fitness value compared to a larger population.

V. Conclusion and Future Work

Our experiments have shown that our genetic algorithm is able to generate complete solutions for solving multi-robot teaming problem with similar qualities as the ASyMTRe approach. However, the algorithm took significantly longer time than the ASyMTRe approach to complete and it is only appropriate for robot teams with relatively small size (around 10). Overall, the genetic algorithm cannot be a replacement to the ASyMTRe approach as its speed and solution quality are not superior over ASyMTRe. Our future work includes the improvement of the fitness function that includes other aspects, for example, a constraint on the coalition size. We will also look into other applications where a genetic algorithm might be more favorable than the ASyMTRe approach.

References

[1] R. C. Arkin. Motor schema based navigation for a mobile robot: an approach to programming by behavior. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 264–271, 1987.

[2] B. R. Donald. Information invariants in robotics. *Artificial Intelligence*, 72:217–304, 1995.

[3] D. M. Lyons and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation*, 5(3):280–293, 1989.

[4] L. E. Parker and F. Tang. Building multi-robot coalitions through automated task solution synthesis. *Proceedings of IEEE*, 2006. Special Issue on Multi-Robot Systems.

[5] O. Shehory. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[6] F. Tang and L. E. Parker. ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2005.

[7] Y. Tang and L. E. Parker. Towards schema-based, constructivist robot learning: Validating an eolutionary search algorithm for schema chunking. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2008.

Parameter Analysis for Differential Evolution on Loop Flow Problem in Power System

Gulcihan Ozdemir Dag
Informatics Institute
Istanbul Technical University
Email:gulcihan.ozdemir@be.itu.edu.tr

Mustafa Bagriyanik
Electric - Electronics Engineering
Istanbul Technical University
Email:bagriy@elk.itu.edu.tr

***Abstract*—In electrical power systems preventing and regulating the loop flow phenomena is very important issue especially after the de-regulation. The problem should be solved very efficiently. We have formulated the loop flow problem in fuzzy environment, as a multi-objective optimization problem using fuzzy set theory and fuzzy decision making rules. Then the resulted single objected optimization problem is solved using differential evolution (DE). Then a sets of simulations are done to figure out the most efficient parameter values of DE to fit our problem. DE is one of the evolutionary search methods. The parameter settings play an important role on reducing the required time and getting better solution to the problem. We applied our loop flow method to IEEE 30 bus test system and presented the results.**

***Index Terms*—Differential evolution, fuzzy set, fuzzy decision-making, interconnected power systems, loop flows, unscheduled flows.**

I. Introduction

The high power losses, low efficiencies, or the long path power travelling through (occupying transmission lines) before arriving to the loads have not been problem in government controlled power systems. Since rising costs of electrical energy were directly adjusted to the customers' bill or were partly subsidized by the governments the loop flows inherited in interconnected power systems (or the heat loses due to loop flows) was not seen as a serious problem [1]. After the privatization (de-regulation) the issues such as; how much power flows on which transmission lines, which company uses the other's transmission lines and/or the amount and the time of the transmission line usage have become important. If a system runs into a problem due to a heavy transmission line usage it is important to identify the responsible parties (systems causing unscheduled power flows). The path electrical power takes depends on the physical laws. That is, Kirchhoff's current and resistance laws determine the path for the electrical power to flow. It takes the shortest path (in terms of resistivity) instead of a contracted path.In this case a third party between a buyer and a seller of electrical energy may come into the picture. In such a case the question becomes who is to pay for the transmission line usage between a seller and a buyer [2], [3].

Unscheduled flows that refer to the deviation of actual electric power flows in transmission circuits from the scheduled (expected) power flows, may result in blackouts and affect cross border trading in the electricity markets. Therefore, unscheduled flows which are also termed inadvertent interchange or loop flows, should be managed/controlled to improve both the operation conditions of the electric network and the market. The effects of parallel paths in system network topology and a survey to explore the present state of practices used to determine transfer capability issues are well studied in [4], [5]. Suryanarayanan et al, proposes an approach based on Lp-norms to estimate the unscheduled flows occurring in a wide area interconnected system [6], [7].

In recent years, there have been a lot of applications of fuzzy set theory to various power system problems [8], [9], [10], [11]. In the past power system optimization problems were dealt with using non-linear and linear programming methods. The optimization problems under an uncertain environment can be reformulated using fuzzy sets. Many interesting applications of fuzzy sets in the optimization of the power system operating and planning stages have been reported.

Differential evolution method was introduced by Price and Storn [16] in 1995. It has gained popularity by years, and has been applied to various scientific problems. Some examples of power systems applications are reactive power optimization [19], power systems planning [20], power system transfer capability assessment [21], and power plant control [22], etc.

In this study, a multi-objective optimization approach based on fuzzy decision making and differential evolution is proposed to manage unscheduled flows. We handle the problem in a fuzzy environment since in practice, the small variations of power systems variables (bus voltages, line currents etc.,) from their limit values can be tolerated, and this can help to obtain one of the best solutions to the problem.

In the next section, a summary regarding unscheduled flows is given. In Section III and IV, the basic principles of fuzzy decision making and differential evolution are introduced briefly. In Section V, the implementation of the proposed approach is described in detail. The simulation results are provided and discussed in the subsequent section. Finally conclusions are provided.

II. Loop Flows(Unscheduled flows)

In an interconnected transmission network, when some amount of the scheduled power flows through an adjacently connected system, a loop flow phenomenon occurs. That is, the loop flow is the difference in between the actual flow and

the scheduled flow in a particular path. It is also referred to as the parallel path flow, unscheduled flow or circulating flow. The main reason of this phenomenon is that the Kirchhoff's laws that determine the path for the electrical power to flow. Loop flow could exist in an interconnected power transmission network depending on the system topology and operating conditions.

It has been known that, without exceeding power transfer limits of lines (not overloaded) and disturbing system reliability, for the sake of efficient operation, neighbouring systems can buy and sell power to and from each other through the transmission system that exists between them. But unscheduled flows affect the operation of the electric power system and the market.

Unscheduled flows can play an important role in causing blackouts and creating the cross-border bottlenecks, they need to be managed.

III. Fuzzy Decision Making

This section summarizes the basic concepts of fuzzy sets used for the fuzzy model, and offers brief information about the multi-objective fuzzy model and the essentials of the techniques for solving the multi-objective fuzzy model.

Fuzzy set theory is a generalization of traditional crisp set theory. The idea is to replace the concept that each variable has a precise value by the fuzzy concept that each variable is assigned a degree of membership for each possible value of the variable. A fuzzy set in the universal set U, is a generalization of a classical set, and it can be characterized by a membership function, $\mu(x)$, that takes real values in the continuous interval [0,1]. A fuzzy set A, in U can be represented by an ordered pair composed by a generic element x and its membership value, that is,

$$A = \{(x, \mu_A(x)), x \in U\} \tag{1}$$

A fuzzy set can be characterized by a membership function to map a parameter to membership grade between the scaled intervals. For modelling the objectives and the constraints in fuzzy environment, initial step is the fuzzification process, are assigned membership values using fuzzy membership functions. The closer the membership is to one the better the solution is for that objective or constraint. Fuzzy sets representing the objectives and constrains may vary considerably. The membership functions may be similar in the sense that numbers outside the interval are excluded from the associated fuzzy sets. Generally a triangular membership function is selected for representing fuzzy sets. The other most common shapes are trapezoidal, exponential, and Gaussian.

In fuzzy decision making fuzzy objective functions and constraints can be characterized by the membership function of the fuzzy objectives, $\mu_g(x)$ and the membership function of the fuzzy constraints, $\mu_c(x)$, respectively. The optimal solution, which is the fuzzy decision μ_D, is given as an intersection of the fuzzy sets describing the constraints and the objectives. Using the membership functions, the overall membership function value is obtained as

$$\mu_D = \min\lfloor \mu_g(x), \mu_c(x) \rfloor \tag{2}$$

The optimal solution is defined to be the one with the highest degree of membership, and thus the optimization problem becomes that of maximizing the satisfaction with the solution, subject to the crisp and fuzzy constraints [12].

IV. Differential Evolution

Differential evolution (DE) is a population based, inherently parallel, heuristic search method. It is powerful to handle non linear and non differentiable functions.

DE procedure is similar to other evolutionary algorithms, such as genetic algorithms, particle swarm optimization, tabu search, simulated annealing, etc. Main parts of the algorithm is shown below.

Algorithm 1 Main Parts of DE Algorithm

```
Initialization
Evaluation
repeat
  Mutation
  Crossover (Recombination)
  Evaluation
  Selection
until Stopping criterion is satisfied
```

Since, DE is a population based method, at every iteration, it operates on a population of Np candidate solution vectors. The first step of the algorithm generates a random solution vector. A population can be represented as shown below.

$$P^i = [X_1^i, \cdots, X_{N_p}^i] \tag{3}$$

where i represents the iteration number, and X is a candidate solution vector. Each solution candidate vector consists of n objective function parameters, where n is the number of unknowns in the function to be optimized. The solution candidates must be initialized within the lower and upper bounds of the unknowns.

The second step of the algorithm creates mutant vectors, by adding a weighted difference vector of two randomly indexed vectors to the third one. There are several versions for this process [18]. Famous scheme DE/rand/1 process can be mathematically represented as shown below.

$$x_j^{'i} = x_{r3}^i + F(x_{r1}^i - x_{r2}^i) \tag{4}$$

where r_1, r_2, and r_3 are randomly selected integers from 1 to N_p and $j \neq r_1 \neq r_2 \neq r_3$. The mutant vector is represented by $x_j^{'i}$. F is the scaling factor, which has effect on the difference vector, within the range of [0,2].

The third step creates trial vectors by mixing the parent vectors, and the mutant vectors. Mathematical representation for this process is given below.

$$x_j^{\text{trial}(G)} = \begin{cases} x_{kj}^{'(i)} & \text{if rand}(0,1) \leq (CR) \text{ or } k = q, \\ x_{kj}^{(i)} & \text{otherwise.} \end{cases} \tag{5}$$

where, q is a random parameter chosen for each j, CR represents crossover constant, within the range of [0,1], and rand is a randomly generated number between 0 to 1.

Decision of inclusion of the trial vector in the next generation is made in the selection step, by comparing the fitness values of the trial vectors with the associated target vectors. This process can be represented as shown below.

$$x_j^{(i+1)} = \begin{cases} x_j^{\text{trial}(i)} & \text{if} f(x_j^{\text{trial}(i)}) \leq f(x_j^{(i)}), \\ x_j^{(i)} & \text{otherwise.} \end{cases} \quad (6)$$

Finally if the stopping criterion is met the algorithm stops otherwise it goes to the second step.

V. Problem Formulation

A classical general optimization problem formulation is given below.
Minimize

$$f(x, u)$$

Such that

$$g(x, u) = 0$$
$$h(x, u) \leq 0$$

where x represents system variables, u represents control variables for the objective function $f(x, u)$ with the equality constraints $g(x, u) = 0$ and the inequality constraints $h(x, u) \leq 0$.

In our formulation, the objective function is the minimization of fitness function

$$fitness = \frac{1}{1 + \mu_D} \quad (7)$$

which is the maximization of the minimum satisfaction value of fuzzy memberships . In the fuzzy environment both the objective functions (minimization of both total active losses and total reactive losses and the scheduled line flow on a contracted path) and the constraints (voltages remaining within the limits, line flows remaining within the limits etc.,) are modelled as fuzzy sets. The intersection of both membership sets, μ_c and μ_g, is the overall satisfaction, μ_D needs to be maximized.

We use control variables such as tap changing transformers tap ratios, generation bus voltages, active power generations, and if available reactance of series compensation with their upper and lower limits to create candidate solutions to established the initial population for DE. Each candidate solution in the population is evaluated by load flow program. The results of load flow (voltages, line flows, loses, etc.) are passed to fuzzy decision making process, where a membership value is assigned for all constraints and objectives. The minimum of those membership values is then tried to be maximized by DE. The process continues until all population is exhausted or a pre-set number of iterations is reached.

The exponential membership function, see 2, has been selected for the fuzzification of the unscheduled flows since our earlier study showed that more satisfying results can be obtained using this kind of membership function [13], [14], [15]. A trapezoidal membership function could have also been used.

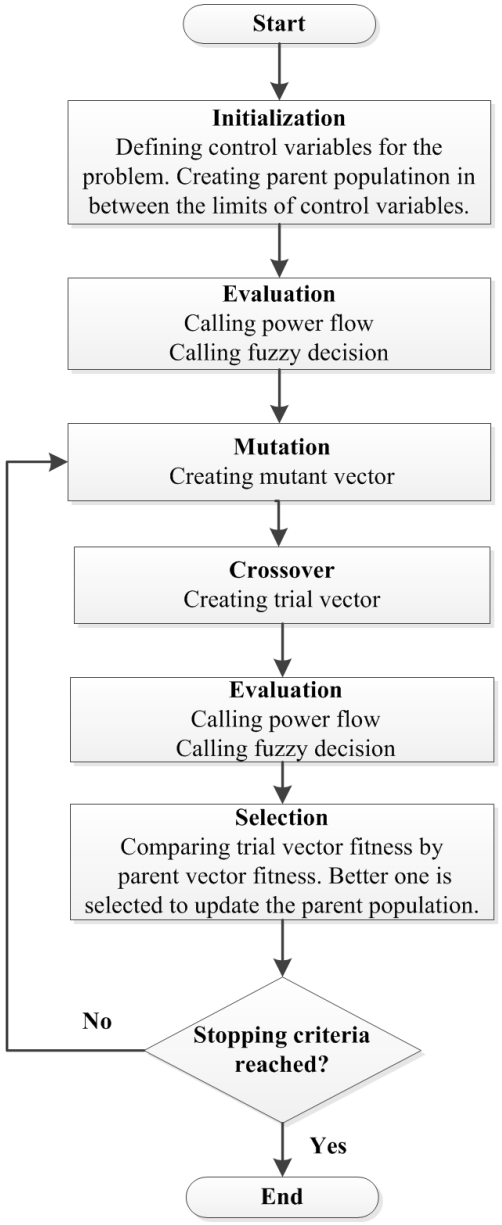


Fig. 1. Flow chart of the problem.

The function in 2 can be described by four parameters (a, b, c, d) with four breakpoints of the shape. The membership function $\mu_{g,ij}(P_{ij})$ belongs to the MW flow (line flow) through the line between bus i and bus j. The system operators taking into account the amount of the scheduled power flowing through the contracted paths can determine the four parameters of the function. The membership function $\mu_{g,ij}(P_{ij})$ is defined as

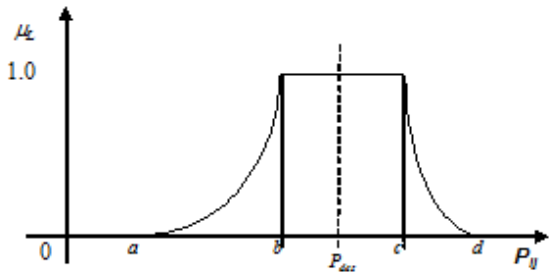


Fig. 2. The fuzzy memberships function for the loop MW flows (the exponential form).

$$\mu_{g,ij}(P_{ij}) = \begin{cases} \frac{P_{ij}-a_{ij}}{b_{ij}-a_{ij}} & a_{ij} < P_{ij} < b_{ij} \\ 1 & b_{ij} < P_{ij} < c_{ij} \\ 1 + \frac{d_{ij}-P_{ij}}{d_{ij}-c_{ij}} & c_{ij} < P_{ij} < d_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where, $a_{ij} < b_{ij} < c_{ij} < d_{ij}$ must hold.

VI. Simulation Results

The chosen values of the parameters of DE is as follows.
F will be vary in between [0.0,1.0]
CR will be vary in between [0.0,1.0]
$n = 14$
$Np = 100$
$ub = [1.1, 1.1, 1.1, 1.1, 1.06, 1.06, 1.06, 1.06, 1.06, 140, 100, 100, 100, 100];$
$lb = [0.9, 0.9, 0.9, 0.9, 0.94, 0.94, 0.94, 0.94, 0.94, 0.0, 0.0, 0.0, 0.0, 0.0];$
ub is the upper bound of the unknowns (power system parameters).
lb is the lower bound of the unknowns (power system parameters).

The paths chosen to control power flows are given below:

- Path 1, branch between buses 2 and 6,
- Path 2, branch between buses 2 and 5,
- Path 3, branch between buses 6 and 7.

The fuzzy membership function parameters (a,b,c,d) are chosen as 50,59,61,70 respectively for the path 1.
The fuzzy membership function parameters (a,b,c,d) are chosen as 55,59,61,65 respectively for the path 2.
The fuzzy membership function parameters (a,b,c,d) are chosen as 15,21,23,29 respectively for the path 3.

