SESSION

GENETIC + EVOLUTIONAY ALGORITHMS

Chair(s)

TBA

Universal Adaptable GA with Engineering Help Tutorial

James G. Schiiller, Greg L. Vitko, and Christian P. Wagner
Department of Industrial and Systems Engineering, Oakland University, Rochester, MI, United States

Abstract - *The Many treatises on genetic algorithms (GA) analyze particular methods of developing problem specific GA and few try to develop a comprehensive analysis of combinations of all methods into a universal framework. Such a universal framework could be used to develop what could be called a Universal Adaptable GA. One of the problems of GA is that no one exactly knows all the details. These can be the ever-changing parts and cannot be made static at this time until they are more fully understood. Other areas are well understood, these should be the static parts in the universal adaptable GA. The goal of this work is to explain both general and problem specific areas by an example with a variant of Tetris as case study to demonstrate how a GA can solve problems. This example will make GA easier to understand and replicate.*

Keywords: Artificial Intelligence in Games, DNA, Genetic Algorithm, Tetris

1 Introduction

The development of faster computers in the twenty-first century has made it possible for computers to explore their environments and create solutions for many types of problems without much human intervention. The computer is no longer seen as a big storage bin, but as a useful supplement to help people solve complex problems, on condition: we have to give the computer the right set of instructions. Coming up with these right set of instructions is a problem. The instructions given have to be both general and specific enough to satisfy the minimum requirements for responding to many different types of problems. Responses do not have to be perfect. The universal adaptable genetic algorithm (GA) has been proposed as a suitable method to apply to get the computer to respond to many different types of problems for us.

Before continuing on to having the computer respond to some challenging problems, we have to have a clear understanding of how a computer GA exactly works, absent from all the common misunderstandings. After understanding, we can proceed to deal with more difficult aspects of the GA and see why developing the universal adaptable GA is a good idea. It is the high hopes of this research to start to assemble a universal adaptable GA, for other researchers to adjust and expand and apply to larger scale problems. How this is done is by demonstrating an application of GA to a variant of the game of Tetris.

2 Quality of Solutions

The computer can be used as a resource to solve real problems by applying a GA and running it to speed up millions of years of evolution by simulation within an hour's amount of time. The result is the creation of an "individual" who can solve problems in a real environment. In order for the computer to be resourceful for us, we have to first understand the GA. Knowing what quality of solution it seeks is a good place to start.

The GA aspires to converge on the equilibrium solution to satisfy the minimum requirements of a problem. This is an adequate compromise when facing new problems. The GA does not normally seek to converge on perfection, so there has to be this compromise of a robust solution but not necessarily the optimum solution. For example, a genetically bred thoroughbred horse can be seen as a genetic solution for short distance speed, but even champion thoroughbreds have been defeated in extreme distance horse races by "mutt" American mustangs without as specialized short race breeding. The point is the realistic goal of any artificial general intelligence method including GA is to find the mutt.

3 Genetic Algorithm

John Holland discovered the GA in terms of AI for applying it to computers [1]–[3]. Holland first started looking around at how the GA even works. We know it takes place from sex. There is the DNA strand from the father and the DNA strand from the mother. Somehow two DNA strands come together and create a child. The first questions he asked then were "What's the mechanism by which this happens? What do you do? How do you get these DNA strands to work? What's the mechanism behind it?"

3.1 Genetic Algorithm Outline

The overall idea of what is going on with genetic learning is shown in the steps below. These are the steps in genetics given to the computer to use genetics to create intelligence.

The steps are as follows:

1) Define what a DNA strand is for our setting AND we have to define an interpreter that transforms any DNA strand into an individual that performs in the environment.

2) Randomly create a population of *N* individuals (e.g.,1,000 individuals).
3) Now we have to determine the fitness of any individual.
4) We create a new population of 1,000 by creating pairs of parents to produce one offspring 1,000 times. Pick two parents at random where the fitness of each parent affects the likelihood that it is selected (the better the fitness, the more likely it is selected). When the two parents are selected, the new child is determined by genetic operators: e.g., crossover, inversion and mutation.
5) Keep the new population of 1,000 and discard the old (many variations exist – e.g., keep the 5% best, can one individual be father and mother?).

The GA has a simple outline, yet the devil is lurking in the details. Keep in mind this is an oversimplification of how real DNA works in biology.

3.2 Mapping the Outline

Step one in the preceding outline asks us to define the DNA strand. The definition of the DNA strand in the universal adaptable GA is shown below. The words DNA strand, chromosome, organism, and individual are interchangeable words. The length and number of organisms are scaled down for illustration. Contrast the number of individuals below with the example number given in the outline. There would normally be more than 10 initial organisms in the initial population, as the steps in the GA specified there should be 1,000 individuals. Additionally, the DNA strands below only have a chromosome length of 10. The size of the organisms listed is a gross simplification of DNA in GA. Remember the goal is to avoid all the common misunderstandings. Some definitions have to be defined and simplifications made on smaller scales, to see the respective computer model part matched to the equivalent part in the biological genetics.

$organism_0$ = (0000000000)
$organism_1$ = (0101010101)
$organism_2$ = (1010101010)
$organism_3$ = (1110000111)
$organism_4$ = (0011001100)
$organism_5$ = (1100110011)
$organism_6$ = (1010101010)
$organism_7$ = (0011111100)
$organism_8$ = (0000000000)
$organism_9$ = (0101010101)

Each gene in the organism is equivalent to a pair of zeros in the first $organism_0$ = (0000000000) <=> (|00|00|00|00|00|), where the '|' delimits the beginning and end of a gene. The full four letter genetic alphabet ACGT can be represented the same way in a computer using the following binary assignments: A = 00, C = 01, G = 10, T = 11.

Everything we perceive can be encoded in "0's" and "1's." Computers are only limited by the hardware, thanks to current software implements of larger integer values.

This completes the illustration of the DNA strand of an organism from the outline. There is however two other things in the outline we need to map for step 1 to be complete. First, the interpreter must be defined, but we find quickly we don't know what to interpret – there is no problem to respond to – yet. It will have to wait until later when more information is gathered on the specific example problem given in a later section. Second, the environment has to be defined.

Stated more implicitly in the outline is the environment (or input) to define, or the part of the environment to pay attention to. These implicit details are one of the places the devil lurks. Are the environmental inputs words, sentences, shapes, etc.? It likewise will have to wait until a specific problem is given. It's the same situation the interpreter had. This lack of information to define these parts tells us something important. The Environment (Inputs) and the Interpreter need to be replaced for every different problem encountered, making them changeable parts. There are a few other similar problem specific elements to go on the changeable parts list. Altogether, they are the 1) Inputs, 2) DNA Instructions, 3) Interpreter, 4) Goals, and the 5) Outputs. These elements all need to be implemented differently for each unique problem.

Reproduction can also be done many different ways. It is reasonable to use a cumulative distribution function for the selection of parents, stochastic crossover, and 5% mutation rate for each bit position in the gene; these are as good as any other methods to use. These genetic operators are candidates to be the sixth part on the changeable parts list. They will be static here for simplicity of keeping the changeable parts list down to five items.

The other remaining steps can be mapped. Mapping the rest of the steps outlined above to something closer to computer language yields the following algorithm. Notice, this step would not be necessary if a universal adaptable GA was available as proposed. For now there is no universal GA, so the following steps are an idea about how to map the steps given earlier as a main function for the universal adaptable GA. These steps are the fixed parts to the Universal Adaptable GA.

Algorithm 1 Universal Adaptable GA Main

Input: Environmental inputs, initial chromosomes.
Output: Highest rated individual.

```
1: repeat for number of GENERATIONS (e.g., 2)
2:   repeat for number of ENV INPUTS (e.g., 14 shapes)
3:     repeat for number of CHROMOSOMES (e.g., 10)
4:       interpret (chromosome, environmental input)
5:   rate chromosomes
6:   if generation < GENERATIONS
7:     reproduce
8. save highest individual
```

If the individual is to perform in an actual real life environment, there needs to be another small addition made after the outline so the individual can perform in the environment, as shown below.

Algorithm 2 Universal Adaptable GA Perform

Input: Environmental inputs, chromosome.
Output: Response to problem.

repeat for number of ENV_INPUTS (e.g., 14 shapes)
interpret_real_life
(chromosome, real environmental input)
rate chromosome

Everything is mapped besides the changeable parts. Actually, these are the hardest parts to get right to make use of genetics. Before defining these parts, we need a problem to solve. In order to get something more concrete, an example problem is given.

4 Example Problem

What we have chosen for our fist problem to solve is the fairly well-known problem of dropping dies (or parts) on a cloth to make seat covers. The problem has many varieties. All varieties of the problem generally try to fit shapes on a surface, without overlapping, trying to get the least amount of waste as possible. You could add as many constraints to the problem as you like. You have to stop somewhere to get something done. The point being, the added constraints you impose should be implementable within a day's work. This is the right level of generalization without having to start over each time with a fresh implementation whenever presented with a new constraint, or problem.

4.1 Tetris Variant

We look at most hard problems as games. Instead of looking at the problem as cloth cutting, it can be fitted into something more entertaining, into a variant of the game of Tetris, with the same inherent problems. See Figure 1 for an arrangement of how the Tetris shapes should be placed on the board (or cloth) in a compressed format.

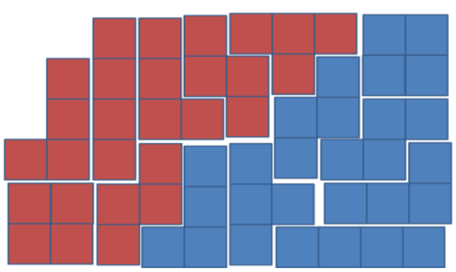


Figure 1: Tetris blocks arranged in compressed format.

The object of the "Tetris" game is to create an individual player, which places the different parts, the Tetris blocks called tetriminos, "tets" for short, on the board, opting to place the most pieces on the board as possible. It is easier to start with a smaller scale problem and solve the problem on this smaller scale before scaling it up to the same problem with larger size. You should eventually be able to scale it up to an inexhaustible size, and get a good enough solution using the same universal GA and adapting it to the larger board size with a little engineering help. Of course, first you have to understand how to solve the smaller scale problem before it will take a day's work to expand it to a larger scale problem. For our concerns the size of the board is a sheet of 6x10 squares. See Figure 2 for the board representation used.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2: Inner board of 6x10 squares.

The border of four extra squares surrounding all sides of the inner board is to ease with implementation details, and seems to be a recurring way to make problems more implementable, instead of having to do level logic, such as, if this, else if that, else this, and so on. Exactly four outer board squares is constant for any sized board, for reasons shown later. It may have its genetic equivalence in biology as part of the extra bits material that doesn't seem to be used, in the non-coding regions of DNA, termed the dead code as opposed to the inner board, termed the active DNA or active code. It has definitely been useful when placing the proximity numbers on the smaller contained board, not having to worry about if the shapes go over the inner board's area.

4.2 Changeable Parts

The variant of the game of Tetris is the specific problem needed before being able to define the abstract parts into something concrete. The parts of GA on the changeable parts list, the 1) Inputs, 2) DNA Instructions, 3) Interpreter, 4) Goals, and 5) Outputs will now be defined.

5 Mapping Tetris Variant

5.1 Environmental Inputs

All shapes have four squares. There are a total of seven different types of tets. All types can be seen looking back to Figure 1: one of each tet shape in each symmetrical triangular group split by the diagonal. The actual environmental inputs will come in as shown in the following sequence.

square, s, z, T, L, J, l, square, s, z, T, L, J, l

Any amount of shapes which can cover at least 90% of the board space is reasonable to consider. Anything less is too

trivial. There are 14 shapes, since 14 fills over 90% of the board. The tets could come down in random order, whatever the constraints they come down in; here they come down in fixed order. However elements you are sensing or suppressing usually have random behaviors.

5.2 DNA Instructions

The DNA strand (or genetic string) has been defined in a previous section, but without DNA instructions, DNA cannot do anything. Recall that DNA instructions are one of the different changeable parts for each implementation. Each gene has a different set of DNA instructions. The first DNA instructions that we define for the first gene are called proximity numbers. When the chromosome recognizes a shape, it categorizes it according to its DNA. Specifically, its gene value can be |00|, |01|, |10|, or |11|. The method each chromosome uses for proximity number instructions is decided by the chromosome's DNA by use of a case statement 0-3 with strategy names: TRIVIAL, SIMPLE, COMPLEX, MIXED (See Figure 3). There are a total of seven different tets. Strategy names are listed at the bottom. The proximity numbers keep the tets glued together by directing placement of next tets.

Tet(0, 1)	Tet(1, 1)	Tet(2, 1)	Tet(3, 1)
(000000000)	(000000000)	(000000000)	(000000000)
(000000000)	(000000000)	(000000000)	(000000000)
(000000000)	(000000000)	(000000000)	(000000000)
(000000000)	(000001000)	(000011000)	(000001000)
(0000##000)	(0000##000)	(0001##100)	(0000##100)
(0000##000)	(0000##000)	(0001##100)	(0001##000)
(000000000)	(000000000)	(000011000)	(000010000)
(000000000)	(000000000)	(000000000)	(000000000)
(000000000)	(000000000)	(000000000)	(000000000)
TRIVIAL	SIMPLE	COMPLEX	MIXED

Figure 2: Definition of the square tet. The # symbols represent shapes. The numerical values surrounding the shapes are proximity numbers.

The second set of DNA instructions for the second gene values to choose from is called Search Procedures (SP). One tricky part of studying DNA is in knowing the difference between the genes and the instructions carried out by the genes. For the second gene on the Tetris DNA strand, the |00| decides which SP the organism uses, meaning which set of instructions to carry out when searching for an open square to place the incoming tet. In other words, what set of DNA instructions does it select? Here there are only four different choices, SP0, SP1, SP2, and SP3; however, there can be more, but four is sufficient to gain an understanding. Notice there are 60 squares on each SP to match the size of the inner board. See Figure 4 for the four SP instructions serving as the second set of DNA Instructions for the DNA to select 0-3.

SP_0

1	6	15	29	37	38	30	16	7	2
5	22	28	42	48	55	49	43	23	8
14	27	36	47	54	59	56	50	31	17
13	26	35	46	53	60	57	51	32	18
12	21	25	41	45	58	52	44	24	9
4	11	20	34	40	39	33	19	10	3

SP_1

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60

SP_2

1	2	4	10	11	21	22	33	34	45
3	5	9	12	20	23	32	35	44	46
6	8	13	19	24	31	36	43	47	54
7	14	18	25	30	37	42	48	53	55
15	17	26	29	38	41	49	52	56	59
16	27	28	39	40	50	51	57	58	60

SP_3

11	16	46	37	30	52	9	39	20	4
29	35	2	15	24	56	47	3	33	27
43	59	49	38	31	14	41	26	44	54
17	12	8	23	1	32	10	53	58	18
7	55	51	40	19	60	34	21	13	48
22	28	5	6	45	57	25	50	42	36

Figure 4: Search Procedures

When following the DNA instructions for the second gene, the chromosome starts by trying to drop pieces on the inner board beginning at the corners with squares labeled 1, 2, 3, 4, and so on until 60, maybe moving in an inward pattern towards the center or maybe starting at a corner (1) and moving towards the other side on the diagonal if the DNA instruction specifies. The search procedures can be defined with any pattern, even random ones. The creator decides what to use for DNA instructions. The SPs in the figures are only examples.

5.3 Genes

It is important to notice there are only two genes |00|00| being actively used on the DNA strand. The other three genes are not being used for anything, although they could be. Some of the other organisms have some of their extra gene bits set to 1. Still they are not being used for anything at the moment. They may be activated for a different problem. Five total genes is a gross simplification of how DNA really works. Recall the mile high stack of procurement documents needed just to build the C-5 aircraft. Imagine the amount of information needed in DNA to build a human, or even a big toe. The DNA is doing much more than it first appears, and therefore should definitely be far longer than 10 bits in length.

5.4 Interpreter

This is entering into interpreter territory, another one of the changeable parts to be made abstract. The interpreter is the key part to the GA. Despite the recent “solving of the genome” [4], with the many correlations found of genetic material to body characteristics, it appears that even with the DNA “source” code, we still do not comprehend exactly how this code is interpreted. We emphasize the interpreter, looking for a general purpose one, but once one is found, then the secret is figured out. For now all we have is the option to plug our dreams in there for how the interpreter works. The interpreter goes from the DNA strand to the full formed individual that works in the environment. Here, “interpreter” means roughly the same thing as it does in a language such as java. Something has to make the programs work. Something has to interpret the DNA strand to work in the environment. It is the problem of going from genotype to phenotype [5]. The cell in biology depends on the context it finds itself in. In Genetic Programming [6] the interpreter is done for us. Not everything can be written as nice little programs, and the GA is pulled off the shelf again to tackle harder problems. In the GA, this interpreter is difficult to define [1] for each problem. Since no one knows how it is exactly done in genetics, all ideas are just wild guesses and dreams. For us, the unknown has to point to some address in a universal GA, to where different interpreters can plug in as any genetic code interpreter you want. The interpreter interprets the given chromosome in the given environment to make the transformation into an individual that performs in the environment; in our case, a player that plays the variant game of Tetris. For an analogy, a book or blueprint is nothing without someone to read and interpret the book or interpret the blueprint to build the house. The perfect example to think about as an interpreter is the mother’s womb. Somehow the mother’s womb forms a baby out of the DNA instructions.

In our case there are only two pieces of data to interpret, the 1) Proximity Numbers, and the 2) SP. The interpreter, as was shown in the previous algorithms, takes the following form. Depending on the situation, there may need to be two interpreter(s) defined.

interpret (chromosome, environment)
interpret_real_life (chromosome, real environment)

5.5 Goals

The goal for this Tetris problem is fairly simple; to have the greatest number of tets placed on the board. Placing all 14 shapes on the board would be a perfect solution. There would be a minimum waste of only four squares. Goals are also related to constraints. The constraints for this problem are: shapes cannot overlap, and shapes have to be placed within the inner board area. If any portion besides the proximity numbers goes outside of the inner board’s area, it is not a valid placement, and therefore is not placed. For example, the following in Figure 5 could not take place.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	C	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	#	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	#	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	#	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	#	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 5: Invalid Placement of Line Tet.

The line shape ‘l’ runs outside of the inner board. Now it can be seen why the square padding around the outside of the inner board needs to be size four. This is the minimum amount needed to contemplate placing the longest shape ‘l’ on the borders of the inner board. Anything larger would be wasteful, and anything shorter would not suffice. This contemplation is the farthest case where one square is on the board as close to the edge as possible. The two miniature squares are where placement starts and ends when contemplating placement. The 9x9 squares show the placement of the whole shape’s area for proximity number. The square labeled ‘C’ is the center of another furthest possible proximity number to contemplate shape placement. Starting from the left top corner, using coordinates (1, 1) as the first miniature square on the corner extends to the (6, 10) second miniature square, spans the exact same size as the inner board.

The ranking computation sweeps over the entire inner board of chromosomes at the end of generations. For each ‘#’ sign it adds 1,000 points. These are the places where the shape’s area is placed on the board. The proximity numbers falling within the inner board are added on to this number to give the total score for the chromosome. The proximity numbers add when overlapped. For instance, placing 14 shapes on the 6x10 cloth would sum to 56,000 points (14 shapes x 4 squares per shape) + overlapping proximity numbers, say 15 more points, totaling 56,015 points, a perfect solution.

5.6 Outputs

The output is the last changeable part on the list. It is possibly more effective to understand the output by seeing a demonstration of the algorithm in action.

6 Demonstration

Even though the outline of the GA is not very complicated, it is helpful to be able to see everything sketched out at a detailed level to explain. So the demonstration uses this strategy to show the GA in process.

6.1 Mapping Outline

1) Problem: Place 14 tets on a 6x10 (60 square) board with the least amount of waste.
2) Tets come in to the cognitive system in the order: square,s, z, T, L, J, l, square, s, z, T, L, J, l.
3) One by one, shapes are perceived by each chromosome.
4) The chromosome recognizes the first shape as being a square. The interpreter translates what the |00| on the first chromosome's first gene means for the shape coming down. It means select the TRIVIAL strategy, or naked shape, meaning no proximity numbers around it.
5) Then, according to the chromosome's DNA value at the second gene, it selects one of the four search procedures to start trying to place the incoming tet. The range of places to try is from 1 to 60. If it cannot place the tet, it tries the next place in sequential order. Once it reaches 60, there is nothing more it can do for this shape. There is nowhere else to try to place the tet. Its board squares are all used up.
6) The above are done for two generations with 10 initial chromosomes. At the end of each *N*-1 generation, the individuals are ranked. Then parents are selected and offspring are reproduced. For the *N*th generation, selection and reproduction does not occur. The highest ranking individual from the last generation is saved to perform in the real environment.

The first attempt at placing 14 tets on a 6x10 board resulted in many isolated squares on the inner board. The isolated squares were greatly fragmented, not a good characteristic, and a low ranking best individual was produced, which took one hour to complete. The slow execution time is offset by being able to visually see and able to visually debug the GA in a spreadsheet G.U.I. and to allow quick and automated adjustments. There are a couple of things to try in order to correct generating these bad results.

Tet(2, 4)	Tet(2, 4)	Tet(2, 4)	Tet(2, 4)
(000000000)	(000000000)	(000000000)	(000000000)
(000000000)	(000000000)	(000010000)	(000000000)
(000000000)	(000000000)	(0003#1000)	(000000000)
(000010000)	(001110000)	(000##1000)	(00003#300)
(0001#3000)	(01###1000)	(0003#1000)	(0001###10)
(0001##000)	(003#30000)	(000010000)	(000011100)
(0001#3000)	(000000000)	(000000000)	(000000000)
(000010000)	(000000000)	(000000000)	(000000000)

Figure 6: Rotation of 'T' Shape with COMPLEX Strategy.

6.2 Corrections

The first idea is to add rotations. This took no more than a day's worth of programming work. See Figure 6 above for an example of one shape's rotation, the rotation of the 'T' shape for the COMPLEX strategy.

Something similar had to be done for the other six shapes and for all different strategies. Once the work was done, an unexpected and considerable speed-up in time occurred. Adding rotations sped up processing time significantly. The reason is because when trying each number from its SP on the board, there are more possibilities to try from the added three directions for open squares to drop the piece. However, the results were similar to the first attempt and not satisfactory. Something else needed to be applied to converge on a reasonable response.

The second idea was to add more search procedures. Another gene could be used or the alphabet of the current gene changed. The latter method was chosen. The problem was the gene only allowed for four different SPs to choose from, since the alphabet was binary. The idea was to convert from binary to trinary (00=0, 01=1, 02=2, 10=3, 11=4, 12=5, 20=6, 21=7, 22 = 8) to provide more search procedures to choose from without overwriting previous SP.

Applying trinary to the first gene on all the organisms yielded something similar to a modified definition of the DNA Strand. The trinary modifications can be seen on the DNA strands. Notice the remaining genes still use the binary alphabet.

Modified DNA Strands:	*Offspring DNA Strands*:
$organism_0$ = (0000000000)	$organism_0$ = (2100000000)
$organism_1$ = (0101010101)	$organism_1$ = (0001010101)
$organism_2$ = (0210101010)	$organism_2$ = (1011101010)
$organism_3$ = (1010000111)	$organism_3$ = (2210000111)
$organism_4$ = (1111001100)	$organism_4$ = (1110001100)
$organism_5$ = (1200110011)	$organism_5$ = (1000110011)
$organism_6$ = (2010101010)	$organism_6$ = (2001101010)
$organism_7$ = (2111111100)	$organism_7$ = (2000111100)
$organism_8$ = (1000000000)	$organism_8$ = (0200000000)
$organism_9$ = (2201010101)	$organism_9$ = (2211010101)

Increasing the first gene's alphabet to nine letters with trinary allows an extra possibility for each bit position (instead of only having two, this gives five extra spaces for five more search procedures) totaling nine search procedures to choose from, where there used to be only four. The best performing individual's SP after reproduction is shown in Figure 7.

SP_4

12	25	33	38	6	24	23	13	32	14
19	36	37	7	20	5	39	40	42	43
35	48	8	50	55	30	4	41	31	22
18	9	52	47	59	21	54	3	46	44
10	27	60	57	53	49	58	29	2	28
26	17	34	16	56	11	51	15	45	1

Figure 7: Best Performing Individual's SP.

The offspring of the individual resulted in the new DNA strands. Of these offspring, $organism_4$ performed the best, where it choose the first newly added fifth SP_4 and the COMPLEX strategy as its proximity number.

This combination of two changes resulted in a satisfactory solution. Only one shape from the environment did not make it onto the inner board area. This response is adequate for giving a minimal answer to the problem. The best individual produced was organism$_4$ = (1110001100) with ranking = 52,023. Output can be seen below in Figure 8.

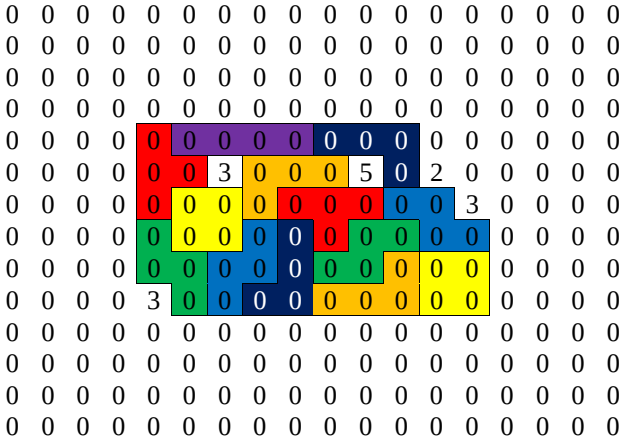


Figure 8: The Best individual solution produced.

7 Conclusions

Some areas of the DNA are well understood, and those should be the fixed areas in the universal adaptable GA. The fixed areas have already been figured out for the most part, and only need to be assembled into a main algorithm. These areas do not have to be reinvented every time a new problem is presented. The areas where there are always uncertainties are where there is always debate. These parts cannot be fixed at this time until they are more fully understood. Otherwise, the same basic GA will end up being reinvented over and over again by a new author for possibly the same problem yet all implementations would be done differently. These areas need to be abstracted away, left up to each unique realization. These ever-changing parts are the only things that need to change for each new problem encountered. The GA should be easily modifiable to satisfice many new problems without more than a day's worth of engineering work required for each new problem.

8 Future Work

The current implementation for SPs does not scale up to larger board sizes. For example, if the size of the inner board was increased to 50x50 squares, all SPs would have to be 2,500 cells. This would be infeasible, even with the number drag automation a spreadsheet provides. One idea is to change the DNA instructions; instead of SPs, use a command list, such as move left, right, up, down, choosing to place the tet as close to the left of the starting position as possible, which could be the closest right. This approach would make the individual more relative than absolute. Similar to how our hands are connected to our arms connected to our shoulders and neck relative to our head and so on.

The highest level goal in AGI is to build an AGI. This work has that goal in mind. The ultimate far reaching goal of this work is to disseminate the idea of learning how we can obtain intelligence from the brain or otherwise by means of GA. We already know the microbiology and the fundamental macro primitives of the brain; we only do not know how they all fit together. We understand why we have failed to learn how the brain works itself. The solution is to bring an outsider in, the computer, to crank away at the problem by giving it primitive functions, then using millions of years of evolution by speeding it up inside the computer to fit the known fundamental pieces together and generate new ideas via mutation.

If the A.I. went out to a rich environment such as the Internet, where there is constant change, read some text, and looked at pictures on a webpage, it should be able to be asked questions about what it perceived, and give an intelligent answer. Then we could declare with confidence we have figured out how to obtain intelligence. The idea is to use GA as control strategy to figure out how intelligence works, given sound primitives, such as NETL [8], hierarchy, lookup table, short term memory with 5+/-2 chunks, large matrix, production rules, an HTM for the five senses [9], even another GA itself (whatever you can dream up until we get it right) then letting GA work on it. All it takes is one person with the right idea.

Most of this research work was dedicated to solving a pseudo-Tetris problem with the GA as the learning mechanism to create the individual for playing the "game" well. In the background is this higher-level goal of creating a more general GA, to solve new problems never seen before.

9 References

[1] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier Systems and Genetic Algorithms"; Artificial Intelligence, 40, 1/3, 1989, 235—282.

[2] J. H. Holland, "Adaptation in Natural and Artificial Systems". The University of Michigan Press, 1975.

[3] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. "Induction: Processes of inference, learning, and discovery". MIT Press, 1986.

[4] J. C. Venter. "A Life Decoded: My Genome: My Life". Penguin Group, 1998.

[5] D. S. Burke, K. A. De Jong, J. J. Grefenstette, C. L. Ramsey, and A. S. Wu, "Putting More Genetics into GA", Evolutionary Computation, 6, 4, Oct 1998, 387—410.

[6] J. R. Koza, et al. "Genetic Programming IV: routine human-competitive machine Intelligence". Kluwer Academic Publishers, 2003.

[7] F. Gobet, et al. "Chunking Mechanisms in Human Learning"; Trends in Cognitive Science, 5, 6, 236—243.

[8] S. E. Fahlman. "NETL: A System for Representing and Using Real-World Knowledge". MIT Press, 1979.

[9] J. Hawkins, "Vision Framework Guide NuPIC 1.7.1" [Online]. Available: http://numenta.com
Oct 2011.

Stock price prediction using genetic algorithms and evolution strategies

Ganesh Bonde
Institute of Artificial Intelligence
University Of Georgia
Athens,GA-30601
Email: ganesh84@uga.edu

Rasheed Khaled
Institute of Artificial Intelligence
University Of Georgia
Athens,GA-30601
Email: khaled@uga.edu

Abstract:-

To many, the stock market is a very challenging and interesting field. In this paper we try to predict whether the prices of the stocks are going to increase or decrease on the next day. We are predicting the highest stock price for eight different companies individually. For each company six attributes are used which help us to find whether the prices are going to increase or decrease. The evolutionary techniques used for this experiment are genetic algorithms and evolution strategies. Using these algorithms we are trying to find the connection weight for each attribute, which helps in predicting the highest price of the stock. The input for each attribute is given to a sigmoid function after it is amplified based on its connection weight. The experimental results show that this new way of predicting the stock price is promising. In each case the algorithms were able to predict with an accuracy of at least 70.00%. Since this approach is new any further study in this field can definitely give better results.

Keywords:

Machine learning,stock market, genetic algorithm, Eovolutionary Strategies.

I. Introduction

The prediction of stock prices has always been a challenging task. It has been observed that the stock price of any company does not necessarily depend on the economic situation of the country. It is no more directly linked with the economic development of the country or particular area. Thus the stock prices prediction has become even more difficult than before.

These days stock prices are affected due to many reasons like company related news, political events, natural disasters … etc. The fast data processing of these events with the help of improved technology and communication systems has caused the stock prices to fluctuate very fast. Thus many banks, financial institutions, large scale investors and stock brokers have to buy and sell stocks within the shortest possible time. Thus a time span of even few hours between buying and selling is not unusual.

To invest money in the stock market we need to have an idea whether the prices of stocks are going to increase or decrease on the next day. Thus in this project we are trying to predict whether the highest price of a stock is going to increase or decrease on the next day. In this paper we are trying to predict the price of stocks of eight different companies. For each company we are predicting whether its highest price is increasing or decreasing next day. Thus it is a classification problem with only two classes involved. Thus we have tried to make the problem as simple as possible.

Kyoung-jae Kim and Won Boo Lee [13] developed a feature transformation method using genetic algorithms. This approach reduces the dimensionality of the feature space and removes irrelevant factors involved in stock price prediction. This approach performed better when compared with linear transformation and fuzzification transformation. This GA based transformation looks promising when compared with other feature transformations. Another research done on genetic algorithms (GAs) by Kyoung-jae Kim [4] again to predict stock market is to use a GA not only to improve the learning algorithm, but also to reduce the complexity of the feature space. Thus this approach reduces dimensionality of the feature space and enhances the generalizability of the classifier. Also Ajith Abraham [15] developed a hybrid intelligent system, which consists of a neural network, fuzzy inference systems, approximate reasoning and derivative free optimization techniques. That system also gives promising results but was not compared with any other existing intelligent systems.

Frank Cross [16] tries to find relationships that could exist between stock price changes on Mondays and Fridays in the stock market. It has been observed that prices on Friday have risen more often than any other day. It has also been observed that on Monday the prices have least often risen compared to other days. Boris Podobnik [17] tried to find cross-correlation between volume change and price change. For the stock prices to change, it takes volumes to move the stock price. They found two major empirical results. One is power law cross-correlation between logarithmic price change and logarithmic volume change and the other is that the logarithmic volume change follows the same cubic law as logarithmic price change.

Abdüsselam Altunkaynak [1] used a genetic algorithm for the prediction of sediment load and discharge. Not many have tried to use only genetic algorithms to predict

stock prices. Since the genetic algorithm can perform reasonably well in many cases there has to be a way to predict stock price using GA as well. Hyunchul Ahn [2] suggested that the genetic algorithm can be used to predict in financial bankruptcy. We have also tried to use a similar approach to predict the stock. The method used in this experiment is completely novel and looks very promising.

Many machine-learning techniques are used for predicting different target values [5,6,10]. This could be even to predict stock price. The genetic algorithm has been used for prediction and extraction important features [1,4]. Lot of analysis has been done on what are the factors that affect stock prices and financial market [2,3,8,9]. There are different ways by which stock prices can be predicted. One way is to reduce the complexity by extracting best features or by feature selection [7,11,12,13,14]. This approach will help us predict stock prices with better accuracy as the complexity reduces.

In this project the method used for predicting the highest price is novel. We try to find the connection weights of each attribute used for predicting the stock price. There are a total of six attributes used for each company. Hence we use six connection weights, one for each attribute. Each connection weight value defines the contribution given by each attribute in predicting the stock price. For example it could happen that the volume attribute contributes more than other attributes. Thus more importance is given to that attribute. Thus obviously this attribute will have a higher connection weight compared to other attributes. This concept is explained in more detail below.

Feature discretization of each input:-

The main concept in discretization is that we try to normalize each input attribute with respect to each other attribute. Thus we try to find the connection weight for each attribute that decides on the contribution given by that attribute. The summation of each attribute after multiplying by the connection weight is given to a sigmoid function. This function is used to classify the next stock price into increasing or decreasing class.

The sigmoid function in terms of mathematical expression is given below. It is used when we do not have detailed information of the input we are trying to predict. This function will classify each input into mainly two classes. So it can be used for binary classification problems.

$$P(t) = \frac{1}{1+e^{-t}} \qquad (1)$$

The two evolutionary techniques used for predicting the stock price are given below:-

Genetic Algorithm:-

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to search and optimization problems. Genetic algorithms are a particular class of evolutionary computation that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. A genetic algorithm finds the potential solution to a specific problem as a simple chromosome like data structure so as to preserve the critical information.

Its implementation begins with the selection of a population of chromosomes, which is a set of solutions to problems that could occur for a particular scenario. One evaluates its fitness and then does its reproduction to get better solutions with respect to the target problem. The chromosomes, which represent better solutions, are given more chance for reproduction than those which represent poorer solutions. This process continues for a number of generations after which we get the optimal solution.

The operators used for this experiment are two-point crossover and creep mutation. The crossover is a genetic operator used to vary chromosome gene structure where gene information is interchanged between selected parents by selecting two points in the gene structure of each parent.

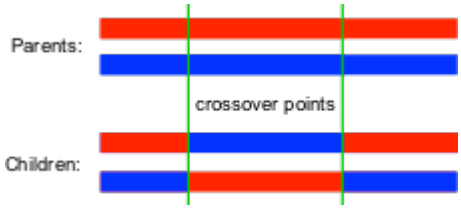


Figure 1. Two point crossover

The creep mutation used works by adding a small value to each gene with probability p. The selection method used to select the population is roulette wheel selection. In this method the fitness assigned to each individual is used for the selection process. This fitness is used to associate a probability selection with each individual. This can be given as below:-

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \qquad (2)$$

Where *fi* is the fitness of the ith individual and N is the population size.

Evolution Strategies:-

The evolution strategy (ES) is also an idea inspired by concepts of adaptation and evolution. This type of algorithm is mainly used for continuous parameter optimization. The representation of the gene is vector. The intermediate recombination technique is used in this algorithm. In this the selected parent values are averaged to give the child and one of the other parents is selected randomly so that two individual can go to the next generation.

The algorithm for evolutionary strategies is given below:

1. Randomly create an initial population of individuals.

2. From the current population generate offspring by applying a reproduction operator (described below).

3. Determine the fitness of each individual.

4. Select the fittest individuals for survival. Discard the other individuals.

5. Proceed to step 2 unless the number of generations have been exhausted.

In this experiment we are using a (μ , λ)-ES strategy in which the parents (candidate solutions) produce offspring (new solutions) by mutating one or more problem parameters. Offspring compete for survival; only the best (i.e., those with the highest fitness) will survive to reproduce in the next generation. If done properly, the population will evolve towards increasingly better regions of the search space by means of reproduction and survival of the fittest.

The mutation technique used is based on a Gaussian distribution requiring mainly two parameters the mean ξ and the standard deviation σ. In this small amounts of f(x) are randomly calculated using the Gaussian distribution N(ξ, σ). This is given as

$$f(x)= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (3)$$

The new value of x is calculated as the sum of previous gene value and some small random value calculated using the above equation.

$$Xnew = Xold + N(\xi, \sigma) \qquad (4)$$
where ξ=0 and σ=1.

II. Experimental Setup

Dataset used:

The dataset used for this experiment consists of data for the last five years. A total of six attributes for each company are used for prediction. These are opening price, closing price, highest price, lowest price, volume and adjusted closing price. The eight companies used for this experiment are Adobe, Apple, Google, IBM, Microsoft, Oracle, Sony and Symantec.

Two datasets are used for the experiment. One training dataset is used for finding the connection weights for each attribute used. We used another testing dataset so that we can verify the result. Thus we can check if over fitting is occurring or not. The results obtained actually showed that no over fitting occurred.

The representation for the problem is floating point so each connection weight used for a particular attribute is a floating point number. The fitness used in this problem is the number of times the connection weights result in predicting stock price correctly. So if it was able to predict the stock price correctly in 500 data points, then its fitness is 500. There are a total of 620 data entries for each dataset, which we need to predict. We first use the training dataset to find the exact connection weight for each attribute and then using these connection weights we try to predict the testing data. The different parameter settings for each algorithm are given below:

The parameter settings for the Genetic algorithm are:-

No.	Parameters	Values
1	Population Size:	100
2	Crossover Probability:	0.5
3	Mutation Probability:	0.013
4	Selection:	Roulette Wheel
5	Stopping Criteria:	1000 generations

Chart 1: Parameter settings for the genetic algorithm

The parameter settings for the Evolution strategy algorithm are given below:-

No	Parameters	Values
1	Population Size with (μ , λ)-ES strategy	20-100
2	Crossover Probability:	0.6
3	Mutation Probability:	0.015
4	Selection:	Roulette Wheel used only for initial population.
5	Stopping Criteria:	1000 generations

Chart 2: Parameter settings for the evolutionary strategies

III. Results

Tables 1 and 2 show the optimal connection weights used for predicting stock price in each algorithm. Table 3 shows the best fitness values evaluated for each company. Table 4 shows the accuracy of the algorithm to predict the highest price. The connection weights are calculated using the training dataset and is tested on the testing dataset. This protects against any over-fitting occurring in the model. From the results shown in Table 3 and 4 it can be seen that over-

fitting is not occurring. The fitness also indicates the number of times it actually predicted the stock price correctly. The total number of entries present in each set is 620.

It can be seen from Table 4 that we were able to predict the stock price with considerable accuracy. The search space for this problem is very large. This is because the connection weight can range from zero to even a million or more. Since we have restriction on space search we have kept the upper end to be 1000 only for floating representation.

From table 4 it can be seen that the connection weight evaluated for each attribute do not get over-fitted. In fact in some cases the accuracy for prediction is higher for testing data than training data. The highest accuracy obtained using the genetic algorithm is 73.87% and using the evolutionary strategies is 71.77%.

Company	Open price	Closing price	Highest price	Lowest price	Volume	Adjusted closing price
Adobe	995.0	10.0	27.0	83.0	929.0	38.0
Apple	98.0	12.0	85.0	18.0	30.0	17.0
Google	89.0	12.0	18.0	15.0	87.0	21.0
IBM	87.0	5.0	39.0	44.0	71.0	23.0
Microsoft	1212.0	135.0	223.0	138.0	218.0	148.0
Oracle	963.0	1.0	24.0	18.0	989.0	28.0
Sony	921.0	7.0	54.0	37.0	975.0	38.0
Symantec	976.0	8.0	23.0	18.0	55.0	2.0

Table 1: Connection weights for each company using the genetic algorithm.

Company	Open price	Closing price	Highest price	Lowest price	Volume	Adjusted closing price
Adobe	804.0	36.0	767.0	18.0	601.0	727.0
Apple	309.0	20.0	116.0	8.0	158.0	111.0
Google	890.0	15.0	27.0	46.0	43.0	830.0
IBM	247.0	23.0	35.0	8.0	907.0	72.0
Microsoft	285.0	5.0	70.0	42.0	24.0	183.0
Oracle	842.0	1.0	769.0	7.0	103.0	281.0
Sony	856.0	9.0	861.0	44.0	854.0	42.0
Symantec	778.0	13.0	161.0	302.0	938.0	23.0

Table 2: Connection weights for each company using the evolutionary strategy.