The averaged values are the results of twenty runs.
According to the table I and II we see that the solutions reached from the 0.4 value of F is better. But the deviations from the targeted values are high.
According to the table III and IV we see that the solutions reached from the 0.4 value of F is the best one.
According to the table V and VI we see that the solutions reached from the 0.2, 0.4 and 0.6 values of F are not really different than each other. The iteration number for the case 0.4 of F is the lowest.
According to the table VII and VIII we see that the solutions reached from the 0.2, 0.4 and 0.6 values of F are not really different than each other. The iteration number for the case 0.6 of F is the lowest.
According to the table IX and X we see that the solutions reached from the 0.4 and 0.6 values of F are fairly good. Iteration number is lower for the 0.4 of F then the 0.6 of F. The 0.4 value of F provides better fuzzy membership values for the paths as well.
Overall conclusion is that we have obtained a value as 0.4 for the parameter F and two values as 0.6 and 1.0 for the parameter CR. When the value of CR is high DE might converge fast and arrive at a local minimum. That is why we concluded the parameter settings for DE as 0.4 for the F and 0.6 for the CR.

After parameter analysis of DE our problem is solved using the best case parameter to compare to base case solution of the system. The results are given from the tables XI and XII.

TABLE I
AVERAGED OBJECTIVE VALUES.

CR=0.2	fitness	generation	path 1 (MW)	path 2 (MW)	path 3 (MW)
F=0.2	0.7099	344.2857	47.2075	51.2710	22.3625
F=0.4	0.7026	288.4286	53.6023	57.0964	22.0552
F=0.6	0.7437	224	44.5404	54.6725	27.1232
F=0.8	0.7679	203.7143	35.9105	43.1106	21.5093
F=1.0	0.7918	213.5714	12.7435	25.7443	25.3571

TABLE II
AVERAGED FUZZY MEMBERSHIP VALUES FOR OBJECTIVES AND CONSTRAINTS.

CR=0.2	min fitness	path 1	path 2	path 3	P_{loss}	Q_{loss}
F=0.2	0.3122	0.3357	0.6508	0.6055	0.5152	0.4315
F=0.4	0.3528	0.4928	0.6643	0.6934	0.4450	0.3978
F=0.6	0.1040	0.3655	0.2415	0.1946	0.2663	0.2550
F=0.8	0.0022	0.0109	0.0252	0.0459	0.3156	0.2761
F=1.0	0	0	0	0.0664	0.3447	0.2429

TABLE III
AVERAGED OBJECTIVE VALUES.

CR=0.4	fitness	generation	path 1 (MW)	path 2 (MW)	path 3 (MW)
F=0.2	0.7372	299.2857	52.7136	54.1790	19.4998
F=0.4	0.6919	335.2857	58.2337	59.7074	20.3662
F=0.6	0.7315	230.4286	52.3205	53.2260	21.1080
F=0.8	0.8504	209	38.4646	44.9840	23.0930
F=1.0	0.8213	201.7143	20.1798	51.5315	48.4237

VII. Conclusion

Regulating or controlling loop flow is formulated as a multi-objective problem subject to operational and electrical

TABLE IV
AVERAGED FUZZY MEMBERSHIP VALUES FOR OBJECTIVES AND CONSTRAINTS.

CR=0.4	min fitness	path 1	path 2	path 3	P_{loss}	Q_{loss}
F=0.2	0.3779	0.3837	0.6998	0.4184	0.3780	0.3791
F=0.4	0.4420	0.4649	1.0000	0.5307	0.4426	0.4426
F=0.6	0.1540	0.2419	0.5065	0.4138	0.2603	0.2643
F=0.8	0.0133	0.0237	0.0970	0.1774	0.2110	0.1834
F=1.0	0	0	0	0.0870	0.1569	0.1306

TABLE V
AVERAGED OBJECTIVE VALUES.

CR=0.6	fitness	generation	path 1 (MW)	path 2 (MW)	path 3 (MW)
F=0.2	0.6963	367.4286	58.1737	58.9501	20.1867
F=0.4	0.6915	252	58.2259	59.3840	20.3931
F=0.6	0.6994	296.8571	58.3525	60.3773	21.3662
F=0.8	0.8072	231	47.6865	57.5251	28.1167
F=1.0	0.8375	201	36.3017	45.1145	28.8744

TABLE VI
AVERAGED FUZZY MEMBERSHIP VALUES FOR OBJECTIVES AND CONSTRAINTS.

CR=0.6	min fitness	path 1	path 2	path 3	P_{loss}	Q_{loss}
F=0.2	0.4361	0.4377	0.9052	0.4411	0.4362	0.4363
F=0.4	0.4437	0.4611	1.0000	0.5553	0.4437	0.4451
F=0.6	0.4144	0.5245	0.9782	0.8188	0.4204	0.4150
F=0.8	0.1309	0.2873	0.3836	0.5651	0.2354	0.2366
F=1.0	0.0110	0.1313	0.3214	0.2840	0.2910	0.1686

TABLE VII
AVERAGES OBJECTIVE VALUES.

CR=0.8	fitness	generation	path 1 (MW)	path 2 (MW)	path 3 (MW)
F=0.2	0.6954	341.1429	58.1755	58.8678	20.2524
F=0.4	0.6904	314.2857	58.2037	59.116	20.2139
F=0.6	0.6996	260.2857	58.4694	60.2493	20.9606
F=0.8	0.7453	219.1429	55.7606	65.9365	27.9187
F=1.0	0.9104	203.8571	50.5513	51.8493	18.6307

TABLE VIII
AVERAGED FUZZY MEMBERSHIP VALUES FOR OBJECTIVES AND CONSTRAINTS.

CR=0.8	min fitness	path 1	path 2	path 3	P_{loss}	Q_{loss}
F=0.2	0.4381	0.4384	0.7539	0.4776	0.4383	0.4382
F=0.4	0.4481	0.4509	0.9925	0.4532	0.4482	0.4483
F=0.6	0.4174	0.5795	0.9548	0.7241	0.4223	0.4179
F=0.8	0.1810	0.2343	0.5679	0.8239	0.2868	0.2607
F=1.0	0	0.1325	0.0692	0.1429	0.1927	0.0938

constraints in fuzzy environment. After fuzzy decision resulted single objective optimization problem is solved by using differential evolution. The primary goal of our problem is to obtain the highest satisfaction level(maximizing the fuzzy membership value) to reach the targeted flow levels. The problem is tested for the many different cases of the parameters of DE to obtained the best solution to our problem.

TABLE IX
AVERAGED OBJECTIVE VALUES.

CR=1.0	fitness	generation	path 1 (MW)	path 2 (MW)	path 3 (MW)
F=0.2	0.8440	211.1429	52.5332	57.8209	23.1920
F=0.4	0.7062	213.8571	58.2072	59.9108	20.6898
F=0.6	0.6904	362.2857	58.2042	59.1771	20.2409
F=0.8	0.7117	238.7143	58.4986	60.0911	21.5948
F=1.0	0.8257	218.4285	53.6428	63.4230	30.1184

TABLE X
AVERAGED FUZZY MEMBERSHIP VALUES FOR OBJECTIVES AND CONSTRAINTS.

CR=1.0	min fitness	path 1	path 2	path 3	P_{loss}	Q_{loss}
F=0.2	0.1967	0.2104	0.4583	0.3823	0.3296	0.3036
F=0.4	0.4163	0.4537	1.000	0.7096	0.4208	0.4163
F=0.6	0.4479	0.4511	0.9860	0.4665	0.4480	0.4482
F=0.8	0.3568	0.6278	0.8173	0.9171	0.3899	0.3684
F=1.0	0.0773	0.2650	0.3486	0.4782	0.1585	0.1294

TABLE XI
AVERAGED OBJECTIVES AND CONSTRAINTS, (BEST FITNESS = 0.6915).

	path 1 (MW)	path 2 (MW)	path 3 (MW)	P_{loss} (MW)	Q_{loss} (MVAr)
test	58.2259	59.3840	20.3931	8.0104	35.0194
target	60.00	60.00	20.00	1.8x7.999	1.8x35.06
base case	52.68	37.80	5.05	7.999	35.06

TABLE XII
AVERAGED FUZZY MEMBERSHIP VALUES FOR OBJECTIVES AND CONSTRAINTS, 3 PATHS.

	min fitness	path 1	path 2	path 3	P_{loss}	Q_{loss}
test	0.4437	0.4611	1.0000	0.5553	0.4437	0.4451
base case	0.00	0.0016	0.00	0.00	0.4444	0.4444

REFERENCES

[1] J. N. Janssens and A. Kamagate, "Loop flows in a ring ac power system," *Elsevier, Electrical Power and Energy Systems*, vol. 25, pp. 591–597, 2003.

[2] H. G. Dreyer, C. B. Miller, and M. B. J. Spence, "Opportunities, benefits, and issues relating to the deregulation of the electric power industry," *IEEE Power Engineering Society, Summer Meeting*, vol. 2, 1999.

[3] R. Azami, M. S. Javadi, H. Monsef, and H. Lessani, "A methodology for active power transmission loss allocation in power markets," *International Review of Electrical Engineering (IREE)*, vol. 4, no. 4, pp. 577–582, Aug 2009.

[4] S. Suryanarayanan, R. G. Farmer, G. T. Heydt, and S. Chakka, "Estimation of unscheduled flows and contribution factors based on lp norms," *PAS*, vol. 19-2, pp. 1245–1246, may 2004.

[5] S. Suryanarayanan, G. T. Heydt, R. G. Farmer, and S. Chakka, "An estimation techniques to contribution factors for loop flows in an interconnected power system," *Electric power components and systems*, vol. 19, no. 2, may 2004.

[6] S. Suryanarayanan and G. T. Heydt, "Modification to contribution factor formula for unscheduled flow," *PAS*, vol. 23, no. 2, may 2008.

[7] S. Suryanarayanan, D. C. Montgomery, and G. T. Heydt, "Considerations for implementing tag schedules in transmission circuits," *PAS*, vol. 20, no. 1, Feb 2005.

[8] J. A. Momoh and K. Tomsovic, "Overview and literature survey of fuzzy set theory in power systems," *PAS*, vol. 10, no. 3, pp. 1676–1690, Aug 1995.

[9] T. Hiyama and K. Tomsovic, "Current status of fuzzy system applications in power systems." IEEE international conference on systems, man, and cybernetics, 1999.
[10] T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Its Applications*. Academic Press, 1992.
[11] D. Srinivasan, A. C. Liew, and C. S. Chang, "Applications of fuzzy systems in power systems," *Electric power systems research*, vol. 35, pp. 39–43, 1995.
[12] G. J. Klir and T. A. Folger, *Fuzzy Sets Uncertainty And Information*. Prentice-Hall International, Inc., 1988.
[13] G. O. Dag and M. Bagriyanik, "Preventing loop flows using fuzzy set theory and genetic algorithms," *IEEE PowerTech09, Bucharest, Romania*, 2009.
[14] G. O. Dag and M. Bagriyanik, "The effect of different fuzzy membership function forms on controlling loop flows," *The 44th International Universities Power Engineering Conference, UPEC 2009, University of Strathclyde, Glasgow, Scotland*, September 1-4 2009.
[15] G. O. Dag and M. Bagriyanik, "Controlling unscheduled flows using fuzzy set theory and genetic algorithms," *International Review of Electrical Engineering (IREE)*, vol. 5, no. 1, Jan-Feb 2010.
[16] R. Storn, and K. Price, "Differential Evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, ICSI, http://http.icsi.berkeley.edu/ storn/litera.html.
[17] R. Storn, and K. Price, "Minimizing the real functions of the ICEC 96 contest by differential evolution", *Int. Conf. on Evolutionary Computation*, Nagoya, Japan, May 1996.
[18] R. Storn, "Diferential evolution design of an IIR-Filter with requirements for magnitude and group delay", Technical Report TR-95-026, International Computer Science Institute, Berkeley, CA, June 1995.
[19] X. Zhang, W. Chen, C. Dai, and A. Guo "Self-adaptive differential evolution algorithm for reactive power optimization", Fourth International Conference on Natural Computation 2008. vol. 6., pp. 560–564 Oct. 2008
[20] G.Y. Yang, Z.Y. Dong, and K.P. Wong "A modified differential evolution with fitness sharing for power systems planning", IEEE Trans. on Power Systems. 2008. vol. 23(2)., pp. 514–522 May 2008
[21] K.P. Wong and Z.Y. YangDong " Differential evolution an alternative approach to evolutionary algorithm", 13th International Conference on Intelligent Systems Application to Power Systems (ISAP). Nov. 2005., pp.73–83
[22] J.H. Van Sickel, K.Y. Lee and J.S.. Heo " Differential evolution and its applications to power plant control", International Conference on Intelligent Systems Application to Power Systems (ISAP). Nov. 2007.

Simulated Docking of Oseltamivir with the 1918 Pandemic Strain Influenza A/H1N1 Neuraminidase Active Site

Jack K. Horner
P.O. Box 266
Los Alamos NM 87544 USA

Abstract

Neuraminidases are glycoproteins that facilitate the transmission of the influenza virus from cell to cell. The neuraminidase inhibitors osteltamivir and zanamivir are currently the most widely used anti-flu therapeutics. Oseltamivir was ineffective against the dominant H1N1 strains in the 2008 flu season and decreasingly effective against the dominant influenza H1N1 mutants in the US in the 2009 "Spring/Fall" pandemic. Here I provide a computational docking analysis of oseltamivir with the active site of the neuraminidase of the 1918 strain (A/Brevig Mission/1/18 H1N1). The docking uses a Lamarckian genetic algorithm. The computed inhibitor/receptor binding energy suggests that oseltamivir would not be effective against that strain.

Keywords: Influenza, H1N1, neuraminidase, oseltamivir

1.0 Introduction

Neuraminidases are glycoproteins that facilitate the transmission of the influenza virus from cell to cell. The most widely used anti-influenza therapeutic, oseltamivir (Tamiflu™, [4]), was ineffective against the dominant H1N1 mutants in the 2008 flu season and was decreasingly effective against the dominant influenza mutant (Influenza A/H1N1) in the US in the 2009 "Spring/Fall" pandemic ([7]).

In the World Health Organization serotype-based influenza taxonomy, influenza type A has nine neuraminidase-related sero-subtypes, and these subtypes correspond at least roughly to differences in the active-site structures of the flu neuraminidases. The subtypes fall into two groups ([3]): group-1 contains the subtypes N1, N4, N5 and N8; group-2 contains the subtypes N2, N3, N6, N7 and N9. Oseltamivir was designed to target the group-2 neuraminidases.

The available crystal structures of the group-1 N1, N4 and N8 neuraminidases ([1]) reveal that the active sites of these enzymes have a very different three-dimensional structure from that of group-2 enzymes. The differences lie in a loop of amino acids known as the "150-loop", which in the group-1 neuraminidases has a conformation that opens a cavity not present in the group-2 neuraminidases. The 150-loop contains an amino acid designated Asp 151; the side chain of this amino acid has a carboxylic acid that, in group-1 enzymes, points away from the active site as a result of the 'open' conformation of the 150-loop. The side chain of another active-site amino acid, Glu 119, also has a different conformation in group-1 enzymes compared with the group-2 neuraminidases (8]).

The Asp 151 and Glu 119 amino-acid side chains form critical interactions with neuraminidase inhibitors. For neuraminidase subtypes with the “open conformation” 150-loop, the side chains of these amino acids might not have the precise alignment required to bind inhibitors tightly ([8]). The active site

of the 1918 strain has the 150-loop configuration.

The difference in the active-site conformations of the two groups of neuraminidases may also be caused by differences in amino acids that lie outside the active site. This means that an enzyme inhibitor for one target will not necessarily have the same activity against another with the same active-site amino acids and the same overall three-dimensional structure ([17]).

2.0 Method

he general objective of this study is straightforward: to computationally assess the binding energy of the active site of crystallized 1918 pandemic strain neuraminidase with oseltamivir. Unless otherwise noted, all processing described in this section was performed on a Dell Inspiron 545 with an Intel Core2 Quad CPU Q8200 (clocked @ 2.33 GHz) and 8.00 GB RAM, running under the *Windows Vista Home Premium (SP2)* operating environment.

Protein Data Bank (PDB) 3BEQ is a structural description of most of the crystallized neuraminidase of Influenza A/Brevig Mission/1/18 H1N1 (the principal 1918 pandemic mutant). 3BEQ consists of two identical chains, designated Chain A and Chain B.

3BEQ was downloaded from PDB ([6]) on 31 January 2011. A PDB description of oseltamivir was extracted from PDB 2HU4 using Microsoft *Word*. The automated docking suite *AutoDock Tools* v 4.2 (ADT, [9]) was used to perform the docking of oseltamivir to the receptor. More specifically, in ADT, approximately following the rubric documented in [12]

-- Chain B, and the water in Chain A, of 3BEQ were deleted

-- Chain A's active-site was extracted. (3BEQ identifies the active site of Chain A as 14 amides: ARG118, GLU119, ASP151, ARG152, ARG156, TRP178, ARG224, GLU227, SER246, GLU276, GLU277, ARG292, ARG371, and TYR406.)