Company	Fitness Value Using GA		Fitness using Evolutionary Strategy	
	Training data	Testing data	Training data	Testing data
Adobe	447	454	450	434
Apple	457	439	460	445
Google	465	430	462	435
IBM	438	439	452	442
Microsoft	467	436	472	440
Oracle	445	452	434	444
Sony	412	431	421	441
Symantec	440	458	431	439

Table 3: The best fitness calculated for each company.

Company	Fitness Value Using GA		Fitness using Evolutionary Strategy	
	Training data	Testing data	Training data	Testing data
Adobe	72.09%	73.22%	72.58%	70.00%
Apple	73.70%	70.80%	74.19%	71.77%
Google	75.00%	69.35%	74.51%	70.16%
IBM	70.64%	70.80%	72.90%	71.29%
Microsoft	75.32%	70.32%	76.12%	70.96%
Oracle	71.77%	72.90%	70.00%	71.61%
Sony	66.45%	69.51%	67.90%	71.11%
Symantec	70.96%	73.87%	69.51%	70.80%

Table 4: The accuracy with which the stock price was predicted for each company.

IV. Conclusion and Future Work

The novel method of predicting stock prices using the genetic algorithm and evolutionary strategies looks promising. It was found that the genetic algorithm and evolution strategies have performed almost evenly. The best accuracy found using the genetic algorithm was 73.87% and using evolutionary strategies was 71.77%. The genetic algorithm was able to predict better than the evolutionary strategies in five cases. The evolutionary strategy reached an accuracy of 70% or better in all cases.

We used two different datasets for predicting the stock prices. The first one acts as training set and the other acts as testing set. This division is required so that we can test if over-fitting is occurring or not. The results show that over-fitting has not occurred.

There are many aspects we can consider in the future. We need to include more attributes to predict stock prices. The six attributes used are very similar to each other hence we need more attributes, which are not similar but affect the prices.

We can try different activation functions for classification. Thus instead of using the sigmoid function we can use some other function.

This method can be compared with other popular algorithms used for stock price prediction such as neural networks and support vector machines.

Future Work:-

The evolutionary algorithms used for this experiment looks very promising. Therefore, further research is required in this field. We can even try to use attributes of other companies to predict the prices to check whether they help in predicting the prices. Thus we can use only those company's data, which will help in predicting the data in a better way. There is a high chance that the accuracy for prediction will be above 80.0% if we used other companies' data also instead of using just individual company's data.

Since the results obtained are above 70.0% in every case then we can test the performance on real time data as well. This will give us an idea whether only historical data is good enough to predict data or not. If not, then we need to find the factors other than historical data which affect the prices. This information can also be fed to the algorithms we used for this experiment. There is a high chance that the accuracy will increase.

The companies used in this experiment were big companies. We can check the performance of those algorithms on small size companies as well.

REFERENCES:

[1] Abdüsselam Altunkaynak, Sediment load prediction by genetic algorithms Advances in Engineering Software, Volume 40, Issue 9, September 2009, Pages 928–934

[2] Hyunchul Ahn , Kyoung-jae Kim[b]. Bankruptcy prediction modeling with hybrid case-based reasoning and genetic algorithms approach, Applied Soft Computing,Volume 9, Issue 2, March 2009, Pages 599–607

[3] Po-Chang Ko, Ping-Chen Lin. An evolution-based approach with modularized evaluations to forecast financial distress, Knowledge-Based Systems,Volume 19, Issue 1, March 2006, Pages 84–91

[4] Kyoung-jae Kim, Ingoo Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert systems with Applications, 2000.

[5] Chung-I Chou, You-ling Chu and Sai-Ping Li . Evolutionary Strategy for Political Districting Problem

Using Genetic Algorithm, Lecture Notes in Computer Science, 2007, Volume 4490/2007, 1163-1166.

[6] Guangwen Li, Qiuling Jia, Jingping Shi , The Identification of Unmanned Helicopter Based on Improved Evolutionary Strategy, Intelligent Computation Technology and Automation, 2009. ICICTA '09. Second International Conference on, 205-208

[7] Chih-Fong Tsai , Yu-Chieh Hsiao . Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches, Decision Support Systems**,** Volume 50, Issue 1, December 2010, Pages 258–269.

[8] Xiaodong Li, Chao Wang, Jiawei Dong, Feng Wang, Xiaotie Deng, Shanfeng Zhu. Improving stock market prediction by integrating both market news and stock prices

[9] F. Mokhatab Rafiei, Manzari, S. Bostanian, Financial health prediction models using artificial neural networks, genetic algorithm and multivariate discriminant analysis: Iranian evidence, Expert Systems with Applications**,** Volume 38, Issue 8, August 2011, Pages 10210–10217

[10] George S. Atsalakis, Kimon P. Valavanis . Surveying stock market forecasting techniques – Part II: Soft computing methods, Expert Systems with Applications**,** Volume 36, Issue 3, Part 2, April 2009, Pages 5932–5941

[11] Kyoung-jae Kim. Financial time series forecasting using support vector machines, *Neurocomputing*, Volume 55, Issues 1-2 (September 2003), Pages 307-319.

[12] Ping-Feng Pai, Chih-sheng Lin. A hybrid ARIMA and support vector machines model in stock price forecasting, Omega **,**Volume 33, Issue 6, December 2005, Pages 497–505.

[13] Kyoung-jae Kim, Won Boo Lee. Stock market prediction using artificial neural networks with optimal feature transformation. Neural Computing and Applications (2004),
Volume: 13, Issue: 3, Publisher: Citeseer, Pages: 255-260

[14] Kyoung-jae Kim, Ingoo Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert Systems with Applications, Volume 19, Issue 2, August 2000, Pages 125–132.

[15] Ajith Abraham, Baikunth Nath and P. K. Mahanti. Hybrid intelligent systems for stock market analysis. Proceedings of the International Conference on Computational Science Part 2, Pages 337-345.

[16] Frank Cross. The behavior of stock prices on Fridays and Mondays. Financial Analyst Journal Vol. 29 No. 6, pages 67-69.

[17] Boris Podobnik, Davor Horvatic, Alexander M. Peterson and Eugene Stanley. Cross-correlations between volume change and price change. Proceedings of the National Academy of Sciences of the United States of America, Vol. 106, No. 52, pp. 22079-22084, December 2009

Fitness Proportionate Niching: Maintaining Diversity in a Rugged Fitness Landscape.

Abrham Workineh, Abdollah Homaifar
Autonomous Control and Information Technology Center
North Carolina A & T State University
Greensboro, NC 27411
atworkin@ncat.edu, homaifar@ncat.edu

Abstract: *Traditional Genetic Algorithms (GAs) fail to maintain useful diversity in the population as a result of a genetic drift due to selection pressure, selection noise and operator disruption. Genetic drift leads to early convergence making simple GAs suitable only for discovering the optimum of unimodal functions. However, most real world optimization problems often deal with multi-modal functions and hence require a technique to discover the location of multiple optima in the search space. The conventional fitness sharing scheme based on the niche count has a limitation when there is a high gap between the peaks of the multimodal function. This paper introduced a new niching technique based on fitness proportionate resource sharing to overcome this limitation. A comparison is made both using mathematical analysis and simulations on well known multi-modal test functions with unequal peaks. Unlike the conventional sharing scheme, the difference in the fitness of the highest and lowest peaks does not affect the performance of the proposed niching scheme.*

Key Words: Evolutionary algorithm, fitness sharing, genetic algorithms, multimodal optimization, niching methods.

1. Introduction

GAs are a class of computerized search procedures that are based on the mechanics of natural genetics [1]. Traditional GAs suffer from early convergence and in the case of multimodal functions, they evolve the whole population towards convergence to discover only one of the solutions. This makes them suitable only for locating the optimum of unimodal functions as they converge to a single solution of the search space. Most optimization problems, however, often lead to multimodal domains and so require the identification of multiple optima, either global or local. For this purpose, niching methods extend the application of simple GAs by promoting the formation of stable subpopulations in the neighborhood of optimal solutions [2]. The whole purpose of niching is to promote useful diversity in a population. Conventional GAs tend to lose diversity due to selection pressure, selection noise and operator disruption. In the case of multimodal functions with unequal peaks, the simple GA converges to the best peak. Also, for a function with multiple equal peaks, the population converges to only one of the peaks at random due to the selection noise and operator disruption (mutation and crossover operations).

Niching in GA is analogous to the mechanics of natural ecosystems where animals compete to survive by performing a differentiated role (hunting, grazing, feeding etc) at distinct ecological niches [3]. A niche can also be viewed as an organism's task, which permits species to survive in their environment. For each niche, the physical resources are finite and must be shared among the population of that niche. The subdivision of environment on the basis of an organism's role reduces interspecies competition for environmental resources, and this reduction in competition help stable sub-population to form around different niches in the environment [4]. A niche is commonly referred to as an optimum of the domain, the fitness representing the resources of that niche [3].

By the same analogy, niching methods tend to achieve a natural emergence of niches and species in the search space. In multi-modal GA, a niche represents the location of each optimum in the search space [5]. Niching enables the standard GA to encourage the formation of subpopulations representing locally optimized solutions. It provides a restoring force for the GA to counterbalance the impact of genetic drift due to the selection pressure. Maintaining useful diversity avoids early convergence and hence allows sufficient exploration of the search space and enables the GA to locate multiple optima at the same time. An algorithm that locates multiple optima will definitely delay convergence whereas one that intends to delay convergence may not necessarily locate multiple solutions. For instance, increasing the mutation rate protects loss of genetic material by maintaining diversity in the population. The role of mutation is to allow sufficient exploration of the search space but this diversity may not necessarily lead to the formation of stable subpopulations at multiple optimum points. Effective application of niching extends the success and power of GAs for multimodal function and multi-objective optimization, machine learning and classification problems [6]. In our previous work [7], we applied this niching technique for evolving hierarchical cooperation in learning classifier systems. The focus of this paper is to emphasize its significance for multimodal

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454.

function optimization with unequal peaks of high fitness variation and comparing its performance with the existing fitness sharing scheme.

The remaining section of the paper is organized as follows. Section 2 investigates previous work on various types of niching techniques. Section 3 presets the modification to the existing fitness sharing scheme. Mathematical analysis of the fitness proportionate niching scheme is also presented in this section. Section 4 discusses the simulation results obtained by comparing the proposed niching technique with the existing approaches. Finally, the last section concludes the paper highlighting the achievements obtained.

2. Review on Niching

The need for maintaining useful diversity in a population to reduce the effect of genetic drift in the standard GA has been emphasized by several researchers in previous work [4-8]. To date, various niching strategies have been proposed in literatures and this section presents a brief survey of the art on the three most notable niching techniques: crowding, fitness sharing and clearing.

2.1. Fitness Sharing

Fitness sharing is the most well-known method for creating stable subpopulations of individuals around the multiple local or global optimum points in the search space [8, 9]. The inspiration for adapting the sharing technique to the traditional GAs emanates from the natural ecosystem where individuals of the same species share a finite natural resource in an environment. The hierarchical organization of species in a competing world of limited resources is shaped by the location and distribution of these resources. Traditional GAs assume an infinite resource model where there is no need for competition of resources and all individuals can comfortably coexist on the same peak and receive the same fitness that they would have if they were the only individual on that peak. Hence, in the case of multimodal functions of unequal peaks, all individuals tend to seek the highest peak and converge to that point. Also, in multimodal functions of equal peaks, the population will converge to one of the peak locations arbitrarily. The feasibility of resource sharing in evolutionary algorithms was first pointed out by Holland [10]. But the first implementation of fitness sharing to model a resource contention with in a simple GA was given by Goldberg and Richardson [8]. It is based on the idea that a point in a search space has limited resources which must be shared by all individuals that occupy similar search space [11]. As more and more individuals get attracted to the highest peak, the resource at that peak gets depleted and other lower peaks in the search space begin to attract individuals. Sharing in an Evolutionary Algorithm (EA) is implemented by scaling the fitness of an individual based on the number of "similar" individuals present in the population [12]. It lowers each individual's fitness by an amount nearly equal to the number of similar individuals in the population. The raw fitness of the individual is reduced by the number of similar solutions in the population belonging to the same niche [8, 9].

Derating an individual's fitness is controlled by two operations, a similarity function, which measures the distance between two individuals in either the genotypic or phenotypic space, and a sharing function [11]. The sharing function is shown in equation (1).

$$sh(d_{i,j}) = \begin{cases} 1-(\frac{d_{i,j}}{\sigma_{sh}})^{\alpha} & ,if \quad d_{i,j} < \sigma_{sh} \\ 0, & otherwise \end{cases} \tag{1}$$

And the niche count is calculated by summing a sharing function over all individuals of the population as:

$$n_i = \sum_{j=1}^{m} sh(d_{i,j}) \tag{2}$$

Now, the shared fitness of an individual i is given by the following equation

$$F_{sh,i} = \frac{F_i}{n_i} \tag{3}$$

Where F_i is the raw fitness of the individual, $d_{i,j}$ is the distance between individual i and j, n_i is the niche count, m is the number of individuals occupying the same niche and σ_{sh} is the niche radius. The constant α is usually set to 1 for a triangular sharing function.

As can be seen from the equation above, the degree to which two individuals are considered to belong to the same niche is controlled by the sharing radius. And, the performance of the fitness sharing highly relies on the proper choice of the niching radius. This is one of the limitations of the fitness sharing technique. In general choosing the optimum niche radius requires a priori knowledge of the distribution of the peaks in the objective function [13].

2.2 Crowding

The standard crowding method was first introduced by De Jong to promote useful diversity in the population to prevent premature convergence of the GA [14]. In this method, a fraction of the total population called the generation gap is allowed to reproduce at each generation [14]. The crowding factor (CF) determines the number of individuals selected from the population for comparing the similarity of the new offspring. Similarity of individuals can be determined by means of a distance measure, either

genotypic or phenotypic distance between individuals. The new offspring then replaces the most similar individual taken from this randomly drawn subpopulation of size CF. Later, Mahfoud introduced a modified crowding technique termed "Deterministic Crowding" [5, 15] to improve the standard crowding by introducing competition between children and parents of identical niches [11]. In a deterministic crowding, the new offsprings replace the nearest (phenotypic distance) parent provided it has a higher fitness [11].

2.3 Clearing

The clearing type niching is essentially similar in principle to the explicit fitness sharing technique. But, instead of uniformly distributing the resource to all subpopulation in a given niche, it allocates the whole resource only to the best members of the subpopulation. It is based on a winner takes all strategy where it preserves the fitness of the best individuals of each niche and resets the fitness of the others with in the niche radius [16]. This convergence to only one of the alternatives is undesirable in multimodal optimization of real problems, because we are interested on getting information about good points and better solutions.

3. Fitness Proportionate Niching (FPN)

This section presents the proposed niching technique. Like the traditional fitness sharing based on niche counts, this technique is also based on the notion of limited resources where individuals in a given niche share the resource of that niche. But, resource sharing here is in proportion to strength. According to this sharing scheme, the sharing function is given in equation (4) and the derated fitness of individual i is given by equation (5).

$$sh(d_{i,j}) = \begin{cases} F_j & ,if \quad d_{i,j} < \sigma_{sh} \\ 0, & otherwise \end{cases} \tag{4}$$

$$F_{sh,i} = \frac{F_i}{\sum_{j=1}^{M} sh(d_{i,j})} \tag{5}$$

Where M is the number of individuals in a given niche, $d_{i,j}$ is the phenotypic distance between individuals *i* and *j*.

3.1 Test Functions

For testing the performance of the algorithm, three well known multimodal functions of different difficulty are considered for simulation [3-6]. All the three functions are defined on [0, 1] interval and maxima (labeled P1 to P5 in the figures) are located at approximate values of 0.1, 0.3, 0.5, 0.7 and 0.9. The first function has equal peaks where the other two have unequal peaks (see equations 14-16).

$$F_1(x) = Sin^6(5\pi x) \tag{14}$$

$$F_2(x) = \exp(-2\log 2(\frac{x-0.1}{0.8})^2)\sin^6(5\pi x)) \tag{15}$$

$$F_3(x) = \begin{cases} 10\exp(-2\log 2(\frac{x-0.1}{0.8})^2)\sin^6(5\pi x), & \text{if } 0 \le x \le 0.2 \\ \exp(-2\log 2(\frac{x-0.1}{0.8})^2)\sin^6(5\pi x), & 0.2 < x \le 1 \end{cases} \tag{16}$$

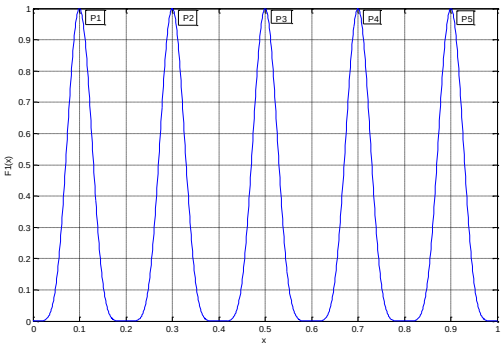


Figure 1: F1(x)-a multi-modal function with 5 equidistant peaks of equal height.

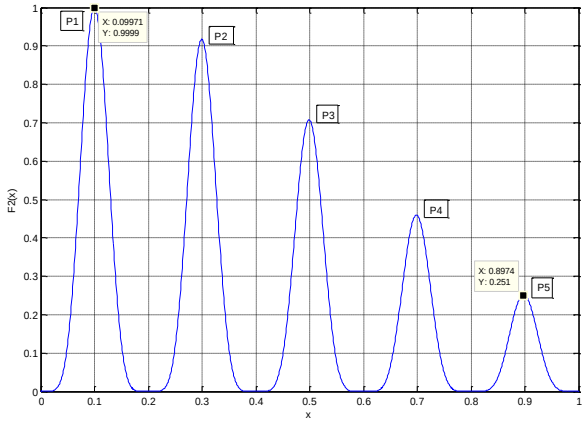


Figure 2: $F_2(x)$-A multi-modal function with 5 unequal peaks and a small variation between the peak values.

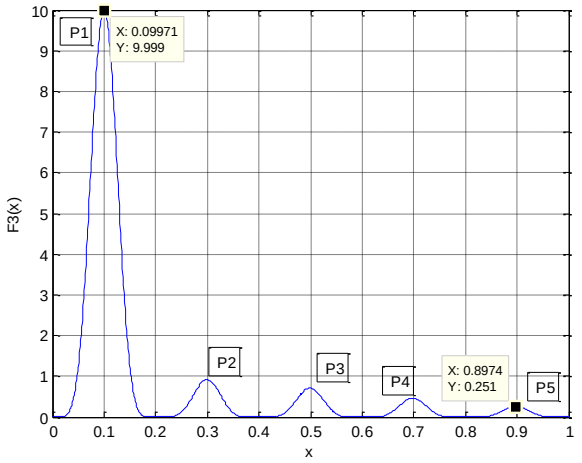


Figure 3: $F_3(x)$-A multi-modal function with 5 unequal peaks and a large fitness variation at the peaks.

3.2 Mathematical Analysis

The feasibility of the proposed niching scheme can be verified both mathematically and using simulation. It can be demonstrated analytically that FPN is indeed insensitive to the difference in fitness of the peaks. We want to show that unlike the traditional fitness sharing, the FPN will tend to form a stable subpopulation around all the niches with no additional restriction on the size of the population for multi-modal functions of unequal peaks.

To get an insight on its performance as compared to the existing sharing technique, consider the three multi-modal test functions given above with corresponding fitness values of P_1 to P_5 at the 5 niche locations. Let the subpopulation size at each of the niches is denoted by n_1 to n_5 respectively.

Using the traditional fitness sharing scheme, the shared fitness of an individual at the k^{th} niche is given by equation (6).

$$P'_k = \frac{P_k}{n_k} \tag{6}$$

Assuming that after sufficient iteration almost all the population distributes around the five peaks, we get equation (7).

$$n_1 + n_2 + n_3 + n_4 + n_5 = N \tag{7}$$

To discover all the peaks, it is required that the shared fitness values at each niche should be approximately equal (i.e. $P'_1=P'_2=P'_3=P'_4=P'_5$).

Substituting and rearranging terms, the number of individuals at the k^{th} niche is governed by equation (8).

$$n_k = \frac{P_k}{\sum_{i=1}^{5} P_i} * N \tag{8}$$

If a niche size of at least two individuals is required at the lowest peak (i.e. n_5>=2), the minimum population size required to discover all the peaks using the traditional sharing technique is given by equation (9).

$$N \geq 2 * \left(\frac{\sum_{i=1}^{5} P_k}{P_5}\right) \tag{9}$$

This indicates that the traditional sharing scheme based on the niche count has a threshold requirement on the minimum population size to discover all the peaks when the objective function has unequal peaks. As the gap of the peak values increases, the required minimum population size also increases drastically.

But using FPN, the shared fitness of an individual in the k^{th} niche is given by equation (10).

$$P'_k = \frac{P_k}{n_k \sum_{i=1} P_i} \tag{10}$$

Where n_k is the subpopulation size at the k^{th} niche (location of a peak). After sufficient generations, individuals in each niche will have approximately equal fitness (i.e. $f_i=f_j$, for two individuals i and j in the same niche). Hence a simplified form of equation (10) is shown in equation (11).

$$P'_k = \frac{1}{n_k} \tag{11}$$

From equation (11), for the shared fitness values to be equal, the population has to be evenly distributed among all the peaks, irrespective of the difference in the fitness value at the peaks (i.e. $n_1=n_2=n_3=n_4=n_5=N/5$). In general, for a multimodal function having M optimum points, the expected number of individuals at the k^{th} peak using the traditional fitness sharing scheme is given by equation (12). And FPN distributes the population around the optimum points uniformly as shown in equation (13).

$$n_k = \frac{F_k}{\sum_{i=1}^{M} F_i} * N \tag{12}$$

$$n_i = \frac{N}{M} \tag{13}$$

Where F_k is the fitness value representing the k^{th} niche, M is the number of peaks of the multimodal function and N is the population size.

It can be observed that, for a multimodal function with equal peaks (e.g. $F_1(x)$), equation (12) would degenerate to equation (13). Hence, for multimodal functions with equal peaks, FPN is essentially the same as the traditional sharing scheme. The underlying principle in FPN looks contrary to the concept of ecological niches where most of the population settles at a niche of more resources. But from the perspective of the GA, what is important is whether the niching scheme is able to form a stable sub population around all the multiple optimum points. In other words, FPN considers all the peaks as equally important for the GA and hence the proportion of the population at the different peaks does not really matter. Once all the peaks are discovered, a preference between the different peaks can be made by arranging the final population based on the fitness value.

$F_2(x)$ and $F_3(x)$ have 5 unequal peaks as shown in Figures 2 and 3 respectively. $F_3(x)$ has a large fitness gap between its highest and lowest peaks. Plugging in the values, the expected subpopulation size at P_5 for $F_2(x)$ is given by:

$$\begin{aligned} n_5 &= \frac{P_5}{P_1+P_2+P_3+P_4+P_5} * N \\ &\cong \frac{0.251}{1+0.91+0.7+0.45+0.251} * N \\ &= 0.0758 * \mathrm{N} \end{aligned} \tag{17}$$

This implies that for the niching technique to locate all the peaks with at least two individuals at the lowest peak, a population size of at least 27 is required. In practice, the desired population size has to be much larger than this ideal mathematical threshold. The optimum population size to discover all the peaks is largely dependent on the ratio of the fitness at the peaks. The higher the fitness ratio between the peaks, the larger is the size of the population required to discover all the peaks. This is more evident from $F_3(x)$ (see Figure 3). $F_3(x)$ has a much higher fitness gap between its highest and lowest peaks as compared to $F_2(x)$.

Using the same expression given above and plugging in the numerical values from the plot, the expected number of individuals at P_5 will be:

$$\begin{aligned} n_5 &= \frac{P_5}{P_1+P_2+P_3+P_4+P_5} * N \\ &\cong \frac{0.251}{10+0.91+0.7+0.45+0.251} * N \\ &= 0.02 * \mathrm{N} \end{aligned} \tag{18}$$

Quantitatively speaking, a population size of at least 100 is required to have at least two individuals at the lowest niche (P_5). FPN overcomes this requirement on the minimum size of the population by uniformly distributing the total population among the various peaks, irrespective of the difference in the fitness value (see equation (13)). From equation (13), it only requires a population size of at least twice the number of peaks to have at least two individuals at each of the niches (i.e. one tenth of the population size required by the traditional sharing scheme).

3.3 Performance Criteria

To verify the performance of FPN and compare it with the existing approach, two criteria are used. The first criterion is the percentage of number of peaks discovered by the niching algorithm as a function of search cycle (generation). This in effect is equivalent to comparing the ratio of the sum of the fitness of the local optima identified by the niching technique divided by the sum of the fitness of the actual optima in the search space. The other criterion is the distribution of the population around the optimum points. This shows whether the niching technique is able to evolve a stable and diverse subpopulation.

4. Simulation Results

The proposed fitness proportionate niching scheme is applied for the optimization of multimodal functions both with equal and unequal peaks and its performance is compared with the traditional fitness sharing scheme. Simulation is carried out with various population sizes to investigate how the niching techniques behave as the population size varies. As the goal of a niching technique is to discover multiple peaks in parallel, one possible way of measuring system performance is displaying the number of peaks discovered as the search process goes on. Figure 4 shows the performance for equal peaks. As can be seen from this figure, there is no significance difference in performance between the two algorithms for this function. For instance, for a population size of 50, both algorithms discovered almost all the 5 peaks. This result is normal and expected as the two approaches are essentially the same for multimodal functions having equal peaks. However, for multimodal functions having unequal peaks, there is a significant difference in performance (see Figures 5 and 6). Two categories of simulation are carried out here. The first is for a function with a small fitness ratio among the different peaks ($F_2(x)$ function). For this

scenario, FPN has a reasonably fair performance even at a small population size. For a population size of 30, the traditional niching scheme discovered nearly 75% of the peaks whereas the fitness proportionate niching discovered about 95% of the peaks. As the population size increases, there is an improvement in performance of both. For a population size of 50, both algorithms discovered almost all the peaks. In the simulation results (both tables and figures), TN refers to the traditional fitness sharing scheme where as FPN stands for the fitness proportionate niching.

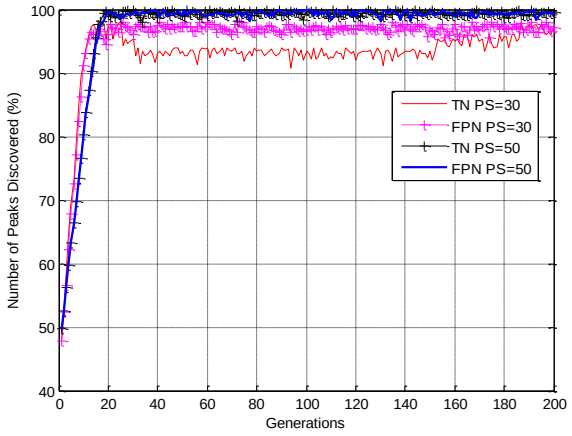


Figure 4: total number of peaks discovered out of a total of 5 peaks (in %).

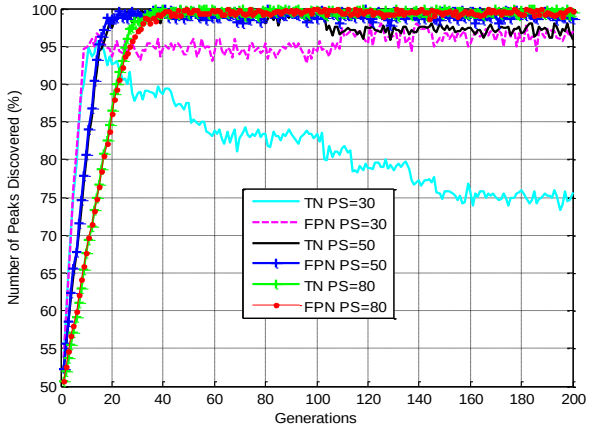


Figure 5: Percentage of the number of peaks discovered for $F_2(x)$, Averaged over 10 runs.

Figure 6 displays the simulation result for $F_3(x)$ for various population sizes. As can be seen from the first subplot in Figure 6, the traditional fitness sharing technique discovers only the location of the highest peak (only 1 peak out of a total of 5 peaks). But the fitness proportionate niching discovered almost all of the peaks. The results for $F_3(x)$ show how the traditional niching technique is sensitive to the difference in fitness at the peaks. To discover all the peak locations, it requires a very large population size which depends on the fitness ratio at the highest and lowest peaks. In this particular simulation, the traditional niching technique requires a population size of 150 to discover all the peaks as compared to 50 or lower population size for the fitness proportionate niching scheme.

Tables 1 and 2 show the distribution of the population among the various peaks for $F_2(x)$ and $F_3(x)$ functions. The values given are the average number of individuals at each of the five peaks over 10 runs. As can be seen from both tables, FPN tends to distribute the population among the various peaks uniformly irrespective of the fitness difference at the peaks.

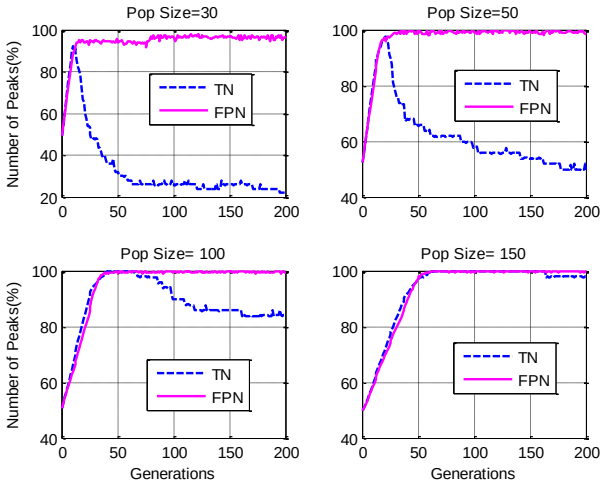


Figure 6: Percentage of number of peaks discovered for $F_3(x)$, averaged over 10 runs.

Table 1: Population distribution at the five different peaks, averaged over 10 runs for $F_2(x)$.

PS		Peak1	Peak2	Peak3	Peak4	Peak5
30	TN	10	9.7	7.1	3	0
	FPN	7.3	6.8	6.7	5.3	3.6
50	TN	15.9	14.4	10.9	5.7	2
	FPN	11	10.9	10.4	9.7	7.3
80	TN	24.6	23.1	17	10.3	4.6
	FPN	18	16	16	15	15

Table 2: Population distribution at the five different peaks, averaged over 10 runs for $F_3(x)$.

PS		Peak1	Peak2	Peak3	Peak4	Peak5
30	TN	29.8	0.2	0	0	0
	FPN	7.9	6.9	6.3	5.5	3
50	TN	47.1	2	0.7	0.1	0
	FPN	12.1	10.7	9.7	9.3	7.5
100	TN	88	5.7	4.5	1.4	0.2
	FPN	21.6	20.6	20	19.9	17.1
150	TN	128.4	9.8	7	3.5	1.3
	FPN	31.9	31.5	30.5	28.9	26.9

5. Conclusion

This work has shown the feasibility of fitness proportionate niching scheme for multimodal function optimization. When the objective function has several unequal peaks with a large peak ratio, the traditional niching techniques tend to discover only the location of the highest peak or require a very large population size in order to discover all the peaks. This demand of large population size added with the distance comparison between individuals makes the traditional sharing techniques computationally cumbersome.

The performance of the fitness proportionate sharing method was compared with the existing niching method for multimodal functions of unequal peaks. Both simulation results and mathematical analyses showed that the performance of the proposed niching technique is insensitive to the fitness difference at the various peaks. When individuals share the resource of a given niche in proportion to the strength, the population distributes around all the peaks uniformly irrespective of the fitness variation at the niches.

6. References

1. D. Goldberg, "Genetic algorithms in search, optimization, and machine learning," Reading, MA: Addison-Wesley, 1989.
2. D. Beasley, eta'l, "A sequential niche technique for multimodal function optimization," *Evol. Comput.*, vol. 1, no. 2, pp.101–125, 1993.
3. Cheol-Gyun Lee;eta'l, "Niching genetic algorithm with restricted competition selection for multimodal function optimization", IEEE Transactions on Magnetics, Vol.35, Issue 3, pps 1722-1725, May 1999.
4. Sareni, B. Krahenbuhl, L." Fitness Sharing and Niching Methods Revisited", IEEE Transactions on Evolutionary Computations, Vol. 2, Issue 3, pps.97-106, 1998.
5. S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D.dissertation, Univ. of Illinois, Urbana-Champaign, 1995.
6. K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf.Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42–50.
7. Abrham Workineh and Abdollah Homaifar, "Evolving Hierarchical Cooperation in Classifiers via Fitness Proportionate Niching", IEEE Transactions on Evolutionary Computations (submitted).
8. D. E. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception and genetic algorithms," in *Parallel Problem Solving from Nature (PPSN-2)*, R. Manner and B. Manderick, Eds. North Holland: Amsterdam, pp. 37–46, 1992.
9. D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genetic Algorithms*, J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 41–49.
10. J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.
11. Y. Davidor, "A naturally occurring niche & species phenomenon: The model and first results," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo, CA: Morgan Kaufmann, 1991,pp. 257–263.
12. Antonio Della Cioppa, etal, "Where are the Niches? Dynamic Fitness Sharing", IEEE Transactions on Evolutionary Computation, Vol. 11, No. 4, August 2007.
13. Grant Dick and Peter A. Whigha, "Spatially-Structured Evolutionary Algorithms and Sharing: Do They Mix?", Proceedings of Proceedings of the 5th International Conference on Simulated Evolution and Learning (SEAL), pp. 457-464, 2006.
14. K. A. DeJong, "An analysis of the behavior of a class of genetic adaptative systems," Ph.D. dissertation, Univ. of Michigan, Ann Arbor,1975.
15. Mahfoud, S.W, "Crossover Interaction Among Niches," Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pps 188-193.
16. A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in Proc. 1996 IEEE Int. Conf. Evolutionary Computation, Nagoya, Japan, 1996, pp. 798–803.

ACKNOWLEDGMENT

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Using Simple Ancestry to Deter Inbreeding for Persistent Genetic Algorithm Search

Aditya Wibowo and Peter Jamieson
Dept. of Electrical and Computer Engineering
Miami University

Abstract—*In this work, we explore how a mechanism for recording ancestry helps avoid inbreeding and, ultimately, convergence for persistent optimization problems. We focus our experimentation on the traveling salesman problem and introduce a tabu search "like" mechanism in a CHC algorithm and preselection genetic algorithm. We then compare how this mechanism improves the diversity within the solution population. We compare this mechanism to a basic genetic algorithm and show how the quality of results is improved and convergence is delayed. Our results indicate that the CHC algorithm with the inbreeding avoidance mechanism is the current best implementation for persistent optimization problems in maintaining diversity of solutions and to find the best solutions. Preselection shows improvement with our mechanism, but does not seem to have sufficient exploitation to find quality results. Our overall goal is to find the best way to maintain diversity while finding good solutions for single-threaded genetic algorithms.*

1. Introduction

There is a small subset of optimization problems that we call persistent optimization problems (POPs), and these problems are characterized by problems that can have their solution space continuously searched for better solutions. One of the most recent of these types of problems is persistent computer aided design (CAD) for Field-Programmable Gate Arrays (FPGAs). FPGAs are programmable chips that can be updated in the field with new and better designs. The placement stage of FPGA CAD, which tries to pack hardware structures that are connected to one another, can be algorithmically solved using genetic algorithms (GAs), and researchers have explored persistently searching for better placement solutions (improving power consumption) using a GA [1], [2]. Other examples of where persistent optimization algorithms may be useful include energy control and distribution, financials, and data mining. In each of these domains, the solution space is dynamically changing over time, and better optimizations for the problems may result in saved money and higher efficiency. The key question for POPs is whether the additional run-time costs justifies the benefit of potential improved solutions and the incremental cost savings.

POPs fit well into GA frameworks since GAs can be manipulated in terms of exploration versus exploitation phases to continually cross a solution space. Still, convergence [3] [4] defined as the lack of diversity in a population such that new offspring are not sufficiently diverse, therefore, resulting in suboptimal solutions, is a major concern for POPs in addition to all GAs. For problems such as the FPGA placement problem [5] and the Traveling Salesman Problem (TSP), where a genome is expressed as unique string of individual genomes, traditional algorithms such as CHC [6], which are built to preserve diversity, are not directly applicable to these problems since the hamming distance measure of familiarity does not apply.

In this work, we implement versions of CHC and preselection GAs to solve the TSP, and we include an inbreeding avoidance technique inspired by Tabu Search [7]. Our goal is to develop a single threaded GA that avoids convergence for the longest period possible while still generating good results. With the improvements of these algorithms, we plan to further investigate divergence techniques such as the island model [8] to build a larger system for POPs using GAs.

Our results show that our inbreeding avoidance mechanism does achieve higher diversity for both preselection GAs and the CHC algorithm based on a greater number of generations before the the problem stabilizes. Preselection algorithms with inbreeding avoidance last the most number of generations, but it seems that this crowding technique loses some of the advantages of competition/exploitation that the CHC algorithm achieves.

The remainder of this paper is organized as follows. Section 2 describes various techniques to maintain diverse populations for GAs and the relevance of crossbreeding operator to our problems. Section 2.2 describes our inbreeding avoidance technique and the two algorithms that they are implemented within. Section 4 describes our experimental setup and Section 5 shows our results. Finally, Section 6 concludes this work.

2. Background

In this section, we examine various approaches to avoiding premature convergence and the crossover operator for ordered chromosomes.

2.1 Approaches to Avoiding Premature Convergence

Premature convergence is a well known problem with GAs [3] [4]. This problem has been addressed using a number of methods including basic approaches such as:

1) Increasing population size

2) Island models/ Niching [8]
3) Crowding [9]
4) Preselection [10]
5) Inbreeding prevention [6], [11]

The first approach, increasing population size, impacts memory usage and algorithmic run-time, but this approach can be used in combination with all the other approaches to the premature convergence problem. The second approach, Island models, divides the search into a number of parallel solutions where each is run independently of one another. In this way, there is no sharing of genetic material of individuals from one island population to another. This approach can also be applied to any other approach for improving diversity. Therefore, the first two approaches can be used to improve our results and will be considered, in a larger system, once we wish to build a larger system to solve POPs, but our focus for this work is single-threaded approaches at fixed population size that maintain genetic diversity while finding good solutions.

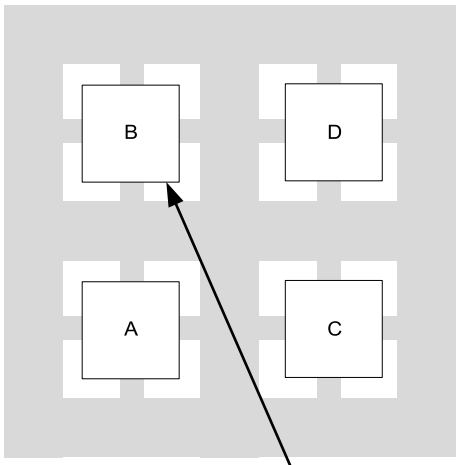
The last three schemes in the list are some examples of algorithmic approaches to the premature convergence problem. Crowding works by having offspring compete against individuals in the population that are most similar and thus maintaining niche lineages within the populations. Similar individuals are found by making a comparison of their genomic strings, and this comparison comes at a computation cost. Preselection is a similar approach to crowding, but to avoid the computational cost an assumption is made; the assumption is that a parent will be a similar individual, and therefore parent and child will compete against each other. Finally, the CHC algorithm takes a number of steps towards avoiding premature convergence, and of main interest to this work, parents of similar genome structure are not bred together (inbreeding prevention). These three approaches have been extensively studied for problems that have a binary encoded genome, and this work looks at two of these approaches for genomes that have unique chromosome encoding (described in the next section).

As mentioned before, the CHC algorithm is a non-traditional GA that was created to avoid premature convergence. In this paper, we implement the CHC algorithm [6] as one of the comparison points for our inbreeding avoidance technique. The CHC algorithm has four main components:

1) Parents and children combined together in competition for next population
2) Inbreeding avoidance by comparing binary encoded genomes hamming distance
3) Highly disruptive crossover operator
4) Full restart (from the best individual) once no new offspring are created in a generation

Since our work is focused on genomes that are not a simple binary encoded string, we take a modified approach to this algorithm. In particular, we use a crossover method described in the next sub section and our inbreeding avoidance technique is described in the section following the background.

2.2 Crossover Operators for Permutation Based Genomes



Design Logic	A	B	C	D
Location	(0,0)	(1,0)	(0,1)	(1,1)

Fig. 1
SAMPLE OF A SMALL PLACEMENT GENOME

In the case of both the TSP and the FPGA placement problem (among other problems) the genome consists of a permutation in which each chromosome is unique. These types of chromosomes are called ordered chromosomes. In the TSP problem, this genomic string represents the order of a tour; for example, for a four city problem we might see the string A, C, B, D which means this solution will go from city A to C, C to B, and B to D in that order. For the placement problem this permutation string indicates which pieces of a circuit are located in a 2D plane. For the previous example, piece A would be placed at x=0, y=0, piece C is at x=0, y=1, piece B is at x=1, y=0, and piece D is at x=1, y=1. Figure 1 shows this example of the 2D placement and the respective genome. This genome structure was originally proposed by Venkatraman *et. al.* [12].

For these types of strings, crossover operators that simply copy the genome of parent 1 and take parts of the genome from parent 2 and map them into the child cannot be used. Instead, careful consideration must be used to perform the crossover. A number of crossover operators of this nature have been proposed and studied ([4], [13], [14], [15], [16], [17], [18], [19]). Cicirello *et. al.* [18] provide a useful classification of these crossover methods by first classifying them as problem dependent or general crossover operators. Cicirello *et. al.* further classify crossovers into three categories (a) position-based crossover (e.g. [16]), (b) order-based crossover (e.g. [19]), and (c) hybrid crossover operators (e.g. [15]).

Recently, the success of a problem dependent crossover operator proposed b Whitley *et. al.* for the TSP suggests that careful thought should be given to a problem with ordered chromosomes. The same is, likely, true for the FPGA placement problem among other problems, and this is an are for future work if we pursue FPGA placement POPs.

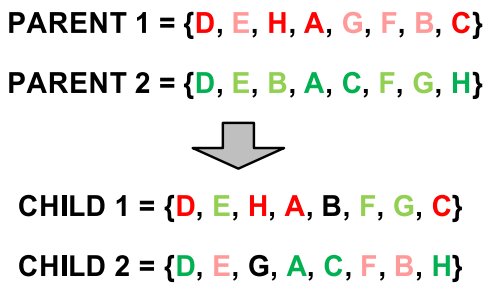


Fig. 2
SAMPLE PMX MUTATION

For this work, we are attempting to find a general single-threaded framework for solving POPs that do not use any of the more modern domain specific solutions that would target the TSP. Instead, we use the partially mapped crossover (PMX) [15], which randomly selects a set of parent genes to be copied to the new child from parent 1. Then, the remaining genes are transferred from parent 2 unless this gene has already been assigned by parent 1. If it has, a reverse mapping using the the information in parent 1 is used to find an appropriate gene to be copied to the empty spot. Figure 2 shows a simple example of the PMX operator where a set of chromosomes have been selected to be copied from parent 1 to child 1 and parent 2 to child 2. The remaining genes are copied over from the opposite parent using a remapping process when needed. Note how, in the figure, child 2 has some chromosomes from parent 1 (illustrated in red), some from parent 2 (illustrated in green), and one remapped chromosome (illustrated in black) . Note that the highly disruptive aspect of the CHC algorithm is not specifically explored in this work, and we simply assume that our crossover operator is sufficient, and we leave this issue as future work, if necessary.

3. Inbreeding Prevention for Permutation Based Genomes

In Eshelman's [6] original work on the CHC algorithm, he introduced inbreeding prevention for binary encoded chromosomes, and in this work we look at a technique to avoid inbreeding for ordered chromosomes and apply this technique both to preselection GAs and our implementation of the CHC algorithm. Our technique is inspired by tabu search [7] where a simple history is kept for previous solutions. In the same way, we can keep ancestry records for each individual by keeping am ancestor tree to a certain depth of generations.

Figure 3 shows how two parents ancestral trees that contain three past generations are combined together in a respective

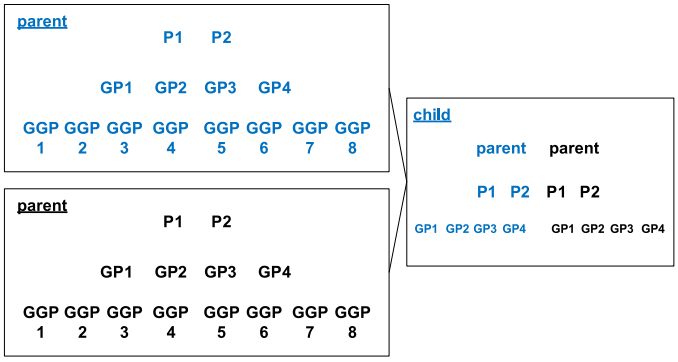


Fig. 3
SHOWS HOW TWO PARENTS ANCESTRY IS RECOMBINED FOR A CHILD

child. Based on our crossbreeding operator, we can assume that both parents contribute roughly 50% of their genetic information to the child. Similarly, grandparents will contribute 25% of their genetic material to the child, and so on for older generations where 4th generation contributes 12.5%, 5th generation contributes 6.25%, and 6th generation contributes 3.125%. Therefore, from a perspective of our inbreeding avoidance it doesn't make much more sense to record deeper than 6 generations, where 6 generations costs us only 128 data locations in memory. This memory cost for ancestry records is small for each member of the population in comparison with the size of their genomic information. Therefore, the memory cost is not significant. We will explore the depth of ancestry in the experimental section.

With the ancestors recorded, avoiding inbreeding is done by comparing two candidate parents and checking if they share any common ancestors. If they do share common ancestors, depending on the GA, a new suitable pair of parents is found or the crossbreeding operation is skipped. In terms of the computation cost to search for shared ancestry, we simply do an exhaustive search of both family trees. We have implemented these trees as arrays, and therefore, the search is very simple. The cost for this comparison is similar to that of calculating the hamming distance between individual genomic strings.

Our inbreeding avoidance mechanism, however, differs from the original approaches in CHC and crowding GAs where the goal is to compare individuals based on how similar they are to one another. Instead, our inbreeding mechanism makes a similar assumption to preselection GAs, where children sharing ancestry will be similar just based on lineage. The problem in implementing a comparison of individuals with ordered chromosomes, like that of the original CHC and crowding GA, is identifying similar solutions, which will lead to implementations of subgraph isomorphism problems [20]. For example, a tour in the TSP might include the substring "A, R, C", and other population solutions with the substring "A, R, C" at some point might be considered similar. Searching for all such string matches would be expensive.

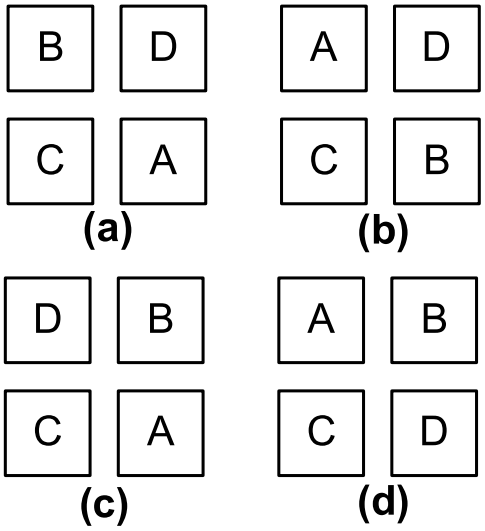


Fig. 4
SHOWS FOUR EXAMPLES OF WHAT MIGHT BE CONSIDERED SIMILAR PLACEMENTS

In the case of the 2D placement problem the problem of finding similar individuals is even harder than finding subgraphs. Figure 4 shows four examples of how 4 things (from a set of, potentially, thousands) can be placed relative close to each other, and in each case we might consider examples (a), (b), (c), and (d) similar to one another since the components are only one hop away from one another. These four components, however, will not be adjacent to one another in the genomic string. For example, example (a) would have a genome of the form "..., B, D, ..., C, A, ..." and (b) would have the form "..., A, D, ..., C, B, ..." meaning sub-string matching approaches cannot be used. Our proposed mechanism, however, can deal with both problems by trading off the measure of similarity for the simplicity of comparing ancestry.

It may also be possible to implement comparison of individuals using some sort of clustering technique, but again, the computational complexity of these approaches is high.

In the original implementation of the CHC algorithm, as the population becomes more and more similar the algorithm relaxes the similarity comparison. This relaxation, eventually, activates a restart condition for the algorithm. Using our ancestry mechanism we implement a similar mechanism in our implementation of the algorithm for ordered chromosomes. In our case, relaxation is implemented by changing the depth of generations explored for similar ancestors. Once the number of generations drops below one (which just compares parents) we activate the same restart mechanism as CHC where the best individual is mutated to create a new restarted population. During this restart, we eliminate all ancestry information and completely start over.

4. Experimental Setup

With our inbreeding avoidance technique, our goal is to delay convergence using a single-threaded algorithm for as long as possible while generating good solutions to the problem. In the case of POPs, run-time is not, necessarily, the most important concern, and instead, the number of generations before convergence occurs is what we are hoping to extend in this work. To study if our inbreeding mechanism achieves this we will compare a GA and a preselection based GA to an implementation of the CHC algorithm and a preselection based GA with inbreeding prevention. Our comparison will be based on how many generations each of the algorithms generates before the best solution exists for 500 new generations. Additionally, we will experiment with the number of generations of ancestors to be recorded to see how this impacts the results.

Before showing data from the results of these experiments, we will describe some of the details for our TSP and each of the candidate GAs.

4.1 TSP instance

Instead of using a particular benchmark such as TSPLIB [21] we build our benchmarks with randomly created cities and the distances are based on Euclidean distance measurements. The reason for this approach is our need for large problems with a high number of nodes to look at POPs where the likelihood of finding a global optimum is unlikely. For each experiment we use the same benchmark to fairly compare each of the algorithmic approaches.

4.2 Common Algorithmic Parameters

To keep our experiments fair, there are a few parameters and operations that are common for all of the GAs in this work. Population size for all of the algorithms is set to 500 individuals per generation. The crossover operator is as described in section 2 and is the PMX based crossover. The mutation operator is a random swap between two locations in the genomic string and this value is set 5% of the number of cities in the TSP. The initial population for each of the algorithms is generated randomly.

4.3 Base GA

The base GA consists of previously described parameters and operators with the following additional aspects. The GA creates each new generation with approximately 20% of the population from crossbreeding, 79% from crossbreeding and mutations, and 1% random new individuals. Crossbreeding and mutations are taken from the best 25% of individuals in the previous population, and no parents are kept from one generation to the next. References to this algorithm will use the name "*base_ga*".

4.4 Preselection GA

Our implementation of the preselection GA has all the previously described common parameters. The crowding aspect of this algorithm is implemented based on the assumption

that children of parents are the most similar (without doing a formal comparison), and therefore, parents will compete directly with their children for the next generation. In our implementation, two parents are selected and the crossover operation is implemented. The two resulting children are then mutated and are grouped with the parents, and the best two individuals are propagated to the next generation.

In the case where inbreeding prevention is part of this algorithm, parents are only selected when they share no ancestry. If they do share ancestry then a new pairing is found by keeping one of the candidate parents and randomly finding a new second candidate parent. There may be a concern that depending on the depth of ancestry there will be no suitable pairing, but given the population size and depth of ancestors to be recorded, this does not occur for our implementation. However, algorithm designers should consider this when implementing a similar mechanism for their own problems.

References to these two algorithms will be "*preselect_ga*" and "*preselect_no_inbreeding_ga*" where the later has the mechanism to prevent inbreeding.

4.5 CHC algorithm

Our implementation of the CHC algorithm is derived from the original publication by Eschelman and a brief description is included for each of the 4 main concepts within his algorithm.

1) Parents and children combined together in competition for next population - This is implemented by ranking both children and parents together and then destroying the lower half of these individuals. In the case of a tie, the parent individuals are chosen first.
2) Inbreeding avoidance - this mechanism was described in the previous section, and when the population of individual remains the same from one generation to the next then the depth of ancestry search is reduced by one.
3) Highly disruptive crossover operator - as described in the background we make the assumption that the PMX crossover, where approximately 50% of the genetic material comes from each of the two parents, is sufficiently disruptive.
4) Full restart (from the best individual) once no new offspring are created in a generation - once the depth of search in the inbreeding avoidance mechanism reaches zero, we take the best individual and copy and mutate (with a 35% chromosome mutation rate) to create a new population. The algorithm then resumes normal operation.

References to this algorithm will be "*chc_algorithm*".

5. Experimental Results

In this section, we will look at two experiments. First, what happens to *chc_algorithm* as we change the number of past generations to record. These results will show us if there is any clear advantage to having a deeper record of ancestry. Next, we will look at how all three algorithms compare to each other observing how our mechanism improves the perseverance of diversity.

5.1 Impact of Ancestors on Diversity

For this experiment, we vary the depth of the ancestry tree between 3 and 8 generations for the *chc_algorithm*. In this experiment, the depth of ancestry search is controlled by the algorithm as described earlier in section 4.5. Each instantiation of the algorithms are executed until 500 iterations of the algorithm provide no improvement on the cost function and the run exits.

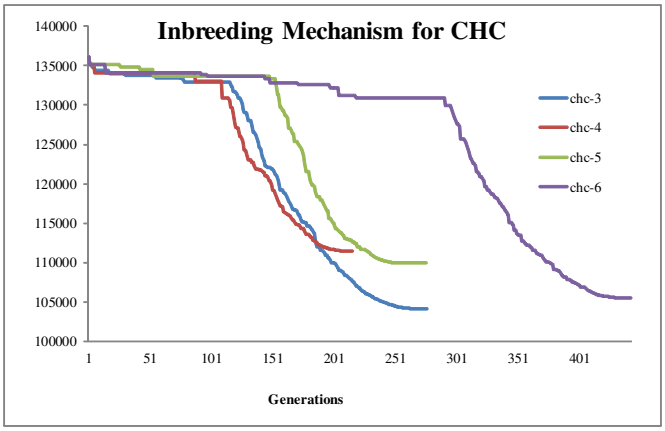


Fig. 5
SHOWS THE RESULTS FOR CHC WITH DIFFERENT ANCESTRY DEPTHS

Figure 5 shows a graph of our CHC implementation for the different ancestry depths. The y-axis shows the cost function measure for a TSP consisting of 2000 cities. The x-axis is the number of generations, where the last generation is reported before the 500 repetitions of no improvement. A legend is provided and the number for each corresponds to the maximum depth of ancestry to be recorded. Note that for each instance of the algorithm the random starting population is the same (chc-3, chc-4, chc-5, and chc-6 all have the same initial population).

From this example, we can see that as the ancestry depth is increased, the number of generations tends to increase. This tendency, however, is not the case for the 3 ancestor generation run labeled as *chc-3*. Also, the instance chc-3 finds one of the best solutions to the problem. This type of result is possible based on the randomness of the algorithm, and the more general result that diversity is maintained based on the trend that increasing the depth of generations recorded tends to increase the number of generations before the run exits.

To get a more thorough picture of what is happening, Figure 6 shows more runs of the CHC algorithm with inbreeding avoidance mechanism and a maximum ancestral depth of eight. For each of the five runs, we have colored the respective ancestry depth runs with the same coloring. The lower number of generations (3, 4, 5) are in dark colors, and the higher number of generations are in the lighter colors. The graph clearly shows that there is randomness for each run as expected. In terms of trends, the higher number of generation

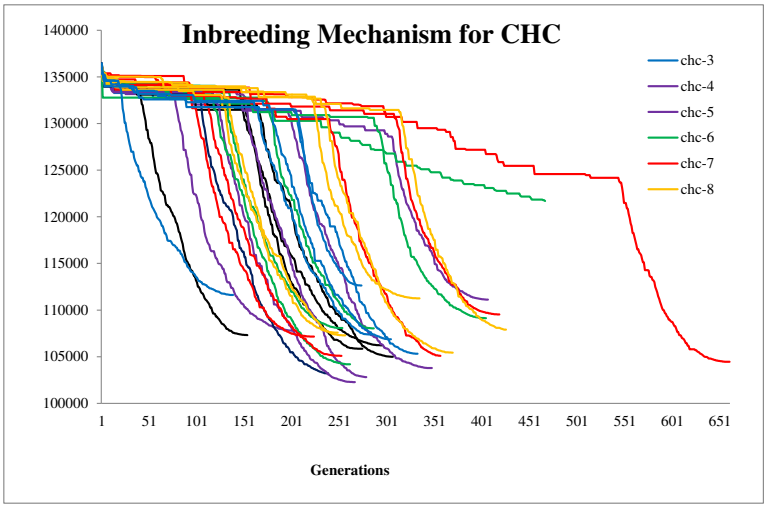


Fig. 6
SHOWS THE RESULTS FOR CHC WITH DIFFERENT ANCESTRY DEPTHS FOR 5 INSTANCES

based algorithms find a path to the low-energy solution later. The best solutions are found by execution runs by chc-5, chc-4, chc-6, and chc-7 in that order. We conclude that the depth range between 5 through 7 seems to be the best choice for ancestry records based on the results and the intuition that algorithms maintaining generations past 6 are unnecessarily restricting mating for unlike individuals.

One other thing we have observed in our CHC algorithm experiments is the lack of effectiveness for the restart mechanism. Only in 2 of the 30 runs of the algorithm was the restart effective in finding better solutions, noting that all 30 instances do initiate the restart mechanism. We hypothesize that random mutations for ordered chromosomes at low energy solution spaces is not effective, and other mechanisms need to be employed. We leave this to future work.

5.2 Comparison of All Algorithms

In this experiment, we compare all our implementations. Our comparison points include those algorithms with an inbreeding avoidance mechanism including the preselection algorithm with an ancestor depth of 6 previous generations. The exit condition is the same as previously described, and our overall goal is to maintain diversity as long as possible while finding good solutions.

Figure 7 shows a graph of our algorithmic implementations with a selection of CHC runs from the previous experiment. Similar to the previous graphs, the y-axis shows the cost function measure for a TSP consisting of 2000 cities, and the x-axis shows the number of generations, where the last generation is reported before the 500 repetitions of no improvement. The legend shows the name for each of the algorithm implementations with the numbering showing an algorithm with inbreeding prevention according to the depth of ancestors recorded. Note that the *preselect_no_inbreeding_ga* algorithm uses a depth of 6 ancestral generations to be recorded and when a pair of parents are selected, all ancestors are searched for common relations.

From the graph, we see a number of trends. First, preselection GAs maintain diversity for the greatest number of generations. This is to be expected since the crowding nature of this algorithm maintains highly diverse pockets of evolution. The best results generated by the preselection GAs, however, are not as good as the CHC algorithm. This is partially due to the pockets of evolution, which maintains diversity at a cost of less competition. The addition of the inbreeding avoidance mechanism in the preselection algorithm improves the quality of the solution and extends the diversity (number of generations), but not by a significant amount. For our purposes, the small crowding pockets do not seem to allow sufficient exploitation to find good solutions.

Overall, the best solutions are found by the more diverse algorithms that do not allow inbreeding based on the mechanism we have introduced. From our experimental data, this suggests that our inbreeding avoidance technique is providing the desired outcome, and overall, we observe significant improvement on diversity and quality for all the algorithms compared to our *base_ga*.

6. Conclusion

In this work, we introduced the concept of POPs and how GAs can play a valuable roll in solving these types of problems. We then explored how to maintain genetic diversity within a single-threaded GA run targeting POPs. We introduced an inbreeding avoidance technique inspired by tabu search, and we described how such a mechanism can be used both with the CHC algorithm and preselection algorithm for genomic strings that have ordered chromosomes. These types of chromosomes can be used to solve problems such as TSP and the FPGA placement problem.

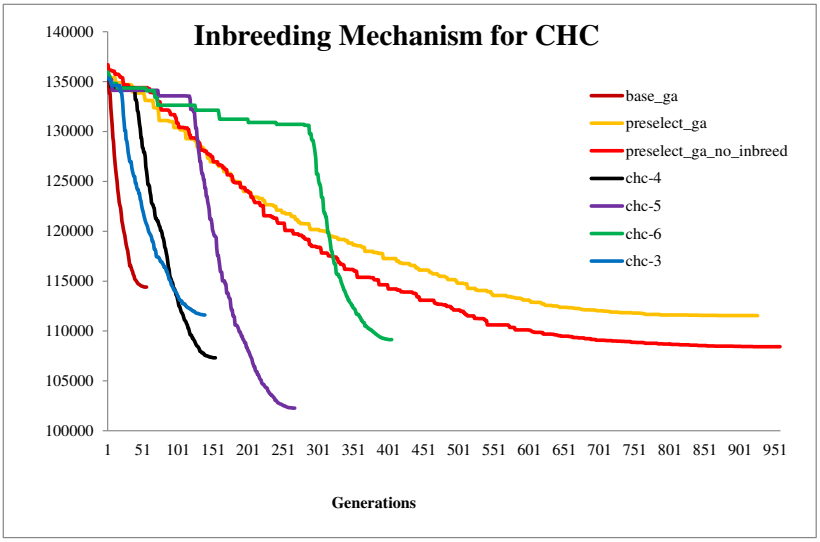


Fig. 7
SHOWS THE RESULTS ALL OUR THE IMPLEMENTATIONS.

The experiments with our inbreeding avoidance mechanism shows that the more levels of recording for generations improves the diversity of the population, and tends to improve the quality of results generated by the GAs. We believe that these types of mechanisms are not only valuable for POPs, but this could be exploited by other GAs for other types of problems.

References

[1] P. Jamieson, "Exploring inevitable convergence for a genetic algorithm persistent fpga placer," in *GEM*, 2011, pp. 1–8. [Online]. Available: http://www.users.muohio.edu/jamiespa/html_papers/gem_11.pdf

[2] ——, "Persistent cad for in-the-field power optimization," in *ERSA*, 2010, pp. 267–270. [Online]. Available: http://www.users.muohio.edu/jamiespa/html_papers/ersa_10.pdf

[3] 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

[4] 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

[5] K. Roy and C. Sechen, "A timing driven n-way chip and multi-chip partitioner," in *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on*, 1993, pp. 240 –247. [Online]. Available: http://www.eecg.toronto.edu/~jayar/pubs/sankar/fpga99sankar.pdf

[6] 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.

[7] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, pp. 533–549, May 1986. [Online]. Available: http://dl.acm.org/citation.cfm?id=15310.15311

[8] 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

[9] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems." Ph.D. dissertation, Ann Arbor, MI, USA, 1975, aAI7609381.

[10] D. J. Cavicchio, "Adaptive Search Using Simulated Evolution," Ph.D. dissertation, University of Michigan, 1970.

[11] S. De, S. K. Pal, and A. Ghosh, "Genotypic and phenotypic assortative mating in genetic algorithm," *Inf. Sci.*, vol. 105, pp. 209–226, March 1998. [Online]. Available: http://dx.doi.org/10.1016/S0020-0255(97)10035-4

[12] 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. [Online]. Available: http://doi.acm.org/10.1145/330855.330986

[13] D. Whitley, D. Hains, and A. Howe, "A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover," in *Proceedings of the 11th international conference on Parallel problem solving from nature: Part I*, 2010, pp. 566–575. [Online]. Available: http://dl.acm.org/citation.cfm?id=1885031.1885092

[14] ——, "Tunneling between optima: partition crossover for the traveling salesman problem," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 915–922. [Online]. Available: http://doi.acm.org/10.1145/1569901.1570026

[15] D. E. Goldberg and R. Lingle, Jr., "Alleles, loci, and the traveling salesman problem," in *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985, pp. 154–159. [Online]. Available: http://dl.acm.org/citation.cfm?id=645511.657095

[16] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 1987, pp. 224–230. [Online]. Available: http://dl.acm.org/citation.cfm?id=42512.42542

[17] B. A. Julstrom, "Very greedy crossover in a genetic algorithm for the traveling salesman problem," in *Proceedings of the 1995 ACM symposium on Applied computing*, 1995, pp. 324–328. [Online]. Available: http://doi.acm.org/10.1145/315891.316009

[18] V. A. Cicirello, "Non-wrapping order crossover: an order preserving crossover operator that respects absolute position," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 1125–1132. [Online]. Available: http://doi.acm.org/10.1145/1143997.1144177

[19] L. Davis, "Applying adaptive algorithms to epistatic domains," in *IJCAI*, 1985, pp. 162–164.

[20] H. G. Barrow and R. M. Burstall, "Subgraph isomorphism, matching relational structures and maximal cliques," *Information Processing Letters*, vol. 4, pp. 83–84, 1976.

[21] G. Reinelt, "TSPLIB — a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

Optimal Design of Islanded Microgrid Using Genetic Algorithm

*Farzad Razavi, **Reza Torani, ***Iman Askarian, ****Alireza Asgharizadeh, *Nima Masoomi
*Department of Electrical Engineering, Qazvin Branch, Islamic Azad University, Qazvin
** Department of Electrical Engineering, Tafresh University
*** Department of Electrical Engineering, Tehran University
**** Department of Electrical Engineering, Amirkabir University
(Corresponding author: Farzad Razavi, tell: +98912177 3241; email: farzad.razavi@qiau.ac.ir)

Abstract—**This paper focuses on an optimization method for a droop controlled microgrid in islanding operation. The ultimate goal is to optimize droop coefficient to minimize frequency variation. Also, gains of PI controllers are optimized to ensure good behavior of the controller. Optimizations are implemented in MATLAB software using Genetic Algorithm (GA). Stability of optimized PI gains of voltage and current controllers are analyzed**.

Keywords- Microgrid; Distributed Generation(DG); Droop Control; Islanding operation; V/f control; power sharing; Voltage Source Inverter (VSI); Genetic Algorithm(GA).

I. INTRODUCTION

Advancement in Distributed Generations (DGs) systems and power electronic devices led to concept of Microgrid. It can integrate renewable energy and other forms of DG and also increase reliability and efficiency [1]. Many forms of DG such as fuel cell and photovoltaic are interfaced to the network through power electronic devices. These interfaced devices make the system more flexible in their operation and their control compared to the conventional electrical machine [2- 4]. Consequently, control strategy of parallel connected inverters is important for microgrid operating.

The basic objective in microgrid control is to achieve an accurate power sharing and regulation of the microgrid voltage and frequency. Centralized control of microgrid is proposed in [5]. However, this method is impractical and costly in microgrid with long distance between DG units. To overcome this limitation, decentralized controllers based on the droop control method are proposed [6-9]. This method does not need any communication.

In normal condition, Microgrid operates in grid-connected mode and the main grid can support the system frequency and voltages by supplying the power mismatch immediately. When a fault occurs in the main grid, Microgrid needs to disconnect from the main grid to provide uninterrupted power to the loads. In islanding operation mode, DGs operate in Vf control for supplying microgrid load and controlling voltage and frequency [10]. To this purpose, droop control that assigns the amount of power sharing for changes of load without communication is used [11].

After transition from grid-connected to islanding, the frequency of microgrid is determined by the droop coefficients of DGs. Since the droop control changes the system frequency to supply the power mismatch, the frequency variation occurs. To maintain frequency close to the nominal value, droop coefficients should be determined properly. The droop coefficient and sharing ratio may be dictated by economic interested of the system operators [12]. The choice of droop coefficients in such case is analyzed in [13].

The main problems for controller parameter optimization are nonlinearity and complexity of the system. Small signal linearization is a usual method for designing of controller parameters. But this method depends on the operation point [14]. Hence, in this paper simulation model in MATLAB/Simulink is employed as a replacement for small signal method for optimization.

This paper concentrates on optimization of microgrid controller. Genetic algorithm is used to optimize the droop coefficients. The proportional and integral gains of voltage and current controller are optimized to achieve the system stability.

II. MICROGRID CONFIGURATION

A microgrid configuration with two DG is shown in Fig.1. Each DG consists of DC source, voltage source inverter (VSI) and LC filter for rejecting high frequency harmonics. Load 1 is sensitive load. Load 2 is non sensitive load.

Under normal operation, the microgrid is a part of main grid. In this mode, DGs injected predefined active and reactive powers and main grid regulate voltage and frequency of microgrid. When disturbance such as fault occurs in the main grid, the switch k opens and microgrid operate in islanding mode. Hence, increase the reliability of the microgrid. In islanding operation mode, due to absence of main grid, DGs should be able to share the power mismatch to supply loads and to maintain power quality. In this situation, DGs operate in V/f control for controlling the voltage and frequency of microgrid and feeding the loads.

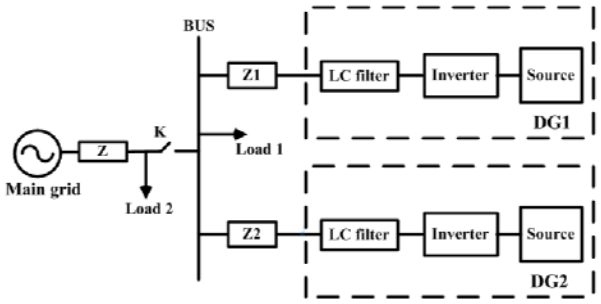


Fig. 1. Microgrid Configuration

III. CONTROL STRATEGY

This section presents the control strategy for islanding operation mode of microgrid. Fig.2 shows the V/f controller and Fig.3 shows the power controller that consists of power calculation and droop controller. The droop control is used for sharing power between DGs in islanding operation.

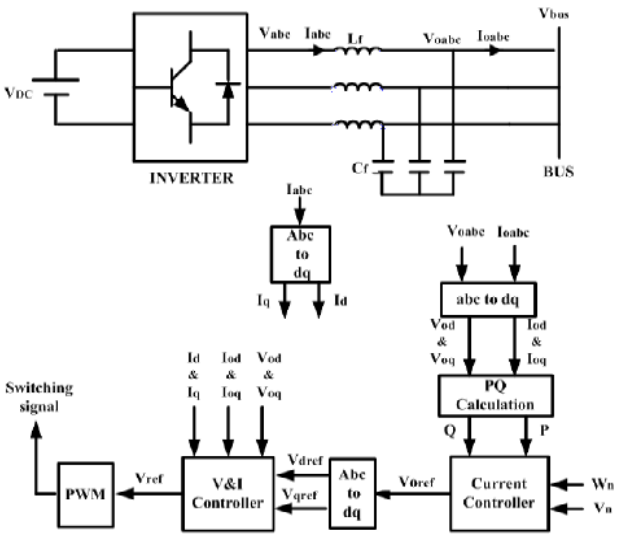


Fig. 2. V/f Control

Power calculation block calculates active and reactive powers from measured instantaneous values of d axis and q axis voltage and current. Equations 1 and 2 show the calculation procedure of powers:

$$P = \frac{w_c}{S + w_c}(V_{od} I_{od} + V_{oq} I_{oq}) \tag{1}$$

$$Q = \frac{w_c}{S + w_c}(V_{od} I_{oq} - V_{oq} I_{od}) \tag{2}$$

The basic idea behind the droop control is to mimic the governor of synchronous generator. In a conventional power system, synchronous generators will share any increase in the load by decreasing the frequency according to their governor droop characteristic. This principle is implemented in inverters by decreasing the reference frequency when there is an increase in the load. Similarly, reactive power is shared by introducing a droop characteristic in the voltage magnitude. In islanding operation, droop method can be used to share loads and controlling voltage and frequency in special range [14], [15].

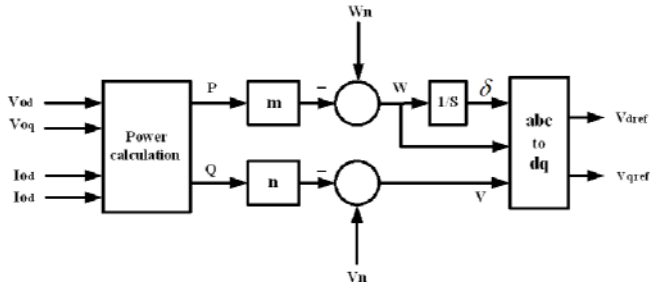


Fig. 3. Droop Control

Two coefficients control the changes slope of frequency and voltage against active and reactive power.

$$w = w_n - mP \tag{3}$$

$$V = V_n - nQ \tag{4}$$

where w_n, V_n, m, n, P and Q are rated frequency, rated voltage, active power droop coefficient, reactive power droop coefficient, output active power and output reactive power of VSI, respectively. Droop coefficients are defined below:

$$m = \frac{\Delta w}{P_{\max}} \tag{5}$$

$$n = \frac{\Delta V}{Q_{\max}} \tag{6}$$

where Δw and ΔV are maximum allowable deviations of frequency and voltages. Also $P_{\max}$ and $Q_{\max}$ are maximum output active and reactive powers of DG.

The reference frequency and amplitude of the output voltage can be obtained by the droop control. Then V_{dref} and V_{qref} are obtained by dq transformation.

Fig.4 shows the voltage and current controller loops. Voltage and current control loops use PI controllers.

IV. OPTIMIZATION ALGORITHM (GA)

A. Genetic Algorithm

In this paper, Genetic Algorithm is used to optimize the objective function. To optimize a problem, using the GA, a population is required to be defined at the first step. This population is formed by binary accidental quantization of chromosomes. In the next step, produced population is applied to the objective function and the fitness of chromosomes is obtained, using equation 7. Some of the best answers are chosen and new generation is produced by the genetic operators of crossover and mutation. In the first type, two gens, that should be combined, are placed beside each other and are divided from a specified point. Then, the sides that are placed in front of each other are combined together. In the second type, a percent of chromosomes are substituted by another value of their allowable confine, in order to make the optimization, global and not local. To have a global and the fastest answers, both of these genetic operators are used in this paper [16].

$$Fitness = \frac{1}{OF} \quad (7)$$

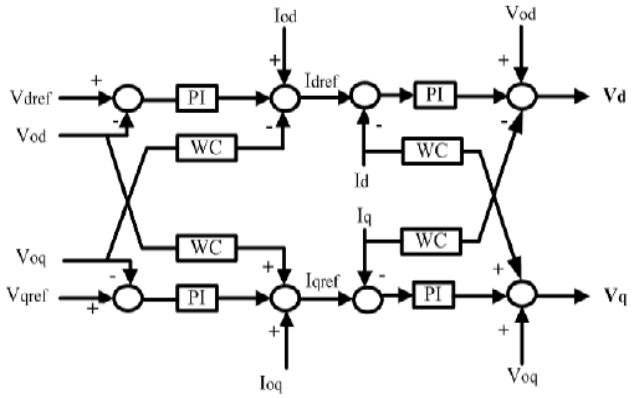


Fig. 4. Voltage and Current Controller

B. Controller Parameters Optimization

The main problems in control optimization are nonlinearity and complexity of the system. A method for this purpose is small signal linearization. But linearization of microgrid for optimization of droop controller may result in difference with the actual microgrid [14, 17]. Hence in this paper, simulation model in MATLAB/Simulink is used for optimization.

After transition from grid-connected to islanding, the frequency of microgrid is determined by the droop coefficients of DGs. Since the droop control changes the system frequency to supply the power mismatch, the frequency variation occurs. To maintain frequency close to the nominal value, droop coefficients should be determined properly.

The control problems are formulated as optimization problem. The criteria of droop control optimization are:

1) Frequency of microgrid should be maintained near nominal value. It means that, frequency variation of microgrid after transition to islanding and load changes should be minimal.

2) DGs output power should be equal to load power and power sharing between DGs should be based on the droop control.

The proportional and integral constants of PI controllers for voltage and current controller are determined by GA to obtain good response and stability of system.

C. Technical Constraints

I. Frequency Deviation

Selection of appropriate droop coefficient results lower deviation of microgrid frequency. Frequency deviation after change of load is determined by following equation, as explained in Appendix.

$$\Delta f = \frac{m_1 m_2}{m_1 + m_2} \Delta P_{Load} \quad (8)$$

According to equation (8), droop coefficient should be optimized for minimizing the frequency deviation. The constraint for droop coefficient is presented bellow.

$$0 < m_{optimized} < m \quad (9)$$

where m value is determined according to equation (5).

I. Power Sharing

In optimization process, load should be supplied and shared between DGs correctly. Following equations shows the power sharing mechanism between DGs after any change of the load, as explained in Appendix.

$$\Delta f = \frac{m_1 m_2}{m_1 + m_2} \Delta P_{Load} \quad (10)$$

$$\Delta f = \frac{m_1 m_2}{m_1 + m_2} \Delta P_{Load} \quad (11)$$

According to equations (10) and (11), change of the load between DGs should satisfy the following equation.

$$\Delta f = \frac{m_1 m_2}{m_1 + m_2} \Delta P_{Load} \quad (12)$$

D. Objective Function

The proposed objective function of this paper consists of droop coefficient and power sharing. The first part of objective function represents frequency deviation, as in equation (13).

$$F_1 = \min(\Delta f) \quad (13)$$

If the constraints of F1 are violated, the output would be infinite value.

The second part of objective function represents the power sharing accuracy between DGs.

$$F_2 = \min(P_1 - \frac{m_2}{m_1} P_2) \quad (14)$$

It should be note that, in this paper it's assumed that DGs have the same droop coefficients. So, the purpose of second part is to equally share load between DGs.

V. Simulation Results

The control method for islanded microgrid of Fig.1 have been modeled and simulated in MATLAB/Simulink. System parameters are presented in TABLE I. For verifying the power sharing between DGs, load is changes from 6 KW to 10 KW at t=0.3 s. Results are presented in two cases. In these cases, the droop coefficients of DGs are chosen equally, so that the power is shared between them equally.

TABLE I. SYSTEM PARAMETERS

Parameters		Values
DG1 & DG2	DC-link Voltage	580V
	Inverter filter inductance	1.35mH
	Inverter filter capacitance	50μF
	Inverter switching frequency	8KHz
	S_{rate}	10KVA
Controller	m	6.25e-5
	n	1.83e-5
	Wn	50Hz
	Vn	220V
Parameters of Lines1and 2		0.03+ j0.11Ω
Load		6KW
RMS line voltage		220√3

Case1:

In this case, selection of droop coefficients are based on the equations 5 and 6. Allowable frequency deviation is considered to 0.5 Hz for determination of droop coefficient. Parameters of PI controller for Voltage and current controller are obtained using try and error. These parameters are presented below.

$$K_{PV} = 0.19,\ K_{IV} = 398,\quad K_{PI} = 0.5,\ K_{II} = 800$$

Fig.5 depicts the output active powers of DGs. Since DGs have a same droop coefficient, change of the load is shared between them equally.

Fig.6 shows the frequency of islanded microgrid. After transition to islanding and change of load in this mode, frequency of microgrid is determined by droop coefficients of DGs in microgrid. It can be seen from Figs.5-6 that the system does not have a good behavior. Although the deviation of frequency in islanding mode is in allowable limit, but it can be minimize using Genetic algorithm.

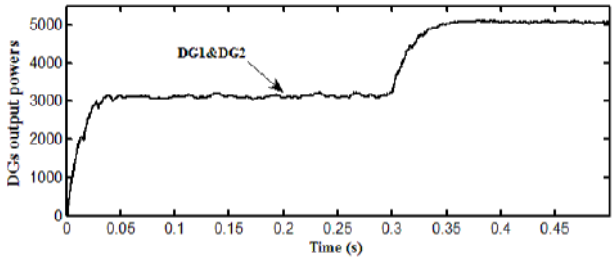


Fig. 5. Output power of DGs

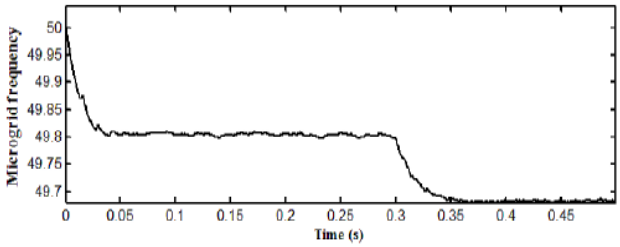


Fig. 6. Frequency of the microgrid

Case 2:

Selection of droop coefficient in this case is based on the Genetic Algorithm. Constraints on the droop coefficient for optimization are 0 and 6.25×10^{-5}. Value of 6.25×10^{-5} is obtained based on equation (5) for 0.5Hz allowable deviation. Optimized parameters are:

$$m = 0.0206\times10^{-5},\quad K_{PV} = 0.4296,\ K_{IV} = 81.946,$$
$$K_{PI} = 1.3123,\ K_{II} = 309.08$$

Fig.7 shows the output active powers of DGs with optimized parameters. Since DGs have a same droop coefficient, change of the load is shared between them equally.

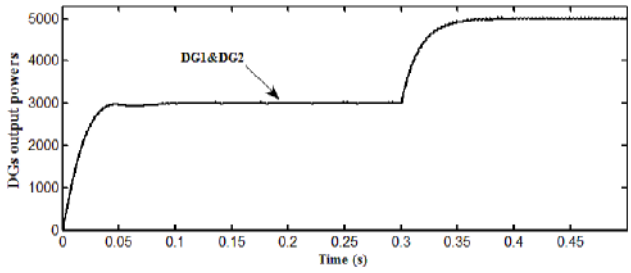


Fig. 7. Output power of DGs

Fig.8 shows the frequency of islanded microgrid. It can be seen from Fig.8 that the frequency variations of microgrid in this case become very smaller than the case1. Also it can be seen from Figs.7-8 that the system behavior with optimized PI gains is better than case1. As a result, with optimization of droop coefficient and PI gains, frequency deviation become smaller and the system behavior is improved.