-- the hydrogens, charges, and torsions in the ligand and active site were adjusted using the ADT-recommended defaults and finally, the ligand, assumed to be flexible wherever that assumption is physically possible, was auto-docked to the active site, assumed to be rigid, using the Lamarckian genetic algorithm implemented in ADT.

The ADT parameters for the docking are shown in Figure 1. Most values are, or are a consequence of, ADT defaults.

```
autodock_parameter_version 4.2       # used by autodock to validate parameter set
outlev 1                             # diagnostic output level
intelec                              # calculate internal electrostatics
seed pid time                        # seeds for random generator
ligand_types C HD OA N               # atoms types in ligand
fld 3BEQ_receptor.maps.fld           # grid_data_file
map 3BEQ_receptor.C.map              # atom-specific affinity map
map 3BEQ_receptor.HD.map             # atom-specific affinity map
map 3BEQ_receptor.OA.map             # atom-specific affinity map
map 3BEQ_receptor.N.map              # atom-specific affinity map
elecmap 3BEQ_receptor.e.map          # electrostatics map
desolvmap 3BEQ_receptor.d.map        # desolvation map
move 3BEQ_Ligand.pdbqt               # small molecule
about 0.5292 81.1637 109.1143        # small molecule center
tran0 random                         # initial coordinates/A or random
axisangle0 random                    # initial orientation
dihe0 random                         # initial dihedrals (relative) or random
tstep 2.0                            # translation step/A
qstep 50.0                           # quaternion step/deg
```

```
dstep 50.0                          # torsion step/deg
torsdof 7                           # torsional degrees of freedom
rmstol 2.0                          # cluster_tolerance/A
extnrg 1000.0                       # external grid energy
e0max 0.0 10000                     # max initial energy; max number of retries
ga_pop_size 150                     # number of individuals in population
ga_num_evals 2500000                # maximum number of energy evaluations
ga_num_generations 27000            # maximum number of generations
ga_elitism 1                        # number of top individuals to survive to next
generation
ga_mutation_rate 0.02               # rate of gene mutation
ga_crossover_rate 0.8               # rate of crossover
ga_window_size 10                   #
ga_cauchy_alpha 0.0                 # Alpha parameter of Cauchy distribution
ga_cauchy_beta 1.0                  # Beta parameter Cauchy distribution
set_ga                              # set the above parameters for GA or LGA
sw_max_its 300                      # iterations of Solis & Wets local search
sw_max_succ 4                       # consecutive successes before changing rho
sw_max_fail 4                       # consecutive failures before changing rho
sw_rho 1.0                          # size of local search space to sample
sw_lb_rho 0.01                      # lower bound on rho
ls_search_freq 0.06                 # probability of performing local search on
individual
set_psw1                            # set the above pseudo-Solis & Wets parameters
unbound_model bound                 # state of unbound ligand
ga_run 10                           # do this many hybrid GA-LS runs
analysis                            # perform a ranked cluster analysis
```

Figure 1. ADT parameters for the docking in this study

3.0 Results

The interactive problem setup, which assumes familiarity with the general neuraminidase "landscape", took about 20 minutes in ADT; the docking proper, about 25 minutes on the platform described in Section 2.0 The platform's performance monitor suggested that the calculation was more or less uniformly distributed across the four processors at ~25% of peak per processor (with occasional bursts to 40% of peak), and required a constant 2.9 GB of memory.

Figure 2 shows the oseltamivir/receptor energy and position summary produced by ADT. The estimated free energy of binding is ~ -6.8 kcal/mol; the estimated inhibition constant, ~11 microMolar at 298 K.

```
MODEL        3
USER    Run = 3
USER    Cluster Rank = 1
USER    Number of conformations in this cluster = 4
USER
USER    RMSD from reference structure       = 127.033 A
USER
USER    Estimated Free Energy of Binding    =   -6.77 kcal/mol  [=(1)+(2)+(3)-(4)]
USER    Estimated Inhibition Constant, Ki   =   10.92 uM (micromolar)  [Temperature =
298.15 K]
USER
USER    (1) Final Intermolecular Energy     =   -8.86 kcal/mol
USER        vdW + Hbond + desolv Energy     =   -5.53 kcal/mol
USER        Electrostatic Energy            =   -3.32 kcal/mol
USER    (2) Final Total Internal Energy     =   -0.83 kcal/mol
```

```
USER    (3) Torsional Free Energy           =   +2.09 kcal/mol
USER    (4) Unbound System's Energy  [=(2)] =   -0.83 kcal/mol
USER
USER
USER
USER    DPF = 3BEQ.dpf
USER    NEWDPF move    3BEQ_Ligand.pdbqt
USER    NEWDPF about   0.529200 81.163696 109.114304
USER    NEWDPF tran0   8.551498 16.101909 -1.664349
USER    NEWDPF axisangle0     -0.077969 -0.447424 -0.890917 157.187877
USER    NEWDPF quaternion0    -0.076430 -0.438587 -0.873322 0.197761
USER    NEWDPF dihe0   -123.77 137.09 57.32 -80.84 72.77 -173.98 76.38
USER
USER                              x       y       z    vdW   Elec        q     RMS
ATOM      1  C2  G39 A 800      11.180  16.277  -1.152 -0.26 +0.07    +0.091 127.033
ATOM      2  C3  G39 A 800      10.774  17.200  -2.439 -0.19 +0.01    +0.050 127.033
ATOM      3  C4  G39 A 800       9.409  16.835  -3.177 -0.19 -0.03    +0.209 127.033
ATOM      4  C5  G39 A 800       8.339  16.597  -2.111 -0.26 -0.03    +0.143 127.033
ATOM      5  C6  G39 A 800       8.792  15.389  -1.175 -0.14 +0.05    +0.147 127.033
ATOM      6  C7  G39 A 800      10.177  15.431  -0.587 -0.16 +0.03    +0.049 127.033
ATOM      7  O7  G39 A 800       7.734  15.216  -0.127 -0.28 -0.06    -0.379 127.033
ATOM      8  C8  G39 A 800       7.830  14.301   1.041 -0.14 +0.06    +0.121 127.033
ATOM      9  C9  G39 A 800       7.539  14.896   2.446 -0.27 +0.01    +0.027 127.033
ATOM     10  C91 G39 A 800       8.557  15.914   2.989 -0.34 +0.00    +0.007 127.033
ATOM     11  C81 G39 A 800       6.902  13.148   0.710 -0.14 +0.02    +0.027 127.033
ATOM     12  C82 G39 A 800       6.273  12.570   1.937 -0.22 +0.01    +0.007 127.033
ATOM     13  N5  G39 A 800       7.073  16.258  -2.868 -0.10 +0.07    -0.352 127.033
ATOM     14  H5  G39 A 800       6.243  16.838  -2.746 -0.31 -0.10    +0.163 127.033
ATOM     15  C10 G39 A 800       7.029  15.199  -3.701 -0.27 +0.15    +0.214 127.033
ATOM     16  C11 G39 A 800       5.741  14.944  -4.393 -0.41 +0.15    +0.117 127.033
ATOM     17  O10 G39 A 800       8.001  14.420  -3.927 -0.75 -0.34    -0.274 127.033
ATOM     18  N4  G39 A 800       9.048  17.944  -4.058 -0.03 +0.09    -0.073 127.033
ATOM     19  H42 G39 A 800       9.432  18.836  -3.744 -0.06 -0.66    +0.274 127.033
ATOM     20  H41 G39 A 800       9.334  17.707  -5.009 -0.15 -0.14    +0.274 127.033
ATOM     21  H43 G39 A 800       8.059  18.187  -3.996 -0.35 -0.76    +0.274 127.033
ATOM     22  C1  G39 A 800      12.507  16.289  -0.582 -0.27 +0.22    +0.177 127.033
ATOM     23  O1B G39 A 800      13.039  17.366  -0.157 -0.02 -0.91    -0.648 127.033
ATOM     24  O1A G39 A 800      13.140  15.196  -0.518 -0.24 -1.23    -0.648 127.033
```

Figure 2. ADT's oseltamivir energy and position predictions.

Figure 3 is a rendering of the active-site/inhibitor configuration computed in this study.

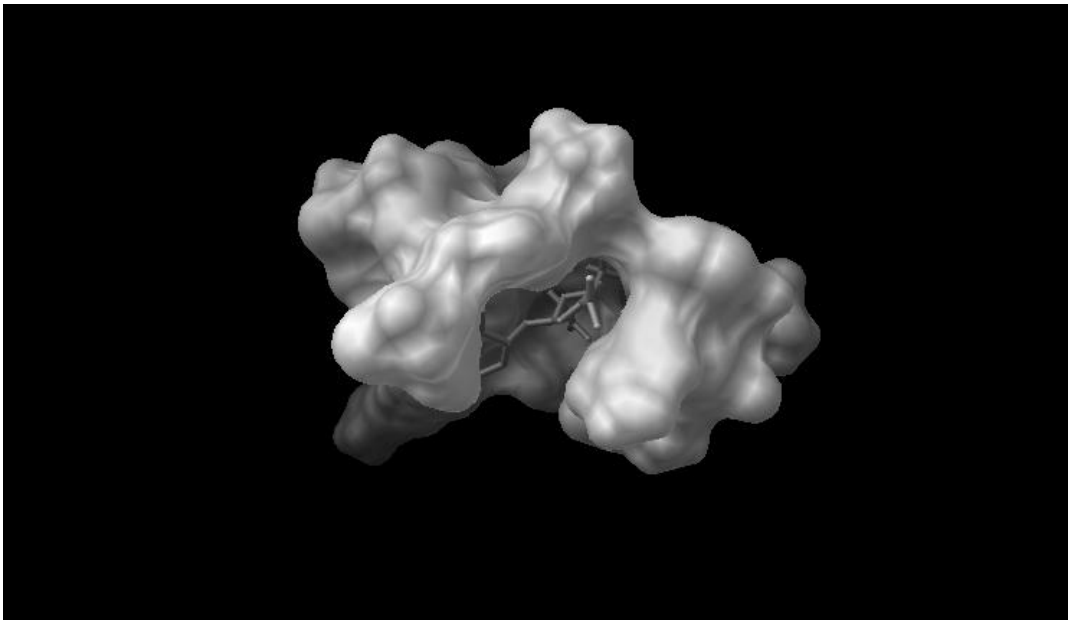


Figure 3. Rendering of oseltamivir computationally docked with the active site of PDB 3BEQ. The inhibitor is shown in stick form. Only the interior, inhibitor-containing region of the molecular surface of the active site can be compared to *in situ* data: the surface distal to the interior is a computational artifact, generated by the assumption that active site is detached from the rest of the receptor.

4.0 Discussion

The method described in Section 2.0 and the results of Section 3.0 motivate several observations:

1. The inhibition constant computed in this study (~11 microMolar at ~298 K) is comparable to the inhibition constant of oseltamivir/neuraminidase interactions that are not clinically effective ([11], [13]). This suggests that oseltamivir would not be effective against the principal 1918 pandemic mutant, A/Brevig Mission/1/18 H1N1.

2. The docking study reported here assumes that the receptor is rigid, and as a result, calculation does not reflect any energy contributions of receptor "flexing" to the interaction of the ligand with native unliganded receptor. Future work will analyze the docking with a flexible receptor

3. The analysis described in Sections 2.0 and 3.0 assumes the neuraminidase is in a crystallized form (isolated at ~278 K). *In situ*, at physiologically normal temperatures (~310 K), the receptor is not in crystallized form. The ligand/receptor conformation *in situ*, therefore, may not be identical to their conformation in the crystallized form.

4. Minimum-energy search algorithms other than the Lamarckian genetic algorithm used in this work could be applied to this docking problem. Future work will use Monte Carlo/simulated annealing algorithms.

5. A variety of torsion and charge models could be applied to this problem, and future work will do so.

5.0 Acknowledgements

This work benefited from discussions with Tony Pawlicki. For any problems that remain, I am solely responsible.

6.0 References.

[1] Russell RJ et al. The structure of H5N1 avian neuraminidase suggests new opportunities for drug design. *Nature* 443 (6 September 2006), 45-49.

[2] Johnson NP and Mueller J. Updating the accounts: global mortality of the 1918-1920 "Spanish " influenza pandemic. *Bulletin of the History of Medicine* 76 (2002), 105-115.

[3] World Health Organization. A revision of the system of nomenclature for influenza viruses: a WHO memorandum. *Bulletin of the World Health Organization* 58 (1980), 585-591.

[4] Ward P et al. Oseltamivir (Tamiflu) and its potential for use in the event of an influenza pandemic. *Journal of Antimicrobial Chemotherapy* 55, supplement 1 (2005), i5-i21.

[5] Butler D. Avian flu special: The flu pandemic: were we ready? *Nature* 435 (26 May 2005), 400-402. doi: 10.1038/435400a.

[6] Xu X, Zhu X, Dwek RA, Stevens J, and Wilson IA. Structural characterization of the 1918 Influenza virus H1N1 neuraminidase. *Journal of Virology* 82 (November 2008), 10493-10501. http://www.pdb.org/pdb/explore/explore.do?structureId=3BEQ.

[7] US Centers for Disease Control. *Summary: Interim Recommendations for the Use of Influenza Antiviral Medications in the Setting of Oseltamivir Resistance among Circulating Influenza A (H1N1) Viruses, 2008-09 Influenza Season.* 19 December 2008. URL http://www.cdc.gov/flu/professionals/antivirals/summary.htm.

[8] Luo M. Structural biology: antiviral drugs fit for a purpose. *Nature* 443 (7 September 2006), 37-38. doi:10.1038/443037a, published online 6 September 2006.

[9] Morris GM, Goodsell DS, Huey R, Lindstrom W, Hart WE, Kurowski S, Halliday S, Belew R, and Olson AJ. *AutoDock* v4.2. http://autodock.scripps.edu/. 2010.

[10] Drug Bank. *Zanamivir*. http://www.drugbank.ca/drugs/APRD00378.

[11] Govorkova EA et al. Comparison of efficacies of RWJ-270201, zanamivir, and oseltamivir against H5N1, H9N2, and other avian influenza viruses. *Antimicrobial Agents and Chemotherapy* 45 (2001), 2723-2732.

[12] Huey R and Morris GM. *Using AutoDock 4 with AutoDock Tools: A Tutorial.* 8 January 2008. http://autodock.scripps.edu/.

[13] Cheng Y and Prusoff WH. Relationship between the inhibition constant (K_i) and the concentration of inhibitor which causes 50 per cent inhibition (I_{50}) of an enzymatic reaction. *Biochemical Pharmacology* 22 (December 1973), 3099–3108. doi:10.1016/0006-2952(73)90196-2.

[14] Protein Data Bank entry 2HU4. http://www.rcsb.org/pdb/explore.do?structureId=2hu4. Russell RJ et al. The structure of H5N1 avian neuraminidase suggests new opportunities for drug design. *Nature* 443 (6 September 2006), 45-49.

CYCLOHEXIMIDE INDUCED CHROMATID EXCHANGES IN BONE MARROW CELLS OF MICE

AMARJOT CHHABRA

NEW LIFE INFERTILITY RESEARCH CENTRE,DEFENCE COLONY,JALANDHAR,144001,PUNJAB,INDIA

KEY WORD;CYCLOHEXIMIDE,ABERRATIONS , CHROMOSOMES,EXCHANGES

ABSTRACT-*Mice were injected i.p.with single dose of 0.5,2.5,5 mg/kg bodyweight of cycloheximide.cells were sampled for mitotic chromosome analysis 4,16 and 24 h after treatment.More than 60% of the cells carried at least one chromatid exchanges.The doses showed a time and dose related increase in aberration frequencies in linear manner.The majority of these exchanges derived from breaks in the centromeric heterochromatin.*

Cycloheximide is regarded as effective protein synthesis inhibior it can effect and can have an opposite effect and can accurately induce the production of several proteins at both transcriptional level(ringold et al.,1984;Israel et al.,1985).A large numbers of reports have been published on mammalian cells regarding their protein synthesis inhibition properties by affecting 80s ribosomal units(cooper et al.,1987.Although some biochemical effect of this drug have been studied,the cytological and cytoplasmic effects have not yet been elucidated.The present investigation deals with the effects of cycloheximide in mouse bone marrow cells.

MATERIALS AND METHODS

As many as three sets each containing three mice were injected intraperitoneally with cycloheximide(dissolved in distilled water)at 00.5,2.5and 5mg/kg body weight.The first set of mice was sacrificed after 4 hr.,the second set after 16 hrand third set set after 24 hr.The mice received an injection of colchicines 4mg/kg three hours before sacrificing .Control mice received distilled water .The bone marrow suspension was collected and hypotonic treatment and cell swelling were done in usual way(Ghosh and Ghosh,1969).Slides were prepared by air technique and stained with giemsa.Microscopic analysis was carried for each concentration and time duration and 150 metaphases were observed for treated groups and 300 for control animals..The mitosis were analysed for chromatid type aberrations such as gaps,breaks and exchange.The selection was based on uniform staining quality ,lack of overlapping chromosome and chromosome number(40+2).

RESULTS

Different type of aberrations induced by cycloheximide.Cells with only gaps were notsignificantly more frequent in the experimental groups than in control groups.Cells with only breaks were frequent.The majority of the effected cells contained chromatid exchanges.Presence of translocations and ring chromosomes are also notable feature.The lowest dose yield a maximum number of 16 hr after treatment.Whereas the maximum number for highest dose occurred at

TABLE; Frequencies of different types of chromatid aberrations induced by cycloheximide in mouse bone marrow cells

Dose Mg/kg	Time (h)	Number Of animals	Total no Of cells observed	Number of cellswith breaks	Number of cells with gaps	cells with exchanges	Percentage of exchanges
0.5	4	3	150	1	8	1	0.67
	16	3	150	1	5	20	11.00
	24	3	150	1	2	13	9.00
control			300	1	1	2	0.01
2.5	4	3	150	0	6	2	0.36
	16	3	150	2	4	50	31.00
	24	3	150	2	5	55	35.00
control			300	0	1	2	0.01
5.0	4	3	150	2	0	3	1.33
	16	3	150	1	6	62	44.00
	24	3	150	2	3	75	55.00
control			300	0	1	0	0.00

DISCUSSION

A perusal of the results shows that cycloheximide induced chromosome and chromatid type aberrations were seen in all treatments indicate that the fungicide acted effectively during G1,SandG2 phase of cell cycle.Nevertheless ,the frequency of aberrations being maximum at the longest period of exposure indicates that fungicide remained active throughout the culture period and acted additively with increasing hours of exposure.The exposure of cultured human lymphocytes pre-irradiated with ICGR of X-rays and their subsequent exposure to cycloheximide prevented their adaptation to higher doses of X-rays for the induction of chromosomal aberrations which may be due to the induction of repair enzymes(youngbloom et al.,1989).The positive correlation with respect to dose and durations confirm the clastogenicity of the drug.From the appearance of the exchanges it can be concluded that the breaking leading to these exchanges occur in the centrometric heterochromatin.(vadhuri et al.,1984)

REFERENCES

[1]Cooper,O.,Banthrope,D.VandWilkie,D;.Modified ribosome conferring resistance to cycloheximide in mutants of saccharomycescerevisiae.J.Mol.Biol.26:347_350(1967).

[2]Ghosh,SandGhosh,J;Chromosomal alterations in the evolution of substrains of Enrlich Letter mouse ascites tumour,Chromosoma,28;62-72(1969).

.

[3]Israel.,D.I.,Estolano,M.G.,Galeazzi,D.RandWhittock,J.P.J r;.Superinduction of cytochrome P_450 gene transcription synthesis in wild type and variant mouse hepatoma cells.J.Biol.Chem.260;5848-5853(1985).

[4]Ringold,G.M.,Dieckmann,B.,Vannice,J.L.,Trahey.,M and McCormick,F;Inhibition of protein synthesis stimulate the transcription of human beta interferon gene s in chinese hamster ovary cells.Proc.Natl.Acad.Sci(U.S.A)81;3964-3968.(1984).

[5]Vadhuri,A.,Chaudhari,M.Mand Ghosh,S;Rifampicin chromatid exchanges in sarcoma -180 mouse Ascites cells.Mutres:141;171-173(1984)

[6]Youngbloom,J.h.,Wiencke,J.KandWolff.,S(1989) Inhibition of adaptive response of human lymphocytes to very low doses of ionizing radiation by the protein synthesis inhibitor cycloheximide.Mutat.res.227;257-261(1989).

SESSION

FEATURE SELECTION + OPTIMIZATION + TIME SERIES

Chair(s)

Prof. Hamid R. Arabnia

Applying GECs for Feature Selection and Weighting using X-TOOLSS

Tamirat Abegaz[#], Gerry Dozier[#], Kelvin Bryant[#], Joshua Adams[#], Vincent Mclean[#] ,Joseph Shelton[#], Aniesha Alford[#], Karl Ricanek^, Damon L. Woodard*

#North Carolina A&T State University

^University of North Carolina at Wilmington

**Clemson University*

Abstract— **In [1], Abegaz et. al compared hybrid genetic and evolutionary feature selection (GEFeS) and weighting (GEFeW) on feature sets obtained by Eigenface, LBP, and oLBP feature extraction methods. GEFeS and GEFeW were implemented using a Steady-State Genetic Algorithm (SSGA). In this paper, we extend the work performed in [1] and compared GEFeS and GEFeW implementations using SSGAs and Estimation of Distribution Algorithms (EDAs). Our results show that GEFeS and GEFeW enhance the overall performance of both the Eigenface-based and LBP-based methods. Comparing the hybrids, our results show that both LBP and oLBP-based hybrids performed better in terms of accuracy than the Eigenface-based hybrids. In addition, the results also show that the EDA implementation of GEFeS (for the LBP and oLBP hybrids) has the best overall performance.**

Keywords— Local Binary Pattern (LBP), overlapped Local Binary Pattern (oLBP),Eigenface, Steady State Genetic Algorithm (SSGA), Estimation and Distribution Algorithm (EDA), Feature Selection.

I. INTRODUCTION

Biometric Systems, recognition of individuals based on distinctive traits, has become an integral part of many applications [2]. Biometric systems generally consist of five commonly accepted steps: Sample Acquisition, Normalization, Feature Extraction, Feature Selection, and Classification [17]. Sample acquisition is used to obtain biometric traits such as fingerprints, iris, periocular, or facial images. Normalization is used to isolate the area of biometric region from any obscuring elements such as noise and homogenize variations such as size and color. From the normalized images, feature extraction is performed to create a discriminatory feature vector used for recognition. The Eigenface and LBP methods are commonly used feature extractors. Feature Selection step is used to select a smaller subset of features. Classification is used to provide the similarity measure among different subjects to determine individual identity by measuring the distance of the features vectors

Genetic & Evolutionary Computation (GEC) is a search strategy aimed to find an optimal (or near optimal) solution for a specific problem domain [4, 5, 6, 7,8] based on natural selection. First, a number of individuals or candidates Feature Subsets are generated to form an initial population. Each candidate Feature Subsets (CFSs) is then evaluated and assigned a fitness value received from an evaluation function specific to the problem at hand. New CFSs are produced from the selected parents by the processes of reproduction. Survivors are selected from the previous generation and combined with the offspring to form the next generation. This evolutionary process continues for user specified number of cycles.

This work is an extension of the research performed by Abegaz et. al [1]. In their work, Abegaz et al. used Genetic and Evolutionary Feature Selection (GEFeS) and Weighting (GEFeW) along with the Eigenface, Local Binary Pattern (LBP), and overlapped LBP (oLBP) methods. They created Eigen-GEFeS and Eigen-GEFeW hybrids that were instances of a Steady State Genetic Algorithm (SSGA). Their results showed that both LBP and oLBP-based GEFeS and GEFeW hybrids performed better in terms of accuracy than the Eigenface hybrids. Their results also showed that that both LBP-GEFeS and oLBP-GEFeS hybrids fell in the best equivalence class with respect to accuracy.

In this paper, we extend the work of Abegaz et. al by comparing the performances of GEFeS and GEFeW that are instances of EDAs. In this paper, we will compare two instances of GEFeS (implemented using an SSGA and an EDA), and two instances of GEFeW (again, implemented using an SSGA and an EDA). The SSGA and EDA are based on the eXploratory Toolset for the Optimization Of Launch and Space Systems (X-TOOLSS) [9] which is suite of 12 GECs which interface with evaluation functions expressed as executables of any programming language or script.

This work is motivated by the research of Gentile et. al [10, 11]. Gentile et. al proposed a two-stage hierarchical process to reduce the number of feature checks required for an iris-based biometric recognition system. Our target is a similar system for Face Recognition (FR) based on short length biometric templates that are able to achieve higher recognition accuracies.

The remainder of this paper is as follows. In Section II, we explain the feature extraction techniques used as input for GEFeS and GEFeW. In Section III, we provide an overview of GEFeS and GEFeW. In Section IV, we present our

experiment. In Section V, we present our results. Finally, our conclusions and future work are presented in Section VI.

II. Feature Extraction using Eigenface, LBP, and oLBP

The Eigenface method [13] is a feature extraction method that uses Principal Component Analysis (PCA). In this method, the gallery images are used to construct the face space using the eigenvectors as axes. Each image (both the gallery and the probe) is then projected into the face space spanned by eigenvectors (also known as Eigenfaces). This method usually ignores those Eigenfaces with less eigenvalues

In the standard LBP method [14], an image is first divided into several patches (blocks) from which local binary patterns are extracted to produce a set of histograms from every non-border pixel. Each histogram obtained from a patch is concatenated to construct the global histogram that represents the feature set.

oLBP based feature extraction [17] is a variant of LBP that attempts to include the internal border pixels that are left out during the process of logical partitioning on the standard LBP method. This is done by logically overlapping the patches horizontally, vertically, and both horizontally and vertically with arbitrary number of pixels.

III. GEFeS and GEFeW

GEFeS and GEFeW [18, 19, 20, 21, 22, 23] were designed to select and weight high discriminating features used for biometric identification. In identification [24], the principal objective is to rank the gallery (set of known individual images and their associated feature vectors) by similarity to the probe (the newly acquired images and their feature vectors) by comparing the probe features with each of the gallery features. Such ranking is computationally expensive particularly for real world applications that involve large number of images in a dataset. GEFeS and GEFeW were developed to reduce the number of features by keeping those feature set combinations that have a higher discriminatory power.

Consider the vector shown in Figure 1.a as feature set. Furthermore, consider the vector shown in Figure 1.b as a candidate real-coded feature mask. For GEFeS case, a masking threshold value is used to create a binary-coded candidate feature mask. A threshold value of 0.5 is applied to real coded-value to produce the binary-coded value. If a real-coded mask is greater than the threshold, then the value in the corresponding binary-coded vector is set to 1. Otherwise, the binary-coded value is set to 0. Figure 1.c shows the candidate binary-coded feature mask vector based on the real-coded feature mask shown in Figure 1.b. Figure 1.d shows the result of the features in Figure 1.a when feature masking (Figure 1.c) is applied to a feature vector.

For GEFeW, the real-coded candidate feature mask is used to weight features within a subset of the feature set. The real-coded candidate feature mask value is multiplied by each feature value to provide a weighted feature. If the number generated is 0 (or approximately equal to 0) the feature value is 0, which basically means that the feature is masked. Figure 1.e shows the weighted feature vector

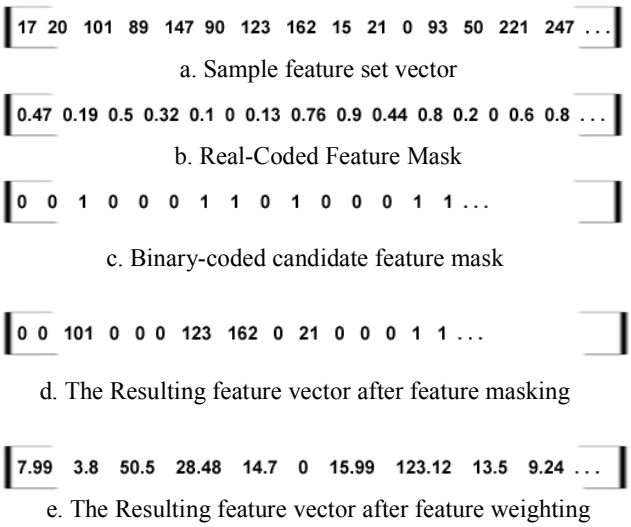


Figure 1 Sample steps of feature selection and weighting

The computation of the fitness for a candidate feature mask for GEFeS and GEFeW is shown in Equation 1. The fitness returned by the evaluation function for a feature mask is simply the number of errors associated with the feature mask (FM) multiplied by 10 plus the percentage of the original feature set that is used. The objective of the evaluation function is to minimize the number of recognition errors (i.e to increase accuracy) as well as to reduce the number of features needed for recognition.

$$\text{Fitness} = (\text{number of errors}) * 10 + \%\text{Features Used} \quad (1)$$

IV. Experiment

The dataset used in this research is a subset of the Face Recognition Grand Challenge (FRGC) dataset [25]. In our dataset, 280 subjects were used, with each subject having a total of 3 associated images with it. This subset was selected because it contains a variety of imaging conditions such as different ethnic origins, frontal images that were neutral and frontal images that had facial expressions. Out of 840 images, 280 were used as the probe set and 560 images were selected for gallery. The images passed the pre-processing stages such as eye rotation alignment, histogram equalization, masking resizing (each with 225 by 195), and conversion of the images into greyscale. GEFeS and GEFeW were used on feature sets extracted using the Eigenface, LBP, and oLBP feature methods. oLBP based feature extraction methods provides information to determine whether including the border pixels has an impact on the recognition rate of the LBP based face recognition algorithm.

V. RESULTS

For our experiment, a total of 12 GEFeS and GEFeW instances were compared. Six of the GEFeS and GEFeW were instances of a SSGA and the other six were instances of an EDA. The SSGA instances evolved population of 20 FMs, had a Gaussian mutation rate of 1 and a mutation range of 0.2. The Mutation rate value of 1 implies that all children must undergo mutation. The mutation range provides a window from the current value (obtained value after recombination) that the new value will be mutated. EDAs uses the probability density/distribution function (PDF), instead of crossover and mutation, to create new population. The PDF is formed using a percentage (25% in our case) of the top performing candidate solutions within the current population. Then, X offspring are created by sampling the PDF. Based on their fitness values, offspring are evaluated and a new population is created using the Y best performing individuals within the current population and the X - Y best offspring.

The hybrids were each run a total of 30 times with a maximum of 1000 function evaluations. Our results are shown in Table I. In Table I, the columns represent the method used, average number features, the average percent accuracy, and the best accuracy obtained.

ANOVA and t-Tests were used to divide the GEFeS and GEFeW instances as well as the baseline algorithms into equivalence classes. Comparing the baseline algorithms, the Eigenface method ($Eigenface_{Baseline}$) performs best.

As can be seen in Table 1, applying GEFeS on the feature set extracted by the standard LBP method significantly improves accuracy from 70.36% to 96.75 % (in the case LBP_GEFeS_{EDA}) and 96.62% (in the case of LBP_GEFeS_{SSGA}). This shows that GEFeS dramatically improves recognition accuracy. This improvement in accuracy also comes with a reduction in the number of features used for recognition.

Compared to the GEFeS hybrids, the results show that the GEFeW hybrids used a larger number of features. Using a larger number of features brings a better result in the case of Eigen-GEFeW as compared to Eigen-GEFeS. Surprisingly, in the case of LBP-GEFeW and oLBP-GEFeW the result is the opposite. Utilizing a significantly larger number of features actually decreases the accuracy for both LBP-GEFeW and oLBP-GEFeW as compared to their corresponding methods. In terms of accuracy, $LBP\text{-}GEFeS_{EDA}$ and $oLBP\text{-}GEFeS_{EDA}$ outperformed all the other hybrids and are in the best equivalence class. All methods performed well in terms of reducing the number of features needed and in producing a significant improvement in accuracy from their corresponding baseline methods.

TABLE I
EXPERIMENTAL RESULTS OF THE LBP, OLBP AND EIGENFACE METHODS

Methods	Number of Features Used	Average % of Accuracy	Best Accuracy
$LBP_{Baseline}$	2124	70.36	70.36
$oLBP_{Baseline}$	2124	70.71	70.71
$Eigenface_{Baseline}$	560	87.14	87.14
$Eigen\text{-}GEFeS_{SSGA}$	291.2	86.67	87.85
$LBP\text{-}GEFeS_{SSGA}$	1022.1	96.62	97.14
$oLBP\text{-}GEFeS_{SSGA}$	1018.46	96.43	96.79
$Eigen\text{-}GEFeS_{EDA}$	277.45	87.05	88.21
$LBP\text{-}GEFeS_{EDA}$	938.763	96.75	97.14
$oLBP\text{-}GEFeS_{EDA}$	936.23	96.73	97.14
$Eigen\text{-}GEFeW_{SSGA}$	492.8	91.42	92.5
$LBP\text{-}GEFeW_{SSGA}$	1865.29	95.33	95.71
$oLBP\text{-}GEFeW_{SSGA}$	1865.08	95.33	96.07
$Eigen\text{-}GEFeW_{EDA}$	548.8	93.19	94.64
$LBP\text{-}GEFeW_{EDA}$	2108.23	94.64	95.36
$oLBP\text{-}GEFeW_{EDA}$	2106.9	94.56	95.36

Figure 2 shows the Cumulative Match Characteristic (CMC) curve for the LBPBaseline , oLBPBaseline, EigenfaceBaseline, and for the methods that fall in the first equivalent class. LBP-GEFeSEDA, $oLBP\text{-}GEFeS_{EDA}$ obtain approximately 97.50% accuracy at rank 10. However, $Eigenface_{Baseline}$ performed well (approximately 96%) at rank 10. $LBP_{Baseline}$ performed relatively poor in terms of accuracy.