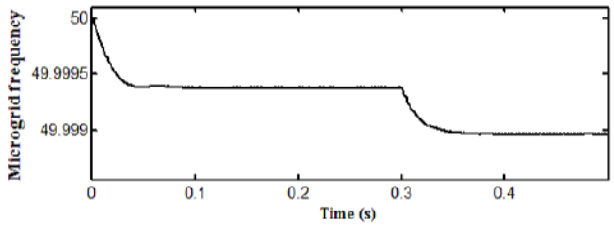


Fig. 8. Frequency of the microgrid

VI. STABILITY ANALYSIS

In this section, stability of voltage and current controllers with optimized PI gains are verified. For this purpose, their transfer functions should be determined.

1) Current Controller Transfer Function:

Fig. 9 shows the block diagram of the current controller for Islanding operation. V_o is the disturbance input. The inverter stage does not have any significant transient time associated with it [18], and hence, it modeled as an ideal gain. This ideal gain can be given by $G_{inv}(s) = 1$. Block diagram of current controller is shown in fig.9.

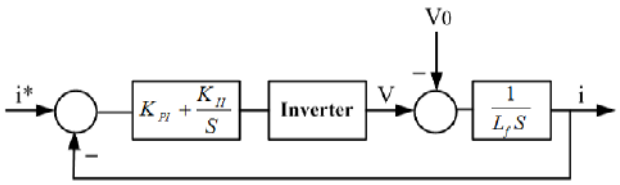


Fig. 9. Block diagram of current controller

The transfer function of the current controller is given by equation (15). It can be seen from equation (15) that the system is stable based on the conventional control theory.

$$T(S) = \frac{1.3123S + 309.08}{1.35\times10^{-3}S + 1.3123S + 309.08} \qquad (15)$$

Fig.10 Shows bode plot of the current controller. It can be seen that the system have positive phase margin and is stable.

Fig.11 shows step response of the controller. In Fig.11, Rise time (tr) is 0.00151, Overshoot is 13.3% and steady state error is zero. We can find out that the system has appropriate performance.

For analysis response of current controller to disturbance, the unit step is applied to disturbance input (V_0) in Fig.12. It can be seen that the system have good response and disturbance is damped very soon. Settling time is less than 15ms.

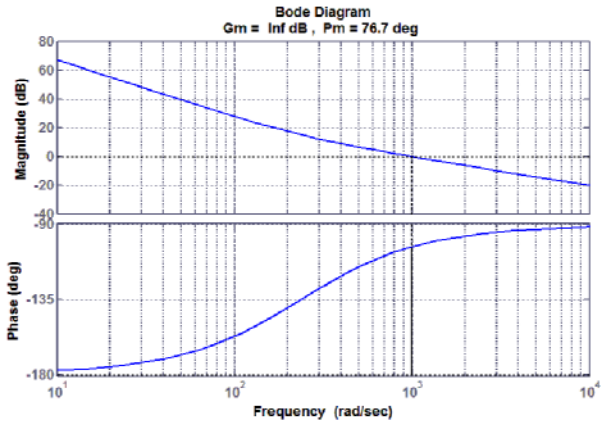


Fig. 10. Bode diagram of the current controller

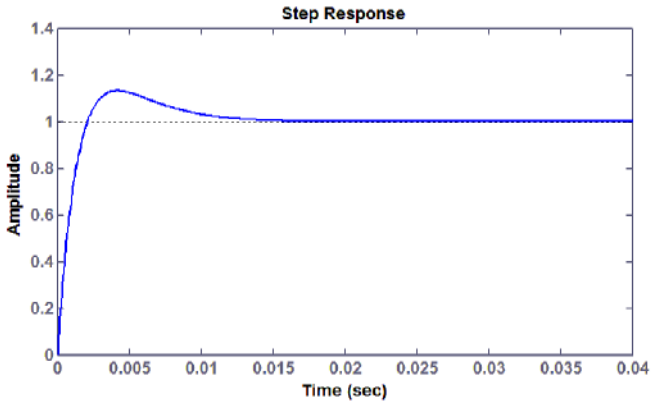


Fig. 11. Step response of the current controller

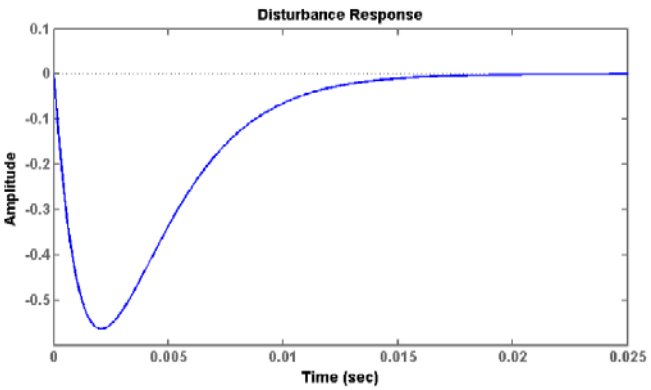


Fig. 12. Disturbance response of the current controller

2) Voltage Controller Transfer Function:

Block diagram of voltage controller is shown in Fig.13.

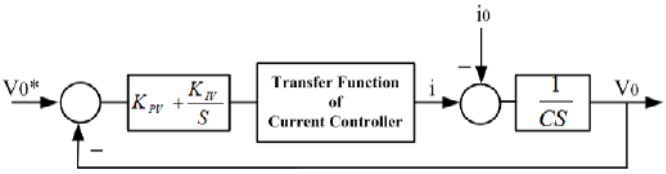


Fig. 13. Block diagram of voltage controller

The transfer function of this controller system is given by equation (16). According to equation (16), the system is stable based on the conventional control theory.

$$T(S) = \frac{4.511S^2 + 3438S + 491676}{6.75\times10^{-8}S^4 + 5.25\times10^{-4}S^3 + 4.811S^2 + 3438S + 491676} \quad (16)$$

Fig.14 Shows bode plot of the voltage controller. It can be seen that the system have positive phase margin and is stable. Fig.15 shows step response of the controller. In Fig.15, settling time (ts) is 0.00144, Overshoot is 24% and steady state error is zero. We can find out that the system has appropriate performance.

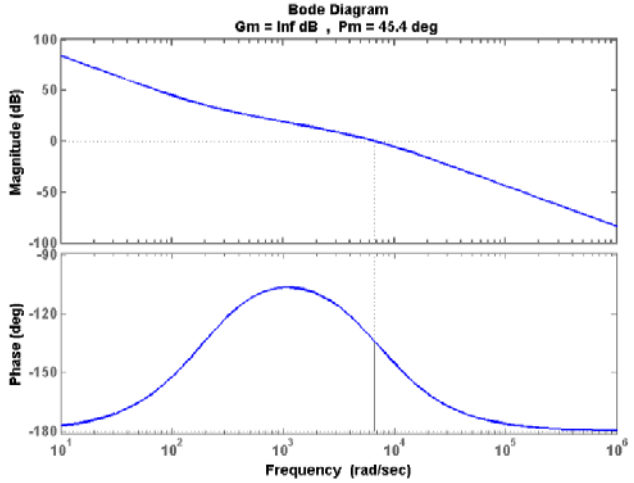


Fig. 14. Bode diagram of the voltage controller

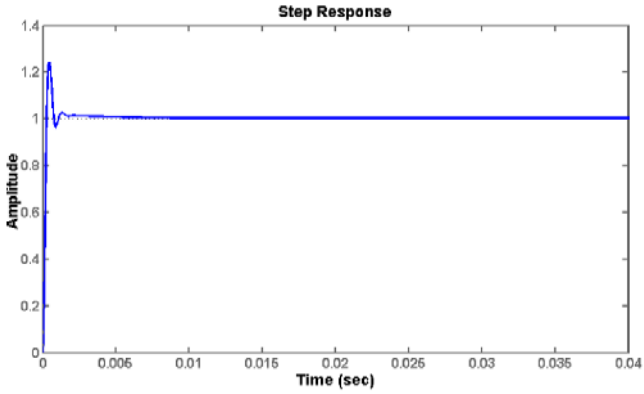


Fig. 15. Step response of the voltage controller

For analysis the response of voltage controller to disturbance, the unit step is applied to disturbance input (i_0) in Fig.16. It can be seen that the system have good response and disturbance is damped very soon. Settling time is less than 19ms.

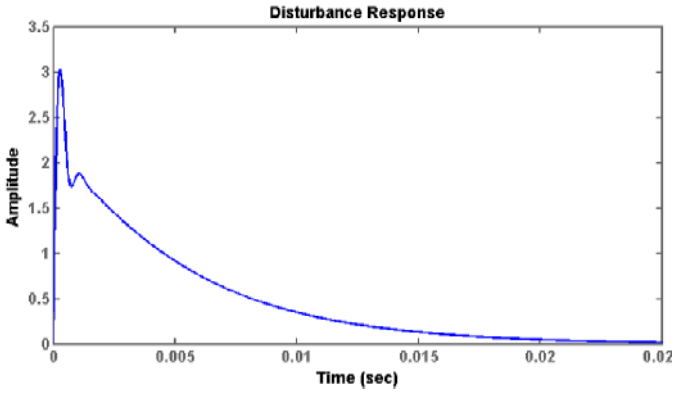


Fig. 16. Disturbance response of the voltage controller

VII. CONCLUSION

This paper described an optimization method for a droop controlled islanded microgrid based on the Genetic Algorithm, which successfully implements optimal frequency deviation by selecting droop coefficients from a region where the frequency deviation is in an allowable range. The proportional and integral gains of PI controller are optimized to achieve good response and stability of the system.

Particular emphasis has been paid to the impact of droop coefficient on frequency deviation, because this coefficient plays a significant role in the microgrid frequency deviation after transient to island and change of load in this situation.

Simulation results are presented that support validity of this optimization. A comparison has been done between optimized and common method for coefficient selection. Finally, stability analyses for optimized PI gains are presented.

APPENDIX

If the load of microgrid is changed in islanding mode, the frequency of microgrid and share of DGs are determined according to following equation.

$$\Delta P_{DG1} + \Delta P_{DG2} = \Delta P_{load} \tag{1A}$$

According to droop equation:

$$-\frac{\Delta f}{m_1} - \frac{\Delta f}{m_2} = \Delta P_{load} \tag{2A}$$

where m_1 and m_2 are droop coefficients of DG_1 and DG_2, respectively.

Hence, the frequency deviation and power sharing between DGs for load change in islanding operation is obtained.

$$\Delta f = -\frac{m_1 m_2}{m_1 + m_2}\Delta P_{load} \tag{3A}$$

$$\Delta P_{DG1} = -\frac{m_2}{m_1 + m_2}\Delta P_{load} \tag{4A}$$

$$\Delta P_{DG2} = -\frac{m_1}{m_1 + m_2}\Delta P_{load} \tag{5A}$$

REFERENCES

[1] R. H. Lasseter, "Microgrids," in *Proc. Power Eng. Soc.Winter Meeting*, Jan. 2002, vol. 1, pp. 305–308.

[2] A. Arulapalam, M. Barnes, A. Engler, A. Goodwin, and N. Jenkins, "Control of power electronic interfaces in distributed generation microgrids," *Int. J. Electron.*, vol. 91, no. 9, pp. 503–523, Sep. 2004.

[3] Y. Li, D. M. Vilathgamuwa, and P. C. Loh, "Design, analysis and real time testing of a controller for multibus microgrid system," *IEEE Trans. Power Electron.*, vol. 19, no. 5, pp. 1195–1204, Sep. 2004.

[4] Nagaraju Pogaku, Milan Prodanovic´, and Timothy C. Green" Modeling, Analysis and Testing of Autonomous Operation of an Inverter-Based Microgrid" IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 22, NO. 2, MARCH 2007

[5] J. A. P. Lopes, C. L. Moreira, and A. G. Madureira, "Defining control strategies for microgrids islanded operation," IEEE Trans. Power Syst., vol. 21, no. 2, pp. 916–924, May 2006.

[6] Yasser Abdel-Rady Ibrahim Mohamed and Ehab F. El-Saadany" Adaptive Decentralized Droop Controller to Preserve Power Sharing Stability of Paralleled Inverters in Distributed Generation Microgrids" IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 23, NO. 6, NOVEMBER 2008

[7] Il-Yop Chung, Wenxin Liu, David A. Cartes, Emmanuel G. Collins, Jr, and Seung-Il Moon," Control Methods of Inverter-Interfaced Distributed Generators in a Microgrid System" IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, VOL. 46, NO. 3, MAY/JUNE 2010

[8] W. Yao, M. Chen, J. Matas, J. M. Guerrero, Z. M. Qian, "Design and Analysis of the Droop Control Method for Parallel Inverters Considering the Impact of the Complex Impedance on the Power Sharing". IEEE Trans. Ind. Electron., vol.58, pp.576-588, February 2010.

[9] X. T. Zhang, Q. C. Zhong, H. Zhang, X. k. Ma, "Proportional load sharing method for parallel connected inverters", in Proc. IEEE ISIE'10, 2010, pp. 2261- 2265.

[10] F. Z. Peng, et al., "Control and protection of power electronics interfaced distributed generation systems in a customer-driven microgrid," in Power & Energy Society General Meeting, 2009. PES '09. IEEE, 2009, pp. 1-8.

[11] K. De Brabandere, B. Bolsens, J. Van den Keybus, A. Woyte, J. Driesen, and R. Belmans, "A voltage and frequency droop control method for parallel inverters," IEEE Trans. Power Electron., vol. 22, no. 4, pp. 1107– 1115, Jul. 2007.

[12] M. A. Hassan and M. A. Abido" Optimal Autonomous Control of an Inverter- Based Microgrid Using Particle Swarm Optimization" IEEE 2010

[13] Barklund, N. Pogaku, M. Prodanovi´c, C. Hernandez-Aramburo, and T. C. Green, "Energy Management in Autonomous Microgrid using Stability-constrained droop control of inverters," IEEE Trans. Power Electron., Vol. 23, No. 5, pp. 2346–2352, Sept.
2008.

[14] Il-Yop Chung, Wenxin Liu, David A. Cartes, Emmanuel G. Collins, Jr, and Seung-Il Moon" Control Methods of Inverter-Interfaced Distributed Generators in a Microgrid System" IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, VOL. 46, NO. 3, MAY/JUNE 2010

[15] M. C. Chandorkar, et al., "Control of parallel connected inverters in stand-alone AC supply systems," in Industry Applications Society Annual Meeting, 1991., Conference Record of the 1991 IEEE, 1991, pp. 1003-1009 vol.1.

[16] S.A. Hosseini, M. Karami, S.S. Karimi Madahi, F. Razavi and A.A. Ghadimi, "Finding the optimal capacity and location of distributed generation resources and analyzing the impact of different coefficient factors", J. Basic Appl. Sci. Res., 1(12), 2578-2589, 2011

[17] I. Chung, W. Liu, D. A. Cartes, and K. Schoder, "Control parameter optimization for a microgrid system using particle swarm optimization," in Proc. IEEE ICSET, Singapore, Nov. 2008, pp. 837–842.

[18] Irvin J. Balaguer, Qin Lei, Shuitao Yang, Uthane Supatti, and Fang Zheng Peng, "Control for Grid-Connected and Intentional Islanding Operations of Distributed Power Generation," IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, vol. 58, no. 1, January2011.

A Fitness Proportionate Reward Sharing: a Viable Default Hierarchy Formation Strategy in LCS

Abrham Workineh and Abdollah Homaifar
Department of Electrical and Computer Engineering,
North Carolina A&T State University, Greensboro, North Carolina, USA
atworkin@ncat.edu, homaifar@ncat.edu

Abstract: *The learning task in a Learning Classifier System (LCS) is aimed at building a set of rules that work in coordination to accurately model a given environment. The addition of the hash symbol ('#') in LCS's condition provides varying degree of coverage to environmental niches. Building a hierarchical set of rules, where accurate and more specific rules respond to a subset of the situations covered by more general but less accurate default rules will be vital to achieve a compact rule set size, especially when dealing with an environment that has huge numbers of states. However, the formation of viable default hierarchy in LCS has been a nightmare in this research area for decades. This paper presents a new resource allocation scheme that leads to the formation of a default hierarchy in LCS. A fitness proportionate reward sharing scheme is introduced and the performance of the algorithm is tested using known test functions.*

Keywords: LCS, Default Hierarchy, Genetic Algorithm, Reward Sharing.

1. Introduction

A Learning Classifier System (LCS) is a machine learning paradigm where an agent learns to perform a certain task by interacting with a partially known environment. Learning is via the guidance of a reward signal that indicates the quality of action taken by the learning agent [1]. Classifiers are rules in the form of "if condition then action" format. In LCS, learning is an iterative process that begins with a randomly initialized population of candidate classifiers and evolves the initial population to build an accurate model of the environment as iteration goes on. The intention of the classifier in the learning process is to accumulate as much reward as possible. The reinforcement program guides the search for solution by rewarding classifiers that propose a correct action.

To date, several modifications have been made to the traditional LCS. Wilson introduced a strength based learning classifier system known as the zeroth-level classifier system (ZCS) in 1994 [2]. A year later, he introduced accuracy based classifier system (XCS) which brought a major change in LCS's rule fitness calculation [3]. The fitness is made to represent the accuracy of the prediction instead of the prediction itself. In this work, we considered a stimulus-response (SR) based LCS system where an immediate reward or punishment is provided at each computational time step by the external environment [4]. For such a system, there is no need for a complex credit assignment algorithm like the bucket brigade [5] and the message list in Holland's formulation of LCS is also omitted.

In a strength based LCS, the past performance of a classifier is measured by the amount of its current strength [6]. Strength is used both as a means of resolving conflicts and as a fitness for the Genetic Algorithm (GA). In our previous work, we introduced a modified bidding strategy in LCSs by allowing classifiers to get a loan from a central loaning agent during auctions [7]. The loaning approach followed was centralized in the sense that there is only one central bank issuing the loan. A bid history variable that gives classifiers a clue on the potential of competent classifiers was also introduced. A more compact, less complex and more realistic distributed loaning approach where loaning occurs among classifiers in the system was also suggested for an improved performance [8]. The generalization capability of LCS by means of using hash symbols gives it the potential to develop a compact representation of the concepts learned.

The remaining part of the paper is organized as follows: Section 2 discusses LCS in brief. Section 3 presents default hierarchy, the challenges and survey of the art. Section 4 details the system formulation followed in this paper. The fitness proportionate reward sharing scheme is also presented in this section. Section 5 summarizes the learning cycle for LCS. Section 6 discusses the simulation set up and results obtained while the last section concludes the paper by highlighting the achievements obtained in this work.

2. LCS Overview

An LCS is a machine learning system based on reinforcement learning and genetic algorithms. Like an expert system, it utilizes a knowledge base of syntactically simple production rules that can be manipulated by a genetic algorithm [9]. The use of a rule-based system allows an LCS to conveniently represent and refine complex control strategies [10]. The robust search ability of the GA enables effective discovery of new rules on the basis of performance only feedback. The reinforcement learning technique determines the rule fitness and enables

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454.

the system to learn from its environment based on a reward signal that implies the quality of its action.

There are two major types of LCS: Michigan and Pittsburgh style LCSs. In Pittsburgh formulation of LCS, individuals in a population are complete solutions to the problem [11]. An individual is a rule set and the length of a rule is fixed while the number of rules in one rule set varies. Individuals in the population compete among themselves to correctly classify the training samples. The working principle of Pittsburgh LCSs is essentially similar to GA. The fitness of an individual in the population is measured by the classification accuracy of the rule set. But in Michigan style LCS, individuals are rules and the solution to the problem is the whole population. An individual rule covers part of the solution and coordination among rules and a mechanism to evaluate the performance of rules in the form of reward or punishment is essential. The use of the term LCS in this work adheres to Michigan style LCS.

The standard LCS consists of three major components: the message and rule system, the apportionment of credit and the discovery component [1, 12]. The apportionment of credit subsystem in an LCS addresses the issue of credit assignment which serves as a measure of the classifier's performance. It is based on an economic analogy where a classifier garners credit in the form of strength (a kind of capital). It involves a bid competition among classifiers that match to the current environmental input. Accordingly matched classifiers bid a certain proportion of their strength and rule conflicts are resolved based on a probability distribution over the bids [13,14]. Also, to promote the exploration of the classifier space, a random noise is added to the deterministic bid [15].

3. Default Hierarchy

In a Michigan style LCS, an individual classifier in the population represents part of a solution to a given problem. There is no single rule that adequately models the environment. Instead, an accurate modeling of the working environment requires coordination among sets of rules in the population. Consider a learning system that tries to model an environment with huge number of states. There is no rule of thumb to determine the optimum number of rules that sufficiently model the environment. An LCS that is to operate in such an environment can be modeled in either of two ways. The first is to build a model of the environment using a set of rules that never make mistakes. This homomorphic approach, however, is practically unfeasible as it requires a vast number of rules to model realistic environments [16]. Besides, an environment exhibiting perpetual novelty combined with a limited sampling of it adds another order of complexity to this homomorphic approach [12, 17]. The other alternative is to build a hierarchical model where the task of the learning system is to categorize the states into groups that can be treated in a similar way [13, 16]. A hierarchical rule set provides a multi-level structure in which rules at the bottom of the hierarchy are very general and those at the top are very specific (see Figure 1). Hierarchy can occur at any level within the rule sets. Here, the term default hierarchy refers to a hierarchical set of rules that contain a default rule for the default class along with other exception rules.

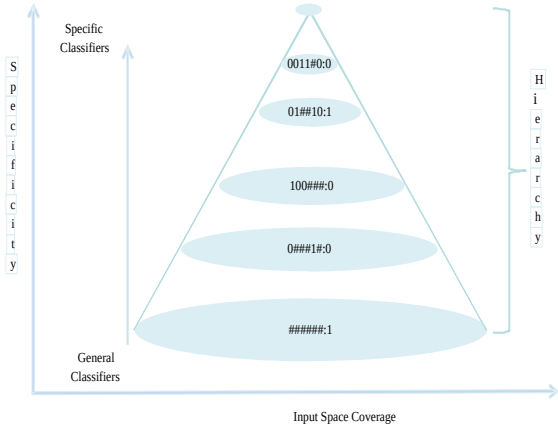


Figure 1-A hierarchical structure in LCS

Consider the 6-bit Boolean multiplexer (6-mux) problem whose disjunctive normal form is as follows:

$$Y = \bar{s}_1 \cdot \bar{s}_2 \cdot a_1 + s_1 \cdot \bar{s}_2 \cdot a_2 + \bar{s}_1 \cdot s_2 \cdot a_3 + s_1 \cdot s_2 \cdot a_4 \qquad (1)$$

Where $a_1 \ldots a_4$ are the address lines, s_1 and s_2 are the select inputs and Y is the output. The system's decision is correct when its output value is the same as the value of Y in equation 1 for a given input.

Non hierarchical set	Hierarchical set
000###:0	000###:0
001###:1	01#0##:0
01#0##:0	10##0#:0
01#1##:1	11###0:0
10##0#:0	######:1
10##1#:1	
11###0:0	
11###1:1	

Figure 2- Default Hierarchy for 6-mux problem

The 8 rules in the non-hierarchical set (the left column in figure 2) are the perfect solutions to the 6-mux problem. With default hierarchy, the same problem can be solved with a more compact hierarchical rule set (right column in figure 2). The last rule in the hierarchical set is a default rule. The default rule matches to all inputs but it makes a correct decision only 50% of the time. A working default hierarchy provides a great parsimony of the required rules to model the environment. In addition, the system's performance can be improved by adding more exception

rules to the hierarchy. The essence behind achieving a working default hierarchy is therefore to build a more compact rule set with a reasonably fair accuracy as compared to the homomorphic model. This requires the coexistence of exception and default rules in the system so that the exception rules provide protection to the default when it is wrong.

3.1 Starvation versus Protection

The starvation-protection dilemma has been a bottleneck to the research in attaining a working default hierarchy. The objective here is to protect the default rule from firing when it is wrong without starving it. This requires a bidding strategy that favors the exception when both match to a given input. In literatures [13, 14, 16, 18], a bid amount proportional to the specificity is proposed. In this kind of bidding strategy, exception classifiers bid a higher amount as compared to more general classifiers in the system. The shortcoming of this type of bidding is the consequent starvation of the default classifier when it is right. Protection is usually associated with an immediate starvation of the default.

To solve the starvation problem, Wilson, in his detailed experiment using Boole, proposed key modifications to the standard LCS formulation [19]. One modification to reduce starvation is extending system decision to the match set instead of limiting it to the active set. In Holland's formulation of LCS, only classifiers in the active set make decision on the system and any reward by the external environment goes to the active set. Extending the decision making and reinforcement to a rather bigger set of classifiers (match set) improved the systems performance [19]. But under the same bidding and paying policy as proposed in Smith [13] and Riolo [16], avoiding the active set might lead to rampant overgeneralization. To overcome this problem, Wilson suggested a different bidding and paying policy in which specificity is retained in the bid calculation but eliminated when calculating the classifier's payout [18, 19]. The bid amount is scaled by the specificity while the actual pay out of the classifier depends entirely on its strength and the bid coefficient.

3.2 Reward Sharing

The environment provides a reinforcement signal to the learning system when it responds correctly. In a stimulus response LCS, the system takes action on its environment directly and receives an immediate reward or punishment as a consequence of its action.

There are two major notions on how to distribute the reward among classifiers in the active set. Holland [1] first suggested that all classifiers in the active set should receive a constant reward R and pay out their bid. This notion of sharing however does not lead to the formation of default hierarchy as it does not distinguish between correct and wrong classifiers. The limitation of this kind of reward sharing on the formation of default hierarchy can be explained this way. Consider for instance a scenario where a default classifier of action zero exists in the system. Assume also that its strength is high enough to outbid other specific classifiers in the match set and join the active set. This classifier may or may not agree with the winner classifier's decision but is going to receive a reward from the environment either ways. This kind of indiscriminate rewarding leads to the emergence of sneaky classifiers that survives on the bear of other reward generating classifiers and results in a poor system performance.

The theory of equal reward sharing ignored the whole notion of competition in LCS. From the classifier's perspective, the intention is to build up its strength by garnering as much reward as possible from its environment. A good analogy here is a competitive market economy where whoever strived hard should be rewarded and accumulates wealth. Wealth in classifiers is measured in terms of strength. Hence instead of using equal resource sharing, the rewarding scheme must somehow be biased towards stronger classifiers. Strength is a measure of quality of a classifier and the rewarding scheme has to adjust the strength to reflect the classifier's overall usefulness to the system. The higher the strength, the more influential the classifier is on the system's decision. We applied a fitness proportionate reward sharing scheme where classifiers proposing the same action as the winner will get credit proportional to their strength. The proposed rewarding scheme resulted in a remarkable improvement in the performance of the system and produced a viable and robust default hierarchy.

4. System Formulation

4.1 Classifier Format

The classifier format is shown in Figure 3 and has 5 parameters: condition, action, strength, experience (Exp) and creation time (Ctime). The condition is a string from the ternary alphabet (0, 1 or #) and the action is binary (0 or 1). The hash symbol (#) in the condition is "don't care" and matches to any input. The experience and creation time parameters are added for better understanding of the learning process. Experience (Exp) indicates the participation of a classifier in decision making process (i.e. match set) and the creation time refers to the iteration time at which the classifier is created. It helps to investigate whether a hierarchy once evolved can be sustained for generations.

Condition	Action	Strength	Exp	Ctime

Figure 3 – Classifier format

4.2 Learning System

Learning in an LCS is an ongoing adaption to a partially known environment and not an optimization problem as in most reinforcement learning systems. The learning system includes the following major components: the auction, clearing house (CH), fitness proportionate reward sharing (FPRS) and the GA.

4.2.1 *Auction*

This is the part where classifiers in the match set participate in auctions by bidding a fixed proportion of their strength. The bid amount depends on the value of its current strength and the specificity. The deterministic potential bid (PB) of a classifier *i* during auction is given in equation (2).

$$PB_i = C_{bid} * S_i * \underbrace{(1 - \frac{NH}{CL})}_{Specificity} \qquad (2)$$

Where *NH* is the number of hashes in the condition string, C_{bid} is the bid constant (see Table 1) and *CL* is the condition length. The specificity parameter is the ratio of the number of non hash symbols to the condition length.

The deterministic bid is not used directly to determine the auction winner. Instead, it is slightly perturbed by adding a random noise to promote exploration of the classifier space.The effective bid (EB) is computed by adding a random noise to the bids submitted by each competing classifier using equation (3).

$$EB_i = PB_i(1 + rand() * EBID) \qquad (3)$$

Where EBID is a constant used during simulations (see Table 1).

4.2.2 Fitness Proportionate Reward Sharing (FPRS)

The learning system continuously interacts with its environment through its detectors and effectors. It uses a feedback about the effect on the environment to learn from experience. The learning agent is blind without a proper guidance by a reward signal. A trainer is therefore necessary to determine whether the environmental modification was beneficial or detrimental.

The reinforcement program (RP) determines the rule's fitness by generating a signal in the form of a reward or punishment. If the whole learning system is a water fall, the RP is the pipe that guides it to a point of interest. This work introduced a novel fitness proportionate reward sharing scheme given in equation (4), for the formation of a viable default hierarchy.

$$r_i(t) = R * \frac{S_i(t)}{\sum_{k=1}^{M(t)} S_k(t)} \qquad (4)$$

Where *R* is the total reward provided by the environment whose value is initialized once, *M(t)* is the number of classifiers in the advocate list at iteration *t*, $r_i(t)$ is the fraction of the total reward (*R*) that goes to classifier *i* at iteration *t* and $S_i(t)$ is the strength of classifier *i* at iteration *t*. The constant reward provided by the external environment is shared proportionally among classifiers in the advocate list.

4.2.3 Clearing House (CH)

The CH is the part of the learning system that deals with the modifications in strength of classifiers as the classifier system learns. All classifiers pay existence tax and classifiers in the match set pay an additional overhead tax while classifiers in the advocate list has to pay also the bid amount. Assuming correct decision is taken by the system at iteration *t*, the strength of a classifier *i* in the advocate list at the next iteration is governed by equation (5).

$$S_i(t+1) = S_i(t) * (1 - C_{ext} - C_{oh} - C_{bid}) + r_i(t) \qquad (5)$$

Where C_{ext} and C_{oh} are the existence and overhead tax constants respectively, C_{bid} is the bid constant, and $S_i(t)$ and $r_i(t)$ are the strength and reward for classifier *i* at iteration time *t*. For the same reason mentioned earlier, the pay out of a classifier in equation (5) is different from the bid amount given in equation (2).

4.2.4 Genetic Algorithm (GA)

The GA discovers new rules among a population of candidate rules based on the experience of existing rules. Each GA operation brings two new classifiers to the existing population of classifiers. It diversifies the population using mutation and cross over operators. A roulette wheel selection method is used to select parents for reproduction. The strength of new classifiers emerging from GA is initialized to a value that is neither too high (so that they do not dominate experienced classifiers) nor too low (to make them competent with the relatively more experienced classifiers in the system during auctions).

Table1: List of simulation parameters with their optimum values.

Parameter	Value	Meaning
Pop size	200:6-mux 400:11-mux	Number of classifiers
Cexs	0.001	Existence tax
Coh	0.005	Over head tax
Cbid	0.1	Bid coefficient
Px	0.35	Probability of crossover
Pm	0.008	Probability of mutation
EBID	0.1	Ebid constant

5. Results and Discussion

For the sake of comparing results with previous research work, the proposed algorithm is applied to the 6 & 11-multiplexer problems. Classifiers are initialized to an initial strength of 100. The payoff for a correct decision (R) by the system was set to 1000 while absence of a reward was considered as punishment for a wrong response. The specificity value for the default classifier is set to 0.1 for 6-mux and 0.05 for 11-mux. The values for the other parameters are given in Table 1.

The performance of the system is measured by the accuracy of its response to a given input. Figures 4 and 5 show the percentage of correctly identified environmental inputs by the system and the solution count as a function of the number of epochs. For the 6-mux problem an epoch stands for one complete presentation of the environmental inputs to the system. So an epoch represents the average system response on the past 64 inputs. But for the 11-mux, an epoch represented only 512 (25% of the inputs) iterations or input presentations. In general, for an *n*-bit string, there are a total of 2^n different environmental inputs to the system. Simulation is carried out 20 times and as can be seen from Figures 4 and 5, the percentage accuracy of the system averaged over the 20 runs is well over 95% after the 100th epoch for the 6-mux and 250th epoch for the 11-mux. The solution count is the percentage of population that contains instances of the perfect solution set (refer to Figure 2) averaged over the size of an epoch. For instance, 90% of the population contains instances of the perfect solution at the 350th epoch (see Figure 4). The high percentage accuracy and solution count achieved is an indication of how well the system learns its environment. The effect of varying the mutation and cross over rates is also investigated and best results are obtained for the values given in Table 1.

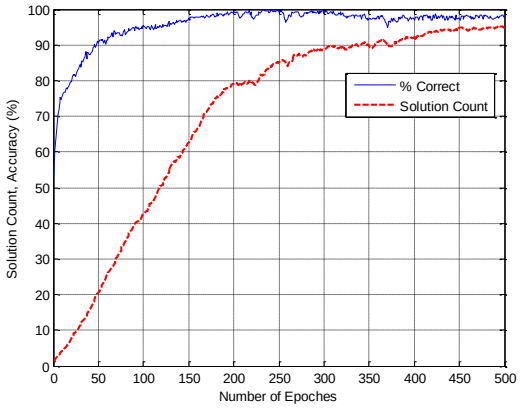


Figure 4- Simulation result for the 6-mux problem averaged over 20 runs. The upper curve represent the percentage of correct decision by the system, the lower curve is the percentage of the population that contains the perfect solution set.

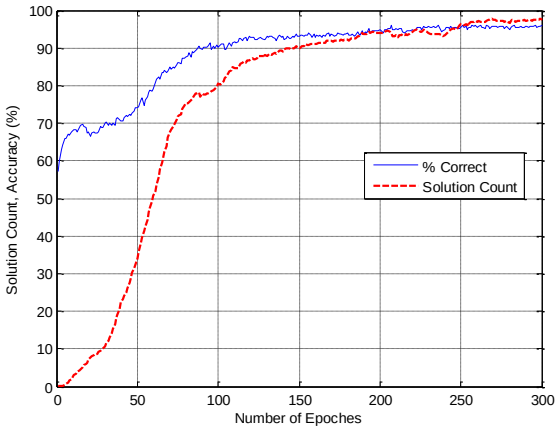


Figure 5-Simulation result for the 11-mux problem averaged over 20 runs. The upper curve represents the percentage of correct decision by the system; the lower curve is the percentage of the population that contains the perfect solution set.

Figures 6 and 7 display the average bid amount of winner classifiers for the 20 runs as iteration goes on for 6 and 11 multiplexer problems respectively. The bid interaction helps to get an insight on how the fitness of the population varies with time. It gives a qualitative clue on the steady state strength of classifiers that influence the system's decision. At the start of the iteration, the population is more likely to be packed with specific classifiers. But as time goes on, the hierarchical set dominates the population resulting in a decline of the bid amount as the bid amount is dependent on the specificity. This trend is clearly evident from the plots in Figure 5 having a high peak at the start and declining abruptly until it finally settles to some steady state value.

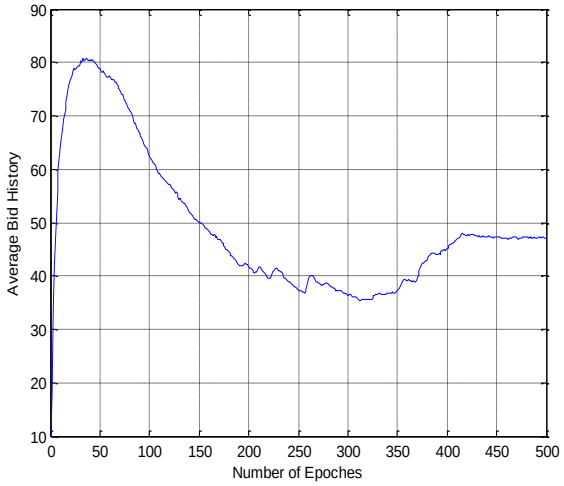


Figure 6- The average bid amount of winner classifiers at each epoch for 6-mux, averaged over a total of 20 runs.

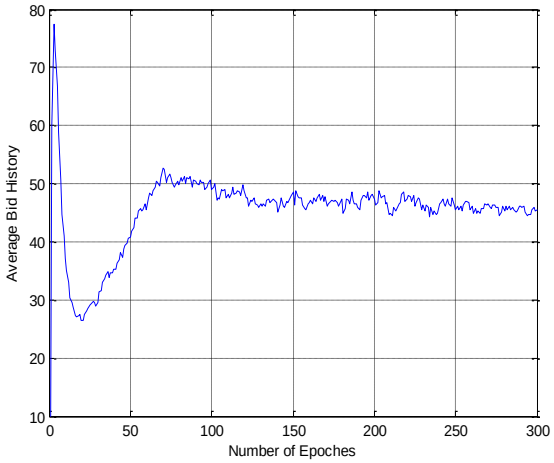


Figure 7-The average bid amount of winner classifiers at each epoch for 11-mux, averaged over a total of 20 runs.

To check whether the system has evolved to a default hierarchy and its ability to sustain it once formed, a tabular result showing a sorted list of the top nine classifiers in the population with their creation time and numerosity is displayed in Tables 2 and 3. A perfect default hierarchy has been achieved in 19 (9 of them with a default of action 0 and the remaining 10 with a default of action 1) out of a total of 20 runs. In one of the runs, a default hierarchy is also attained but one of the 4 perfect solutions is missed. Table 2 shows a scenario where a default of action 0 and the 4 perfect solutions of action 1 dominating the total population. As can be seen from the number of copies column, these classifiers comprises 96% of the population (192 instances out of a total population of 200 classifiers). Table 3 displays the same statistics for a default of action 1 and the other 4 perfect solutions of action 0 case. Again, these hierarchical set contains 193 instances of the total population. The creation time (Ctime column in the tables) gives an insight on the time of emergence of a hierarchical set and whether the learning system was able to maintain it. It is measured in terms of iteration, not epoch. In one iteration, a single input is given to the system and its response to it is evaluated. Hence, 500 epochs is tantamount to a total of 32000 (i.e. 500*64) iterations. The time of formation of the default hierarchy can be inferred by looking at the creation times of individual classifiers in the hierarchical set. In table 2 for instance, considering the top 5 classifiers that comprise a default hierarchy, the highest creation time value is 4330 (nearly at the 9th epoch), which means that the latest classifier that joined the hierarchical set is "11###1/1".

Table 3 shows similar results for the 11-mux problem displaying the top 10 classifiers in the final population. As can be seen from the table, the hierarchical set comprises of 98% of the population (i.e. 392 instances out of a population size of 400). From the creation time, it can also be seen that the hierarchy once formed in the learning process is maintained for generations. For instance, observing the top 9 classifiers that comprise a hierarchical set in Table 3, the default hierarchy was achieved at the 42000th iteration. A creation time of 0 indicates that particular classifier was part of the initial population.

Table 2-A sample pattern of the final population for 6-mux with a default of action 0.

Condition	Action	Str (Max)	Str (Avg)	Num. Copies	Ctime
001###	1	2959	274	29	5
10##1#	1	1935	241	32	3795
11###1	1	1310	244	33	4330
01#1##	1	927	242	33	2460
######	0	471	119	65	1315
###0##	0	193	193	1	23600
#####0	0	184	184	1	3695
##0###	0	175	175	1	18655
#0##1#	1	115	116	1	24475

Table 3-A sample pattern of the final population for 11-mux with default action of 1.

Condition	Act	Str(max)	Copies	Ctime
111#######0	0	2892	38	31990
001#0######	0	2739	34	42000
101#####0##	0	2698	32	31220
011###0####	0	2246	37	24080
110######0#	0	1913	45	17390
0000#######	0	955	44	26790
100####0###	0	790	48	12490
010##0#####	0	635	48	7750
###########	1	261	66	19990
########0##	0	223	1	137630

6. Conclusion

The learning task in an LCS is to build a set of rules that work in coordination to accurately model a given environment. Depending on the complexity of the working environment, adequate modeling of the environment might require a huge number of rules that collectively give a better model of the environment. Building a hierarchical set of rules, where accurate and more specific rules respond to a subset of the situations covered by more general but less accurate default rules will be vital to achieve a compact rule set size, especially when dealing with an environment that has huge numbers of states. This requires the co-existence of exception and default rules in the system so that the exception rules can protect the default rule from

making mistakes without starving it. To the best of our knowledge, the techniques proposed in literatures so far have failed to provide useful protection without a subsequent starvation of the default. This work introduced a novel reward sharing technique based on a competitive market analogy that leads to the formation of a viable default hierarchy. The technique introduced here has led to the evolution of cooperative classifiers by maintaining diversity in the population via a fitness proportionate implicit niching. Classifiers with actions that agree with the system's decision share a reward from the environment in proportion to their current strength. The results obtained for all the simulations proved the effectiveness of the proposed reward sharing technique.

7. References

[1] Holland, J. "Adaptive algorithms for discovering and using general patterns in growing knowledge bases", International Journal of Policy Analysis and Information Systems, vol. 4, no. 3, pp. 245-268, 1980.

[2] Wilson, S., "ZCS: A zeroth level classifier system," Evolutionary Computation, vol. 2, no. 1, pp. 1.18, 1994.

[3] Wilson, S., "Classifier fitness based on accuracy," Evolutionary Computation, vol. 3, no. 2, pp. 149.175, 1995.