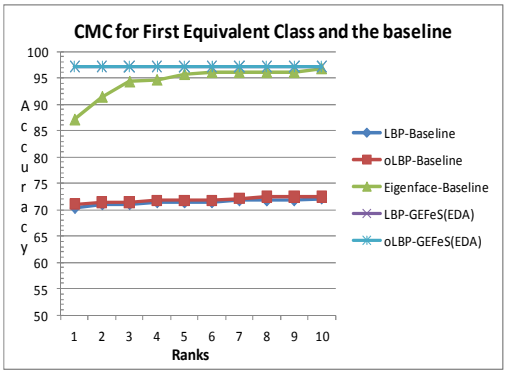


Figure 2: Comparisons of CMC results for baseline and the best performing algorithms

VI. Conclusion and Future Work

The experimental results of applying feature selection and weighting using the concept of GEC on LBP, and oLBP shows that GEFeS and GEFeW enhances the overall performance of the LBP and Eigenface based feature extractions. Feature selection and weighting result indicate that both LBP and oLBP hybrids performed much better than the Eigenface hybrids. Our results suggest that hybrid GECs for feature selection/weighting enhance the overall performance of the Eigenface, LBP, and oLBP-based methods while reducing the number of features needed. Our future work will be devoted towards the investigation of GEFeS and GEFeW based on other forms of Genetic and Evolutionary Computation.

Acknowledgment

This research was funded by the Office of the Director of National Intelligence (ODNI), Center for Academic Excellence (CAE) for the multi-university Center for Advanced Studies in Identity Sciences (CASIS) and by the National Science Foundation (NSF) Science & Technology Center: Bio/computational Evolution in Action CONsortium (BEACON). The authors would like to thank the ODNI and the NSF for their support of this research

References

[1] Abegaz,T, Dozier, G, Bryant, K, Adams, J, Baker, B , Shelton, J, Ricanek, K, Woodard, D, "Genetic-Based Selection and Weighting for LBP,oLBP, and Eigenface Feature Extraction", in Submission to Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), 2011.

[2] Ajay Kumar "ENCYCLOPEDIA OF BIOMETRICS" 2009, Part 6, 597-602, DOI: 10.1007/978-0-387-73003-5_157

[3] M. Dash and H. Liu, Bartlett, Javier R. Movellan, and Terrence J. Sejnowski,, "Feature Selection for Classification" Genetic Algorithms for Feature Selection", Intelligent Data Analysis, vol. 1, no. 3, pp. 131-156, 1997.

[4] Danial Ashlock. "Evolutionary Computation for Modeling and Optimization.", Springer, 2005.

[5] D. Guillamet, & J. Vitri`a, "*Evaluation of distance metrics for recognition based on non-negative matrix factorization*", Pattern Recognition Letters, 24(9-10), 2003, 1599-1605.

[6] Fogel, D. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence . IEEE Press, 2nd Edition., 2000.

[7] Goldberg, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts., 1989.

[8] M. L. Tinker, G. Dozier, and A. Garrett, "The exploratory toolset for the optimization of launch and space systems (x-toolss)," http://xtoolss.msfc.nasa.gov/, 2010.

[9] J.E Gentile, N. Ratha, and J. Connell, "SLIC: Short-length iris codes," In Proc. IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems, 2009. BTAS '09, 28-30 Sept. 2009, pp.1-5.

[10] J.E. Gentile, N. Ratha, and J. Connell, "An efficient, two-stage iris recognition system", In *Proc. 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS),* 2009.

[11] Peter T. Higgins, "Introduction to Biometrics", The Proceeding of Biometrics consortium conference 2006, Baltimore", *MD, USA, Sept. 2006.*

[12] M. Turk and A. Pentland, "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, Vol. 13, No. 1, pp. 71-86, 1991.

[13] Caifeng Shan and Tommaso Gritti, "Learning Discriminative LBP-Histogram Bins for Facial Expression Recognition", Proc. of 15th EUSIPCO, Poznan, Poland, September 2007.

[14] Goldberg, Toimo Ahonen, Abdenour Hadid, and Matti Pietik¨ainen " Learning Face Expression Recognition", http://www.ee.oulu.fi/mvg/, visited on sept 10, 2120.

[15] J. Zhao, H. Wang, H. Ren, and S. C. Kee," LBP discriminant alalysi for face verification," in Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol 3, pp. 167-172, June 2005.

[16] Tamirat Abegaz, "*Genetic and Evolutionary feature selection and weighting for Face Recognition*", thesis submitted to North Carolina A&T State University

[17] J. Adams, D. L. Woodard, G. Dozier, P. Miller, K. Bryant, and G. Glenn. *Genetic-based type II feature extraction for periocular biometric recognition: Less is more*. In Proc. Int. Conf. on Pattern Recognition, 2010. to appear.

[18] *Huang C. L. and Wang C. J. "*GA-based feature selection and parameters optimization for support vector machines*",. C.-L. Huang,* C.-J. Wang / Expert Systems with Applications. Vol. 31(2), 2006, pp231–240.

[19] Adams, J., Woodard, D. L., Dozier, G., Miller, P., Glenn, G., Bryant, K. "*GEFE: Genetic & Evolutionary Feature Extraction for Periocular-Based Biometric Recognition*," Proceedings 2010 ACM Southeast Conference, April 15-17, 2010, Oxford, MS.

[20] Dozier, G., Adams, J., Woodard, D. L., Miller, P., Bryant, K. "A Comparison of Two Genetic and Evolutionary Feature Selection Strategies for Periocular-Based Biometric Recognition via X-TOOLSS,", Proceedings of the 2010 International Conference on Genetic and Evolutionary Methods (GEM'10: July 12-15, 2010, Las Vegas, USA).

[21] Simpson, L. , Dozier, G., Adams, J., Woodard, D. L., Dozier, G., Miller, P., Glenn, G., Bryant, K.. "*GEC-Based Type-II Feature Extraction for Periocular Recognition via X-TOOLSS*," Proceedings 2010 Congress on Evolutionary Computation, July 18-23, Barcelona, Spain.

[22] Dozier, G., Bell, D., Barnes, L., and Bryant, K. (2009). "Refining Iris Templates via Weighted Bit Consistency", Proceedings of the 2009 Midwest Artificial Intelligence & Cognitive Science (MAICS) Conference, Fort Wayne, April 18-19, 2009.

[23] Dozier, G., Adams, J., Woodard, D. L., Miller, P., Bryant, K. "A Comparison of Two Genetic and Evolutionary Feature Selection Strategies for Periocular-Based Biometric Recognition via X-TOOLSS", (to appear in) The Proceedings of the 2010 International Conference on Genetic and Evolutionary Methods (GEM'10: July 12-15, 2010, Las Vegas, USA).

[24] Abegaz, T, Dozier, G, Bryant, K, Adams, J Popplewell, K, Shelton, J Ricanek, K. Woodard, D " "*Hybrid GAs for Eigen-Based Facial Recognition"*, accepted for IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011).

P. Jonathon Phillips1, Patrick J. Flynn2, Todd Scruggs3, Kevin W. Bowyer2 Jin Chang2, Kevin Hoffman3, Joe Marques4, Jaesik Min2, William Worek3," Overview of the Face Recognition Grand Challenge", IEEE Conference on Computer Vision and Pattern Recognition, 2005.

A Study of Different Transfer Functions for Binary Version of Particle Swarm Optimization

S. Mirjalili[1], S.Z. Mohd Hashim[1], G. Taherzadeh[2], S.Z. Mirjalili[3], and S. Salehi[1]
[1]Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia, 81300 Skudai, Johor Bahru, Malaysia
[2]Faculty of Information Technology, Multimedia University, Selangor, Malaysia
[3]Faculty of Mathematics, Sharif University of Technology, Tehran, Iran

Abstract - *Particle Swarm Optimization (PSO) is one of the most widely used heuristic algorithms. The simplicity and inexpensive computational cost make this algorithm very popular and powerful in solving wide ranges of problems. However, PSO suffers two problems of trapping in local minima and slow convergence speed. Binary version of this algorithm has been introduced for solving binary problems. Because BPSO uses the same concepts of PSO, it also undergoes the same problems. The main part of the binary version is the transfer function. There is not enough study in the literature focusing on the transfer function. In this study, eight new transfer functions dividing into two families (s-shape and v-shape) for binary particle swarm optimization are introduced and evaluated. Four benchmark optimization problems are employed in order to evaluate these transfer functions in terms of avoiding local minima, convergence speed, and accuracy of results. The results prove that the new introduced v-shape family of transfer functions could improve the performance of original binary PSO based on the above-mentioned drawbacks.*

Keywords: Optimization Algorithm (MOA), Magnetic field theory; Function optimization; Transfer function

1 Introduction

Particle Swarm Optimization (PSO) is one of the most widely used evolutionary algorithms inspired from social behavior of animals [1,2]. The simplicity and inexpensive computational cost make this algorithm very popular. Due to above-mentioned advantages, PSO has been applied to many domains such as medical detecting [3], grid scheduling [4], robot path planning [5], and video abstraction [6]. In spite of these advantages, like all other population-based algorithm, trapping in local minima and slow convergence rate are two unavoidable problems for PSO. With the increase of problems' dimension, these two problems become more complex. PSO is capable of solving problems which have continuous search space. However, some problems have different search spaces.

There are many optimization problems, which have discrete binary search spaces. They need binary algorithms to be solved. Binary version of PSO was proposed by Kennedy and Eberhart in 1997 [7]. Like PSO, binary version of PSO has the problems of trapping in local minima and slow convergence speed because of using the same concepts for solving problems [8]. The only different element between these two algorithms is transfer function that is used to map continuous search space to the binary one. Transfer function is the most important part of binary PSO [9]. In the literature, there is not enough study about the transfer function. In This study, eight different transfer functions for binary version of PSO are introduced and evaluated. The effectiveness of employing these new transfer functions are investigated in terms of avoiding local minima, convergence speed, and accuracy of results.

The rest of the paper is organized as follow. Section II presents a brief introduction to PSO. Section III discusses the basic principles of binary version of PSO. The experimental results are demonstrated in section IV. Finally, section V concludes the work and suggests some researches for future works.

2 The Particle Swarm Optimization

PSO is an evolutionary computation technique which is proposed by Kennedy and Eberhart [10,11]. The PSO was inspired from social behavior of bird flocking. It uses a number of particles (candidate solutions) which fly around in the search space to find best solution. Meanwhile, they all look at the best particle (best solution) in their paths. In other words, particles consider their own best solutions as well as the best solution has found so far.

Each particle in PSO should consider the current position, the current velocity, the distance to *pbest*, and the distance to *gbest* to modify its position. PSO was mathematically modeled as follow:

$$v_i^{t+1} = wv_i^t + c_1 \times rand \times (pbest_i - x_i^t) + c_2 \times rand \times (gbest - x_i^t) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

where v_i^t is the velocity of particle *i* at iteration *t*, *w* is a weighting function, c_j is a weighting factor, rand is a random number between 0 and 1, x_i^t is the current position of particle *i* at iteration *t*, $pbest_i$ is the *pbest* of agent *i* at iteration *t*, and *gbest* is the best solution so far.

The first part of (1), wv_i^t, provides exploration ability for PSO. The second and third parts, $c_1 \times rand \times (pbest_i - x_i^t)$ and $c_2 \times rand \times (gbest - x_i^t)$, represent private thinking and collaboration of particles respectively. The PSO starts with randomly placing the particles in a problem space. In each iteration, the velocities of particles are calculated using (1). After defining the velocities, the position of masses can be calculated as (2). The process of changing particles' position will continue until meeting an end criterion.

3 Binary Version of Particle Swarm Optimization

Generally, there are many problems which have intrinsic discrete binary search space like feature selection and dimensionality reduction [12,13]. In addition, the problems with continuous real search space can be converted into binary problems. However, a binary search space has its own structure with some limitations.

A binary search space can be considered as a hypercube. The agents of a binary optimization algorithm can only shift to nearer and farther corners of the hypercube by flipping various numbers of bits [7]. Hence, for designing binary version of PSO, some basic concepts such as velocity and position updating process had been modified.

In the original PSO, particles can move around the search space because of having position vectors with continuous real domain. Consequently, the concept of position updating can be easily implemented for particles adding velocities to positions using (2). However, the meaning of position updating is different in a discrete binary space. In binary space, due to dealing with only two numbers ("0" and "1"), the position updating process cannot be done using (2). Therefore, we have to find a way to use velocities to change agents' positions from "0" to "1" or vice versa. In other words, we have to find a link between velocity and position, as well as revise (2).

Basically, in discrete binary space, the position updating means a swithcing between "0" and "1" values. This switching should be done based on velocities of agents. The question here is that how the concept of velocity in real space should be employed in order to update positions in binary space. According to [7,14,15], the idea is to change position of an agent with the probability of its velocity. In order to do this, a transfer function is needed to map the velocities values to probability values for updating the positions.

As mentioned above, transfer functions define the probability of changing position vector's elements from "0" to "1" and vice versa. Transfer functions force agents to move in a binary space. According to [14], some concepts should be taken into account for selecting a transfer function in order to map velocity values to probability values.

The transfer function should be able to provide a high probability of changing the position for a large absolute value of the velocity. It should also present a small probability of changing the position for a small absolute value of the velocity. Moreover, the range of a transfer function should be bounded in the interval [0,1] and increased with the increasing of velocity. The function that have been used in [7] are presented as (3). This function is also depicted in Fig.1.

$$S\left(v_{i,j}^k(t)\right) = \frac{1}{1+e^{-v_{i,j}^k(t)}} \tag{3}$$

The above-mentioned transfer function and the new introduced transfer functions in this work are listed in Table I. These transfer functions are also visualized in Fig.1 and Fig.2. We call the first and second groups s-shape and v-shape family transfer functions respectively. According to [7], for the transfer function in Fig.1, we use (4) in order to update position vectors. According to [14], for the transfer function in Fig.2, we use (5) to update position vectors based on velocities. It should be noticed that these transfer functions satisfy all aforementioned concepts.

$$x_{i,j}^k(t+1) = \begin{cases} 0 & \text{If } rand < S\left(v_{i,j}^k(t+1)\right) \\ 1 & \text{If } rand \geq S\left(v_{i,j}^k(t+1)\right) \end{cases} \tag{4}$$

$$x_{i,j}^k(t+1) = \begin{cases} complement\left(x_{i,j}^k(t)\right) & \text{If } rand < S\left(v_{i,j}^k(t+1)\right) \\ x_{i,j}^k(t) & \text{If } rand \geq S\left(v_{i,j}^k(t+1)\right) \end{cases} \tag{5}$$

TABLE I. TRANSFER FUNCTIONS

No	Transfer Functions
1	$S(x) = \frac{1}{1+e^{-2x}}$
2 [7]	$S(x) = \frac{1}{1+e^{-x}}$
3	$S(x) = \frac{1}{1+e^{\frac{-x}{2}}}$
4	$S(x) = \frac{1}{1+e^{\frac{-x}{3}}}$
5	$S(x) = \left\lvert \text{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert = \left\lvert \frac{\sqrt{2}}{\pi}\int_0^{\frac{\sqrt{\pi}}{2}x} e^{-t^2}\,dt \right\rvert$
6 [14]	$S(x) = \lvert \tanh(x) \rvert$
7	$S(x) = \left\lvert \frac{x}{\sqrt{1+x^2}} \right\rvert$
8	$S(x) = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right) \right\rvert$

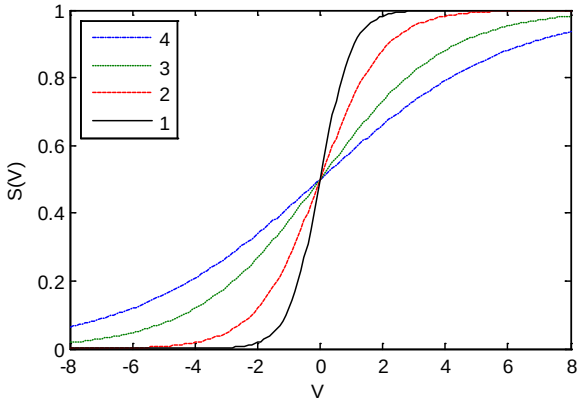


Figure 1. S-shape family transfer functions

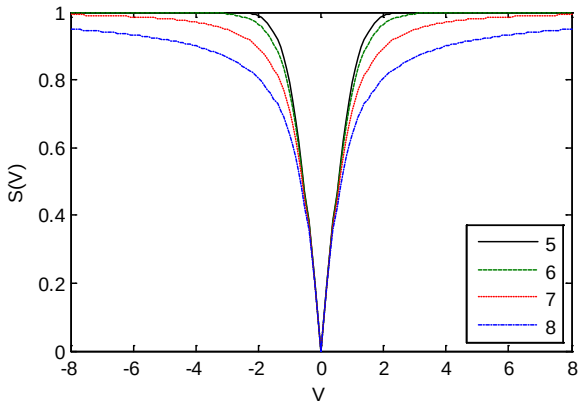


Figure 2. V-shape family transfer functions

The general steps of Binary PSO are as follows.

a) All particles are initialized with random values
b) Repeat steps c-e until the meeting of the end condition
c) For all particles, velocities are defined using (1)
d) Calculate probabilities for changing elements of position vectors based on transfer function's formula.
e) Update the elements of position vectors based on the rules in (4) or (5) based on the type of transfer function

4 Experimental results and discussion

In order to evaluate the performance of new binary versions of PSO called BPSO with different transfer functions, 4 standard benchmark functions are employed [16]. Table II lists down these benchmark functions and the range of their search spaces. Fig.3, Fig.4, Fig.5, and Fig.6 illustrate them, Spherical, Rastrigin, Rosenbrock, and Griewank functions, respectively. Furthermore, function's dimension is set to 5 (m=5). To represent each continuous variable, 15 bits are used. It should be noticed that one bit is reserved for the sign of each functions' dimension. Therefore, the dimension of agents are 75 (Dim=m×15).