[4] Butz, M. V. et al, "Toward a Theory of Generalization and Learning in XCS," IEEE Transactions on evolutionary computation, Vol. 8, February 2004.

[5] Holland, J., "Properties of the Bucket Brigade", Proceedings of an International Conference on Genetic Algorithms and their Applications, 1-7, John J. Grefenstette (Ed), Carnegie-Mellon University, Pittsburg, 1985.

[6] Kovacs, T., "Strength or Accuracy: Credit Assignment in Learning Classifiers Systems", Springer-Verlag, Berlin 2003.

[7] Workineh, A. and Homaifar, A., "Robust Bidding in Learning Classifier Systems Using Loan and Bid History", Journal of Complex Systems, Vol. 19, Issue 3, pps. 287-303, 2011.

[8] Workineh, A. and Homaifar, A., "A New Bidding Strategy in LCS using a Decentralized Loaning and Bid History", IEEE Aerospace Conference, pp. 1-8, Big Sky, Montana, March 03-12, 2012 (in press).

[9] Goldberg, D.E, "Genetic algorithms in search, optimization, and machine learning," Reading, MA: Addison-Wesley, 1989.

[10] Wilson, S. and Goldberg, D.E, "A critical review of classifier systems", in Proceedings of the Third International Conference on Genetic Algorithms and Their Applications, Morgan Kaufmann, 1989, pp. 244-255.

[11] Smith, S. F., "A learning system based on genetic adaptive algorithms," Unpublished Doctoral Dissertation, University of Pittsburgh, 1980.

[12] Booker, L.B. et al, "Classifier Systems and Genetic Algorithms", Artificial Intelligence, Elsevier Science Publishers, pps. 235-282, 1989.

[13] Smith, R.E. and Goldberg, D.E., "Reinforcement learning with classifier systems," in IEEE proceedings of AI Simulation and Planning in high autonomy systems, March 1990, pp.184-192.

[14] Smith, R. and Goldberg, D.E, "Adaptive Default Hierarchy Formation", Applied Artificial Intelligence, Volume 6, Issue 1, 1992.

[15] Homaifar, A. et al, "Boolean Function Learning with A Classifier System", Proceedings of the Applications of Artificial Intelligence VI at the International Society of Optical Engineering and the Computer Society of the IEEE, Orlando, FL, April 1988, pp.264-272.

[16] Riolo, R.L., "Bucket Brigade Performance II: Default Hierarchies", Genetic Algorithms and teir applications: Proceedings of the second internation conference on genetic algortithms, pps. 196-201, MIT, Cambridge, MA, July 28-31, 1987.

[17] Booker, L. B., "Intelligent behavior as an adaptation to the task environment," Dissertations Abstracts International, University Microfilms No.8214966, vol. 43(2), p. 4G9B, 1982.

[18] Holland, J. et al, "What is a Learning Classifier System?,"LCS'99, LNAI 1813, pp. 3-32, Springer-Verlag, Berlin Heidelberg, 2000.

[19] Wilson, S., "Bid Competition and Specificity Reconsidered", journal of Complex Systems, Issue 2, pps. 705-723, 1989.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Evolutionary Refinement of Trading Algorithms for Dividend Stocks

Robert E. Marmelstein, Bryan P. Balch, Scott R. Campion, Michael J. Foss, Mary G. Devito
Department of Computer Science,
East Stroudsburg University
East Stroudsburg, Pennsylvania, USA

***Abstract* - This paper describes a Stock Strategist application which can mine, test, validate and refine complex equity trading strategies based on historical financial data. We address the capabilities and operation of the program and detail the currently implemented strategy to trade stocks that pay high dividend yields. Additionally we will present preliminary results which show the ability of the system to automatically refine a specific trading strategy using genetic algorithms. The methodology of the test will be discussed and the results of two variant genetic algorithm approaches will be compared and contrasted.**

KEY WORDS Data Mining, Simulation, Algorithmic Trading, Genetic Algorithms.

I. INTRODUCTION

This paper describes a Stock Strategist application that is designed to evaluate and refine specific stock trading strategies. There is definitely a very large body of academic work on the application of data mining techniques to discover and validate patterns in the stock market [1]. For this project, we decided to pursue a strategy focused on trading strategies for stocks that pay substantial dividends due to their predictable behavior both before and after the Ex-Dividend (Ex-Div) date. Further, we restricted our stock pool to those stocks which historically have a high dividend yield (6% or more).

A dividend is a payout of profits to owners (shareholders) of a company. The dividend is some percentage of the stock price, typically anywhere from 1% to 12% annually. Most companies that pay dividends do so quarterly. Unlike like certificates of deposits or bonds (where earned interest is proportional to holding time), companies will pay the entire dividend to whoever holds the stock on the Ex-Div date. As a result, the price of a stock tends to rise in the lead up to a dividend payment; often the price will rise by significantly more than the amount of the dividend payment itself. After the Ex-Div date, the stock drops sharply, sometimes by much more than the dividend amount. In these cases, the stock price may recover quickly from the typical Ex-Div drop.

Figure 1 illustrates this pattern for the stock of Vector Group Limited (symbol: VGR). There are two basic strategies for trading the dividend event. The first is riding the wave on the way up and selling at, or near, the predicted peak. The other is buying right after the Ex-Div drop in anticipation of a quick price recovery.

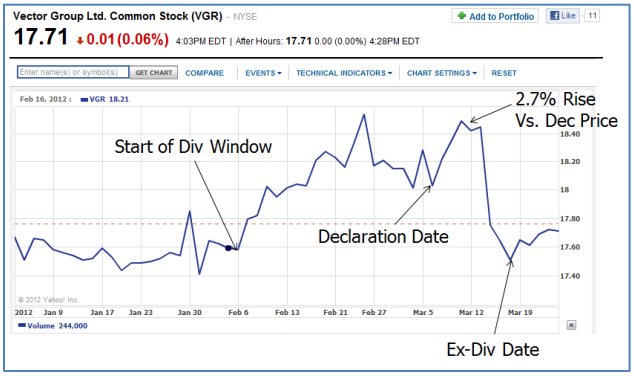


Figure 1. Example Dividend Trading Period

II. SYSTEM DESCRIPTION

The following paragraphs describe how the Stock Strategist program operates for a given strategy. Strategies are implemented as a composite set of C# classes that conform to interface specifications that facilitate extensibility, as well as, support for multiple strategies and comparative analysis. A strategy can be compiled, linked, and executed if it is written according to an Interface specification.

A. Preliminaries

Before running the Stock Strategist, the user must specify the following information:

- Stock Pool – The set of stock symbols of interest to the current strategy. Only stocks in the pool can be purchased for the asset portfolio.

- Start/End Dates – The data range over which a simulation will be run. Typically, the end date is the current date.
- Starting Capital Balance – The initial capital available to purchase portfolio assets.
- Transaction size – The minimum size of a buy transaction. Larger sizes result in less diversity and more risk, while smaller sizes diversify more at the expense of using capital for less attractive positions.
- Commission amount – This is the brokerage fee associated with each buy and sell transaction.

mode which, in turn, makes use of functionality in the Opportunity Identification mode.

C. Backtesting and Decision Cycle

As stated above, the Backtesting mode evaluates how a given strategy would have worked over a selected time period in the past. To accomplish this, a decision cycle is applied to the portfolio on a weekly basis from the specified start date through the end date. The purpose of this cycle is to make buy/sell decisions based on the strategy. The success of the strategy is gauged by the amount of portfolio net profit achieved over the test period after subtracting out transaction fees.

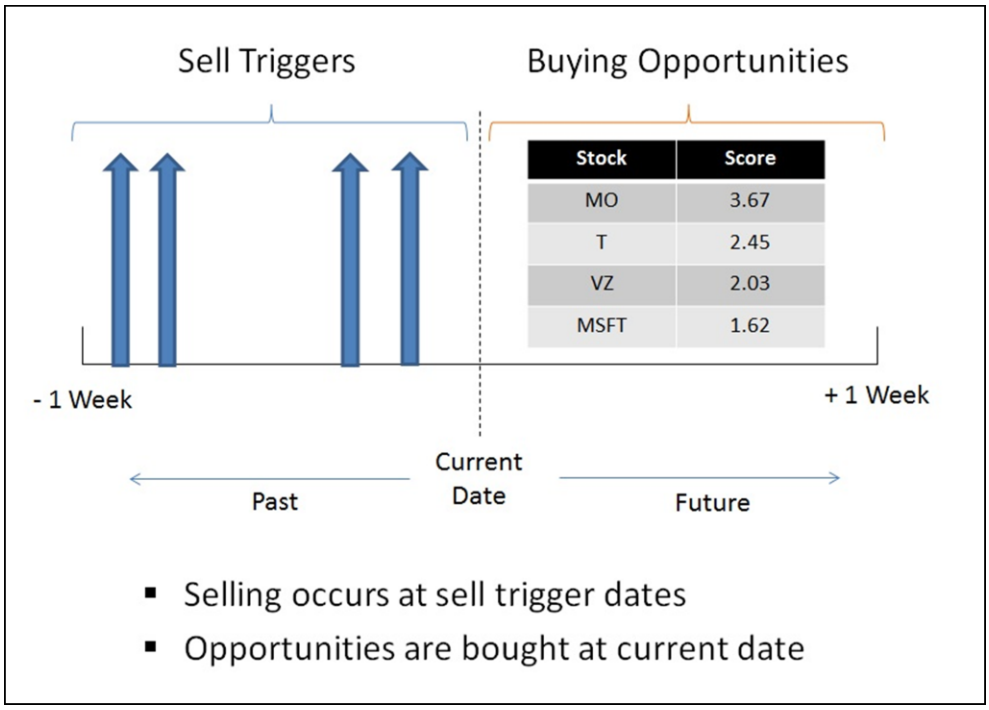


Figure 2. Weekly Decision Cycle for Backtesting

B. Modes

The Stock Strategist has four basic modes: Data Acquisition, Opportunity Identification, Backtesting, and Strategy Refinement. The Data Acquisition mode updates the database for those stocks in the stock pool. Data is incrementally extracted from several financial websites. The Opportunity Identification mode identifies opportunities for purchase within some time window from a specified date. Only stocks within the stock pool are eligible for this evaluation. The Backtesting mode evaluates the performance of a given strategy over a [past] data range. Lastly, the Strategy Refinement mode applies a search technique to improve the performance of a given strategy. This is typically done by optimizing the rule weights for scoring a given candidate stock. Note that these modes are not mutually exclusive. For example, the Strategy Refinement mode makes use of the Backtesting

Per Figure 2, the Stock Strategist looks back one week to determine what holdings in the current portfolio should have been sold based on calculated price targets. If these price targets are not met, the stock is ultimately "sold" the day prior to the Ex-Div using the opening price for that day. The gain (or loss) from these sales are added to the current capital balance. Potential buying opportunities for the next week are then evaluated, scored, and then sorted by score in descending order. Using the available capital balance, stock purchase orders are initiated (per the specified transaction size) by descending order of score until the available balance is below the minimum transaction. Any remaining balance is then used to increase the transaction size of the order with the highest scored stock. This approach ensures that the available capital is always fully utilized and limits exposure to individual stocks by having a diverse set of positions. The

detailed steps in the weekly decision cycle are shown below in Figure 3.

1. Identify candidates for purchase – These are based on known or predicted trigger events in the strategy. Example of trigger events include: dividend declarations, earnings reports and stock splits.
2. Score candidates for purchase – The specific scoring criteria is strategy dependent.
3. Identify current assets for sale – These are strategy dependent but can be based on stop-loss or date triggers, percentage gains, and relative value compared to candidate buy positions.
4. Unload existing assets – Assets are sold once their sell trigger conditions are met. Again, these conditions are strategy dependent.
5. Buy new assets (based on candidate score).
6. Compute portfolio value.
7. Advance simulation time – Advance by one week. Exit loop if past specified end date.
8. Remove expired [buy-related] events.

Figure 3. Decision Cycle Steps

III. System Components

The following paragraphs describe the major components of the Stock Strategist system.

A. Automated Data Acquisition

This component automatically extracts ("scrapes") stock data from a number of investment-oriented websites. From a web programming standpoint, extraction of this data requires a two-step process. Like many contemporary websites, the sites above utilize AJAX to produce dynamic content. As such, attempting to extract information requires reviewing the communication data flow that results in the final displayed page content. This can be achieved using developer browser extensions. When a desired data element of the displayed page is associated with a web resource, the URL of the resource is noted. In most cases, the URL has to be constructed dynamically from an intermediate stock code embedded in the target website's HTML. Once the URL has been constructed and data retrieved, a flexible HTML parser is used to transform the potentially malformed HTML into XHTML which allows the usage of XPATH queries to surgically extract the desired data. Post extraction, the data is then validated and the updates are saved in the database. To avoid unnecessary website bandwidth and processing time, the data acquisition process only performs incremental updates. Following a successful update, the stock is updated with a timestamp for the property that was retrieved, such as quote information or quarterly data. This allows the data acquisition process to operate as a system service or start-up routine for the application to maintain an up to date view of the market.

B. Common Framework & Database Layer

This component contains the classes for each major type of data object (e.g., real-time quote, stock, daily quote, quarterly data, etc.). It also provides the code for associated database stores and queries of these objects. Interactions with the database are performed through a common database provider, which currently targets MySQL. Other databases could be used with minimal code changes such as SQLite.

C. Asset (Portfolio) Manager

The Asset Manager component contains all functionality for maintaining the list of currently held assets. Each of the assets must be one of the stocks listed in the stock pool. During a given decision cycle, the Asset Manager also tracks the various buy and sell decisions made. At the end of each decision cycle, the entire value of the portfolio (cash and currently held assets) is computed. The Asset Manager has a number of wrappers for the identification and scoring of buy or sell candidates. These wrappers rely on method calls to the Strategy component class to accomplish their primary functions.

D. Analytics

This component computes technical indicators (features) and related statistics based on the historical stock performance. While these are independent of a given strategy, they are generated for input to the Strategy component. Analytics output fall into two general categories: features snapshot or statistical distribution. The snapshot is a feature's value at a given point in time. With regard to the statistical distribution, the mean or standard deviation of the distribution is reported. For example, the Price to Earnings (PE) ratio of a stock may be the current PE ratio (snapshot) or the mean PE ratio (of a Normal distribution) over a specified time period.

Another key feature of the Analytics component is to predict stock-related events based on their past occurrence. For example, a strategy may require prediction of a date range when the declaration of a dividend will occur. Since dividends are paid quarterly, it becomes necessary to first cluster these events into quarters (using the K-Means method [2]) to determine the likely time window for this event in any future quarter. *The capability to predict future events is critical to the Strategy component and distinguishes the Stock Strategist from most commercially available trading platforms.*

Once these events are predicted, the Strategy component can set up triggers based on these predictions. In most cases, an event that is (first) predicted is later confirmed by the data. These transitions are discovered as a result of the automated data acquisition process. At this point, the software changes the status of the event from

predicted to known. When this happens, the event may be treated differently by the logic in the Strategy component.

E. Strategy

The Strategy component implements the logic behind the buy/sell decisions of the application. This component determines trigger conditions (for buying or selling) and scores candidate stocks for purchase. In general, Strategy component decisions are based on a series of distinct rules native to each strategy. The output of these rules are categorized, weighted, and summed to compute a single score for decision making purposes. The buy signal computation is shown below in Equation 1.

$$Buy\ Signal\ (stk) = \sum_{i=1}^{n} rule_i\ (stk) * w_i \qquad (1)$$

As previously stated, these rules are supported by selected, lower level snapshot features and statistics available from the Analytics component. For example, one of the rules computes the ratio of the current change in price (after the dividend declaration date) to the historic (mean) change in price after that event. Once computed, the goodness of that ratio is categorized (good, acceptable, or poor) and a weight is applied to that rule. The Strategy component implements a standardized set of classes and methods. This design makes it straightforward to plug different equity trading strategies into the Stock Strategist framework.

F. Simulation

The Simulation component is an essential capability for backtesting a given strategy. It runs the decision cycle once per week over a given date range. After each cycle, the simulation time is advanced forward by one week. This component also logs the weekly portfolio value and all asset buy/sell transactions that occur over the simulated time period. The most critical ingredient of the Simulation is having sufficient and accurate data for a given date range. As previously noted, this task is accomplished by the Automated Data Acquisition component. The online sources we rely on only provide summary data for any given data; intra-day data is only available in real-time and is not stored in the database. The practical impact is that the simulation rarely utilizes real-time current data.

G. Search

The purpose of the Search component is to refine the current strategy in a way that increases (maximizes) portfolio return. The Search is performed with respect to key parameters for a specific strategy. For the most part, these parameters correspond to the strategy rule weights discussed in Section 4E. However, they can also include other parameters, such as the minimum Transaction Size and the size of the time window used for predictions. To conduct the search, these parameters are then passed as input to a given simulation instance. The simulation is then executed and the ending portfolio balance is passed back to the search routine. The fitness of the parameter set is computed simply as the ratio of the ending to starting balances (see Equation 2); the higher the ratio, the greater the fitness.

$$Fitness = \frac{Portfolio\ Value\ (\$) - Starting}{Portfolion\ Value\ (\$) - Ending} \qquad (2)$$

Although the parameter set optimized by the Search component is somewhat strategy dependent, it can be easily reconfigured. Currently, the search techniques utilized are Hill Climbing and evolutionary search using a Genetic Algorithm (GA). Unlike Hill Climbing algorithms, GAs can escape local optima traps [3]. While the GA is a highly effective technique for global search, the biggest impediment to its use is the amount of time to evaluate each generation, even with a small population. This is because each chromosome is a variation in trading strategy that takes several minutes to simulate. Thus, accelerating the Search component without degrading the fitness metric has been one of our top research priorities.

IV. Evolutionary Approach for Buy Signal Computation

Researchers have long sought automated "black box" techniques to identify and trade financial instruments of all varieties. Applying GAs to the evolution of trading strategies is not new—there is a solid body of work in this area. Allen and Karjalainen [4] were among the first to develop a GA-based system for finding technical trading rules. Becker [5] expanded on this using a Genetic Programming (GP) approach and an emphasis on monthly (vs. daily) trading. Schoreels [6] employed GAs to design agent-based systems for trading. Subramanian [7] developed a similar agent-based system, but with an emphasis on reducing trade risk and volatility. In contrast to the focus on trading rules, other researches including Yang [8], Lin [9], and Lai [10] utilized GAs for the selection and optimization of stock portfolios.

The key component of these systems is the signal to buy or sell a given security. The goal of our experiments was to evolve a set of weights (W) for each rule component to improve the performance of the buy signal for our dividend stock trading strategy. In this case, the size of the weight set was 12. Two sets of weights were evolved. The first set ($W_{0\text{-}1}$) had a range of (0, 1). These weights serve to select the rule components of the Buy signal as indicated in Equation 1. The second set ($W_{0\text{-}5}$) has a range of (0, 5). These can both select and amplify the contribution of a given rule to the overall Buy signal. As indicated earlier, the fitness of a given weight set is determined by the overall portfolio return.

In this experiment, each type of weight set was evolved over a simulated one year period for eleven periods (2001-2011). At the beginning of each one year training period, the Stock Strategist has an initial balance of $200K to trade with. After each training period, the best performing

set of evolved weights were evaluated on two sets test scenarios:

Portfolio Continuation (PC) – The evolved weights are used to continue run the simulation where the training left off. Thus, the trading continues over a follow-on trading period (1 and 6 months, respectively) with the portfolio and balance that remained at the end of the trading period. Given the dynamic nature of the modern financial markets, it was judged that six (6) months was the maximum amount of time to safely continue using a set of weights before they became stale.

Portfolio Restart (PR) – The evolved weights are used to start a new trading simulation in the follow-on trading period (one and six months, respectively). In this case, however, the strategist starts from scratch with a $200K balance and no pending transactions. Thus, any decisions made during training (beyond the evolved weights) are forgotten.

The GA library used for this experiment was part of the AForge.NET Framework [11]. A population of 30 chromosomes was utilized over 50 generations. Elitist selection was used with a crossover rate of 0.75 and a mutation rate of 0.01.

V. Results

Figures 4 and 5 show the results for the $W_{0\text{-}5}$ and $W_{0\text{-}1}$, respectively. These bar charts show the relative performance of each weight set vs. those of the default weight set (W_D). In W_D, the Buy signal is computed with each rule selected and given a unity weighting.

These results are noteworthy in two respects. The PC scenarios dramatically outperform PR scenario. In fact, the performance of the PR scenario is not much better than the default approach for computing the Buy signal. This indicates that the evolved rule weights are much better suited to running the existing portfolio versus starting over with a new portfolio using the same general strategy.

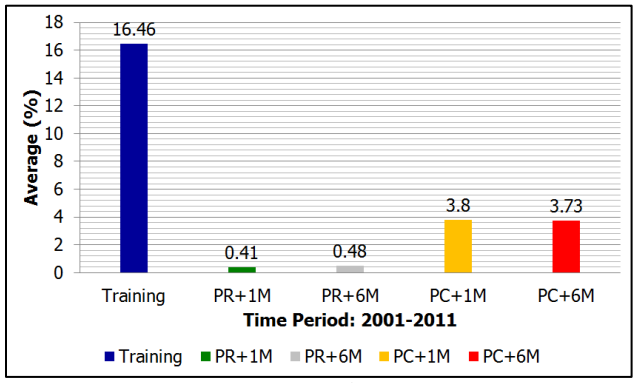


Figure 4. Return of W_{05} vs. W_D

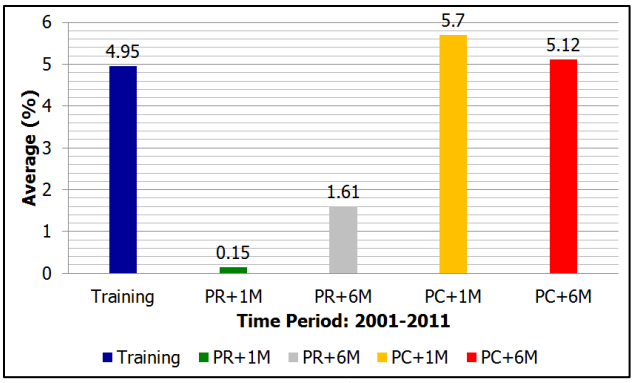


Figure 5. Return of W_{01} vs. W_D

The $W_{0\text{-}5}$ variant evolves a solution that performs much better during training than the $W_{0\text{-}1}$ variant. This is not at all unexpected since the $W_{0\text{-}5}$ variant offers a much bigger search space. Unfortunately, this advantage does not hold up during testing. Indeed, the $W_{0\text{-}1}$ variant is much more consistent than the $W_{0\text{-}1}$ variant's performance (training vs. test) on the PC cases. This is likely due to overlearning on the $W_{0\text{-}5}$ variant. Thus, while the $W_{0\text{-}1}$ variant is simpler, it is also more powerful due to its ability to better generalize its performance into the future.

VI. Conclusion

In this paper, we have introduced the Stock Strategist application. Initially, we are using this program to pursue a dividend stock trading strategy. We chose this strategy due to its relative simplicity, stability, and predictability vs. the myriad of other possible equity trading strategies. Our experiment in automatically refining our trading strategy yielded some interesting results in how to effectively structure GA-based training and apply the results. In particular, it indicates that simple, 0-1 rule selection is a better alternative to more complex weighting factors. The experiment also suggests that the evolved weights be used to continue the existing portfolio momentum, rather than restarting the portfolio. We have also experimented with techniques to accelerate the GA search and made substantial progress in this area (to be documented in an upcoming paper). Our future research seeks to expand the GA to cover additional algorithm parameters (such as the nominal size of a given stock buy) and strategies (such as using a limit strategy when initiating a stock purchase).

References

[1] E. Hajizadeh, H. D. Ardakani and J. Shahrabi, "Application of data mining techniques in stock markets: A survey," *Jornal of Economics and International Finance*, vol. 2, no. 7, pp. 109-118, 2010.

[2] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100-108, 1979.

[3] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm.," Carnegie Mellon University - Computer Science Dept, Pittsburg, PA, 1995.

[4] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial Economics*, vol. 51, pp. 245-271, 1999.

[5] L. A. Becker and M. Seshadri, "GP-evolved technical trading rules can outperform buy and hold," in *Sixth International Conference on Computational Intelligence*, Cary, NC, USA, 2003.

[6] C. Schoreels, B. Logan and J. M. Garibaldi, "Agent based genetic algorithm employing financial technical analysis for marking trading decisions using historical equity market data," in *Intelligent Agent Technology* , Beijing, China, 2004.

[7] H. Subramanian, S. Ramamoorthy, P. Stone and B. Kuipers, "Designing safe, profitable automated stock trading agents using evolutionary algorithms," in *GECCO*, Seattle, WA, USA, 2006.

[8] X. Yang, "Improving portfolio efficiency: A genetic algorithm approach," *Computational Economics*, vol. 28, no. 1, pp. 1-14, 2006.

[9] C.-M. Lin and M. Gen, "An effective decision-based genetic algorithm approach to multiobjective portfolio optimization problem," *Applied Mathematical Sciences*, vol. 1, no. 5, pp. 201-210, 2007.

[10] K. K. Lai, L. Yu, S. Wang and C. Zhou, "A double-stage genetici optimzation algorithm for portfolio selection.," in *Neural Information Processing - Lecture Notes in Computer Science*, Berlin Heidelburg, SpringerLink, 2006, pp. 928-937.

[11] AForgenet.com, "AForge.NET Framework - Genetic Algorithms Library," 23 Feburary 2012. [Online]. Available: www.aforgenet.com.

Template Personalization and Evolutionary Algorithms

H. ElGibreen[1], S. El-Masri[2]
[1] IT Department, King Saud University, College of Computer and Information Sciences, Riyadh, SA, hjibreen@ksu.edu.sa
[2] IS Department, King Saud University, College of Computer and Information Sciences, Riyadh, SA, selmasri@ksu.edu.sa

Abstract – *Due to the rapid and hectic growth of the Web, its access and design have become a challenge. Web personalization has occurred to solve this problem. However, such personalization mostly deals with the visitors rather than other types of website users. Alternatively, Web Content Management System (WCMS) has occurred to facilitate website development, including its design. Nevertheless, designing of such websites are only customized. Thus, website developers need to further choose and refine the appropriate design. Such process needs a lot of effort and wastes a lot of time. Consequently, this paper will present a new approach that extracts and personalize templates by combining web personalization and template extraction in order to automatically extract templates and, thereby, simplify WCMS and increase its scalability. In specific, to accomplish this approach a new technique is used, which mixes Genetic Algorithm, Ant Colony Clustering, and Cluster Tree Matching. This approach is tested with an experiment, which proved its quality, in regard of speed, precision, and accuracy.*

Keywords – Evolutionary Algorithm, Template Extraction, Web personalization, Web Content Management System

1 Introduction

Nowadays, the Web has emerged and become the most wanted method for information sharing and communication. In consequence, to such a rapid and hectic growth, Web access and effective design have become a challenge. Websites that lack in its information and structure make the visitors get lost and feel disoriented [1]. Consequently, web personalization has emerged to better fit the information access and design to the user's need [2]. Originally, it was used in advertisement and promotion personalization for different visitors. Currently, it focuses more on visitors to provide the appropriate information and service access and, thus, make the websites more useful [3]. However, up until now website developers are usually forgotten when it comes to personalization.

Alternatively, Web Content Management System (WCMS) concentrates more on website developers' need rather than the visitors. It is a tool that allows a variety of centralized and decentralized non-technical people to manage, edit, create, and control a large and dynamic collection of HTML content [4]. However, the available WCMS allows for customization of website design rather than personalizations, i.e. developers customize the web pages by themselves. Sometimes WCMS provides template samples but such samples do not automatically consider the developers' needs or the purpose of the website to be developed. In addition, it has been stated in [5] that high productivity of web pages can be accomplished by automatic web page generation using common template and data. Thus, designing effective web page templates is wearying mission [6]. Website developers must consider the feature of web pages and investigate what design structure would work better.

Accordingly, using personalization in WCMS will help both WCMS and website developers. WCMS developers would not have to provide a large repository of template and, thus, no need to study the templates' usability or update it every period of time. In addition, using personalization will reduce the website developers' time for checking similar websites and studying the visitors' need to choose which structure is more appropriate for them. As a result, instead of studying Human Computer Interaction strategies, repeatedly, and searching for usable page structure, using personalization will take advantage of others' experiences. Additionally, it would guarantee covering all possible cases of the dynamic demands of Web users who are excessively changing. To further motivate the use of web personalization in WCMS, it has been found that the spread of the World Wide Web and the increase of website development to match visitors' needs have increased the necessity for WCMS to tailor itself for different web pages design depending on the need.

Consequently, this paper presents a new approach which improves WCMS using web personalization, where WCMS can take advantage of previous developers' knowledge, to automatically offer different template designs for each website developer and, thus, increase the websites productivity and WCMSs scalability and flexibility, by merging web personalization with automatic template extraction. To accomplish such an approach a new technique has also been developed, which mixes hybrid Evolutionary Algorithm (Genetic Algorithm and Ant Colony Clustering) with Cluster Tree Matching.

This paper is organized as follows. First, background is explained and some related works are discussed. Then, the proposed approach is explained in details. Afterwards, an experiment is explained and its results are discussed. Finally, the paper is concluded and some future works are proposed.

2 Background

In this section, the main aspects that are needed to understand the proposed approach are defined and discussed.

2.1 Web Personalization

In [1] web personalization has been defined as "*any action that adapts the information or services provided by a web site to an individual user, or a set of users*". Thus, in personalization all the work is done automatically for the users based on their data. It uses different techniques to collect users' data and then personalize the services by measuring the similarity degree between the current data and other content or users of the website. In general, web personalization must first collect the web data to be pre-processed and modeled. Then, such data must be analyzed and matched to determine what action should be performed [7].

2.2 User Profiling

Discovering users' differences is essential to provide the required personalized service. User profile has been defined in [8] as *"the process of 'discovering' correlations between data in databases that can be used to identify and represent a human or nonhuman subject (individual or group) and/or the application of profiles (sets of correlated data) to individuate and represent a subject or to identify a subject as a member of a group or category"*. In general, user profiles vary in content and types, which usually depend on the application [8] [9]. In order to build and use profiles different techniques have been developed in the last few years, such as machine learning, genetic algorithm, and classification techniques [10].

2.3 HTML Web Pages

Hypertext Mark-up Language (HTML) has been defined in [10] as "*a web programming language used to display web pages*". Usually, browsers provide web pages based on HTML files, which define the structure and represent the pages as tags and attributes. HTML contains tags that are nested in a tree structure. Thus, different web pages can be generated by extracting some nodes of the HTML tree [11].

In order to navigate, modify, add, or delete content or elements of HTML web pages, Document Object Model (DOM) has been developed by W3C as HTML standard interface [12]. Specifically, HTML files are represented by DOM trees in order to apply different similarity measures and extract common templates [5]. Its representation aims to display and show the structure of HTML web pages rather than its content. Web pages are converted into DOM trees using its tag structure, which labels each node in the tree with its matching tag [13]. The hierarchy of the tags is preserved and the content of the page is stored at the leaf nodes.

To extract templates, HTML web page structure must be compared and, thus, DOM tree similarity must be measured. One of the most recent measures in that field is Cluster Tree Matching (CTM). It is developed by Ferrara and Baumgartner [14] to calculate the tree similarity recursively in a matrix and count the weight difference between two trees. CTM clusters the matching process sub-tree by sub-tree and assign a weighted value to give less weight for insignificant sub-trees; such that deep levels of the trees usually indicate an insignificant component of the web pages like table rows or list of items. This measure is used in the proposed approach and, thus, more details will be explained later.

2.4 Evolutionary Algorithms

Evolutionary Algorithm (EA) [15] is an artificial intelligent algorithm that has been inspired by nature. Such an algorithm is used to solve any kind of problem and identify candidate solutions as fast as possible. It is useful when the best solution is unknown while the data repository is very large. Different techniques have been developed under the umbrella of EA, such as Genetic Algorithm (GA) [16] and Ant Colony Clustering (ACC) [17]. Each technique has been tailored in different domains depending on the requirement.

GA is considered as the most popular technique of EA. It is *"a general purpose search algorithm which use principles inspired by natural genetic populations to evolve solutions to problems"* [16]. It typically starts from a collection of random solutions, called population, and then evolves it using selection and reproduction procedures. The best solution is chosen based on a fitness function that measures the solution strength. The population contains elements, called individuals or chromosomes, to represent possible solutions. In GA, four steps are performed to match individuals [18]. First it represents the available dataset to the required data type. Then it initializes the population by randomly assigning different values to the chromosomes. Afterward, it reproduces a new solution by mixing every chromosome with another to produce a new offspring. Finally, it selects the best chromosomes based on their fitness value.

Alternatively, ACC method is a new type of clustering that is based on an algorithm called Ant Colony Optimization. Such an algorithm is an Evolutionary Algorithm that is built based on how ants look for food by spreading their pheromone to find the food path [19]. It starts updating its local pheromone and explores all possible roads to finally emphasize the shortest one. As a result, the global pheromone will be emphasized, while the other pheromone is fading, i.e. the best road will be chosen. ACC is developed to optimize

the process of clustering and improve its time. It is mostly used in web usage mining [20] and basically divided into three steps [21]. First, it updates the local pheromone, i.e. update each ant solution with the suitable cluster. Then, applies the state transition rule, i.e. decide if a certain ant should be dropped in/out of a cluster. Finally, it updates the global pheromone, i.e. update the final solution (road map).

3 Related Work

After discussing the main aspects needed to understand the paper, it is time to discuss other works in that area. To the best of our knowledge, no one yet has considered personalization in template extraction. Thus, the available works in template extraction and personalization are separately discussed.

In [3], a new methodology that allows users to personalize their web page navigation through websites has been presented. They emerged content and usage mining techniques with the field of recommendation systems to suggest the most relevant pages to users. Similarly, in [2] web mining model that dynamically creates personalized web pages is presented; such that users with similar interest are grouped and their visited pages are related. On the other hand, in [22] a complete framework is presented along with web usage mining of real websites. It profiles users explicitly using search query that is stored in the web log data. For further improvement, Eirinaki et al. [1] proposed a semantic web personalization that is applied to semantically annotate websites' content. They integrated content semantic with usage information to improve the semantics of navigational patterns.

Alternatively, when it comes to template extraction, in [23] a new method has been proposed to parse different websites into HTML tag trees to generate a template for each site and identify repeated patterns and extract pages' content. Moreover, in [24] another method is presented to induce web page template. They first select websites randomly, and then transform it into DOM trees, judge its similarity, and cluster the web page to extract templates, correlate it, and extract other pages' content. Furthermore, in [25] an approach of three steps has been proposed. It generates templates, extract data, and then edit the template. Finally, Shui-Lung and Jane Yung-jen [6] introduced a Tree Template Automatic Generator to learn template from given web pages. The template is generated using a top down approach starting from the root and then going down one level at a time.

Regardless of all the evaluation conducted in the previous works, some important deficiencies have been noticed. When it comes to web personalization, all the works discussed have only considered the visitor behavior and the content similarity; i.e. the problem of navigation is only considered and the focus was on the website/visitor relationship while the website designers have been neglected. Consequently, even though it is called web personalization but the focus was mostly directed to navigation personalization and content similarity. Alternatively, when it comes to template extraction, the available works have directed the focus to the dynamic content of the websites rather than the template itself. Templates were generated only to identify the content of different websites without taking advantage of the semi-structured static[1] knowledge that already has been extracted. In addition, time of template extraction and its accuracy was not fully considered in most of the work. Therefore, the next section presents a new approach which is developed to increase the website's productivity and WCMS scalability and flexibility by personalizing template extraction and, thus, automating the process of website design.

4 WCMS Personalization

WCMS personalization is a new approach that personalizes the process of template extraction in WCMS. Such an approach uses a new technique that is based on hybrid EA (GA and ACC) and CTM measure. It mainly applies two steps: pre-personalization and personalization. In the pre-personalization step data are collected from developers then profiles are generated. However, in the personalization step profiles are matched, and common templates are extracted.

4.1 Pre-Personalization

In order to start the personalization process, data used to build the profiles must be first collected explicitly from the developer account. Such data would contain the developers' country, religion, and their business scale, and also the visitor age, language, gender, ability, and nature. In addition, after collecting these data, profiles are generated. Thus, websites' profiles are gathered in Extensible Markup Language (XML) files and all websites that target the same age are grouped together, as illustrated in Figure 1, in order to simplify and speed up the process of matching.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <UserProfile age="kid">
  - <User ID="123456">
      <country>Any</country>
      <religion>Any</religion>
      <scale>Individual</scale>
      <language>Arabic</language>
      <gender>Any</gender>
      <ability>autism</ability>
      <nature>medical</nature>
    </User>
  + <User ID="123459">
  + <User ID="123460">
</UserProfile>
```

Figure 1: Individual profile

It must be noted that changes in the collected data are usually not frequent, i.e. it is a long-term interest. Thus, every period of time this step is repeated, in the background, in order to consider the possible changes in the long-term interest.

[1] Semi-structured documents are documents that have a known content without knowing where such content is placed in the document. In addition, static data are the data that is statically written in the page without deriving it dynamically from the database.

4.2 Personalization

Whenever the developers access WCMS, in order to design the website interface, templates will be personalized by matching profiles and generating the required templates. This step is repeated every time the users want to redesign their website because users and visitors' interest might frequently change; thus, changing in the short-term interest is considered. The pseudo code of this step is illustrated in Figure 2.

```
Personalization {
Input: users profile U, target user T, population size: β, stop condition α,
chromosome size γ, HTML pages HP, Kp, Kd, Number of Cluster cβ.
Output: generated template.
// 1- profile matching (GA algorithm)
//Initialize population and calculate the fitness of each solution
P = GeneratePopulation (U, T, β, γ);
S = best solution in P;
While (α not reached) and (F(S) ≠ 1) do
        //Crossover to produce better solutions
        Pnew = Reproduce (P)
        Calculate Pnew fitness;
        //Selection
        P = Select best β chromosomes from P and Pnew
        S = best solution in P;
End while
// 2- template extraction (ACC and CTM algorithm)
DomT = ParseHTML (HP, S);   //parse HTML of best users to DOM
// Start clustering
Initialize global pheromone C;
// Spread the ants
For each tree do
   Assign cluster randomly based on cβ;
End for
a = 0;
// Start the learning process
While (global pheromone is not filled) OR ( a <= α ) do
   i = j = 0;
   //Apply transition rule and update local pheromone
   For each tree do
      i++;
      For all other trees do
         j++;
         f(i,j)= TreeMatch (DomTi, DomTj);
         //if more than 75% of the characteristic matches, then similar
         if (f(i,j) >= 0.75) then
            Calculate Pp;
              Update DomTi local pheromone with DomTj cluster based on Pp
              //drop down if not similar but same cluster
         Else if (DomTi cluster = DomTj cluster) then
              Calculate Pd;
              Drop down DomTi cluster based on Pd
      End for
   End for
Update global pheromone C;
End while
Generate Templates using C, HP, and S;  }
//CTM measure function
TreeMatch {
Input: two trees T1 and T2.
Output: weighted similarity measure W.
If T1.root ≠ T2.root
   Return 0;
Else
   m = d(T1.root);
   n = d(T2.root);
   // Initialize measure matrix
   M [i][j] = 0 for all i = (0 .. m) and (j = 0 .. n);
   //start matching sub-trees
   For i = 1 to m do
     For j = 1 to n do
       // Recursive call to measure sub trees similarities
       // starting from current node as root
       SupTreeM = TreeMatch (T1[i-1], T2[j-1]);
       M [i][j] = Max ( M[i][j-1], M[i-1][j], (M[i-1][j-1] + SupTreeM) );
     End for
   End for
//Check if leaf nodes is reached
If m ≠ 0 and n ≠ 0 then
   W = M [m][n] / Max (s(T1.root),s(T2.root));
Else
   W = M [m][n]+1 / Max (s(T1.root),s(T2.root));
Return W;
End if }
```

Figure 2: Personalization algorithm

Developers are matched using GA, which typically starts with random solutions then evolves it by repeated selection and reproduction. Chromosomes are used to represent the candidate solutions; for example, if the chromosome size is 4 then it can be represented as illustrated in Figure 3. The content represents the similar users' ID, i.e. possible solution, and each gene in the chromosome is linked to the related characteristic extracted from the XML file.

"123456"	"123789"	"654321"	"987654"

Figure 3: Solution Chromosome

First, the population will be initialized by users of the same age, given the maximum number of genes; such that each chromosome will represent a solution. Maximum number of genes represents the maximum number of similar users who can be considered; it is restricted because the repository of profiles can be very large, which tremendously increases the chromosome length to the extent that might affect the process time without much of improvement in the solution.

The fitness value of chromosome I is calculated using Equation(1), where D(I, T) is the similarity distance between the targeted user profile T and other users included in the chromosome, G is number of genes, and Ch is number of characteristic in profiles. The similarity measure D (I, T) is calculated using Equation (2) for all genes in the chromosome. ChMatch is a function that calculates the similarity between two profile characteristics using Equation (3).

$$F(I) = \frac{\mathrm{D(I,T)}}{[\mathrm{G} \times \mathrm{Ch}]} \quad (1)$$

$$\mathrm{D(I,T)} = \sum_{i=1}^{G} \sum_{w=1}^{Ch} ChMattch(I_{iw}, T_w) \quad (2)$$

$$ChMatch(I_w, T_w) = \begin{cases} 1 & \text{if } I_w = T_w \\ 0 & \text{if } I_w \neq T_w \end{cases} \quad (3)$$

After calculating the fitness of all users in the chromosome the population is reproduced until the stop condition is reached or the target solution, where the fitness is

equal to one, is found. After that, using the resulting solution, templates can be extracted. This extraction is divided into three steps, as follows.

4.2.1 HTML Parsing

Parsing web pages into trees will simplify and speed up the search in such pages. Hence, each website front page is parsed into a tree using DOM representation where each node in the tree represents an HTML tag. In addition, each tree will be identified by the developer (user) id and his profile fitness.

4.2.2 Tree Clustering

After parsing HTML pages to DOM trees, the resulting trees are clustered into groups to identify similar templates and avoid redundant results. To accomplish this step ACC and CTM are used. Each tree represents an ant, clusters represent roads, and a solution is represented by a vector with N (number of trees) elements and W (number of clusters) values. For example, if (N=5) and (W=3) then a solution could be as illustrated in Figure 4; which means that the second, fourth, and fifth trees belong to the same cluster while the first belong to cluster#2 and the third tree belong to cluster#3.

T1	T2	T3	T4	T5
2	1	3	1	1

Figure 4: Cluster solution

After initializing all the attributes, this step works as follows. First, each tree will be assigned randomly to a cluster. Then, while the stop condition is not reached, each tree is matched with all other trees. The similarity measure between two trees Ti and Tj is calculated using the TreeMatch function illustrated in Figure 2, where d(n) is number of first-level children of node n, T[i] is a sub-tree T starting from level i, s(n) is number of siblings of node n including itself, and M is the measure matrix. Such measure selects sub-trees that share the same root and analyze its tag name to return the similarity weighted measure between two trees. TreeMatch function use CTM measure, discussed previously, which calculates tree similarity recursively in a matrix and counts the weight difference between two trees.

After calculating the similarity measure it will be possible to apply the transition rule and decide whether to pick up a neighbor cluster or drop the current tree cluster in order to update the local pheromone. In the proposed approach, a slightly different transition rule has been exploited and used, as follows. If the current tree has a similar clustered neighborhood, then the neighbors' cluster will be picked up, based on probability P_p, and the current tree will join the same cluster. On the other hand, if the current tree has a similar cluster to none neighborhood tree then it means that the clustering is inaccurate; thus, the target tree must drop down its cluster, based on a certain probability P_d.

The probability of picking up (P_P) and dropping down (P_d) a tree out/in a certain cluster is calculated using Equation (4) and (5), respectively, where f is the similarity measure, calculated previously using TreeMatch function, K_p and K_d are threshold constant. Note that if the similarity between two trees is zero while they have the same cluster then P_d will be equal to 1 in order to force the drop down of the cluster.

$$P_P = \left(\frac{K_p}{K_p + f_{(i,j)}}\right)^2 \quad (4)$$

$$P_d = \begin{cases} \left(\frac{f_{(i,j)}}{K_d + f_{(i,j)}}\right)^2 & f \neq 0 \\ 1 & f = 0 \end{cases} \quad (5)$$

After comparing all trees together and updating their local pheromone, the global pheromone will be updated. At the end, this step will be repeated in order to improve the solution and assign trees to the most appropriate cluster. Note that only one tree can update the global pheromone in each round in order to guarantee testing as many roads as possible. In addition, some trees might not find any similar neighborhood; thus, such trees will be left in a separate cluster to make sure that its characteristic does not disappear. Consequently, unique web pages are preserved in case similar websites join the Web.

4.2.3 Template Generation

In order to extract the common template, resulting clusters will be examined and trees with the highest fitness in each cluster will be chosen. After that, attributes such as images and colors will be changed to random values. Then, DOM trees are parsed back to HTML files to be viewed as a personalized template.

5 Implementation

In order to test the performance of the proposed approach an experiment has been conducted. This section discusses the details of the experiment and explains its result.

5.1 Experiment Settings

The proposed approach has been implemented with Java language and executed on a PC with Intel®Core™ i7 CPU, and 2.67 GHz processes. In addition, websites with different languages and characteristics have been used in the experiment. Each website has a full profile which was synthetically build in a database. Moreover, since almost all the websites have unclean structure, an external library called HTML Cleaner [26], was used to extract the websites from its URL, clean it, and then convert it to HTML pages. During the experiment, all the parameters were unchanged, except for the stop condition. Population and chromosome size was set to "10" because only ten templates are wished to be viewed; K_p and K_d were set to "0.5" to give 50% chance of dropping/picking, while the maximum number of group was set to "5". In the experiment ten trials have been recorded, each trial has a different stop condition starting from 10 to 100.

5.2 Evaluation Measure

In order to evaluate the result of the experiment three types of measures have been calculated. The first measure is the algorithm speed, which record the time of personalization step. The second measure, however, records the quality of GA using the fitness of each trial extracted solution to indicate the similarity degree between the users included in the solution and the target user. The final measure, on the other hand, assesses the clustering quality using two measures: inter-cluster and intra-cluster similarity. Inter-cluster similarity measures the similarity between pages in the same cluster using TreeMatch function. If its value is high, it would mean pages in one cluster are very similar. On the other hand, intra-cluster similarity measures the similarity between the clusters. It measures the distance between the most fitted trees of each cluster using TreeMatch function. If it has a low value, it would mean that each cluster is very dissimilar from the other.

5.3 Experiment Result

After conducting the ten trials, using the parameters specified previously, each measure is calculated and recorded. In Figure 5 the execution time of the algorithm is illustrated. As it can be noted, when the stop condition increases the time will also increase. Such result is normal because matching and clustering steps depend on this condition. Nevertheless, the time taken to execute the algorithm, in any trial, is very low. Most of the trails have taken less than one second, which indicates a good performance in terms of speed.

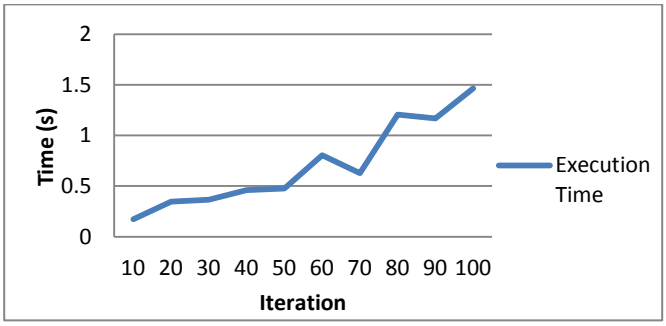


Figure 5: Algorithm speed

Additionally, in Figure 6 the solution fitness (matching quality) and inter/intra cluster similarity (clustering quality) are recorded in each trial. As it can be noted, the quality of the solution extracted in each trial was, on average, 98% match with the target user. Thus, it can be stated that the quality of profile matching is very good.

In addition, inter-cluster similarity indicated high percentage (99%) while intra-cluster similarity, on the other hand, indicated low percentage (0.1%). These results reveal that such algorithm did not group dissimilar trees together and did not separate similar ones. However, we must emphasize on the fact that increasing of the stop condition did not affect the value of the clustering quality while slightly decreased the quality of the solution. Thus, a fair stop condition (equal to the number of pages) is recommended.

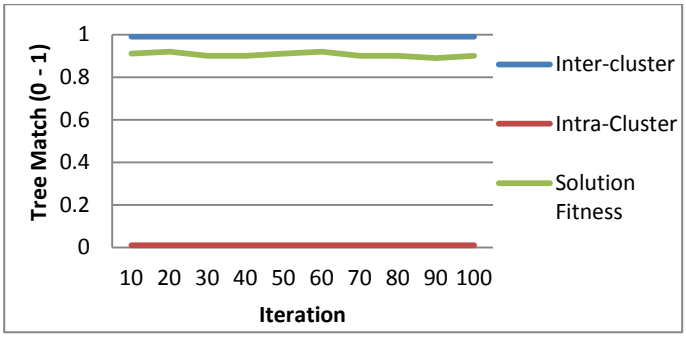


Figure 6: Matching and Clustering quality

5.4 Discussion

After understanding and testing the proposed approach, it is time to summarize its contribution. First of all, such an approach has further generalized web personalization. Instead of only using web personalization on navigation and dynamic data it was applied on the static part of web pages to consider the developer of websites rather than only the visitors. Thus, such an approach has added a value to the area of web personalization to include providers' need, personalizing static content, and consider characteristics when matching rather than keywords.

Moreover, long and short term interest has been considered; such that it does not always update and re-execute the whole process. Only the personalization step is repeated with every query; however, the rest is done once and updated every period of time while its result is stored in XML files. Such decision reduced the time tremendously without affecting the accuracy and improved the scalability of WCMS. In addition, using of XML files simplified and speed up the process of matching because XML is a standard language for describing the data in a tree structure; thus, profiles will be easily searched and information can quickly be fetched.

In addition, to handle web pages' content it was parsed to DOM trees, which simplified and further improved the speed of web page matching and clustering because searching and matching trees is much better than going through the whole web page. Moreover, such an approach is very dynamic due to the use of different parameters, which can be changed depending on the providers' need; for example, one provider could prefer to view only one template; thus, the chromosome length can be set to one. Additionally, pages were not forced to join a cluster to preserve unique web pages and improve the clustering in case similar websites join the Web.

Finally, even though the testing result is mathematically good, but we could not compare it with other methods because no other works of template personalization in WCMS have been found. Thus, as future work, the proposed approach

should be tested with real users to ensure its satisfaction. In addition, ontology, which is a structured knowledge repository, could also be used when acquiring data. Instead of giving limited options when entering the characteristics developers can enter what they wish and translate it using ontology's repository. Therefore, providers will have more freedom in designing their websites.

6 Conclusion

In today's World Wide Web, the use of Web personalization has increased tremendously. However, such use has only concentrated on the visitors' behavior. Alternatively, WCMS has occurred to help website developers to create, edit, organize, and publish the web content. Nevertheless, when designing a website, WCMS has only considered customization; thus, WCMS developers must design some templates in advance. Such process consumes a lot of effort and time. Therefore, the paper proposed a new approach which improves the quality of WCMS and increases its flexibility and scalability. Such an approach use web personalization and template extraction to take advantage of previous developers' knowledge and automatically offer different template designs for each website developer by mixing GA and ACC with CTM. In addition, an experiment has been conducted to finally conclude that this approach has a high quality, in regard to speed, precision, and accuracy. Finally, as future work, it was recommended to use ontology to further enrich the scalability of the proposed approach.

7 References

[1] M. Eirinaki, D. Mavroeidis, G. Tsatsaronis, and M. Vazirgiannis, "Introducing Semantics in Web Personalization: The Role of Ontologies," in *Semantics, Web and Mining*. vol. 4289, M. Ackermann, B. Berendt, M. Grobelnik, A. Hotho, D. Mladenic, G. Semeraro, M. Spiliopoulou, G. Stumme, V. Svátek, and M. van Someren, Eds., ed: Springer Berlin / Heidelberg, 2006, pp. 147-162.

[2] H.-z. Shen, J.-d. Zhao, and Z.-z. Yang, "A Web Mining Model for Real-time Webpage Personalization," in *Management Science and Engineering, 2006. ICMSE '06. 2006 International Conference on*, 2006, pp. 8-12.

[3] S. Flesca, S. Greco, A. Tagarelli, and E. Zumpano, "Non-invasive support for personalized navigation of Websites," in *Database Engineering and Applications Symposium, 2004. IDEAS '04. Proceedings. International*, 2004, pp. 183-192.

[4] L. He and Y. Chen, "Design and implementation of Web Content Management System by J2EE-based three-tier architecture: Applying in maritime and shipping business," in *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*, 2010, pp. 513-517.

[5] K. Chulyun and S. Kyuseok, "TEXT: Automatic Template Extraction from Heterogeneous Web Pages," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, pp. 612-626, 2011.

[6] C. Shui-Lung and H. Jane Yung-jen, "Tree-Structured Template Generation for Web Pages," in *Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on*, 2004, pp. 327-333.

[7] M. Eirinaki and M. Vazirgiannis, "Web mining for web personalization," *ACM Trans. Internet Technol.*, vol. 3, pp. 1-27, 2003.

[8] M. Hildebrandt, "Defining Profiling: A New Type of Knowledge?," in *Profiling the European Citizen*, M. Hildebrandt and S. Gutwirth, Eds., ed: Springer Netherlands, 2008, pp. 17-45.

[9] M. Crossley, N. J. Kings, and J. R. Scott, "Profiles — Analysis and Behaviour," *BT Technology Journal*, vol. 21, pp. 56-66, 2003.

[10] W. A. M. A. Wan Mohd Mahidin, "Dynamic template for lecturers' webpages in FTMSK," Information Technology, Universiti Teknologi MARA, 2003.

[11] B. Christos, K. Vaggelis, and M. Ioannis, "Web page fragmentation for personalized portal construction," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, 2004, pp. 332-336 Vol.1.

[12] (2004). *What is the Document Object Model?* Available: http://www.w3.org/TR/DOM-Level-3-Core/introduction.html

[13] Y. Kim, J. Park, T. Kim, and J. Choi, "Web Information Extraction by HTML Tree Edit Distance Matching," pp. 2455-2460, 2007.

[14] E. Ferrara and R. Baumgartner, "Automatic Wrapper Adaptation by Tree Edit Distance Matching," *CoRR*, vol. abs/1103.1252, 2011.

[15] A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Berlin: Spinger-Verlag, 2002.

[16] J. Wook and S. Woo, "New Encoding/Converting Methods of Binary GA/Real-Coded GA," *IEICE Trans*, vol. E88-A, pp. 1545-156, 2005.

[17] L. Deneubourg, S. Goss, N. Franks, C. Detrain, and L. Chretien, "The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot," in *from animals to animats : proc. of the first int. conf. on simulation of adaptive behavior*, 1991, pp. 356 - 365.

[18] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, second ed.: Prentice Hall, 2003.

[19] S. Wei, L. Jian-Chang, H. Yu-Jun, and L. Jian-Qiang, "Application of neural network model combining information entropy and ant colony clustering theory for short-term load forecasting," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 4645-4650 Vol. 8.

[20] J. Lu, D. Ruan, and G. Zhang, "E-Service Intelligence: An Introduction," in *E-Service Intelligence*. vol. 37, J. Lu, G. Zhang, and D. Ruan, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 1-33.

[21] T. Dinh and A. Mamun, "A combination of clustering algorithms with Ant Colony Optimization for large clustered Euclidean Travelling Salesman Problem," in *WSEAS SOSM 2004*, Miami, Florida, 2004, pp. 484-352.

[22] O. Nasraoui, M. Soliman, E. Saka, A. Badia, and R. Germain, "A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, pp. 202-215, 2008.

[23] X. Ji, J. Zeng, S. Zhang, and C. Wu, "Tag tree template for Web information and schema extraction," *Expert Systems with Applications*, vol. 37, pp. 8492-8498, 2010.

[24] Y. Gui-Sheng, G. Guang-Dong, and S. Jing-Jing, "A template-based method for theme information extraction from web pages," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, 2010, pp. V3-721-V3-725.

[25] H. Haikun, C. Xiaoxin, W. Guoshi, and L. Jing, "Web Data Extraction Based on Tree Structure Analysis and Template Generation," in *E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on*, 2010, pp. 1-5.

[26] net.sourceforge.htmlcleaner. (2010). *HtmlCleaner release 2.2*. Available: http://htmlcleaner.sourceforge.net

A Simulated Docking of TOK-001 with Cytochrome P450 17A1

Jack K. Horner
PO Box 266
Los Alamos NM 87544
jhorner@cybermesa.com

Abstract

Cytochrome P450 17A1 (also known as CYP17A1) catalyses the biosynthesis of androgens in humans. Because prostate cancer cells proliferate in response to androgen steroids, CYP17A1 inhibition can help to prevent androgen synthesis and treat lethal metastatic prostate cancer. Here I report the results of a computational docking of TOK-001, a steroidal inhibitor of CYP17A1 currently under investigation for the treatment of advanced prostate cancer, with the CYP17A1 active site, based on recent X-ray crystallography of the receptor/ligand complex.The docking uses a Lamarckian genetic algorithm.

Keywords: cytochrome P450, CYP17A1, TOK-001, computational docking, prostate cancer, Lamarkian genetic algorithm

1.0 Introduction

Cytochrome P450 17A1 (also known as CYP17A1 and cytochrome P450c17) is a membrane-bound monooxygenase that plays a fundamental role in the synthesis of several human steroid hormones ([5]). The 17α-hydroxylase activity of CYP17A1 is required for the generation of glucocorticoids such as cortisol; the hydroxylase and 17,20-lyase activities of CYP17A1 are required for the production of androgenic and oestrogenic sex steroids. CYP17A1 is thus an important target for the treatment of breast and prostate cancers that proliferate in response to oestrogens and androgens ([6],[7]).

Until recently, steroidal CYP17A1 inhibitors were thought to bind the cytochrome P450 haem iron, more or less parallel to the plane of the haem group in the active site ([8]).

TOK-001 is a steroidal drug currently being investigated for the treatment of metastatic prostate cancer ([8]). Recent X-ray crystallography of TOK-001 complexed with the active site of CYP17A1 shows the drug binds the haem iron in the receptor active site, forming a 60° angle above the haem plane and packing against the central I helix with the 3β-OH interacting with aspargine 202 in the F helix ([1],[3]). This conformation differs substantially from those that are predicted by homology models and from steroids in other cytochrome P450 enzymes with known structures; some features of this conformation are more similar to steroid receptors ([1]).

2.0 Method

The general objective of this study is straightforward: to computationally assess the binding energy of the active site of crystallized cytochrome p450 17A1 with TOK-001. 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) 3SWZ is a structural description of a crystallized cytochrome p450 17A1 bound to TOK-001. 3SWZ has 4 chains, designated A-D.

3SWZ was downloaded from PDB ([3]) on 30 January 2012. The ligand and receptor-active-site portions of 3SWZ Chain A were extracted to separate files, , using *AutoDock Tools* (ADT, [2]). ADT was then used to perform the docking of the ligand to the receptor. More specifically, in ADT, approximately following the rubric documented in [4]

-- all waters and Chains B-D of 3SWZ were deleted

-- the ligand (TOK-001) and Chain A's active-site were extracted to separate files, one for each of the ligand and the receptor (3SWZ identifies the active site of Chain A as 10 residues: ALA113, PHE114, ASN202, ILE205, ALA302, THR306, VAL366, ALA367, VAL482, and HEM600.)

-- the hydrogens, charges, and torsions in the ligand and active site were adjusted using 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.

```
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 A C OA HD N             # atoms types in ligand
fld 3SWZ_A_active_receptor.maps.fld  # grid_data_file
map 3SWZ_A_active_receptor.A.map     # atom-specific affinity map
map 3SWZ_A_active_receptor.C.map     # atom-specific affinity map
map 3SWZ_A_active_receptor.OA.map    # atom-specific affinity map
map 3SWZ_A_active_receptor.HD.map    # atom-specific affinity map
map 3SWZ_A_active_receptor.N.map     # atom-specific affinity map
elecmap 3SWZ_A_active_receptor.e.map # electrostatics map
desolvmap 3SWZ_A_active_receptor.d.map# desolvation map
move 3RUK_A_ligand.pdbqt             # small molecule
about 27.936 -1.9813 32.3924         # 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 2                            # 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 used in this study. The setup uses a Lamarckian genetic algorithm minimum-energy search; all other ADT parameters are defaulted.

The minimum-energy configuration was selected from the configurations generated, and saved. Interatomic distances between ligand and receptor in the computed form were compared to those in [3].

3.0 Results

The interactive problem setup, which assumes familiarity with the general CYP17A1 "landscape", took about 20 minutes in ADT; the docking proper, about 24 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 ligand/receptor energy and position summary produced by ADT for the best-fit conformation obtained under the conditions described in Figure 2.0. The estimated free energy of binding is ~ -7.9 kcal/mol; the estimated inhibition constant, ~1.75 microMolar at 298 K. All distances between receptor and ligand atoms in the computed ligand position lie within 10% of the distances of the corresponding atoms in 3SWZ.

```
        LOWEST ENERGY DOCKED CONFORMATION from EACH CLUSTER
        ___________________________________________________


Keeping original residue number (specified in the input PDBQ file) for outputting.

MODEL        8
USER    Run = 8
USER    Cluster Rank = 1
USER    Number of conformations in this cluster = 5
USER
USER    RMSD from reference structure       = 10.790 A
USER
USER    Estimated Free Energy of Binding    =   -7.85 kcal/mol  [=(1)+(2)+(3)-(4)]
USER    Estimated Inhibition Constant, Ki   =    1.75 uM (micromolar)  [Temperature = 298.15 K]
USER
USER    (1) Final Intermolecular Energy     =   -8.45 kcal/mol
USER        vdW + Hbond + desolv Energy     =   -8.40 kcal/mol
USER        Electrostatic Energy            =   -0.05 kcal/mol
USER    (2) Final Total Internal Energy     =   -0.16 kcal/mol
USER    (3) Torsional Free Energy           =   +0.60 kcal/mol
USER    (4) Unbound System's Energy  [=(2)] =   -0.16 kcal/mol
USER
USER
USER
USER    DPF = 3SWZ_A.dpf
USER    NEWDPF move    3RUK_A_ligand.pdbqt
USER    NEWDPF about   27.936001 -1.981300 32.392399
USER    NEWDPF tran0   20.130862 7.221189 34.800957
USER    NEWDPF axisangle0     -0.007078 0.803179 -0.595696 -140.734418
USER    NEWDPF quaternion0    -0.006666 0.756486 -0.561065 -0.335991
USER    NEWDPF dihe0   2.04 -178.84
USER
USER                             x       y       z    vdW   Elec        q     RMS
ATOM      1  C1  AER A 601      21.585   8.609  31.649 -0.21 +0.00    +0.016 10.790
ATOM      2  C2  AER A 601      21.647   9.403  30.352 -0.17 -0.00    +0.033 10.790
ATOM      3  C3  AER A 601      20.280   9.574  29.742 -0.22 -0.02    +0.122 10.790
ATOM      4  C4  AER A 601      19.351  10.306  30.715 -0.25 -0.03    +0.066 10.790
ATOM      5  C5  AER A 601      19.376   9.713  32.096 -0.41 +0.02    -0.072 10.790
ATOM      6  C6  AER A 601      18.195   9.661  32.707 -0.49 -0.00    -0.023 10.790
ATOM      7  C7  AER A 601      17.917   8.839  33.940 -0.38 +0.01    +0.033 10.790
ATOM      8  C8  AER A 601      19.182   8.454  34.702 -0.46 -0.00    -0.001 10.790
ATOM      9  C9  AER A 601      20.304   8.080  33.721 -0.33 +0.00    +0.003 10.790
ATOM     10  C10 AER A 601      20.670   9.195  32.730 -0.31 +0.00    -0.017 10.790
ATOM     11  C11 AER A 601      21.521   7.442  34.395 -0.29 +0.00    +0.007 10.790
ATOM     12  C12 AER A 601      21.150   6.331  35.386 -0.29 +0.01    +0.014 10.790
ATOM     13  C13 AER A 601      20.167   6.836  36.406 -0.39 -0.01    -0.016 10.790
ATOM     14  C14 AER A 601      18.961   7.212  35.572 -0.42 +0.00    +0.003 10.790
ATOM     15  C15 AER A 601      17.753   7.179  36.506 -0.58 +0.00    +0.010 10.790
ATOM     16  C16 AER A 601      18.058   6.134  37.563 -0.35 +0.03    +0.036 10.790
ATOM     17  C17 AER A 601      19.545   5.867  37.355 -0.35 -0.04    -0.060 10.790
ATOM     18  C18 AER A 601      20.783   7.922  37.307 -0.39 +0.01    +0.020 10.790
ATOM     19  C19 AER A 601      21.471  10.305  33.394 -0.23 -0.01    +0.020 10.790
ATOM     20  C20 AER A 601      20.298   4.767  38.073 -0.31 -0.02    -0.018 10.790
ATOM     21  C25 AER A 601      19.715   3.500  38.254 -0.26 +0.02    +0.014 10.790
ATOM     22  C24 AER A 601      20.422   2.508  38.915 -0.24 +0.02    +0.018 10.790
ATOM     23  C23 AER A 601      21.698   2.812  39.369 -0.22 +0.09    +0.087 10.790
ATOM     24  N22 AER A 601      22.241   4.038  39.185 -0.08 -0.30    -0.375 10.790
ATOM     25  H22 AER A 601      23.180   4.225  39.540 -0.36 +0.09    +0.164 10.790
ATOM     26  C21 AER A 601      21.578   5.012  38.550 -0.14 +0.07    +0.099 10.790
ATOM     27  O3  AER A 601      20.432  10.399  28.591 -0.10 +0.07    -0.395 10.790
ATOM     28  H3  AER A 601      19.566  10.534  28.225 -0.17 -0.06    +0.210 10.790
TER
ENDMDL
```

Figure 2. Coordinates of TOK-001 generated by this study.

Figure 3 is a rendering produced in ADT of the CYP17A1/TOK-001 docking described in Section 2.0.

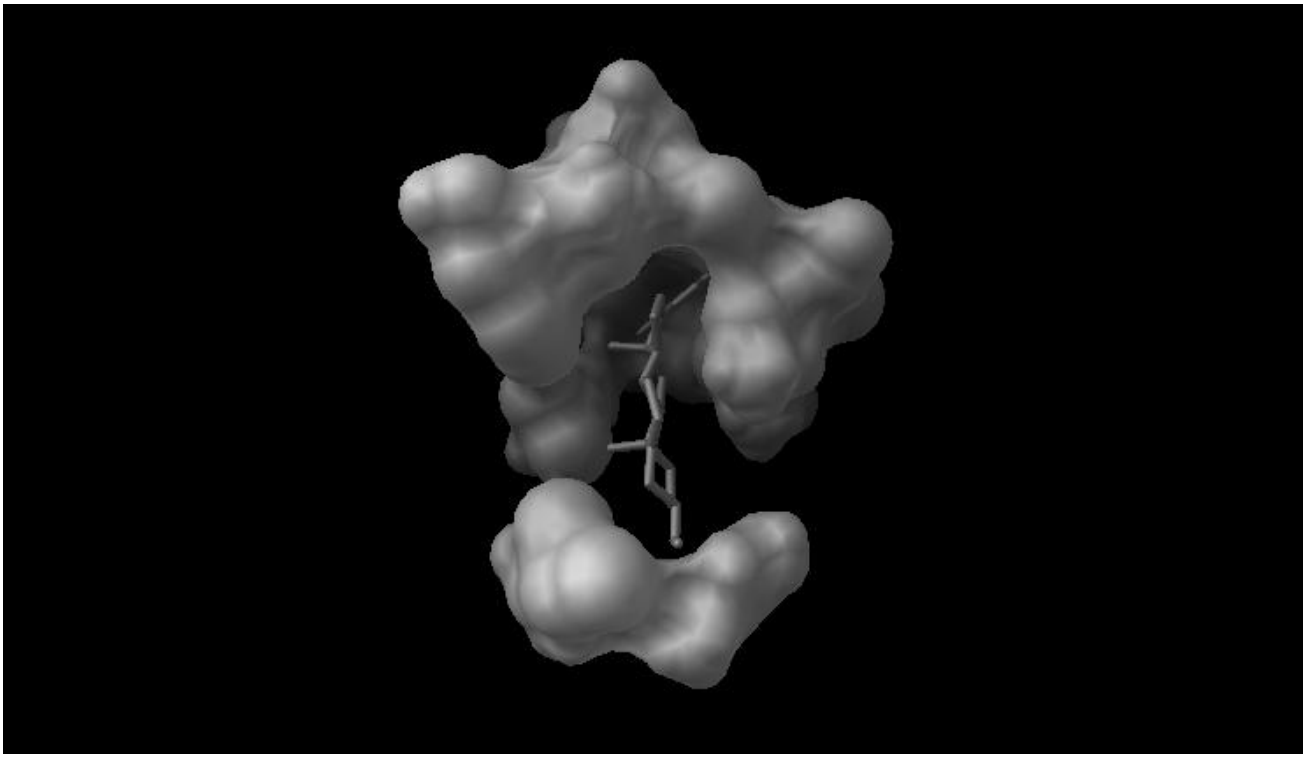


Figure 3. AutoDock Tools (ADT,[2]) rendering of a computational docking of TOK-001 (the ligand, shown in stick-and-ball form) with molecular surface of the active site of Chain A of cytochrome p450 17A1 (shown in white), derived from PDB 3SWZ ([1],[3]). The upper right end of the ligand lies directly below the center of the haem group in the active site.

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 (~1.75 microMolar at ~298 K) is comparable to the inhibition constant of cancer-therapeutic ligand/receptor interactions that are clinically effective.

2. All distances between receptor and ligand atoms in the computed ligand position lie within 10% of the distances of the corresponding atoms in 3SWZ. (For electrostatic forces, a 10% distance difference would correspond to a ~20% difference in electrostatic force and potential energy, in the worst case. 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 3SWZ per se. However, the calculation does not reflect what receptor "flexing" could contribute to the interaction of the ligand with native unliganded receptor.

4. The analysis described in Sections 2.0 and 3.0 assumes receptor is in a crystallized form. *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.

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.

7. 3SWZ has four chains, each with its own active site. The work described in this paper was performed on Chain A only. Chains B-D appear to have active sites highly similar to the Chain A active site. Future work will assess the ligand/receptor binding energies of Chains B-D.

8. CYP17A1 is a membrane-bound protein; 3SWZ describes a conformation that is not bound to a membrane. The membrane-bound conformation of CYP17A1 may differ from the conformation in 3SWZ.

9. The free energy of binding of TOK-001 to CYP17A1 is ~20% larger than the free energy of binding of abiraterone, a steroidal CYP17A1 inhibitor, to CYP17A1, suggesting that TOK-001 may be a more effective CYP17A1 inhibitor than abiraterone ([9]).

5.0 References

[1] DeVore NM and Scott EE. Structure of cytochrome P450 17A1 with prostate cancer drugs abiraterone and TOK-001. *Nature* online pre-publication doi:10.1038/nature10743.

[2] Morris GM, Goodsell DS, Huey R, Lindstrom W, Hart WE, Kurowski S, Halliday S, Belew R, and Olson AJ. *AutoDock Tools v4.2*. http://autodock.scripps.edu/. 2011.

[3] Protein Data Bank. PDB ID: 3SWZ. DeVore NM and Scott EE. Structure of cytochrome P450 17A1 with prostate cancer drugs abiraterone and TOK-001. *Nature* online pre-publication doi:10.1038/nature10743.

[4] Huey R and Morris GM. *Using Autodock4 with AutoDock Tools: A Tutorial*. 8 January 2008.

[5] Miller WK and Auchus RJ. The molecular biology, biochemistry, and physiology of human steroidogenesis and its disorders. *Endocrine Reviews* 32 (2011), 81–151.

[6] Attard G, Reid,AH, Olmos D, and de Bono JS. Antitumor activity with CYP17 blockade indicates that castration-resistant prostate cancer frequently remains hormone driven. *Cancer Research* 69 (2009), 4937–4940.

[7] Yap TA, Carden CP, Attard G, and de Bono JS. Targeting CYP17: Established and novel approaches in prostate cancer. *Current Opinion in Pharmacology* 8 (2008), 449–457.

[8] Vasaitis TS, Bruno RD and Njar VC. CYP17 inhibitors for prostate cancer therapy. *Journal of. Steroid Biochemistry and Molecular Biology* 125 (2011), 23–31.

[9] Horner JK. A simulated docking of abiraterone with cytochrome P450 17A1. Submitted to the *2012 International Conference on Bioinformatics and Computational Biology*.

An Application-Specific Approach in Automotive Network Optimization

Martin Dohr and Bernd Eichberger
Institute of Electronics, Graz University of Technology, Austria

Abstract - *The increasing number of automotive functionalities becomes a significant challenge for in-car communication and network architecture. Our approach provides optimized network architectures by applying evolutionary algorithms and application-specific representations. In this paper, we present a new network encoding supported by feasibility-preserving mutation and routing operators. We show that well-established algorithms can be extended with our operators to efficiently optimize cost and complexity of automotive networks.*

Keywords: Evolutionary algorithm; automotive network; optimization; network encoding; AUTOSAR;

1 Introduction

Increasing amounts of new functionality in modern cars have over the last decade lead to ever more complex architectures. This complexity presents new challenges for the Original Equipment Manufacturers (OEMs) in terms of the automotive development process and especially for communication architecture. Those challenges can be outlined as follows:

1.1 Automotive development process

The state of the art in automotive Electric/Electronic (EE) systems comprises among others the following complexities:

- Multiple bus systems and sub-bus systems
- Extensive gateway functionality between bus systems
- Increased effort for testing and multiple variant management
- Exponential growth of software costs.

To handle these complexities, positive experience from software engineering suggests model driven development and similar paradigms also for the automotive domain [1]. Thus, initiatives like AUTOSAR [2] are addressing a standardized software architecture and model driven EE tools like PREEvision [3] support the OEMs in early architecture decisions and variant management. Furthermore, the modeling of complex functionalities using tools like Matlab/Simulink simplifies the interface from OEM to supplier as well as testing and verification efforts. In summary, a holistic and consistent top-down architecture and development methodology is essential to maintain automotive quality requirements.

1.2 Communication architecture

Nearly 20 years have passed since the introduction of first bus-based communication in cars [4]. While new bus systems like FlexRay [5] and Media Oriented Systems Transport (MOST) [6] have addressed higher bandwidth requirements for vehicle dynamics and multimedia applications respectively, most basic applications still utilize the well established Controller Area Network (CAN) [7] bus. Additionally, sub-systems based on Local Interconnect Network (LIN) [8] have been introduced to reduce costs and complexity of the overall network. When looking at the historical development of the network topology, one can notice an organic growth of buses around an ever more complex central gateway. The reasons for this growth are the repeated usage of legacy hardware combined with the introduction of new functionality as individual Electronic Control Units (ECUs) and bus systems. Another reason for this growth is, that newly added features often lead to the installation of a dedicated bus system while leaving existing communication structures untouched due to bandwidth limitations.

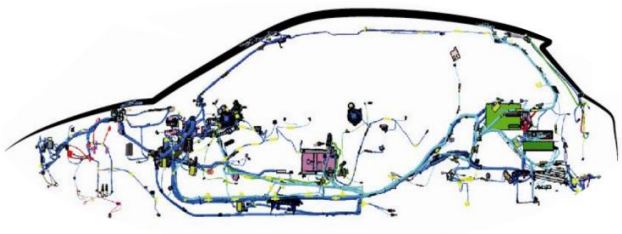


Figure 1. Example car network

Hence, a premium class car nowadays consists of several independent bus systems for the domains powertrain, chassis, body and comfort. In addition to those buses, new features like drive by wire or active suspension imply the need for a safety critical bus system like FlexRay. Furthermore, the multimedia and infotainment cluster is networked with the high-speed bus MOST. With all those new systems it is clear that, apart from power consumption, also the wiring costs have dramatically increased during the last few years.

1.3 Motivation

As new network technologies like Ethernet are making their way into the automotive domain [9], we must now question the suitability of those old network architectures and topologies. Furthermore we need to explore the

optimization potential of new topologies especially in the light of cost reduction and complexity.
To explore the optimization potential, this paper introduces a new heuristic for the following network architecture tasks:

- Mapping of software components onto ECUs
- Layout of network topology
- Routing of communication and creation of gateway tables

These tasks are done using multi objective evolutionary algorithms (MOEAs) and application-specific network encodings to efficiently handle constrains. To take advantage of these encodings, new stochastic operators for mutation and message routing are presented.
The presented methodology is compliant with the system development tasks defined by AUTOSAR and supports automotive development tool chains like PREEvision. Therefore, the challenges of a consistent and model driven automotive development process as described above are met.

2 Related Work

2.1 Application specific encodings

A network encoding with focus on multicast networks has been presented by Ahlswede et al. [10] and used for deterministic topology design by Chi et al. [11]. The cost modeling of automotive electrical architectures was investigated by Quigley et al. [12].
Regarding constraint handling, Coello Coello gave a very good survey in [13], also stating a constraint-consistent GA approach proposed by Kowalczyk [14].

2.2 Automotive network optimization

An automated bus system synthesis for PREEvision was presented by Heinz et al. [15]. Their method based on Hierarchical Clustering and functional nearness of ECUs without considering variations in application mapping.
In contrast to that, Lukasiewycz et al. [16] as well as Glass et al. [17] optimized automotive networks with respect to reliability using a binary Integer Linear Program (ILP) [18].
Furthermore, Kim et al. [19] showed an efficient method to optimize task allocation, ECU positioning and network assignment using a repeated matching method and simulated annealing.

3 Problem Formulation

3.1 Prerequisites

The network optimization problem at hand is defined by a communication description, network constraints and available hardware.
1) In every layered or model driven development, all aspects the resulting product will have are defined by functional and non-functional requirements. The first task is to transform these requirements into technical features and applications, so-called "Software Components" in AUTOSAR. Already, these components form a logical network based on communication requirements. At the time of this task, the timing constraints and required bandwidth of the communications are subject to implementation and therefore not exactly known. We can however approximate the requirements based on previous implementations or estimations on data types and frequency. Furthermore, multicast and broadcast messages, together with their receivers and respective update rates can be identified at this stage.
2) Another aspect of the optimization problem is defined by local or supplier-specific constraints. Local constraints state, that specific software components need to be executed in vicinity to their relevant sensors or actuators. Supplier-specific constraints come from the fact, that the development of some features is often outsourced by the OEM. This outsourcing implies, that the OEM needs to integrate hardware without a reasonable opportunity to manipulate the software components executing on those ECUs. Therefore, some software components are locked to specific ECUs and cannot be remapped.
3) The third input parameter describes the layout of ECUs within the car; providing information about processing unit, available memory, bus connections and peripherals for each unit. It is also possible to consider multiple hardware manifestations and their corresponding costs for the same mounting location.

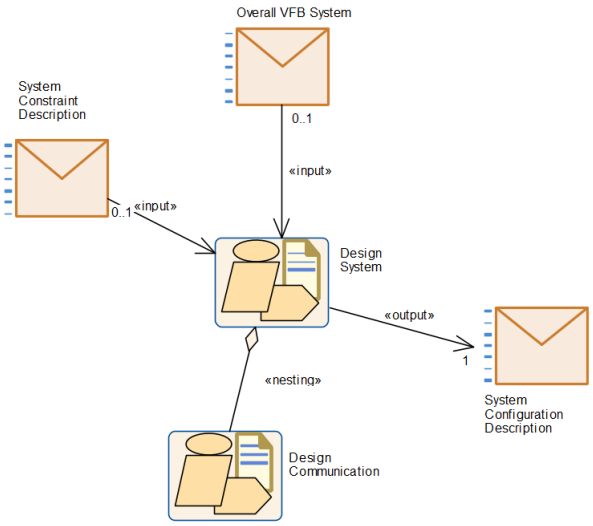


Figure 2. Design System process in AUTOSAR

Using those input parameters, the heuristic has to find a feasible network topology while optimizing several objectives. This process of finding a topology is represented by the "Design System" task in the AUTOSAR specification and metamodel.

3.2 Objectives

The Objectives evaluated by our algorithm are introduced as follows:

- *Monetary costs*: Our main objective is to minimize the amount of ECUs installed by deploying several software components onto the same ECU. Additional costs can be saved by simplifying the bus structure. Thus, hardware costs are modeled as a fixed amount for each used ECU and additional costs for each communication controller and bus coupler.
- *Cable length*: Reduces wiring costs as well as overall weight and manufacturing time.
- *Bandwidth reserves*: Subsequent changes in requirements and communication cannot be ruled out during a typical development time of several years. Therefore it is considered a good practice to reserve some bandwidth for future extensions on each bus system.
- *Gateway complexity*: Gateway routing tables represent additional implementation and testing effort. To minimize this effort we prefer message routing within one network and want to add gateway functionality to as few ECUs as possible.

4 Implementation

The realization of our encodings and operators is written in Java and based on the jMetal [20] framework. Due to the extensible design of the framework, new solution variables and operators can be added easily while providing wide compatibility with already implemented algorithms.

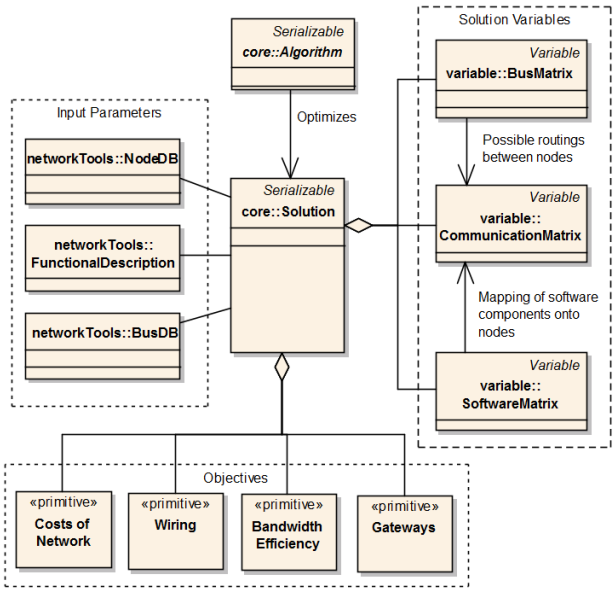


Figure 3. UML relationships between input, parameters, solution and objectives

4.1 Descision variables

Each solution represents a full network and consists of 3 abstract variables.