TABLE II. BENCHMARK FUNCTIONS

Function	Range
$F_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100,100]^m$
$F_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]^m$
$F_3(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12,5.12]^m$
$F_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^m$

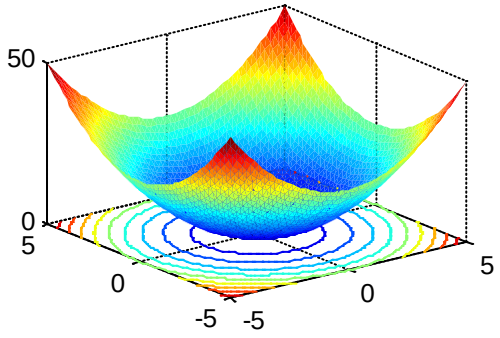


Figure 3. Spherical function (F1)

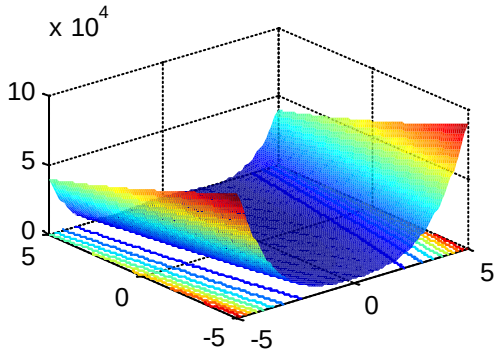


Figure 4. Rastrigin function (F2)

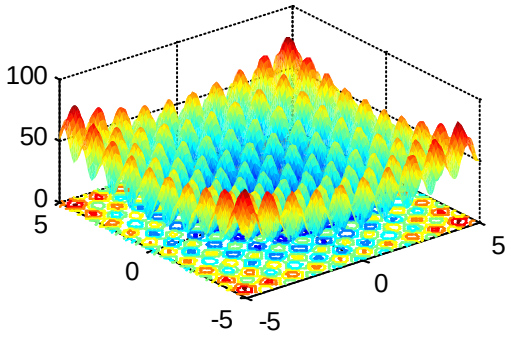


Figure 5. Rosenbrock function (F3)

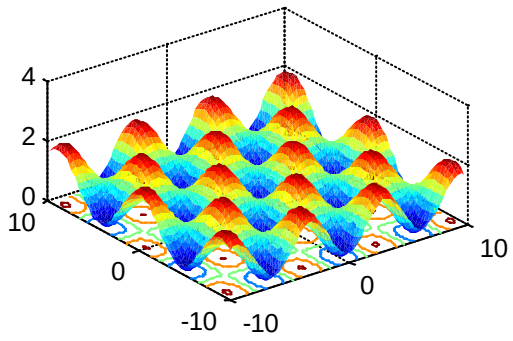


Figure 6. Griewank function (F4)

In this paper, our objective is minimization. The global minimum values for all appeared functions in table I are 0. The number of particles is 30, C_1 and C_2 are set to 2, W is linearly decreased from 0.9 to 0.4, maximum velocity is set to 6, maximum iteration is set to 500, and the stopping criteria is the meeting of maximum number of iteration. According to [14], to achieve a good convergence rate, the velocity should be limited. So, the maximum velocities for all versions of BPSO in this work are set to 6.

The experimental results are presented in Table III. The results are averaged over 30 independent runs, and the best results are indicated in bold type.

TABLE III. MINIMIZATION RESULTS OF 4 BENCHMARK FUNCTIONS OVER 30 INDEPENDENT RUNS

F	Algorithm	ABSF [a]	STDV [b]	MBSF [c]	Best [d]
F1	*BPSO1*	0.4921	0.4165	0.3822	0.0492
	BPSO2	5.2965	2.7657	4.6684	1.7044
	BPSO3	33.3306	17.0770	30.58	2.5258
	BPSO4	71.0918	37.9921	59.313	18.2828
	BPSO5	0.7844	2.9905	0.0044	**0**
	BPSO6	0.3514	0.8997	0.0179	**0**
	BPSO7	0.2663	0.7691	**0.0063**	**0**
	BPSO8	**0.0977**	**0.2725**	0.0064	**0**
F2	*BPSO1*	20.2467	26.1441	8.366	2.0515
	BPSO2	65.3304	79.3444	39.838	5.8852
	BPSO3	437.3886	347.783	310.043	68.5187
	BPSO4	744.8748	623.8992	612.7769	57.498
	BPSO5	26.7199	45.3441	8.5981	1.5156
	BPSO6	21.1831	34.2671	6.7977	07392
	BPSO7	14.8538	33.188	3.2592	0.8658
	BPSO8	**7.3941**	**19.4263**	**2.9353**	**0.1093**
F3	*BPSO1*	2.2054	1.0149	2.2792	0.2207
	BPSO2	4.1528	1.6081	3.6184	2.23
	BPSO3	8.2714	3.3343	8.6903	1.7784
	BPSO4	12.4642	3.2046	12.6385	6.1475
	BPSO5	1.9923	0.8618	1.9932	0.995
	BPSO6	2.0495	0.8105	1.995	0.995
	BPSO7	1.8899	1.2806	**1.7407**	**0.0242**
	BPSO8	**1.6945**	**0.7212**	1.9903	0.9952
F4	*BPSO1*	0.2598	0.0868	0.2776	0.0558
	BPSO2	0.3985	0.1088	0.3753	0.1961
	BPSO3	0.6403	0.1612	0.6409	0.2682
	BPSO4	0.7989	0.1465	0.8471	0.4665
	BPSO5	0.1684	0.1068	0.161432	0.0272
	BPSO6	0.1762	0.1016	0.1623	0.0144
	BPSO7	0.1351	0.0904	0.1154	0.0330
	BPSO8	**0.1032**	**0.0655**	**0.0958**	**0.0154**

a. Indicates average best so far solution over 30 runs in the last iteration
b. Indicates standard deviation of the best so far solution over 30 runs in the last iteration
c. Indicates median best so far solution over 30 runs in the last iteration
c. Indicates the best solution over 30 runs in all iteration

For functions F1 and F2, BPSO8 reaches better results than other algorithms for ABSF and STDV variables in Table II. The functions F1 and F2 belong to family of unimodal functions which are monotonous functions without any local solution. As shown in the Fig.3 and Fig.4, there is only one global solution for these kinds of functions. Hence, the results of the aforementioned statistical variables show BPSO8 improve the ability of exploiting the global minima in original BPSO with its new transfer function.

The results for Best variable in Table III prove that BPSO8 also owns best result accuracy among the other algorithm. Moreover, Fig.7 and Fig.8 prove that BPSO8 possesses good convergence rate in the last iterations.

According to Table III, for functions F3 and F4, BPSO8 outperform other algorithms in all statistical variables. Functions F3 and F4 are multimodal functions that have many local solutions in comparison with unimodal functions. Hence, it can be said that BPSO8 with its new transfer function could enhance the ability of original BPSO to avoid local minima.

The results of Best variable in Table III for BPSO8 also insist on having more accurate results than the other algorithms. Fig.9 and Fig.10 prove that BPSO8 has better convergence speed in multimodal functions for the last iterations as well.

To summarize, results prove that family of s-shape transfer functions with their method of updating position are not suitable for binary version of PSO. In contrary, the new introduced v-shape family of transfer functions with their special method of updating positions is useful for binary version of PSO in terms of avoiding local minima, convergence speed, and accuracy of results. It can be concluded that the new introduced family of transfer function has merit to use in binary algorithms.

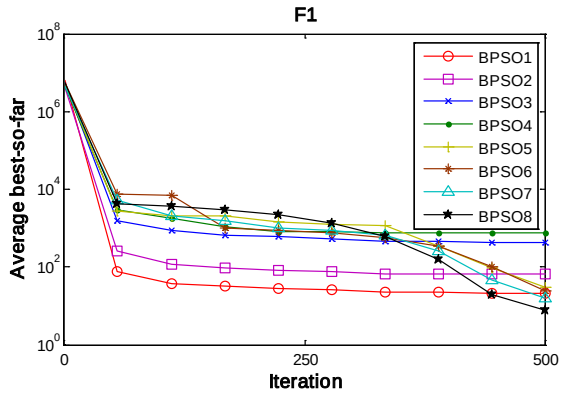


Figure 7. Comparison between BPSOs with diffetent transfer functions on fuction F1

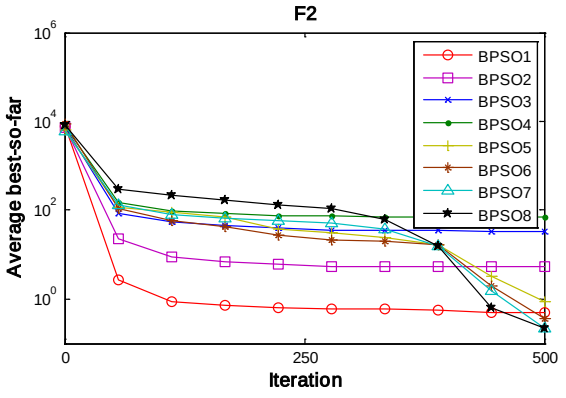


Figure 8. Comparison between BPSOs with diffetent transfer functions on fuction F2

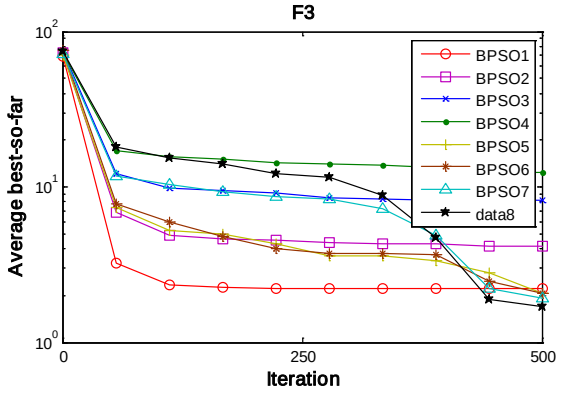


Figure 9. Comparison between BPSOs with diffetent transfer functions on fuction F3

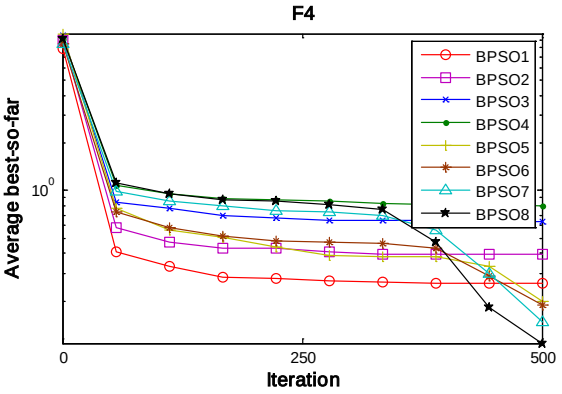


Figure 10. Comparison between BPSOs with diffetent transfer functions on fuction F4

5 Conclution

In this paper, some new versions of binary PSO are introduced utilizing different new transfer functions. Eight new transfer functions dividing into two families (s-shape and v-shape) are introduced and evaluated. In order to justify the performance of all versions, four benchmark functions are employed, and the results are compared together. The results prove that the new introduced v-shape family of transfer functions with their own method of updating positions vector can improve the performance of original binary version of PSO in terms of avoiding local minima, convergence rate, and results' accuracy. The results also show that new introduced v-shape family of transfer function has merit for binary algorithms.

For future studies, it is recommended to use the new introduced family transfer function for the other binary algorithms like Binary Gravitational Search Algorithm.

6 References

[1] R.C. Eberhart and J. Kennedy, "A new optimizer using particles swarm theory," in *Sixth Int. Symp. on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[2] R.C. Eberhart and J. Kennedy, "Particle swarm optimization," in *IEEE Int. Conf. on Neural Network*, Perth, Australia, 1995, pp. 1942–1948.

[3] S. Chandra, R. Bhat, and H. Singh, "A PSO based method for detection of brain tumors from MRI," in *Nature & Biologically Inspired Computing*, Coimbatore, 2009, pp. 666 - 671.

[4] MP. Mathiyalagan, UR. Dhepthie, and SN. Sivanandam, "Grid Scheduling Using Enhanced PSO Algorithm," vol. 2, no. 2, pp. 140-145, 2010.

[5] E. Masehian and D. Sedighizadeh, "A multi-objective PSO-based algorithm for robot path planning," in *IEEE International Conference on Industrial Technology* , Vi a del Mar, 2010, pp. 465 - 470.

[6] MB. Fayk, HA. El Nemr, and MM. Moussa, "Particle swarm optimisation based video abstraction," *Journal of Advanced Research*, vol. 1, no. 2, pp. 163-167, 2010.

[7] J. Kennedy and R.C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *IEEE International Conference on Computational Cybernetics and Simulation*, vol. 5, 1997, pp. 4104 - 4108.

[8] K. Premalatha and A.M. Natarajan, "Hybrid PSO and GA for Global Maximization ," *Int. J. Open Problems Compt. Math.*, vol. 2, no. 4, pp. 597-608, 2009.

[9] Y. Xiaohui, Y. Yanbin, W. Cheng, and Z. Xiaopan, "An Improved PSO Approach for Profit-based Unit Commitment in Electricity Market," in *Transmission and Distribution Conference and Exhibition*, 2005, pp. 1-4.

[10] J. Kennedy and RC. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE international conference on neural networks*, vol. 4, 1995, pp. 1942–1948.

[11] Y. Shi and R.C. Eberhart, "A modified Particle SwarmOptimiser," in *IEEE International Conference on Evolutionary Computation, Anchorage*, Alaska, 1998.

[12] P. Avishek and J. Maitib, "Development of a hybrid methodology for dimensionality reduction in Mahalanobis-Taguchi system using Mahalanobis distance and binary particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1286-1293, March 2010.

[13] X. Zenga, Y. Li, and J. Qina, "A dynamic chain-like agent genetic algorithm for global numerical optimization and feature selection," *Neurocomputing*, vol. 72, no. 4-6, pp. 1214-1228, January 2009.

[14] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "BGSA: binary gravitational search algorithm," *Natural Computing*, vol. 9, no. 3, pp. 727-745, 2009.

[15] S. Mirjalili and S. Z. Mohd Hashim, "BMOA: Binary Magnetic Optimization Algorithm," in *2011 3rd International Conference on Machine Learning and Computing (ICMLC 2011)*, vol. 1, Singapore, 2011, pp. 201-206.

[16] Xin Yao, Yong Liu, and Guangming Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82 - 102, 1999.

Modeling Time series with missing and incorrect values using Self Adaptive Genetic Algorithms

P. Flores[1], **M. G. Cota**[1], **and L. B. Morales**[2]
[1]Departamento de Matemáticas, Universidad de Sonora, Hermosillo, Sonora, México
[2]Instituto de Inv. en Matemáticas Aplicadas y Sistemas,
Universidad Nacional Autónoma de México, Cd. de México, México

Abstract— *In this work it is presented a methodological proposal to build models for Time Series with missing and erroneous values. This methodology consist of two stages: first, it is realized an estimating of the missing and erroneous values of the series; and second, it is built a model for the series. The proposal is based on Self Adaptive Genetic Algorithms that were especially utilized to calculate ARMA models for the NN5-REDUCED problems which results are presented in this work. This methodology here presented can be generalized for the treatment of this type of Time Series by other non linear models that use, for example, neuronal networks, fuzzy logic, etc.*

Keywords: Time Series, Genetic Algorithm, Missing Values.

1. Introduction

Time Series (TS) are modeled to forecast the behavior of real data, but in several occasions, by different causes, values of some series are lost or erroneous values are registered. This is the case of the competence examples NN5 in which all the problems set out to model its behavior come from cash dispensers where in some days, for any reason, it was not recorded the total money withdrawn. Also, some days present the value of zero, that from the process point of view it is an erroneous data since it does not reflect the total money withdraw from the cash dispenser. In real life these problems appear when modeling different phenomena for which are important to have an alternative process to treat them.

In this work we approach the problem of constructing models for this type of TS who present missing and erroneous values. It is necessary to add that in our case it is important to know which the erroneous values are and to have and idea of the interval in which they change. The main idea of this work is to rebuild the missing or erroneous information of the series supposing that the type of model who fits to the series is known.