- A class representing the mapping between each software component and corresponding ECU.
- The definition of all used bus systems and their connected nodes.
- The communication description for nodes and gateway routing tables.

New solutions are created with all software components randomly deployed on allowed ECUs and all network nodes connected to the fastest available bus system. The routing is then straight forward without any gateway functionality. This solutions is always a feasible but very expensive.

4.2 Stochastic Operators

Our approach comprises a set of mutation operators specifically designed for our network optimization problem:

- A mapping mutation deploys software components onto different ECUs within their allowed borders. To maintain feasibility, the corresponding bus variable has to be repaired or reinitialized.
- A bus mutation operator randomly adds ECUs to a bus network. For the sake of simplicity it cannot remove existing connections as this would threaten the feasibility of the system.
- In every case the communication has to be re-routed after changes in other variables. We implemented an efficient algorithm to find the cheapest possible route for each communication requirement. The sequence by which the router iterates through the messages is randomly chosen to add another stochastic influence. Furthermore, this influence allows us to use the router as single operator in order to mutate an existing communication.

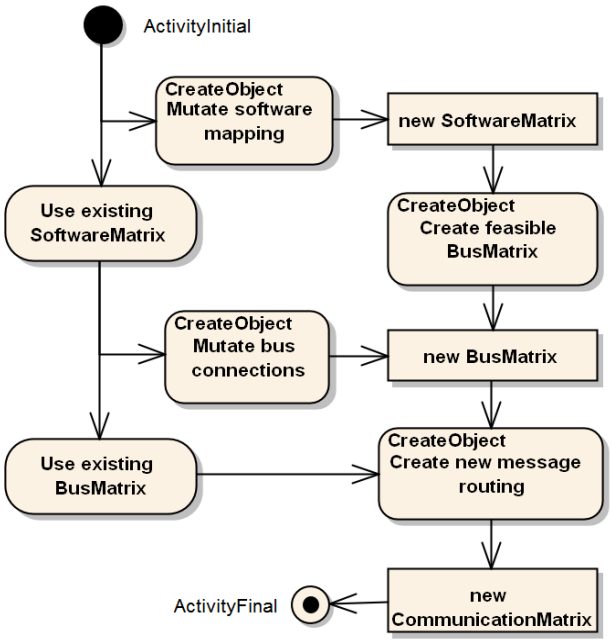


Figure 4. Possible mutations

4.3 Algorithm workflow

Since our network representation is closely related to jMetal´s software design, we could easily adapt algorithms like the SPEA2 [21] for our purposes. The most important adaption was in the algorithm´s variation step. There, we removed the crossover operator and added our own mutation and routing methodologies. Apart from those changes in the variation step, the existing software can be used unaltered. The evaluation of solutions includes a deterministic reduction algorithm, to delete unused bus connections and nodes.

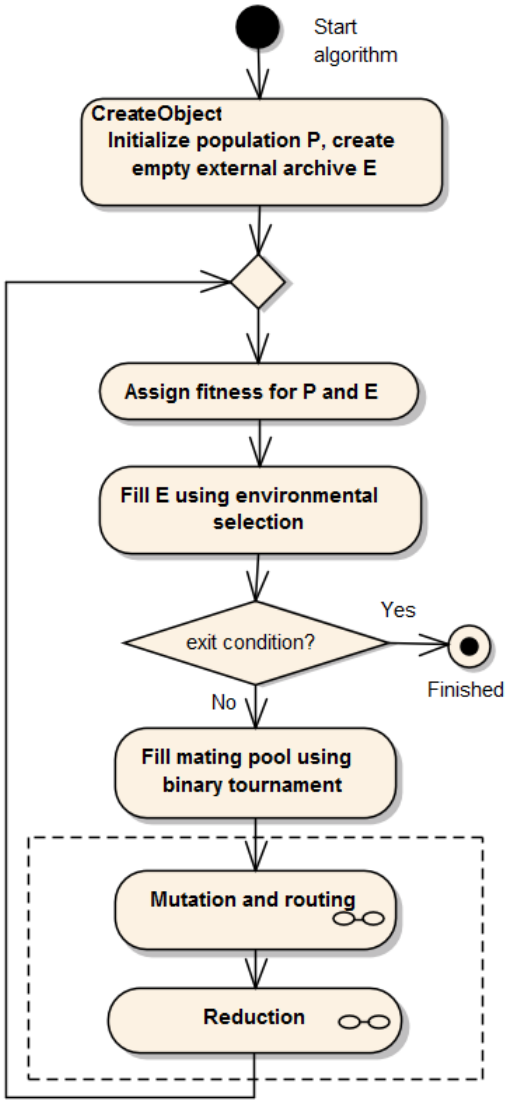


Figure 5. Algorithm workflow

5 Experimental Results

In order to verify the functionality of the presented implementation and characterize it´s behavior under different conditions, experiments with test networks were performed. All experiments were executed on an Intel Core2 Duo T7500 CPU at 2.2GHz with 3 GB Ram running Windows 7.

5.1 Test networks

We created various test networks with 10 to 80 nodes and various constraints. The 10-node network has exactly one known optimal solution and is used for performance benchmarks. The latter 2 networks might represent a low-end and high-end car respectively, but all communication values are purely fictional. Due to their complexity, a optimal solution or true pareto front is not known.

TABLE I. DIMENSIONS OF TEST NETWORKS

Test network	**1**	**2**	**3**
No. Nodes	10	40	80
Location constrained SW components	10	20	70
Unconstrained SW components	0	30	30
Gross bit rate [MBit/sec]	0.2	1	10

5.2 Convergence

Our first experiment series evaluated the mean convergence speed of our optimization. First, we executed 100 independent runs using test network 1. We aborted each run after the optimal solution had been found or 2000 solutions had been evaluated.

82% of our testruns hit the global optimum within 2000 evaluations while the rest was stuck in local optima and, to our observation, would not have succeeded in reasonable time. The results in Fig. 6 support this assumption since the probability of finding the global optimum within a run decreases after 500 evaluations.

TABLE II. CONVERGENCE EXPERIMENT 1

Test network model		**1**
Mutation probability	Software mapping	0.2
	Bus connections	0.2
Archive size		20
Population size		20
Max. evaluations		2000
Runs		100
Optimum hit		**82 %**
Average execution time		**0.68 s**

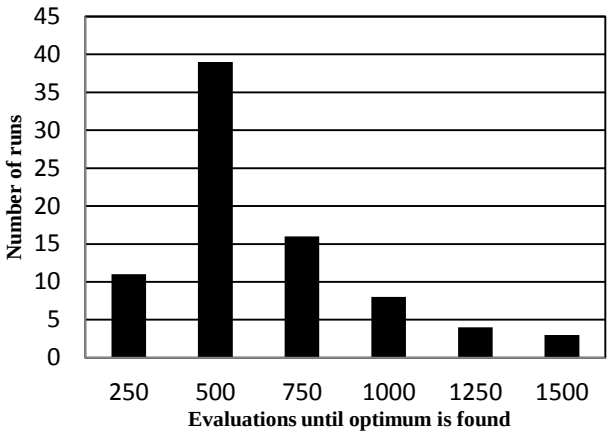


Figure 6. Results for convergence experiment 1

To further examine the convergence behavior we observed the best cost objective of the population while optimizing test network 3. Figure 7 illustrates the optimization process for 5 independent runs. The execution time for each run has been significantly higher due to the larger network model and data output during execution.

TABLE III. CONVERGENCE EXPERIMENT 2

Test network model		**3**
Mutation probability	Software mapping	0.2
	Bus connections	0.2
Archive size		50
Population size		50
Max. evaluations		30000
Runs		5
Average execution time		**36.4 s**

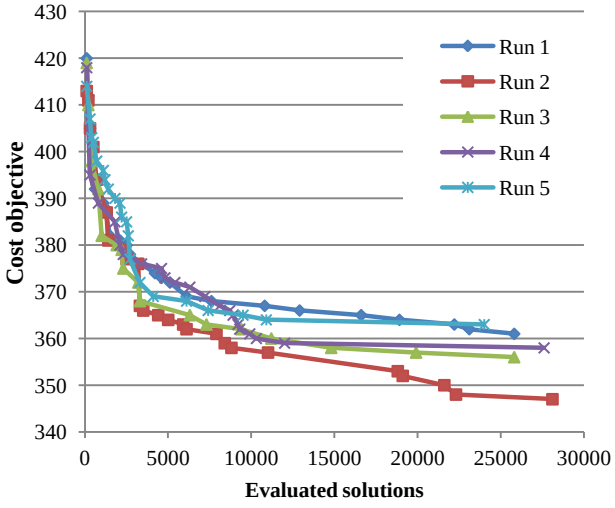


Figure 7. Results for convergence experiment 2

5.3 Performance

In a final step we have studied the influence of archive size and number of evaluations on calculation time. Therefore, for each setting we measured the average execution time of 10 independent runs using test network 2. The results in Fig 8 show that the archive size only influences the overall execution time for large numbers of evaluations. The significant difference in performance for 15.000 and 20.000 evaluations is subject to further investigations.

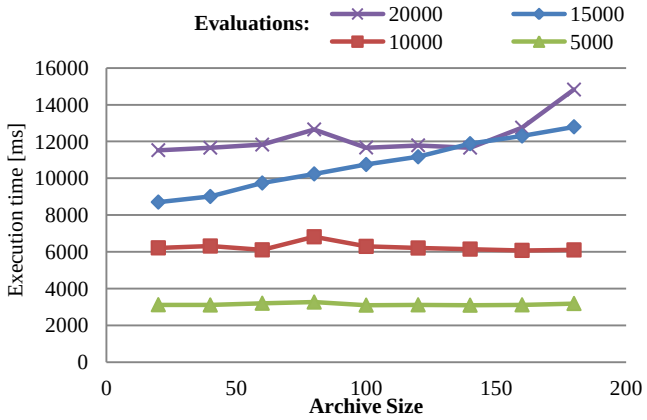


Figure 8. Calculation performance for different amounts of evaluations

6 Conclusion and Further Work

In this paper, we presented a novel approach for automotive network encoding and optimization using evolutionary algorithms.

First, we introduced typical initial situations and challenges for network architects at OEMs and automotive suppliers. Subsequently, we listed requirements and constraints which have to be taken into consideration when developing a new communication network. This network development can be described as a series of tasks: After defining atomic software components and logical links representing transmission requirements, we want to effectively deploy those software components onto corresponding ECUs. Simultaneously to the deployment, we need to interlink these ECUs using automotive specific bus systems while keeping hardware costs, wiring effort and network complexity low.

These tasks can be optimized using our new encoding scheme. In our encoding, we presented 3 objects to represent bus connections, software mapping and network communication respectively. Those objects are varied using established evolutionary algorithms like the SPEA2 to obtain near-optimal network solutions. We ensure the technical feasibility of our solutions by implementing problem specific mutation operators and routing algorithms.

We have shown that the SPEA2 algorithm, as already implemented in the jMetal framework, can easily be adopted to optimize our test networks. Additionally, initial experiments have confirmed a fast and reliable convergence towards optimal results.
Further work will include the introduction of a crossover operator as well as benchmarks regarding common quality indicators used in evolutionary algorithms.
Another interesting task will be the implementation of algorithms like differential evolution [22] or particle swarm optimization [23] since our encodings were designed to utilize different optimization strategies.
We also want to compare our algorithm´s results with currently established car networks to explore the potential of cost optimization in the automotive domain.

7 References

[1] M. Broy, "Challenges in automotive software engineering," New York, NY, USA, 2006.

[2] Automotive Open System Architecture, [Online]. Available: http://www.autosar.org.

[3] Aquintos PREEvision, [Online]. Available: http://www.aquintos.com/.

[4] L. Vlacic, M. Parent and F. Harashima, Intelligent Vehicle Technologies, Butterworth-Heinemann, 2001.

[5] FlexRay, [Online]. Available: http://www.flexray.com/.

[6] MOST Coorperation, [Online]. Available: http://www.mostcooperation.com/home/index.html.

[7] Bosch Controller Area Network, [Online]. Available: http://www.semiconductors.bosch.de/en/ipmodules/can/can.asp.

[8] Local Interconnect Network, [Online]. Available: http://www.lin-subbus.org/.

[9] H.-T. Lim, L. Volker and D. Herrscher, "Challenges in a future IP/Ethernet-based in-car network for real-time applications," 2011.

[10] R. Ahlswede, N. Cai, S. y. Robert and R. W. Yeung, "Network Information Flow," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 46, no. 4, pp. 1204-1216, 2000.

[11] K. Chi, X. Jiang, S. Horiguchi and M. Guo, "Topology Design of Network-Coding-Based Multicast Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 627-640, #may# 2008.

[12] C. Quigley, R. McMurran, R. Jones and P. Faithfull, "An Investigation into Cost Modelling for Design of Distributed Automotive Electrical Architectures," 2007.

[13] C. A. Coello, "A Survey of Constraint Handling Techniques used with Evolutionary Algorithms," 1999.

[14] R. Kowalczyk, "Constraint consistent genetic algorithms," 1997.

[15] M. Heinz, M. Hillenbrand, K. Klindworth and K.-D. Mueller-Glaser, "Rapid automotive bus system synthesis based on communication requirements," 2011.

[16] M. Lukasiewycz, M. Glass, C. Haubelt, J. Teich, R. Regler and B. Lang, "Concurrent topology and routing optimization in automotive network integration," 2008.

[17] M. Glass, M. Lukasiewycz, R. Wanka, C. Haubelt and J. Teich, "Multi-objective routing and topology optimization in networked embedded systems," 2008.

[18] M. Lukasiewycz, M. Glass, C. Haubelt and J. Teich, "SAT-decoding in evolutionary algorithms for discrete constrained optimization problems," 2007.

[19] S. Kim, E. Lee, M. Choi, H. Jeong and S. Seo, "Design Optimization of Vehicle Control Networks," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 7, pp. 3002-3016, sept. 2011.

[20] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760-771, 2011.

[21] E. Zitzler, M. Laumanns and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," 2001.

[22] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.

[23] J. Kennedy and R. Eberhart, "Particle swarm optimization," 1995.

SESSION

PARTICLE SWARM OPTIMIZATION + FIREFLY ALGORITHMS + BEE COLONY OPTIMIZATION

Chair(s)

TBA

Application of a New Multi-Valued Particle Swarm Optimization to Forest Harvest Schedule Optimization

Jared Smythe[1], Walter D. Potter[1], and Pete Bettinger[2]
[1]Institute for Artificial Intelligence, University of Georgia, Athens, Georgia, United States
[2]Warnell School of Forest Resources, University of Georgia, Athens, Georgia, United States

Abstract - *Discrete Particle Swarm Optimization has been noted to perform poorly on a forest harvest planning combinatorial optimization problem marked by harvest period-stand adjacency constraints with the goal of maximizing the even-flow of harvest lumber across harvest periods. Attempts have been made to improve the performance of discrete Particle Swarm Optimization on this type of problem. However, these results do not unquestionably outperform Raindrop Optimization, an algorithm developed specifically for this type of problem. In order to address this issue, this paper proposes a new Roulette Wheel Particle Swarm Optimization algorithm, which markedly outperforms Raindrop Optimization on two of three planning problems.*

Keywords: particle swarm optimization, roulette wheel, forest planning, raindrop optimization

1 Introduction

In [7], the authors evaluated the performance of four nature-inspired optimization algorithms on four quite different optimization problems in the domain of diagnosis, configuration, planning, and path-finding. The algorithms considered were the Genetic Algorithm (GA) [6], Discrete Particle Swarm Optimization (DPSO) [5], Raindrop Optimization (RO) [1], and Extremal Optimization (EO). On the 73-stand forest planning optimization problem, DPSO performed much worse than the other three algorithms, despite thorough testing of parameter combinations, as shown in the following table. Note that the forest planning problem is a minimization problem, so lower objective values represent better quality solutions.

Table 1: Results obtained in [7]

	GA	DPSO	RO	EO
Diagnosis	87%	98%	12%	100%
Configuration	99.6%	99.85%	72%	100%
Planning	**6,506,676**	**35M**	**5,500,391**	**10M**
Path-finding	95	95	65	74

In [2], the authors address this shortcoming of DPSO by introducing a continuous PSO with a priority representation, an algorithm which they call PPSO. This yielded significant improvement over DPSO, as shown in the following table, where they included results for various values of inertia (α) and swarm size.

Table 2: Results from [2]

		PPSO		DPSO	
α	Pop. Size	Best	Avg	Best	Avg
1.0	100	7,346,998	9,593,846	118M	135M
1.0	500	6,481,785	9,475,042	133M	139M
1.0	1000	5,821,866	10M	69M	110M
0.8	100	8,536,160	13M	47M	70M
0.8	**500**	**5,500,330**	**8,831,332**	**61M**	**72M**
0.8	1000	6,999,509	10M	46M	59M

Although the results from [2] are an improvement over the DPSO, PPSO does not have a resounding victory over RO. In [1], the average objective value on the 73-stand forest was 9,019,837 after only 100,000 iterations, compared to the roughly 1,250,000 fitness evaluations used to obtain the best solution with an average of 8,831,332 by the PPSO. Obviously, the comparison is difficult to make, not only because of the closeness in value of the two averages, but also because the termination criteria are not of the same metric.

In this paper we experiment with RO to generate relatable statistics, and we formulate a new multi-value discrete PSO that is more capable of dealing with multi-valued nominal variable problems such as this one. Additionally, we develop two new fitness functions to guide the search of the PSO and detail their effects. Finally, we experiment with a further modification to the new algorithm to examine its impact on solution quality. The optimization problems addressed are not only the 73-stand forest planning problem [1][2][7], but also the 40- and 625-stand forest problems described in [1].

2 Forest planning problem

In [1], a forest planning problem is described in which the goal is to develop a forest harvest schedule that would maximize the even-flow of harvest timber, subject to the constraint that no adjacent forest partitions (called stands) may be harvested during the same period. The goal of maximizing

the even-flow of harvest volume was translated into minimizing the sum of squared errors of the harvest totals from a target harvest volume during each time period. This objective function f_1 can be defined as:

$$f_1 = \sum_{k=1}^{z} \left(T - \sum_{n=1}^{d} a_n h_{n,k}\right)^2 \quad (1)$$

where z is the number of time periods, T is the target harvest volume during each time period, a_n is the number of acres in stand n, $h_{n,k}$ is the volume harvested per acre in stand n at the harvest time k, and d is the number of stands. A stand may be harvested only during a single harvest period or not at all, so $a_n h_{n,k}$ will be either the volume from harvesting an entire stand n at time k or zero if that stand is not scheduled for harvest at time k.

Three different forests are considered in this paper, namely a 40-stand northern forest [1] (shown in Figure 1), the 73-stand Daniel Pickett Forest used in [1][2][7] (shown in Figure 2), and a much larger 625-stand southern forest [1] (not shown). The relevant stand acreage, adjacency, centroids, and time period harvest volumes are located[1] under "Northern US example forest," "Western US example forest," and "Southern US example forest." Three time periods and a no-cut option are given for the problem. For the 40-stand forest the target harvest is 9,134.6 m^3, for the 73-stand forest the target harvest is 34,467 MBF (thousand board feet), and for the 625-stand forest the target harvest is 2,972,462 tons.

In summary, a harvest schedule is defined as an array of length equal to the number of stands in the problem and whose elements are composed of integers on the range zero to three inclusively, where 0 specifies the no-cut option, and 1 through 3 specify harvest periods. Thus, solutions to the 40-, 73-, and 625-stand forest planning problems should be 40-, 73-, and 625-length integer arrays. A valid schedule is a schedule with no adjacency violations, and an optimal schedule is a valid schedule that minimizes f_1.

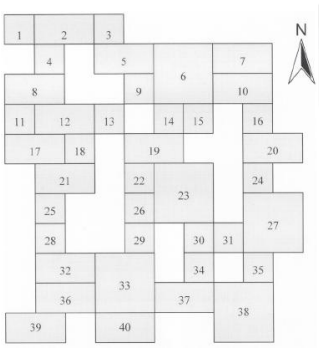


Figure 1: 40-Stand Forest

Figure 2: 73-Stand Forest

Many algorithms have been applied towards this problem. In [1] Raindrop Optimization, Threshold Accepting, and Tabu Search were used on the 40-, 73-, and 625-stand forest planning problems. In [7], a Genetic Algorithm, integer Particle Swarm Optimization, Discrete Particle Swarm Optimization, Raindrop Optimization, and Extremal Optimization were applied to the 73-stand forest planning problem. In [2], a Priority Particle Swarm Optimization algorithm was applied to the 73-stand forest problem.

In this paper, tests will be run with Raindrop Optimization and a new proposed algorithm, Roulette Wheel Particle Swarm Optimization. Comparisons will be made between these test results and the test results from [1], [2], and [7].

3 Raindrop optimization

Described in [1], Raindrop Optimization (RO) is a stochastic point-based search algorithm developed for the forest planning problem and inspired by the ripples in a pond generated from a falling raindrop. It starts with an initial valid schedule, perturbing the harvest period at a random stand and repairing the resulting period adjacency violations in an ever-expanding ripple from the original perturbation, based on the centroids of the stands. The perturbation repeats, and the best solution is reverted to after a certain interval. This perturbation and repair process repeats until the termination criteria are met. The number of intervals until reversion is called the reversion rate. Note that there is no set ratio between the number of iterations and the number of fitness evaluations.

4 Particle swarm optimization

Particle swarm optimization (PSO) [3][4] is a stochastic population-based search algorithm, where each member (called a particle) of the population (called a swarm) has a dynamic velocity v and a location x, which is a point in the search space. The particle "flies" through the search space using its velocity to update its location. The particle "remembers" its (local) best location p so far and is aware of the (global) best current location g for any particle in the swarm. The attraction to the former is called the cognitive influence c_1, and the attraction to the latter is called the social influence c_2. Each iteration of the PSO involves evaluating each particle's location according to a fitness function, updating the local and global bests, updating each particle's velocity, and updating each particle's location. The formula to update the velocity and location of a particle in the i^{th} dimension at time t is specified by the following:

$$\begin{aligned} v_i(t) &= \mu v_i(t-1) \\ &+ r_1 c_1 \left(p_i - x_i(t-1)\right) \\ &+ r_2 c_2 \left(g_i - x_i(t-1)\right) \end{aligned} \quad (2)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (3)$$

where μ is the inertia, r_1 and r_2 are random numbers between 0 and 1, and t is the new iteration step.

[1] Available at: http://www.warnell.forestry.uga.edu/Warnell/Bettinger/planning/index.htm

4.1 The proposed PSO

In the proposed multi-valued algorithm Roulette Wheel PSO (RWPSO), a particle's location in each dimension is generated by a roulette wheel process over that particle's roulette wheel probabilities (here called its velocities) in that dimension; in a given dimension, the particle has a velocity for every permissible location in that dimension. The algorithm deterministically updates the velocity of each permissible location *k* in each dimension *i* for each particle using the following formulas:

$$v_{i,k}(t) = v_{i,k}(t-1) + m\begin{pmatrix} s\big(B(g_i,k) - B(x_i(t-1),k)\big) \\ +(1-s)\big(B(p_i,k) - B(x_i(t-1),k)\big)\end{pmatrix} \quad (4)$$

$$\text{where } B(x_i,k) = \begin{cases} 1 & if\ x_i = k \\ 0 & else \end{cases} \quad (5)$$

and *m* is the maximum step size, *s* is the social emphasis, and (1-*s*) is the cognitive emphasis. The velocity $v_{i,k}$ is limited to the range [0,1], but is initially set to the reciprocal of the number of permissible values in dimension *i* for two reasons: (1) the velocities represent roulette probabilities and so together should sum to 1, and (2) with no domain knowledge of the likelihood of each location *k* in the optimal solution, there is no reason to favor one location over another. As previously mentioned, the particle's location in dimension *i* is determined by a roulette wheel process, where the probability of choosing location *k* in dimension *i* is given by:

$$P(x_i(t) = k) = \frac{v_{i,k}(t)}{\sum_{a=1}^{4} v_{i,a}(t)} \quad (6)$$

RWPSO parameters include the swarm size, the stopping criterion, the maximum step size, and the social emphasis. The maximum step size controls the maximum rate at which the RWPSO velocities will change and thus determines how sensitive the RWPSO is to the local and global fitness bests. The social emphasis parameter determines what fraction of the maximum step size is portioned to attraction to swarm global best versus how much is portioned to attraction to the local best.

RWPSO is applicable to problems with nominal (un-ordered, finite, and discrete) variables. In this forest planning problem, harvest period variables can be considered nominal variables, because not only are they discrete and finite, but there is also no obvious ordering or "between" relation to the periods with respect to the objective value.

The algorithm is robust because it lacks inherent bias in the variable value encodings, and because it is not a specialized algorithm, it is useable as-is on a variety of problems. In this paper, the only (eventual) modification is the biasing of initial velocities, which is a modification applicable to any other problem where there is a little domain knowledge about the likelihood of certain variable values in the optimal solution.

Note that as with any parameterized heuristic method, time must be spent to experiment with various parameter combinations to tweak the algorithm for a given problem. This is addressed for the forest planning problem later in this paper.

5 RWPSO guide functions

Since RWPSO will generate schedules that are not necessarily valid, the original objective function f_1 cannot be used to guide the RWPSO, because f_1 does not penalize such schedules. Thus, two fitness functions are derived that provide penalized values for invalid schedules and also provide values for valid schedules which are identical to those produced by f_1.

5.1 RWPSO guide function f_2

The fitness function f_2 is defined as:

$$f_2 = \sum_{k=1}^{z}\left(T - \sum_{n=1}^{d} V_{n,k}\right)^2 \quad (7)$$

where

$$V_{i,t} = \begin{cases} a_n h_{n,k} & if\ s_n = k\ AND\ \nexists q, s_{q\in adj(n)} = k \\ 0 & Otherwise \end{cases}$$

and

$$adj\ (n) = \{h | h\ is\ a\ stand\ adjacent\ to\ n\}$$

where *s* is the harvest schedule, and s_n is the scheduled harvest time of stand *n*.

Essentially, this function penalizes infeasible solutions by harvesting only those stands that do not share a common scheduled harvest period with an adjacent stand; this is effectively a temporary repair on the schedule to bring it into feasible space for fitness evaluation by omitting all scheduled stand harvests that are part of an adjacency violation. As with f_1, $a_n h_{n,k}$ will be either the volume from harvesting an entire stand *n* at time *k* or zero if that stand is not scheduled for harvest at time *k*.

5.2 Alternate RWPSO guide function f_3

The final fitness function f_3 uses a copy of the harvest schedule *s*, denoted s', and modifies it throughout the fitness evaluation. It is the harmonic mean of f_2 and f_3' defined as:

$$f_3 = 2\frac{f_2 \cdot f_3'}{f_2 + f_3'} \quad (8)$$

Where

$$f_3' = \sum_{k=1}^{z}\left(T - \sum_{n=1}^{d} V'_{n,k}\right)^2 \quad (9)$$

$$V'_{i,t} = \begin{cases} a_n h_{n,k} & if\ s'_n = k\ AND\ best(n) \\ 0 & Otherwise \end{cases}$$

$$best(n) = \begin{cases} T & if\ a_n h_{n,s'_n} > \sum_{q \in adj(n)} a_q h_{q,s'_q}\ ; \\ & side\ effect: (\forall y)\left((y \in adj(n) \wedge s'_y = s'_n) \rightarrow s'_y = 0\right) \\ F & Otherwise\ ; \\ & side\ effect: s'_n = 0 \end{cases}$$

$$adj(n) = \{h | h\ is\ a\ stand\ adjacent\ to\ n\}$$

Fitness function f_3 combines the strict penalizing fitness function f_2 with the more lenient function f_3'. The function f_3' creates a copy of the schedule s and modifies this copy s' during the fitness evaluation. The function iteratively considers each stand in a schedule, and whenever an adjacency violation is reached, the harvest volume of the currently considered stand is compared to the total harvest sum of the adjacent stands having the same harvest period. If the former is greater, then the stands adjacent to the current stand that violate the adjacency constraint are set to no-cut in s'. Otherwise, the current stand's harvest schedule is set to no-cut in s'. As with f_1, $a_n h_{n,k}$ will be either the volume from harvesting an entire stand n at time k or zero if that stand is not scheduled for harvest at time k.

Note that for every feasible schedule, if the schedule is given to all three fitness functions, each will yield identical fitness values, because the difference between them is in how each one temporarily repairs an infeasible schedule in order to give it a fitness value; f_1 does no repair, f_2 does a harsh repair, and f_3 combines f_2 with a milder repair f_3'.

6 Tests

Having a comparable measure of process time poses a problem in determining the statistics of RO, because unlike RWPSO, the number of fitness evaluations is not the same as the number of candidate solutions. In fact, RO may use many fitness evaluations in the process of mitigating infeasibilities before offering a single candidate solution. Thus, two sets of statistics will be offered for RO, where RO_c specifies the case of limiting the number of candidate solutions to 1,000,000, and RO_f specifies the case of limiting to 1,000,000 the number of fitness evaluations over 10 trials on the 40- and 73-stand forests. Each RWPSO parameter combination was allowed to run for 10 trials of 1,000,000 fitness evaluations on the 40- and 73-stand forests. In order to allow the algorithms more time on a more difficult problem, both algorithms were run for 5 trials of 5,000,000 fitness evaluations on the 625-stand forest.

To find good parameter combinations, RO was run for 10 trials with reversion rates from 1 to 10 in steps of 1 on the 73-stand forest. RWPSO was run for 10 trials with f_2 on the 73-stand forest with all combinations of the following parameters:

Swarm Size: {20, 40, 80, 160, 320, 640, 1280, 2560, 5120}
Max Step Size: {0.01, 0.05, 0.09, 0.13}
Social emphasis: {0.0, 0.25, 0.50, 0.75}

Even though RWPSO was tested over roughly 15 times the number of parameter combinations of RO, run-time to complete all combinations was roughly the same between RO and RWPSO. Note also that both algorithms were rather forgiving in terms of performance over parameter combination variations. The best parameter combinations from this were used on the remainder of the tests. Tests where RWPSO used f_2 are denoted $RWPSO_{f2}$. Similarly, tests where RWPSO used f_3 are denoted $RWPSO_{f3}$.

One final variation on the configuration used for RWPSO is denoted $RWPSO_{f3\text{-pb}}$. This configuration involves biasing the initial velocities of the RWPSO using some expectation of the likelihood of the no-cut option being included in the optimal schedule. It is expected that an optimal schedule will have few no-cuts in its schedule. However, there is no expectation of the other harvest period likelihoods. Therefore, the initial probabilities were tested for the following cases:

$$\left(v_{i,0}(0), v_{i,1}(0), v_{i,2}(0), v_{i,3}(0)\right) \in \begin{Bmatrix} (0.01, 0.33, 0.33, 0.33), \\ (0.04, 0.32, 0.32, 0.32), \\ (0.07, 0.31, 0.31, 0.31), \\ (0.10, 0.30, 0.30, 0.30) \end{Bmatrix}$$

7 Results

As with [1], the best reversion rate found for RO was 4 iterations. Similarly, the best parameter combination found for RWPSO was swarm size 640, max step size 0.05, and a social emphasis of 0.25. Of the initial no-cut probabilities tried, 0.04 gave the best objective values.

Table 3 shows the results of running each algorithm configuration on the 73-stand forest. Clearly, the choice of how RO is limited—either by candidate solutions produced or by the number of objective function evaluations—will substantially affect the solution quality. In fact, if the number of objective function evaluations is considered as the termination criterion for both algorithms, then every configuration of RWPSO outperforms RO on the 73-stand forest. However, the use of f_3 improves the performance of RWPSO over RO, regardless of the termination criterion used for RO. Also, note that although the use of biased initial no-cut probability makes $RWPSO_{f3\text{-pb}}$ outperform $RWPSO_{f3}$, the largest gains by RWPSO in terms of average objective values come from using f_3 instead of f_2. Additionally, changing the function that guides RWPSO drastically decreases the standard deviation of the solution quality.

Table 3: 73-Stand Forest Results

	Best Solution	Average Solution	Standard Deviation
RO_c	5,500,330	6,729,995	1,472,126
RO_f	5,741,971	8,589,280	2,458,152
$RWPSO_{f2}$	5,500,330	7,492,459	1,920,752
$RWPSO_{f3}$	5,500,330	5,844,508	450,614
$RWPSO_{f3\text{-}pb}$	5,500,330	5,786,583	437,916

The distribution of the 73-stand forest results from an additional set of trials is given in the following graph:

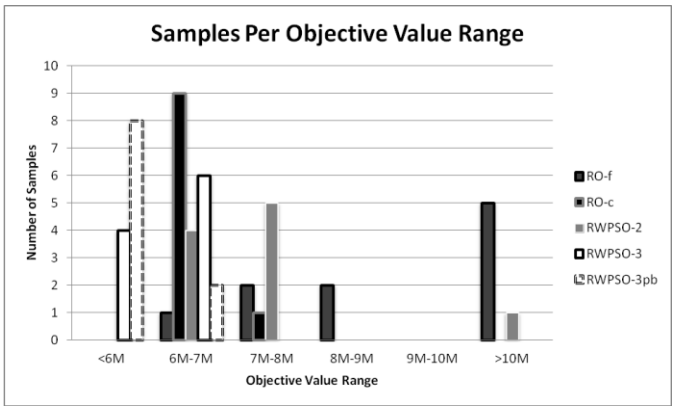


Figure 3: 73-stand forest distribution of samples

The results for the 40-stand forest in Table 4 are similar to those in Table 3, except that every configuration of RWPSO outperforms RO, regardless of the termination criterion used. Additionally, RWPSO's greatest increase in performance came through using f_3 instead of f_2.

Table 4: 40-Stand Forest Results

	Best Solution	Average Solution	Standard Deviation
RO_c	90,490	160,698	46,879
RO_f	98,437	190,567	63,314
$RWPSO_{f2}$	90,490	151,940	46,431
$RWPSO_{f3}$	90,490	123,422	30,187
$RWPSO_{f3\text{-}pb}$	90,490	113,624	23,475

The distribution of the 40-stand forest results from an additional set of trials is shown in the following graph:

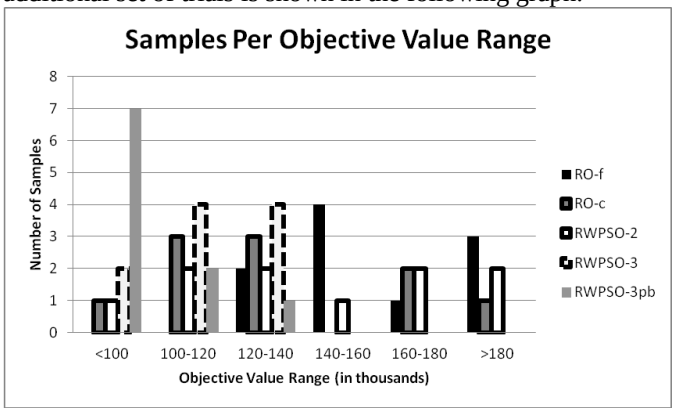


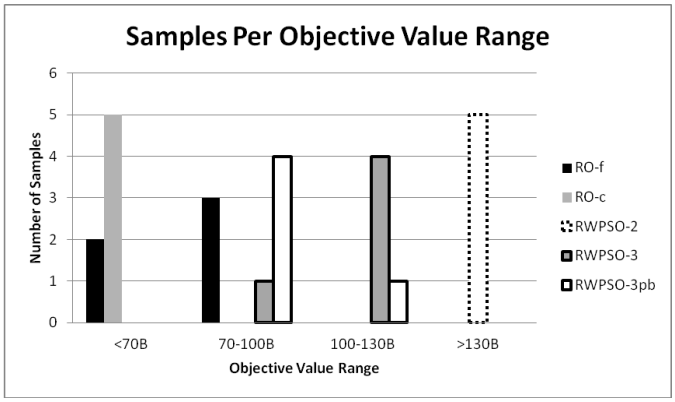
Figure 4: 40-stand forest distribution of samples

The results from Table 5 illustrate that the 625-stand forest is much more difficult for RWPSO than either of the other forests. RO with both termination criteria outperformed every configuration of RWPSO. It can be noted that, just like for the other forests, the use of f_3 improved RWPSO's performance over f_2, and the use of biased initial no-cut probability further improved the quality of the RWPSO solutions.

Table 5: 625-Stand Forest Results

	Best Solution	Average Solution	Standard Deviation
RO_c	61,913,898,152	66,142,041,314	2,895,384,577
RO_f	66,223,010,632	72,552,142,872	3,732,367,645
$RWPSO_{f2}$	118,239,623,212	150,819,800,640	14,487,010,747
$RWPSO_{f3}$	91,224,899,372	100,862,133,880	6,894,894,714
$RWPSO_{f3\text{-}pb}$	87,444,889,432	95,872,673,094	6,808,823,161

The distribution of the 625-stand forest results from an additional set of trials is given in the following graph:



8 Discussion

Some comparisons to the studies in Section 1 may be made, limited by the fact that they limited the runtimes or iterations differently. Additionally, most of those studies deal only with the 73-stand forest.

In [1], comparisons are difficult to make, because the number of fitness evaluations was not recorded. In that paper, the best performance on the 73-stand forest was a best fitness of 5,556,343 and an average of 9,019,837 via RO. On the 40-stand forest, it was a best of 102,653 and an average of 217,470, and on the 625-stand forest via RO, it was a best of 69B and an average of 78B via RO. In [7], the DPSO was allowed to run on the 73-stand forest up to 2,500 iterations with swarm sizes in excess of 1000, which translates to a maximum of 2.5M fitness evaluations for a 1000 particle swarm. With DPSO, they obtained a best fitness in the range of 35M. In that paper, the best fitness value found was 5,500,391 via RO. In [2], the best performance from their PPSO on the 73-stand forest was with 2,500 iterations of a size 500 swarm, which translates to 1,250,000 fitness evaluations. They achieved a best fitness of 5,500,330 and an average fitness of 8,831,332.

In comparison, the best results from this paper on the 73-stand forest are a best of 5,500,330 and an average of 5,786,583 after 1M fitness evaluations. RWPSO obtained the best results of all the studies discussed for the 73-stand forest. Similarly, it outperformed on the 40-stand forest with a best objective value of 90,490 and an average of 113,624. However, on the 625-stand forest, RO outperforms RWPSO, regardless of the termination criterion used.

9 Conclusion and future directions

The functions f_2 and f_3 promote Lamarckian learning by assigning fitnesses to infeasible schedules based on nearby feasible schedules. The use of Lamarckian learning may be beneficial in general on similar problems, and additional research would be required to test this.

RWPSO was formulated specifically for multi-valued nominal variable problems, and it treats velocity more explicitly as a roulette probability than do other probability based PSOs. Additionally, by expressing parameters of the algorithm in terms of their effects on the roulette wheel, parameter selection is more intuitive. Future work needs to be done to determine if this explicit roulette wheel formulation yields any benefit in general to RWPSO's performance over other discrete PSOs.

10 References

[1] P. Bettinger, J. Zhu. "A new heuristic for solving spatially constrained forest planning problems based on mitigation of infeasibilities radiating outward from a forced choice"; in Silva Fennica, vol. 40(2): p315-33, 2006.

[2] P. W. Brooks, and W. D. Potter. "Forest Planning Using Particle Swarm Optimization with a Priority Representation"; in *Modern Approaches in Applied Intelligence*, edited by Kishan Mehrotra, Springer, *Lecture Notes in Computer Science*, vol. 6704: p312-8, 2011.

[3] R. Eberhart, and J. Kennedy. "A new optimizer using particle swarm theory"; in Proceedings *6th International Symposium Micro Machine and Human Science,* Nagoya, Japan, p39-43, 1995.

[4] J. Kennedy, and R. Eberhart. "Particle Swarm Optimization"; in Proceedings *IEEE International Conference Neural Network*, Perth, WA, Australia, vol. 4: p1942-8, 1995.

[5] J. Kennedy, and R. Eberhart. "A Discrete Binary Version of the Particle Swarm Algorithm"; *IEEE Conference on Systems, Man, and Cybernetics*, Orlando, FL, vol. 5: p4104-9, 1997.

[6] J.H. Holland. *Adaptation in Natural and Artificial Systems*; Ann Arbor, MI: The University of Michigan Press, 1975.

[7] W.D. Potter, E. Drucker, P. Bettinger, F. Maier, D. Luper, M. Martin, M. Watkinson, G. Handy, and C. Hayes. "Diagnosis, Configuration, Planning, and Pathfinding: Experiments in Nature-Inspired Optimization"; in *Natural Intelligence for Scheduling, Planning and Packing Problems*, edited by Raymond Chong, Springer-Verlag, *Studies in Computational Intelligence*, vol. 250: p267-94, 2009.