Our proposition consists of two main stages. In the first stage an estimate of the missing and erroneous values is done, to this one we named it: TS correction stage. In the second stage it is work with the original series to which it had been added the estimated values to build an appropriated model. In our own case it will be utilized linear models for both stages, but in this methodological proposal it can be used other non lineal models such as neuronal networks, fuzzy logic, etc.

The problem of finding a good linear model for a series of data requires to determine, first, how many and which are the terms most appropriate to solve this problem. Afterwards, it is necessary to know in which interval are the linear expression coefficients, and finally it is important to know the values of this coefficients that minimize the square error over all the history. This leads us to set out a non linear optimization problem with variables in real intervals which limits it is necessary to specify. This problem has multiple local minimums by which appropriate optimization techniques are needed.

In order to solve this problem, first, the principal results of the TS statistical analysis [3],[6] are applied to calculate the biggest number of terms that will appear in the series, as well as the intervals in which these change.

Later on, to estimate the best coefficients of the linear expression we used Genetic Algorithms (GA) that are optimization techniques that implement global multi-point search, quickly locating areas of high quality. For a successful implementation of the GA it is required to estimate the value of certain parameters which is achieved by means of numerous tests. In this work it were implemented versions of the Self Adaptive Genetic Algorithms (SAGA) since with these there is eliminated the stage of repeated test that are needed for the adjustment of parameters, achieving with this that the same version of the code solves any TS linear modeling problem without the intervention of the user.

The plan of the work is as follow. In the second section the results for the linear models that we will use in our proposal are summarized. In the third section the SAGA characteristics that will be used are described. In the fourth section the algorithm that we use to calculate the autoregressive models of a series are presented. Afterwards, in the fifth section, the procedure to correct the values of the series and to calculate the ARMA models of the same are presented. Finally, in the sixth section are shown the result of modeling the examples of the NN5-REDUCED 2008.

2. Principal Results of Linear Models

The univariated TS were analyzed by Box-Jenkins [3], from the construction of equations in differences with a random additive component named white noise. From the above mentioned models they determined the conditions under which it is presented the stationarity of the series and the scheme that should be followed to estimate the parameters of the particular model. The most general model is named ARMA (p, q) and indicates the presence of autoregressive components in both the observable variables x_t and in ε_t with

$$\varepsilon_t = x_t - \left(\delta + \sum_{i=1}^{p} \phi_i x_{t-i}\right)$$

and

$$\hat{x}_t = \delta + \sum_{i=1}^{p} \phi_i x_{t-1} + \sum_{j=1}^{q} \gamma_j \varepsilon_{t-j},$$

where ϕ_i and γ_j are the AR and MA coefficients, and δ is a constant. This model can be generalized to an autoregressive integrated moving average (ARIMA) by differentiating the time series TS to a certain level.

In our case with regards to the first correction stage of the series only autoregressive models are considered, whereas for the second stage only the models ARMA are calculated.

The autoregressive models of order p are represented by the expression:

$$x_t = \delta + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + a_t, \qquad (1)$$

where traditionally a_t is a white noise. In this work a_t it will not be a white noise, and in the case the model corresponds to a stationary series δ and ϕ_j are constants that satisfy the following relations:

$$|\phi_1| < 1, \qquad \mu = \frac{\delta}{1 - \sum \phi_j}, \qquad (2)$$

and

$$\sum \phi_j < 1.$$

μ represents the series average. The relation (1) and (4) are a consequence of the stationarity property.

As soon as we have an autoregresive model for a series, it can be considered another new series formed by the differences between the original values of the series and the values of the model. When an autoregressive model was calculated for this new series, this is equivalent to estimate the MA part of the original series with which we would have a complete ARMA model.

The most common way to analyze an ARIMA model is calculating the root of the sum of the square (RSS) or to use the mean absolute percentage error ($MAPE$). With the methodology that we propose it can be constructed models with different values of the parameters p and q in such a way that RSS always diminishes when the values of these parameters increase, which can leads to an over fitting that provokes that the model fits very well to the data series but do not forecast well the future values. To avoid this problem it will be used as a measure of the quality of the model the statistical estimator Bayesian Information Criterion (BIC) as it is used in [4], [9].

$$BIC = N * \ln(RSS^2/N) + p * \ln(N), \qquad (3)$$

where N denotes the number of training examples and p the number of parameters.

3. Self Adaptive Genetic Algorithms

The GA were developed by John Holland [7] and are inspired in basic principals that rules the evolution of the species. It have been successfully used to solve several problems [5]. To apply them it is necessary: a genetic representation of the organism, a way of create an initial population, a performance function that evaluates the adaptation of an individual to his environment and separates the better one, according to its performance by a process called selection, and the genetic operators who alter the structure of the children and the values of several parameters that are required by the algorithm.

In our proposal the individuals will be vectors of real components. The initial population it will be randomly created. The performance functions always will be RSS.

The selection process will be carried out by means of the tournament selection [8]. In order to avoid the premature convergence problem in each selection moment there only can be ten copies of the best individual. The genetic operators will be the multiple crossings and mutations. The parameters that will be used are population size, individual crossing probability, crossing repetition, individual mutation probability and mutation repetition. This is necessary because in our model the probability of crossing and mutation are the characteristics of each individual and not of the population as it is traditional in the GA. Besides it is considered that crossing and mutation can be multiple, that is to say, it can act several times with the same individuals. The individuals crossing it will be realized exchanging the components of the individual and the mutation of the individual is the change in value of a component of the individual among his limits.

The SAGA were developed by Thomas Back [1], [2] and have the characteristic that automatically search the best parameter to its functioning. In our case the population size for the problems in the stage correction it will be of 20, and when the SAGA will be utilized to calculate some autoregressive model the population size will be fix an equals 100. Also, in both stages we will use the same values for the four self adaptable parameters that will be: individual crossing probability that varies in the interval $[0.5, 0.95]$, crossing repetition $[1.0, 4.0]$, individual mutation probability that varies in the interval $[0.5, 0.85]$ and mutation repetition in $[1.0, 5.0]$. The individual for each stage it will

be solution proposals for the same and additionally will have four components more where it will be represented the values of the self adaptable parameters. The way of operating with these parameters is similar to the one presented in [1], [2] taking the respective average of the corresponding case. Next are described the way of calculating the autoregressive models using SAGA.

4. Calculus of the Autoregresive Models

The algorithm proposed it is developed considering as the function to minimize: the RSS. First, it is necessary to select the series that will be used among the following ones: the original series, the series of the first differences, the series of the second differences, and in our case it is included the possibility of working with the series of the third differences. In order to decide this is selected the series that presents the minimum variance, what assures that the selected series is stationary. Once decided the series that will be used it is calculated the number of necessary numbers for the linear approximation of the selected series taking into account the autocorrelation function considering all the terms between two consecutive maximums of this function. With the previous information the limits for the coefficient intervals of the chosen series are constructed, and to do this are all the coefficients are taken in $[-1, 1]$ except the independent term whose limits are estimated between zero and the average value of the series. The reason why these limits are established was obtained from the equation . Afterwards, 10 repetitions of 500 iteration were realized for each SAGA and with each results it is constructed a solution by means of the average sum of the absolute values of the 10 results. This gives information about the most important components since these will have the major value.

Later on the components of the previous solution are ordered from major to minor and it is selected the number of values of the model parameters. With the components already chosen and the same limits established beforehand 5 executions of SAGA are realized taking as stop criterion that in 250 previous iterations the optimum has not changed, and from these 5 executions it is selected the best result obtained.

Finally, with the above result, it is constructed a solution for the original series and with this result new limits are built taking in each variable of the solution a vicinity of radium 1. At this moment 5 more executions of SAGA similar to the above paragraph are realized and finally it is taken the best result of these 5. The smallest terms are neglected.

5. Series Correction

To find the missing values of the series it is calculated the minimum of the values different from zero of the series and the maximum of these. The first supposition that will be made with regards to the missing values is that these values are between these limits. For the zero values it is established the supposition that the values are in the interval $[-5, 5]$. With all these limits it can be generated the genetic algorithm that it is applied in the first stage.

Later on it is chosen a number K that is the number of terms of the autoregressive model that will be found for the series. To do this it is generated an initial population of 20 individuals (this really means the assignment of values between the limits to the missing and erroneous values of the series). In order to evaluate each individual it is considered the series that results of the correction of the original series applying the values that correspond to the individual and calculating an autoregressive model with K terms. The RSS it is taken as the fitness value of the individual for the model above mention.

Subsequently 30 iteration of SAGA are realized and the end it was found at a RSS value that indicates if the correction with K terms was good. This is realized by several K values, and ultimately by means of the BIC are chosen the number of terms utilized to modeling and which are the values that will be used to correct the series. It is important to notice that this stage is very short because is extremely costly in computational effort.

Several autoregressive models are calculated for different values of K close to the value before for the corrected series. This is done in this way because the iterations realized in the first stage were very few. With these results and applying again the BIC (5) it is decided what is the best autoregressive model of the series.

Finally, it is constructed the residuals series and for this it is calculated another autoregressive model with the procedure mention above obtaining this way the complete ARMA model.

5.1 Results of NN5-Reduced

In the Table 1 are shown for each example of NN5-reduce the time lags that appear in the component of AR and MA of each value, as well as the correspondent RSS value. The value of zero in some time slag corresponds to δ in the autoregressive part.

Table 1: Results.

Problem	AR	MA	RSS
101	<0,1,3,7,8>	<2,7,14,21>	146.775
102	<0,1,7,8>	<7,21,28>	241.653
103	<0,1,7,14,21>	<2,3>	59.172
104	<0,1,2,5,6,7>	<7,21>	247.434
105	<0,1,2,3,5,7>	<21>	140.760
106	<0,1,7>	<7,14,21>	201.833
107	<0,1,7,8>	<7,14,21>	165.027
108	<0,1,6,7>	<7,21,28>	157.073
109	<0,1,2,4,6,7>	<7,28>	245.326
110	<0,1,2,6,7,8,10, 14,16>	<21>	186.946
111	<0,1,6,7,11,12,14,16,21>	<14,21>	156.629

6. Appendix

Finally in figures 1-11 are presented the series graphs for the last 80 values, as well the corresponding forecast.

References

[1] T. Bäck, "The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm," in *Proc. 2nd Conf. on Parallel Problem Solving from Nature*, Brussels, Elsevier, Amsterdam, 1992.

[2] T. Bäck, "Self-adaptation in genetic algorithms," in *Proc. 1st Eur. Conf. on Artificial Life*, Cambridge, MIT Press, MA, 1992.

[3] G. E. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 1st ed., Oakland, Cal: Holden–Day, 1976.

[4] P. Cortez, M. Rocha, J. Neves, "Evolving Time Series Forecasting ARMA Models," *Journal of Heuristics*, vol. 10, num. 4, pp. 415-429, jul. 2004.

[5] D. E. Goldberg, *Genetic Algorithms in Search, optimization and Machine Learning*, 1st ed., Reading, Addison Wesley, 1989.

[6] V. M. Guerrero, *Análisis Estadístico de Series de Tiempo Económicas*, 1st ed., Cd. México, México: Thompson Ed., 2003.

[7] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 1st ed., Ann Arbor, Mich., University of Michigan Press, 1975.

[8] B. L. Miller, D. E. Goldberg, "Genetic Algorithms, Tournament Selection, and the Effects of Noise," *Complex Systems*, vol. 9, pp. 193-212, 1995.

[9] G. Shwarz, "Estimating the Dimension of a Model,," *The Annals of Statistics*, vol. 6, pp. 461-464, 1978.

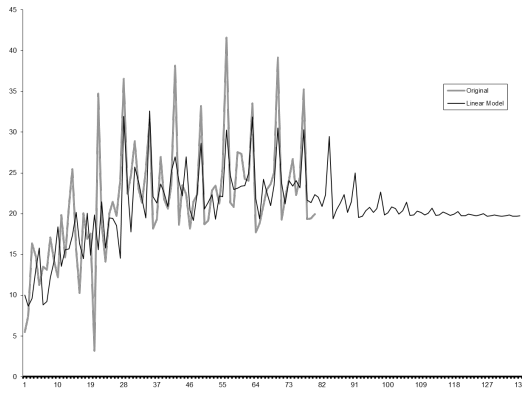


Fig. 1: Example 101.

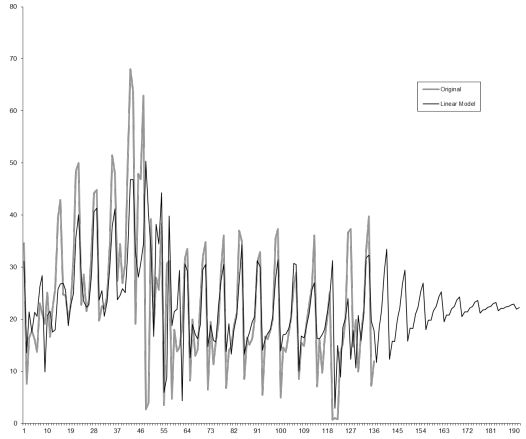


Fig. 2: Example 102.

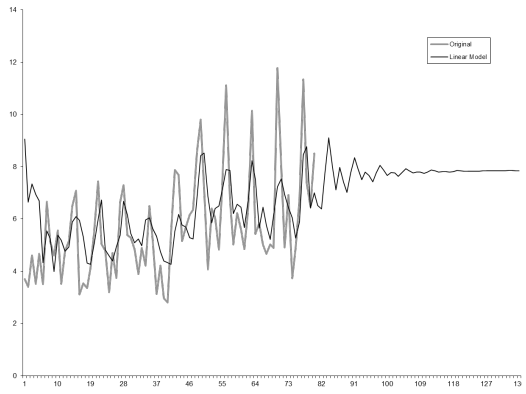


Fig. 3: Example 103.

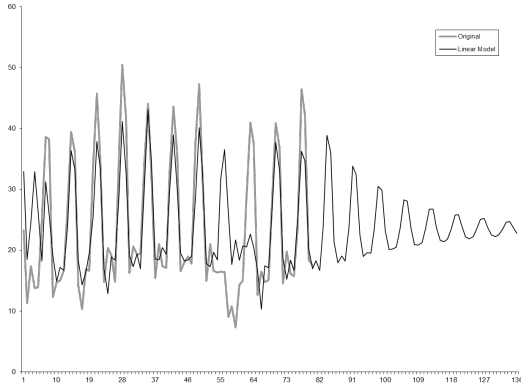


Fig. 4: Example 104.

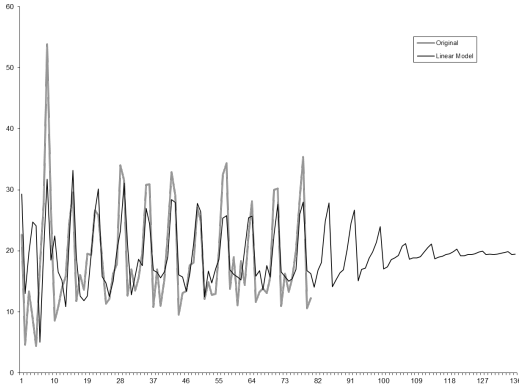


Fig. 7: Example 107.

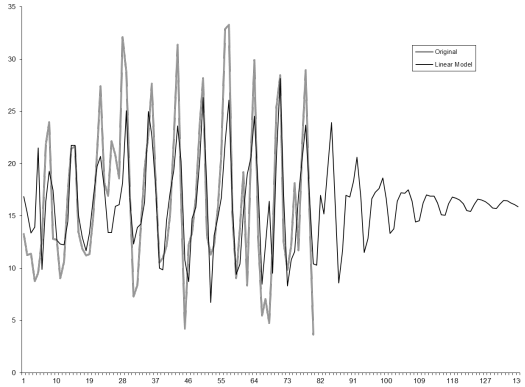


Fig. 5: Example 105.

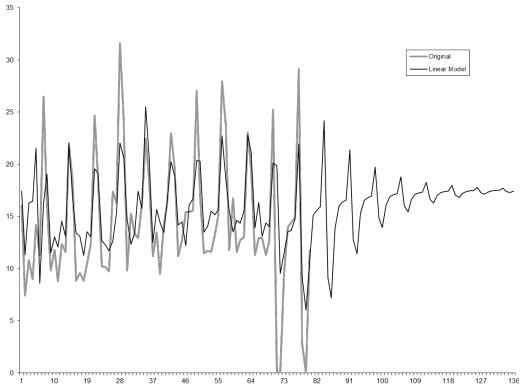


Fig. 8: Example 108.

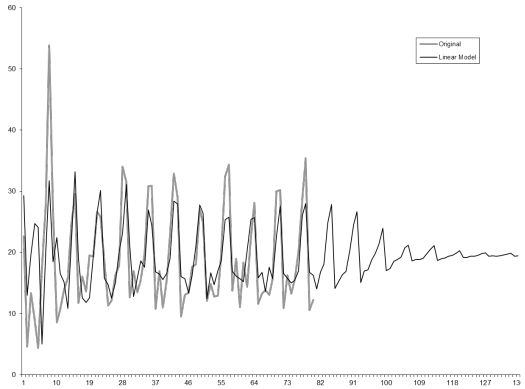


Fig. 6: Example 106.

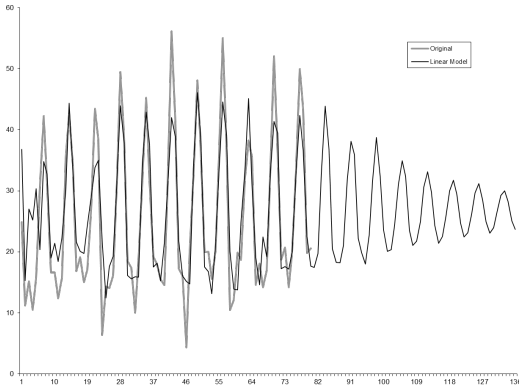


Fig. 9: Example 109.

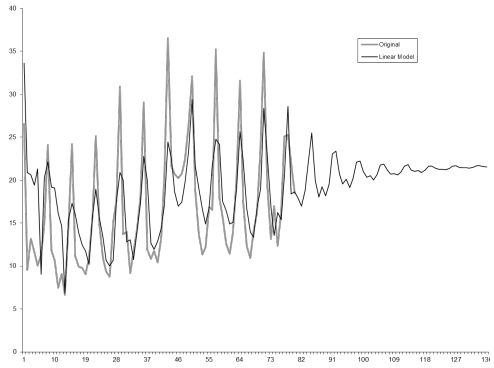


Fig. 10: Example 110.

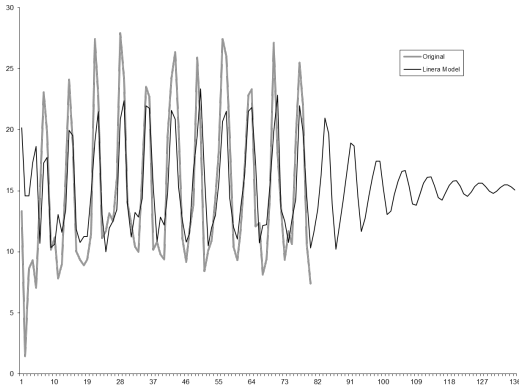


Fig. 11: Example 111.

Solving UAV Routing Problem with a Multi-Chromosome Representation Genetic Algorithm

Kien Ming Ng and Yen Joon Tan
Department of Industrial & Systems Engineering
Faculty of Engineering
National University of Singapore
10 Kent Ridge Crescent, S(119260), Singapore

Abstract - *An important vehicle routing problem in the military context is the routing of unmanned aerial vehicles (UAVs). In this paper, a multi-chromosome representation genetic algorithm (MCGA) is proposed to solve the UAV routing problem that is essentially of the form of multiple travelling salesman problem with time windows. We apply the MCGA to the data based on the Solomon's benchmark instances. The MCGA obtained solutions within 20% of the best known solution in an average time of less than 50 seconds for 100-customer instances. These findings suggest that the MCGA proposed is viable for solving UAV routing problems as it is able to produce good quality results within acceptable computation time for problems with realistic size.*

Keywords: UAV Route Planning, Vehicle Routing Problem with Time Windows, Genetic Algorithm, Multi-Chromosome Representation

1 Introduction

Unmanned aerial vehicles (UAVs) are typically used in the military to provide intelligence sources to commanders so that well-informed decisions can be made in the battlefield. Initial route planning of the UAVs is usually done by human operators manually [2]. Given that there may be many targets and operational constraints to consider, it is desirable to improve the quality and speed of deriving the solution with a heuristic that outperforms manual routing.

A main objective of UAV route planning for reconnaissance missions is to solve the problem of task allocation and routing of multiple UAVs, using the minimum number of UAVs. Thus, the quality of a solution is evaluated based on the number of UAVs used. This is to take into account of the nature of military operations where UAV resources might not always be available because of operational constraints. However, due to the time critical nature of military missions, the quality of the solutions could sometimes be secondary to the time taken to obtain the solutions.

In this paper, the problem of interest is the routing of multiple homogeneous UAVs to complete their missions by visiting all targets at given time windows. It is in the form of the multiple travelling salesman problem with time window constraints (MTSPTW) that minimizes the required number of UAVs. Given the computational difficulty of solving even the basic travelling salesman and vehicle routing problems of large size, the use of heuristics or metaheuristics is common, especially when the problem size exceeds 50 customers [1].

2 The UAV Routing Problem

In this section, we present a model of the UAV routing problem. This model is modified from the model in Tan *et al*. [6]. In particular, the service time at any target is assumed to be equal to the time window of the target in the UAV routing problem as this is the planned surveillance time for the target. This is different from what is known generally for usual routing problems, such as vehicle routing problem with time windows (VRPTW) and MTSPTW, where service time is independent of time windows.

2.1 Notations Used

We first define the sets, parameters and decision variables that are to be used in the formulation of the UAV routing problem.

The sets include the following:
$U = \{1, 2, \ldots, K\}$ is the set of all UAVs.
$T = \{1, 2, \ldots, N\}$ is the set of all targets.
$T' = T \bigcup \{0\}$ where 0 denotes the base.

The parameters include the following:
D_k = maximum endurance of UAV k ($k \in U$).
s_i = time to perform mission at target i ($i \in T$).
a_i = start of the time window for the assigned UAV to perform mission at target i ($i \in T$).
b_i = end of the time window for the assigned UAV to perform mission at target i ($i \in T$).
t_{ij} = time taken to travel from target i to target j ($i, j \in T, i \neq j$).

The decision variables include the following:
t_i = time that the assigned UAV arrives at target i ($i \in T$).
l_i = loiter time for the assigned UAV at target i before the start of the time window for target i ($i \in T$).

$$x_{ijk} = \begin{cases} 1 & \text{if UAV } k \text{ travels from target } i \text{ to target } j \\ 0 & \text{otherwise} \end{cases}$$
$$(i, j \in T,\ i \neq j,\ k \in U).$$

2.2 Mathematical Programming Model

The model modified from [6] is then formulated as follows:

$$\text{Minimize} \quad \sum_{j=1}^{N}\sum_{k=1}^{K} x_{0jk} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{N}\sum_{k=1}^{K} x_{0jk} \leq K \tag{2}$$

$$\sum_{j=1}^{N} x_{0jk} = \sum_{i=1}^{N} x_{i0k} \leq 1 \qquad (k \in U) \tag{3}$$

$$\sum_{\substack{j=0 \\ j\neq i}}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad (i \in T) \tag{4}$$

$$\sum_{\substack{i=0 \\ i\neq j}}^{N}\sum_{k=1}^{K} x_{ijk} = 1 \qquad (j \in T) \tag{5}$$

$$\sum_{i=0}^{N}\sum_{\substack{j=0 \\ j\neq i}}^{N} x_{ijk}(l_i + s_i + t_{ij}) \leq D_k \qquad (k \in U) \tag{6}$$

$$t_0 = l_0 = s_0 = 0 \tag{7}$$

$$a_i + s_i = b_i \qquad (i \in T) \tag{8}$$

$$\sum_{\substack{i=0 \\ i\neq j}}^{N}\sum_{k=1}^{K} x_{ijk}(b_i + t_{ij}) \leq t_j \qquad (j \in T) \tag{9}$$

$$t_i + l_i = a_i \qquad (i \in T) \tag{10}$$

$$t_i,\ l_i \geq 0 \qquad (i \in T) \tag{11}$$

$$x_{ijk} \in \{0,1\} \qquad (i, j \in T',\ i \neq j,\ k \in U) \tag{12}$$

Equation (1) shows the objective function of minimizing the number of UAVs used. Constraint (2) ensures that there are no more than K routes out of the base. Constraint (3) guarantees that every UAV that leaves the base also returns to the base at the end of the tour. Constraints (4) and (5) make sure that each target is visited exactly once only. Constraint (6) ensures that the endurance of any UAV is not exceeded in each tour. Constraint (7) initializes the time parameters at the base to zero. Constraint (8) takes into account of the mission time at each target as the time window duration. Constraints (9) and (10) ensure that precedence constraints are met. Constraints (11) and (12) restrict the range of the variables. In the next section, we describe a multi-chromosome genetic algorithm (MCGA) that can be used to solve the UAV routing problem, taking into account most of the above constraints.

3 The Proposed MCGA

Király and Abonyi [3] proposed a novel chromosome representation based genetic algorithm to solve the multiple travelling salesman problem. It was reported that this multi-chromosome representation can reduce the search space and is easily interpretable. This is in contrast to the single chromosome representation by Tan *et al.* [6], which requires an additional step to retrieve the information on customers that are allocated to each vehicle. However, the objective considered in [3] involves the transportation cost and there is no detailed discussion about how the time window constraints are handled, while only a small scale problem of 25 customers is used to show the effectiveness of the algorithm in solving the problem. Here, the MCGA described in this paper is used to solve the MTSPTW with a larger scale of 100 targets that focuses on minimizing the number of UAVs required.

3.1 Multi-Chromosome Representation

In the multi-chromosome representation, each individual chromosome of a population is subdivided into multiple sub-chromosomes, which represent one UAV each. Thus, each sub-chromosome contains the targets assigned to the corresponding UAV and the order in which the targets are visited. For example, a sub-chromosome k would represent UAV k being deployed to visit targets $i_1^{(k)}, i_2^{(k)}, \cdots, i_{n_k}^{(k)}$ in this order, where n_k is the total number of targets visited by UAV k. Here, $i_p^{(k)} \neq i_q^{(l)}$ whenever $k \neq l$ or $p \neq q$. Also, the sub-chromosomes are only created when a new UAV is required to visit the remaining targets, with the number of sub-chromosomes in chromosome c, $K_c \leq K$ and $T = \bigcup_{k=1}^{K_c}\bigcup_{p=1}^{n_k}\left\{i_p^{(k)}\right\}$. Note that the output of such a representation is easily interpretable and does not require additional decoding or processing to retrieve the solution.

3.2 Initial Population

When forming an initial population for the MCGA, some chromosomes are created using simple heuristics, such as the Earliest Due Time, Earliest Opening Time and Tightest Time Window. The rest of the chromosomes are created and initialized with a random starting feasible route using the Push Forward Insertion Heuristic (PFIH) introduced by Solomon [5]. Here, only feasible solutions with no violation of any constraints would be created and added to the initial population. The PFIH would determine the best insertion position for each unrouted target and choose the target that results in the largest amount of savings, while taking into consideration of both distance and time window feasibility. Therefore, only feasible insertions are being made. If no more feasible insertions can be performed on a UAV, a new UAV would be assigned for insertion. This will repeat until all

targets have been assigned to a UAV or the maximum number of UAVs available has been utilized.

3.3 Crossover

After an initial population has been obtained or after each generation of evolution, selection of chromosomes as parents to the next generation would be done based on the following fitness f_c for each chromosome $c = 1, 2, \ldots, N_C$:

$$f_c = \frac{1}{K_c} \tag{13}$$

where N_C is the number of chromosomes in a population. In the selection process, elitist strategy is used, while the roulette wheel selection method is used for the selection of parents for crossover in which the probability p_c of chromosome c being selected is given by:

$$p_c = \frac{f_c}{\sum_{i=1}^{N_C} f_i}. \tag{14}$$

In our approach, a single parent operator (see e.g., [4]) that takes into account of the time window constraints is used. Firstly, two sub-chromosomes are randomly chosen from the selected parent chromosome. Then instead of simply appending the genes of one sub-chromosome, say A, to the end of the other sub-chromosome, say B, each gene of A would be inserted to the first available and feasible position in B. This would be done until all the targets in A are completely inserted into B, or if there are no more feasible insertion places for the remaining targets of A. In the event that all targets in A are inserted in B, it would resemble the crossover operator of simply combining two sub-chromosomes, leading to a reduction in the number of UAVs required. In situations where not all targets in A were inserted into B, all insertions made previously would be undone and returned to the original state before crossover with a probability equal to the mutation probability.

3.4 Mutation

With the multi-chromosome representation, four different methods of mutation have been adopted in this MCGA. They are classified into in-route and cross-route mutations based on the classification in [3]. In particular, two methods of in-route mutation and two methods of cross-route mutation as described next are implemented. For the first in-route mutation, two genes would be randomly chosen from the same sub-chromosome that was randomly selected initially, and a swap would be performed. For the second in-route mutation, a gene would be randomly chosen from a randomly selected sub-chromosome. Then, a new position in that sub-chromosome would be randomly chosen, and the selected gene would be removed from its original position and inserted into the new position. For the first cross-route mutation, two genes would be randomly chosen from two randomly selected sub-chromosomes, and a swap would be performed. For the second cross-route mutation, a gene would be randomly chosen from a randomly selected sub-chromosome. Then, a new position in another randomly selected sub-chromosome would be randomly chosen, and the selected gene would be removed from its original position and inserted into the new position.

When a chromosome is selected to mutate, one of these four mutation methods would be invoked randomly. If the mutation is feasible, the mutated chromosome would be copied to the new generation. Otherwise, the mutation would not take place.

4 Computational Results

The performance of the MCGA has been tested with the 56 100-customer Solomon instances [5], which are well-known benchmark tests for the VRPTW. Although the UAV routing problem is of the form of the MTSPTW, the best known solutions to the Solomon instances would serve as an indicative bound to the solutions of the equivalent MTSPTW instances, which do not have the VRPTW capacity constraints.

The MCGA was designed and implemented using the GAlib [7]. The code used to implement the MCGA and the associated operators were written in the Visual Studio C++ Express 2008 environment, and the tests were conducted on a 32-bit 2GHz Intel Core 2 Duo Processor with 2GB of memory, running on the Microsoft Windows Vista operating system. The population size for the MCGA computational runs is 30 and the number of generations is 200. The probabilities of crossover and mutation are 0.7 and 0.05 respectively. This set of parameters was selected based on sample trial runs done on selected problems with different sets of parameters.

Table 1 shows a comparison of the number of vehicles required using the proposed MCGA to that of the best known solutions for the Solomon's instances (Best NV). The average number of vehicles required over all the 56 instances is within 12% of the average Best NV value. For each of the problem types, the MCGA results are found to be within 20% of the Best NV value, and MCGA performed the best for problem type C1 with only a percentage difference of less than 2%. Note that the percentages are generally smaller if the best MCGA values have been used in the comparison instead of the average MCGA values used in the table.

Table 1: Comparison of average MCGA objective values and computation time

Problem Type	Best NV	MCGA Average	Percentage Difference	Computation Time (secs)
C1	10.00	10.18	1.78	13.58
C2	3.00	3.48	15.83	63.15
R1	12.25	13.87	13.20	10.82
R2	2.82	3.18	12.90	92.02
RC1	11.63	13.73	18.06	11.50
RC2	3.38	3.80	12.59	69.10
Average	7.36	8.23	11.89	43.11

Table 1 also summarizes the computation time needed by the MCGA. The average computation times required to solve the 56 Solomon's problems is found to be 43.11 seconds. However, the computation times for problem type 2, namely C2, R2, RC2, where time windows are larger and more targets assigned per vehicle, are correspondingly longer than that for problem type 1.

5 Conclusions

In this paper, a multi-chromosome representation genetic algorithm has been applied to solve the UAV routing problem in the form of the MTSPTW with the objective of minimizing the number of UAVs used. The output of the representation is easily interpretable and does not require additional processing to retrieve the solution. From the computational results, it is found that the MCGA is able to provide good quality solutions that are within 20% from the best known solutions. The average computation time of 43 seconds for the 100-customers instances is also encouraging as realistic UAV routing problems are expected to be of comparable problem size.

Future work includes applying the MCGA to solve larger instances to test the scalability of the algorithm. It is also possible to reduce the computation effort required by the MCGA through improving the choice of parameters used in the MCGA.

6 Acknowledgements

This research work has been supported by the Temasek Defence Systems Institute (TDSI) under the following grants: R-266-000-054-232, R-266-000-054-422 and R-266-000-054-592.

7 References

[1] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5), pp. 512-522, 2002.

[2] G. W. Kinney, J., R.R. Hill, and J.T. Moore, Devising a quick-running heuristic for an unmanned aerial vehicle (UAV) routing system. *Journal of the Operational Research Society*, 56(7), pp. 776-786, 2005.

[3] A. Király and J. Abonyi, A novel approach to solve multiple traveling salesmen problem by genetic algorithm, in *Computational Intelligence in Engineering*, I. Rudas, J. Fodor, and J. Kacprzyk, Editors. 2010, Springer Berlin/Heidelberg. pp. 141-151.

[4] S. Liu, W. Huang, H. Ma. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, 45(3), pp. 434-445, 2009.

[5] M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), pp. 254-265, 1987.

[6] K.C. Tan, L.H. Lee, Q.L. Zhu, K. Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering*, 15(3), pp. 281-295, 2001.

[7] M. Wall. GAlib: A C++ library of genetic algorithm components. Massachusetts Institute of Technology, 1996, http://lancet.mit.edu/ga/