Parallel Parametric Optimisation with Firefly Algorithms on Graphical Processing Units

A.V. Husselmann and K.A. Hawick
Computer Science, Institute for Information and Mathematical Sciences,
Massey University, North Shore 102-904, Auckland, New Zealand
email: { a.v.husselmann, k.a.hawick }@massey.ac.nz
Tel: +64 9 414 0800 Fax: +64 9 441 8181

March 2012

Abstract

Parametric optimisation techniques such as Particle Swarm Optimisation (PSO), Firefly algorithms (FAs), genetic algorithms (GAs) are at the centre of attention in a range of optimisation problems where local minima plague the parameter space. Variants of these algorithms deal with the problems presented by local minima in a variety of ways. A salient feature in designing algorithms such as these is the relative ease of performance testing and evaluation. In the literature, a set of well-defined functions, often with one global minimum and several local minima is available to evaluate the convergence of an algorithm. This allows for simultaneously evaluating performance as well as the quality of the solutions calculated. We report on a parallel graphical processing unit (GPU) implementation of a modified Firefly algorithm, and the associated performance and quality of this algorithm. We also discuss spatial partitioning techniques to dramatically reduce redundant entity interactions introduced by our modifications of the Firefly algorithm.

Keywords: optimisation; firefly algorithm; GPU; CUDA; spatial partitioning.

1 Introduction

Research towards metaheuristic optimisation algorithms has begun as far back as 1975, in which John Holland introduced Genetic Algorithms [1]. Several years after this discovery, Simulated Annealing followed in 1983, which is inspired by the annealing process in metallurgy [2]. It was another 12 years before Kennedy and Eberhart developed the Particle Swarm Optimiser (PSO), which lead to the development of the Firefly Algorithm (FA) by Yang [3], which was introduced in 2008. Yang observed better performance in the Firefly algorithm than the standard PSO, as did Aungkulanon and Chai-ead [4]. As can be seen by the great milestones of computational metaheuristic optimisation, algorithms in this domain generally share a natural or biological inspiration.

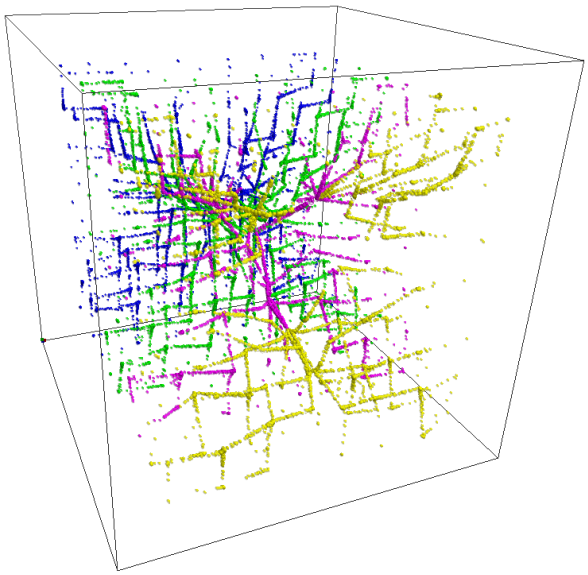


Figure 1: 65536 fireflies attempting to optimise a 3-parameter generalised Rosenbrock Function. The global minimum is at coordinates $(1, 1, 1)$, which is near the centre of the box. The Rosenbrock function is characterised by a low-lying valley, which is easy to find, but the minimum inside this valley is more difficult to find.

Parametric optimisation has been an area of interest for decades however, with early techniques such as linear programming. As computational power increased in latter years, interest and scientific inquiry has increased exponentially in the search for algorithms to effectively exploit this computing power. This is even more so with

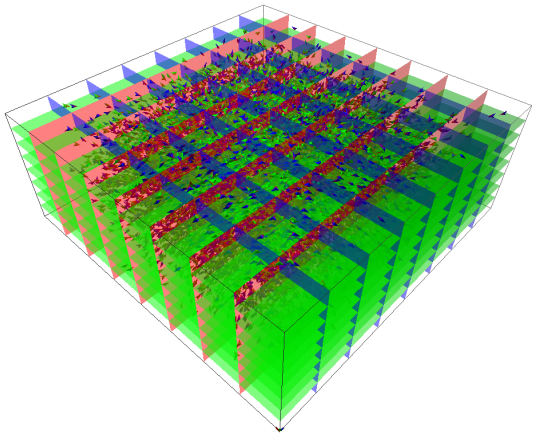


Figure 2: A visualisation of a uniform grid datastructure.

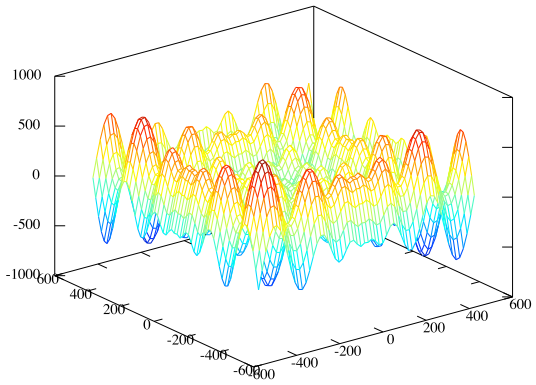


Figure 3: A surface plot of the 3-parameter Schwefel function.

the development of multi-core chips and indeed a massively parallel environment such as NVidia's Compute Unified Device Architecture (CUDA). Graphical Processing Units (GPUs) [5] have shown excellent ability in accelerating agent-based simulations, and continues to impress with newer algorithms.

Problems involving parametric optimisation are plentiful in the areas of image compression [6, 7], manufacturing improvement [4], structure design [8], scheduling problems [9], cryptanalysis [10], object clustering/recognition [11], economics [12–14], structure design [8, 15], Antenna design [16, 17], isospectral systems [18] and more. Generally, in parameter-based optimisation, one attempts to obtain a vector x with d dimensions (parameters) which minimises a scalar-valued function $f(x)$. In practice, the function $f(x)$ is not normally known. This makes parametric optimisation very versatile. However, when evaluating the performance of algorithms such as these, the function $f(x)$ is known exactly. An example of this is the Rosenbrock function, given in Equation 1. For this equation, the global minimum is $f(x, y) = 0$ for $x = y = 1$.

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \qquad (1)$$

To facilitate the evaluation of n-parameter optimisers, these functions are normally generalised in some way. For Rosenbrock's function, a common generalisation is shown in Equation 2. Similar generalisations exist for most of the other evaluation functions.

$$f(x) = \sum_{i=0}^{n-2} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \qquad (2)$$

For the purpose of visualisation, we restricted the number of parameters to 3. This is a severely limiting attribute in real-world applications, but the visualisation of the algorithm is extremely useful.

The Firefly algorithm [3] is a homogeneous, metaheuristic, evolutionary optimisation algorithm, and a recent addition to the particle-based optimiser family. Macroscopic effects of this algorithm is reminiscent of emergence in agent-based systems. It finds several similarities with the standard PSO algorithm, but in the original article, Yang states that the PSO is simply a special case of the Firefly algorithm. In nature, the displays of flashing lights from fireflies is associated with mating habits. This served as the source of inspiration for Yang, who idealised the biological phenomenon with a few assumptions. Firstly, the algorithm would have unisex fireflies which would be attracted to any other fireflies regardless. Secondly, the attractiveness of a firefly must be essentially proportional to the objective function. For minimisation problems which we discuss, the attractiveness of a firefly is inversely proportional to the objective function value.

The use of these homogeneous spatial agents involves each firefly performing a movement calculation based on a deterministic part and a stochastic part in conjunction with other spatially local agents. While not always local, distant fireflies have a degraded influence. Depending on the parameter count, the parameter space is easily modified (albeit less able to be visualised).

By making use of what can be seen as the *de facto* standard naming convention, the α-step and the β-step, every firefly is influenced by every other firefly in a deterministic and stochastic way respectively. The α-step refers to the stochastic space exploration, and the β-step refers to the deterministic bias towards other fireflies with better solutions in the parameter space.

As described by Yang, the update formula for any two fireflies x_i and x_j is shown in equation 3.

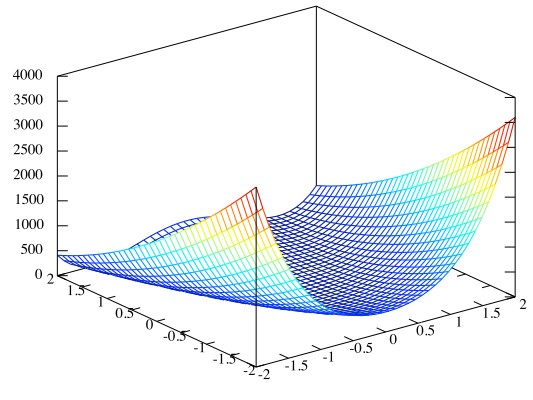


Figure 4: A surface plot of the 3-parameter Rosenbrock function.

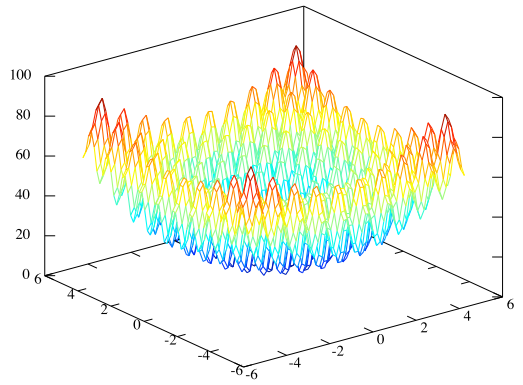


Figure 5: A surface plot of the 3-parameter Rastrigin function.

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(d) \qquad (3)$$

Apart from advancing the firefly based on its previous position, the equation begins with the β-step. β_0 refers to an attractiveness coefficient, while the exponential $e^{-\gamma r_{ij}^2}$ degrades the attractiveness of firefly x_j over distance r_{ij}. The second term is the α-step. The α variable is the random step size, where d refers to a random vector.

As can be seen from this equation, the algorithm has an inherent $\mathcal{O}(n^2)$ complexity, since every firefly x_i must evaluate this equation n times, for every other firefly x_j. This complexity is not normally reduced because of its scaling problems, but because of diversity loss in fireflies in the parameter space [19]. There are a handful of algorithms which attempt to rectify this problem, most of which involve maintaining multiple independent swarms [19, 20]. Our algorithm differs from these approaches by very loosely maintaining separate swarms, and allowing full interaction between these swarms. By limiting the interaction distance, we also permit ourselves to use acceleration data-structures to dramatically increase performance and hence allow the use of much, much larger systems which greatly increase accuracy and reduces computation time necessary.

Some report that the FA responds well to hybridisation, especially with algorithms from the realm of artificial intelligence [19, 21]. As with PSO, the FA does require tuning, but far less than the standard PSO.

Some popular functions for evaluating the performance of optimisation algorithms include the Rosenbrock function, Ackley's Path function, the Schwefel Function and the Rastrigin function. Each of these have vastly different appearances, and their 3-parameter counterparts are shown in Figures 2, 4, 5 and 3. For visualisation, these functions are best constrained to 3 parameters (dimensions) or less.

It is important to note that these have very different characteristics. As we observed from our results in Section 3, unimodal and multimodal test functions can have staggering performance implications.

2 CUDA Fixed-Interaction Firefly Algorithm

There have been various attempts to parallelise swarm intelligence and particularly the PSO [20, 22–24]. A few of these include CUDA implementations.

Our algorithm differs greatly in two aspects from the original FA. Firstly, our algorithm can support large numbers of fireflies due to optimised data-structures. The second difference is that we impose a maximum interaction distance, which is dramatically smaller than the values some authors have suggested. In this limited interaction distance, we still degrade attractiveness of fireflies that are more distance, but at roughly the same rate as the original algorithm would. These two differences go hand-in-hand as the interaction distance allows us to accelerate the simulation. According to authors who implement multi-swarm modifications in the FA, smaller interaction distances improves accuracy when there are local minima to avoid, as is often the case [19].

Following from Equation 3, we first modified the update to incorporate our smaller interaction distance. This is shown in Equation 4.

$$x_{i+1} = x_i + \beta e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(d) \qquad (4)$$

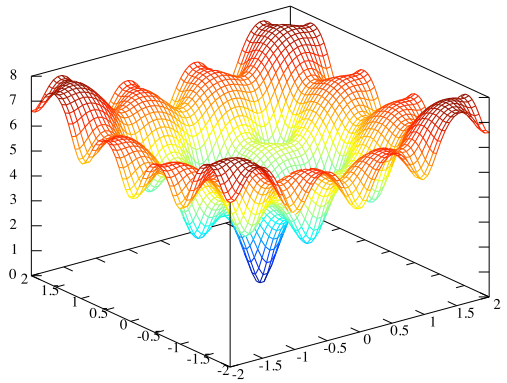


Figure 6: A surface plot of the 3-parameter version of Ackley's Path function.

Where we specifically choose γ to be

$$\gamma = \frac{1}{g^2(1 + ln(\beta))}$$

with g being size of a grid box. This will ensure that the β step will give less notice to fireflies that are at the edge of visibility to a particular firefly. In our experience, this does not seem to provide far better results, but it will preserve some diversity. Instead of this, for simplicity and speed, we multiply the β-step with $N(r)$, as defined below.

$$N(r) = \begin{cases} 0 & r > m \\ r & r < m \end{cases}$$

This is very simple, and by choosing a suitable γ, we need not add extra complexity as indicated above.

When optimising a function, it is beneficial to have as many fireflies or agents as possible to avoid convergence to local minima. We are able to simulate large numbers of fireflies with this algorithm by means of acceleration data-structures which dramatically reduce redundant entity interactions. We use a method named grid-boxing parallelised for CUDA (sometimes known as uniform-grid space partitioning) [25] to localise interactions of fireflies, in order to improve performance greatly. Our single-GPU implementation of this algorithm is similar to the data-structure used by the Particle simulation shipped with the CUDA SDK [26]. This allows separate (yet fully interacting) swarms to search in parallel, while still allowing complete interaction between these swarms. The simulation is not initialised with a random set of independent swarms. Rather, these seemingly separate swarms spontaneously form as a side effect of using grid-boxing with a suitably chosen interaction distance.

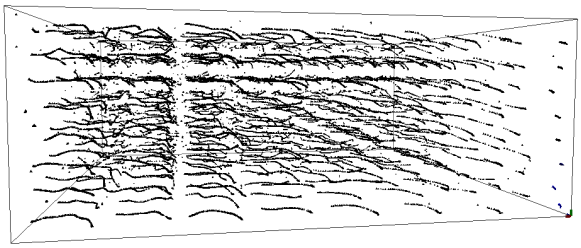


Figure 7: 4096 fireflies solving a 3-parameter De Jong function.

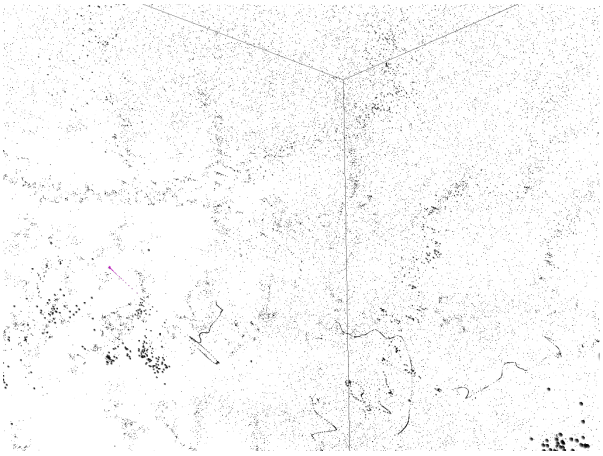


Figure 8: 262,244 fireflies optimising a 3-parameter Rosenbrock function.

The α-step in the update formula requires a random vector d. This is a significant problem, considering that this code executes almost entirely on GPU. Fortunately, a library named CURAND is available directly from NVidia, distributed within the CUDA SDK. This library provides a high performance Mersenne Twistor(MT), which we use to generate enough d vectors to satisfy the α-steps for all the fireflies in the simulation There is a complex tradeoff between quality and raw performance of random number generators [27], but we believe the MT is a suitable compromise for both criteria for teh work reported here.

3 Performance Evaluation

In our performance evaluations, we primarily used an Intel Core i7 server running at 3.4GHz, configured with two NVidia GTX 590 graphics cards. The implementation of the original firefly algorithm was obtained from [28]. This program includes a hybridised Firefly algorithm, which we did not use in our evaluations. This modified algorithm by Mancuso modifies the size of the α step size before every simulation step.

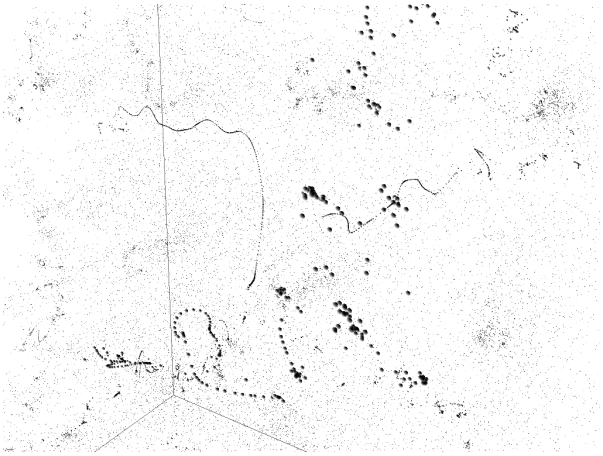


Figure 9: 262,244 fireflies optimising a 3-parameter Rosenbrock function.

	CPU	GPU
Rosenbrock 3D		
Time (msec)	368455.2	9488.725
Minimum	0.000071	0.000045
Rastrigin 3D		
Time (msec)	367329.1	966.5312
Minimum	1.445014	1.174144
Schwefel 3D		
Time (msec)	369934.9	848.6163
Minimum	73.26784	25.47458
Ackley's Path 3D		
Time (msec)	368384.2	949.9359
Minimum	1.138163	3.060646

Table 1: CPU vs GPU Parallel Firefly algorithm in optimising the 3-parameter Rosenbrock function.

To compare algorithms we use the 3-parameter version of the Rosenbrock function, which is shown in Figure 4. This function is as follows:

$$f(x) = \sum_{i=0}^{n} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \qquad (5)$$

In Equation 5, x_i refers to parameter i, n refers to the number of parameters, which in our simulations is set to a constant 3, so that we can easily view the results. For this function, the global minimum is $f(x) = 0$ when $x = (1, 1, 1)$ when $n = 3$.

Table 3 contains some performance data for the comparison between the original CPU Firefly algorithm and our GPU Parallel Firefly algorithm. The data was averaged over 100 runs. Both the CPU and the GPU parallel algorithms were configured with 4096 particles/fireflies, and set to run for exactly 600 full simulation steps. Configuration for each of the evaluation functions are shown in

	Boundary (min,max)
Rosenbrock 3D	-2.0, 2.0
Rastrigin 3D	-5.0, 5.0
Schwefel 3D	-500, 500
Ackley's Path 3D	-20, 20

Table 2: CPU vs GPU Parallel Firefly algorithm in optimising the 3-parameter Rosenbrock function.

Table 3. The optima for each of the four algorithms is as follows:

1. Rosenbrock function:
   for $f(x, y, z)$, $f(1, 1, 1) = 0$.
2. Ackley's Path function:
   for $f(x, y, z)$, $f(0, 0, 0) = 0$.
3. Rastrigin function:
   for $f(x, y, z)$, $f(0, 0, 0) = 0$.
4. Schwefel function:
   for (x, y, z), $f(420.9, 420.9, 420.9) = 0$.

We modified our Schwefel function by adding $3(418.9829)$ to the function, in order to yield a minimum of 0.

The algorithms were both configured to randomly distribute particles in the ranges described in Table 3 in each parameter range. We removed boundary checks in the CPU Firefly algorithm to more closely resemble the implementation of our GPU algorithm, but the CPU algorithm is implemented in double precision, whereas the GPU algorithm is implemented in single precision. This makes comparison more difficult, but with the great margin of performance difference obtained, the differenceb becomes more pronounced.

The results show a clear difference in performance, both in computation time, and accuracy of the solution, albeit, the latter is not true for Ackley's Path function. The accuracy of the algorithm can be attributed to the difference in interaction distance, making it more resistant to converging to a local minimum instead of the global minimum. The decreased computation time can be immediately attributed to the parallel environment. The local minma of the Ackley's Path function is not as pronounced as the Schwefel function or Rastrigin function, so it seems possible that global interaction is indeed more suitable for this function. Another consideration is that all particles in the CPU Firefly algorithm will eventually reach the area around the global minimum, which means there will be more fireflies in that area.

So far we have compared the quality of the optimisers and the performance to some extent, but good per-

formance is possible to achieve with far less fireflies, depending on the application (unimodal or multimodal functions). Our testing revealed that the CPU Firefly algorithm can easily outperform the GPU algorithm with a simple unimodal function such as the Rosenbrock function, by simply having 16 fireflies which obtains an error of less than 0.00005 in approximately 25msec (10-run average). The GPU counterpart is only effective from 4096 fireflies up, and this takes on a 10-run average approximately 338msec for an error less than 0.00005. A clear advantage. However, when facing multimodal functions such as the Schwefel function, Rastrigin function and Ackley's path function with low numbers of fireflies, the CPU algorithm either fails to achieve the global minimum, or takes an inordinate amount of time to compute it (days, weeks).

The GPU algorithm can easily optimise multimodal functions such as the Schwefel function in a fraction of the time it would take the CPU algorithm. Roughly 2048 or more fireflies would be needed to ensure the algorithm saturates the parameter space enough to obtain the global minimum. As seen above, the GPU speed-up over the CPU firefly algorithm is 39 times for 4096 fireflies, this would mean the GPU algorithm is far superior in multi-modal optimisation, whereas the CPU firefly algorithm is far superior in unimodal optimisation, where large numbers of fireflies are not necessary. It is difficult to compare with absolute certainty in meta-heuristic optimisation, but in our observations, we could not use the CPU firefly algorithm for optimising the multimodal functions with less than 2000 fireflies - Schwefel, Rastrigin, and Ackley's Path in reasonable time at all, whereas the GPU algorithm could optimise all three within one second.

4 Discussion

Our algorithm seems to perform more accurately and faster than the original Firefly algorithm. There is however, a clear difference when comparing multimodal functions to unimodal functions. The original Firefly algorithm is very well suited to optimising unimodal functions, as very few fireflies are required, and frame calculation times are dramatically lower. However, for multimodal functions, it is imperative to have a larger number of fireflies to avoid local minima. This results in an $\mathcal{O}(n^2)$ complexity which our algorithm reduces to $\mathcal{O}(n \log n)$ and also parallelises. The acceleration data-structure allows this, thanks to a small modification we made to the original firefly algorithm, namely the smaller and fixed interaction distance.

Using ideas contained within this new GPU algorithm, it is possible to dramatically increase system sizes, and efficiently optimise multimodal functions. However, it is well worth noting that we only discuss 3-parameter functions in this article. In practice, it is common to find functions which require hundreds of parameters. The Firefly algorithm easily adapts to this by simply moving through n-dimensional space towards other fireflies, and having an n-dimensional random step. A CUDA implementation which allows n-dimensional optimisation will require extra considerations to be made, as it will require far more storage among others.

We observe that the greater the number of fireflies, the more likely it is that the global minimum will be obtained. This is certainly the case with Ackley's Path function, as the GPU algorithm could only obtain the global minimum (0) consistently when run with 266,144 fireflies. This serves to saturate the parameter space to the extent that the global minimum is obtained. It is noteworthy however, that with 262,144 fireflies, it takes substantially longer to compute 600 frames, but it is still within two minutes on average.

5 Conclusions and Future Work

We have presented a GPU-based Firefly algorithm with a fixed-interaction distance and a uniform-grid acceleration data-structure. We compared this algorithm in accuracy and performance to the original single-threaded Firefly algorithm, and found a vast performance increase, but only for multimodal test functions. Global interaction in very few numbers of fireflies could still outperform the GPU algorithm in a unimodal function such as Rosenbrock's function. However, we observed that the multimodal test functions could only be optimised with the GPU algorithm, and it consistently does so in less than one second for 4096 fireflies. This is approximately a 39-fold speedup over the same number of fireflies simulated by the CPU-based algorithm.

We conclude that for massively multimodal functions such as the Schwefel function, the GPU algorithm is by far the better choice. This is especially applicable to functions which also have geometrically distant global minima.

In future we will explore methods of effectively parallelising this algorithm across several GPUs in order to increase system sizes with a much smaller computational cost. We will also aim to support n-dimensional optimisation problems. These data-parallelisable optimisation algorithms show great promise across a range of complex systems applications.

References

[1] Holland, J.H.: Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press (1975)

[2] Kirkpatrick, e.a.: Optimization by simulated annealing. Science **220** (1983) 671–680

[3] Yang, X.S.: Firefly algorithms for multimodal optimization. In: Stochastic Algorithms: Foundations and Applications, SAGA. (2009)

[4] Aungkulanon, P., Chai-ead, N., Luangpaiboon, P.: Simulated manufacturing process improvement via particle swarm optimisation and firefly algorithms. Prof. Int. Multiconference of Engineers and Computer Scientists **2** (2011) 1123–1128

[5] Leist, A., Playne, D., Hawick, K.: Exploiting Graphical Processing Units for Data-Parallel Scientific Applications. Concurrency and Computation: Practice and Experience **21** (2009) 2400–2437 CSTN-065.

[6] Horng, M.H.: Vector quantization using the firefly algorithm for image compression. Expert Systems with Applications **38** (2011)

[7] Horng, e.a.: The codebook design of image vector quantization based on the firefly algorithm. Computational Collective Intelligence, Technologies and Applications **6426** (2010) 438–447

[8] Azad, S.K., Azad, S.K.: Optimum design of structures using an improved firefly algorithm. International Journal of Optimization in Civil Engineering **1** (2011) 327–340

[9] Khadwilard, e.a.: Application of firefly algorithm and its parameter setting for job shop scheduling. First Symposium on Hands-On Research and Development **1** (2011)

[10] Palit, S., Sinha, S.N., Molla, M.A., Khanra, A., Kule, M.: A cryptanalytic attach on the knapsack cryptosystem using binary firefly algorithm. Int. Conf. on Computer and Communication Technology (ICCCT) **2** (2011) 428–432

[11] Senthilnath, J., Omkar, S.N., Mani, V.: Clustering using firefly algorithm: Performance study. Swarm and Evolutionary Computation (2011)

[12] Yang, X.S.: Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Applied Soft Computing **12** (2012) 1180–1186

[13] Giannakouris, G., Dounias, V.V.G.: Experimental study on a hybrid nature-inspired algorithm for financial portfolio optimization. SETN 2010, LNAI 6040 **1** (2010) 101–111

[14] Apostolopoulos, Vlachos: Application of the firefly algorithm for solving the economic emissions load dispatch problem. International Journal of Combinatorics **1** (2011)

[15] Gandomi, A.H., Yang, X.S.: Mixed variable structural optimization using firefly algorithm. Computers and Structures **89** (2011) 2325–2336

[16] Chatterjee, A., Mahanti, G.K., Chatterjee, A.: Design of a fully digital controlled reconfigurable switched beam coconcentric ring array antenna using firefly and particle swarm optimization algorithms. Progress in Electromagnetic Research **B** (2012) 113–131

[17] Basu, B., Mahanti, G.K.: Firefly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna. Progress in Electromagnetic Research **32** (2011) 169–190

[18] Dutta, R., Ganguli, R., Mani, V.: Exploring isospectral spring-mass systems with firefly algorithm. Proc. Roy. Soc. A. **467** (2011)

[19] Farahani, S.M., Nasiri, B., Meybodi, M.R.: A multiswarm based firefly algorithm in dynamic environments. Third Int. Conf. on Signal Processing Systems (ICSPS2011) **3** (2011) 68–72

[20] Mussi, L., Daoilo, F., Cagoni, S.: Evaluation of parallel particle swarm optimization algorithms within the cuda architecture. Information Sciences **181** (2011) 4642–4657

[21] Farahani, S.M., Abshouri, A.A., Nasiri, B., Meybodi, M.R.: Some hybrid models to improve firefly algorithm performance. Int. J. Artificial Intelligence **8** (2012) 97–117

[22] Schutte, J.F., Fregly, B.J., Haftka, R.T., George, A.D.: A parallel particle swarm optimizer. Technical report, University of Florida, Department of Electrical and Computer Engineering, Gainesville, FL, 32611 (2003)

[23] Venter, Sobieszczanski-Sobieski: A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. 6th World Congresses of Structural and Multidisciplinary Optimization **6** (2005)

[24] Zhou, Tan: Gpu-based parallel particle swarm optimization. Evolutionary Computing (2009)

[25] Hawick, K.A., James, H.A., Scogings, C.J.: Grid-boxing for spatial simulation performance optimisation. In T.Znati, ed.: Proc. 39th Annual Simulation Symposium, Huntsville, Alabama, USA (2006) The Society for Modeling and Simulation International, Pub. IEEE Computer Society.

[26] Green, S.: Particle simulation using cuda. Technical report, NVIDIA (2010)

[27] Hawick, K., Leist, A., Playne, D., Johnson, M.: Speed and Portability issues for Random Number Generation on Graphical Processing Units with CUDA and other Processing Accelerators. In: Proc. Australasian Computer Science Conference (ACSC 2011). (2011)

[28] Mancuso, N.: Firefly algorithm implementation. http://code.google.com/p/csc6810project/ (2010)

Smooth Path Planning for Mobile Robot Using Particle Swarm Optimization and Radial Basis Functions

Contact Author: Nancy Arana-Daniel,
Alberto A. Gallegos, Carlos López-Franco and Alma Y. Alanis
Department of Computer Science, CUCEI, University of Guadalajara (UDG),
Av. Revolución 1500, Col. Olímpica, C.P. 44430, Guadalajara, Jalisco, México
Email: nancy.arana, carlos.lopez, alma.alanis@cucei.udg.mx
gallegos.alberto.a@gmail.com

***Abstract*—One of the most important tasks to be performed by a mobile robot is to find a collision-free and smooth path to follow. Given a set of initial control points and using a Radial Basis Function (RBF), a method is proposed, in which is used the RBF's property to approximate smooth functions to define a collision-free and short path. In this paper we formulate the training technique of an RBF as an optimization problem and employed Particle Swarm Optimization (PSO) to solve it.**

The path planning problem is equivalent to optimize the parameters of the RBF using a set of trajectory constraints based on coverage control points as input pattern, which can be seen as places where is desirable for the robot to explore. Furthermore, a combined fitness function is proposed with respect to three requirements: (i) achieve minimum mean square RBF- function approximation error ; (ii) avoid collisions and (iii) minimize the length of the obtained path .

Keywords: Path planning, PSO, RBF.

PAPER TO BE SUBMITED TO: The 2012 International Conference on Genetic and Evolutionary Methods (GEM'12)

I. Introduction

Considerable number of research papers exist in the field of robot path planning. Classic methods used to solve path planning include grid based path planning algorithms as the most commonly used methods [5], [16], [1], [18]. Unfortunately, even though algorithms like A* and D* are complete (they'll always find a solutions if there is one), the paths made by this algorithms lack of smoothness; the robots will often have to stop and readjust their trajectory to continue following the path with every drastic change of direction. For the modeling of the environment, the map is discretized, by doing this, lots of solutions may be excluded, and also it could cause non-smooth paths in algorithms like A*, D* and potential fields. Furthermore, in the case of potential fields, the algorithm could be trapped in a local minimum formed by concave obstacles [1].

Smooth paths are important in robotics because nonholonomic mobile robots are commonly used in practice, so, a smooth path will be more suitable for such robots because this kind of paths are more preferable for designing continuous control algorithms to follow the paths.

New methods based on evolutionary computational methods have been used recently to solve path planning problems due to they are relatively simple to implement, they have fast processing speed, few parameters to be adjusted and good performance. In addition to the above, it is known that the path planning problem can be stated as an optimization (multi-objective optimization) problem and evolutionary computational methods, for instance genetic algorithms (GAs) were used in solving the optimization of path planning successfully [4], [7].

PSO, is a method for optimization of continuous nonlinear functions, created by James Kenedy and Russell Eberhart in 1995 [9]; inspired by the social behavior of bird flocks and school of fishes. In PSO, each individual would be the equivalent of a bird on a flock, each 'bird' is named 'particle', and the 'flock' is called a 'swarm'. A particle is analogous to a chromosome in Genetic Algorithms.
Compared to other Evolutionary Algorithms, classic PSO has no crossover and mutation calculation, actually this is one of the things that makes it really easy to implement. PSO only evolve their social behavior and accordingly their movement towards the best solutions [3]. The search can be carried out by the speed of the particle during the development of several generations, and only the most optimist solution can pass their information over iterations.

The main properties of collective behavior, are a few of the characteristics that make algorithms based in swarm intelligence so effective [15]:

- Homogeneity: Every particle in the swarm has the same behavioral model. The swarm moves without a fixed leader, even if a temporary leader appear.
- Locality: Its nearest swarm mates only influence the motion of each individual.
- Collision Avoidance: Avoid colliding with nearby swarm mates.

- Velocity Matching: Attempt to match velocity with nearby swarm mates.
- Flock Centering: Attempt to stay close to nearby swarm mates.

PSO has proven to have good results in path planning to perform obstacle avoidance [8], [17], [14]. An algorithm for path planning for mobile robot using PSO with mutation operator is proposed in [12]. Its strategy consists in three steps: First the MAKLINK graph is built to describe the working space of the mobile robot. Then, the Dijkstra's algorithm is used to obtain a sub-optimal path and finally PSO is adopted to get the optimal path. In order to generate enough particles in PSO, it has to be chosen a parameter $t_i \in [0, 1]$ for each free-link in the MAKLINK graph. A free-link is defined as a line whose two ends are either corners of two obstacles or one of them is a corner and the other is a point on a working space boundary wall. Therefore, complex environments containing large number of obstacles mean great increase in the complexity of the whole path planning system. Besides, the mutation operator used to avoid local minimum problems consists in a random strategy and it does not make good use of the evolution of population and this approach produces non-smooth paths.

In 2010, an algorithm of improved PSO was applied in mobile robotic path planning [10]. This approach proposes a grid method to model the path space which results in nonsmooth paths, it also includes mutation and crossover operators in order to avoid local minimum. But in addition to avoidance of local optimum these two steps add to PSO algorithm computational complexity.

A smooth path planning of a mobile robot using Stochastic PSO is implemented in [2]. It uses a kind of cubic spline in which coefficients are trained with PSO in order to produce smooth paths. The fitness function with respect to obstacle avoidance makes necessary to know each center of each obstacle in the environment, as well as the calculation of critical points defined as points on the trajectory that are at the minimum distance to an obstacle. So, the fitness function includes these calculations for each one of the M obstacles in the environment.

In this paper it is proposed an approach using Particle Swarm Optimization (PSO) technique to train a Radial Basis Function Network (RBF) used to solve a function approximation problem in order to obtain a smooth path of a mobile robot through an environment containing static obstacles. RBF networks were chosen to be trained with PSO to meet the following objectives: 1) to keep relatively simple the implementation of the path planning system and therefore 2)to get an efficient path planner and 3) To obtain smooth paths making use of their excellent capabilities as function approximators.

Next two sections, II, III show the methods which comprise the path planner system. Section IV describes the particle modeling and the fitness function implementation. Last section, V is devoted to show the simulation results and parameter tuning of the PSO-RBF path planning algorithm.

II. Particle Swarm Optimization, PSO

PSO exploits a population of potential solutions, each solution consists of a set of parameters, representing a point in a search space$A \subset \Re^n$. The population of solutions is called swarm and each individual from a swarm is called a particle. A swarm is defined as a set of N particles. Each particle i is represented as a D-dimensional position vector $x_i(t)$. The particles are assumed to move within the search space A iteratively. This is done by adjusting their position using a proper position shift, called velocity $v_i(t)$.

Each iteration t, the velocity changes by applying equation (1) to each particle.

$$v_i(t+1) = \omega v_i(t) + c_1\varphi_1(P_{ibest} - x_i) + c_2\varphi_2(P_{gbest} - x_i), \quad (1)$$

where φ_1 and φ_2 are random variables uniformly distributed within [0,1]; c_1 and c_2 are weighting factors, also called the cognitive and social parameters, respectively; ω is called the inertia weight, which decreases linearly from ω_{start} to ω_{end} during iterations. P_{ibest} and P_{gbest} represent the best position visited by a particle and the best position visited by the swarm till the current iteration t, respectively.
The position update is applied by equation (2) based on the new velocity and the current position.

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (2)$$

The basic algorithm is as follows:

1) Initialize each particle of the swarm, with random values for position and velocity in the search space.
2) Evaluate member of the swarm with the fitness function.
3) Compare the value obtained from the fitness function from particle i, with the value of P_{ibest}. If the value of the fitness function is better than the P_{ibest} value, this new value takes the place of P_{ibest}.
4) If the value in P_{ibest} is better than P_{gbest}, then $P_{gbest} = P_{ibest}$.
5) Modify the velocity and position of the particles using equations (1) and (2), respectively.
6) If the maximum number of iterations or the ending condition isn't achieved, return to step 2.

To solve the uncontrolled increase of magnitude of the velocities (swarm explosion effect), is often used to restrict the velocity with a clamping at desirable levels, preventing particles from taking extremely large steps from their current position [11].

$$v_{ij}(t+1) = \begin{cases} v_{max} & if v_{ij}(t+1) > v_{max}, \\ -v_{max} & if v_{ij}(t+1) < -v_{max} \end{cases}$$

Although the use of a maximum velocity threshold improves the performance, by controlling the swarm explosions,

without the inertia weight, the swarm would not be able to concentrate its particles around the most promising solutions in the last phase of the optimization procedures; even if a promising region of the search space would be detected, no further refinement would be able, with the particles oscillating on wide trajectories around their best positions [11].

III. Radial Basis Functions

The structure of an RBF neural network consists on three layers [13], as seen in Fig. 2:

- The input nodes layer.
- The hidden neuron layer, that provides a nonlinear transformation from the input through RBFs.
- The output layer, is the linear combination from the outputs from the hidden layer.

When a RBF neural network is used to perform a complex pattern classification task, the problem is basically solved by first transforming it into a high dimensional space in a nonlinear manner and then separating the classes in the output layer. The underlying justification is found in Cover's theorem on the separability of patterns, which, in qualitative terms, may be stated as follows:

A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

From Cover's theorem, we can state that a non linear mapping is employed to transform a nonlinearly separable classification problem into a linearly separable one with high probability.
The neural network is designed to perform a nonlinear mapping from the input space to the hidden space, followed by a linear mapping from the hidden space to the output space [6].
The radial basis equation for interpolation consists in selecting F as:

$$F(x) = \sum_{i=1}^{N} w_i \varphi(\| x - C_i \|), \tag{3}$$

$$\varphi(r) = exp(-\frac{r^2}{2\sigma^2}). \tag{4}$$

Where,

- $\varphi()$: Is the set of N nonlinear functions, known as radial basis. Equation (4) is a gaussian function, but there are other type of RBFs like multicuadratics and inverse multiquadratics.
- w_i : Represents the weight of the connection between the neuron i from the hidden layer with the output layer.
- $\|\|$: Represents the euclidean norm.
- C_i : Are the centers of each of the gaussian functions; where $C_i \in \Re^p, i = 1, 2, 3, ..., N$.

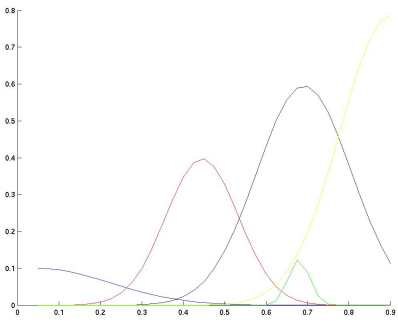


Fig. 1. Set of gaussian functions that conform the RBF neural network output

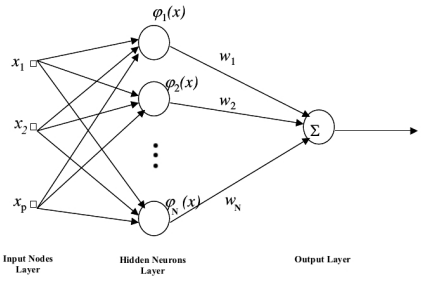


Fig. 2. RBF Neural Network Structure

- σ^2 : Represents the variance.
- x : Is the input signal.

IV. Particle Description and Fitness Function

Path planning for car-like mobile robots can be realized through a search space of functions [7], [8]. In this case we reduce the space to a sub-space of RBFs.
Usually, RBFs are trained by algorithms like k-means and a totally supervised learning method; but by using PSO we substitute this phases.
Each particle in the swarm is composed by the Ci, σ^2 and wi parameters to be used by the RBF function, to approximate a function that passes by a predefined set of points; this set of points is taken as the RBF input points for the input nodes layer. They represent trajectory constraints, i.e. coverage control points can be seen as places where is desirable for the robot to explore in the environment.

The input points are not fixed, excepting for the first and the last point, that are the start and goal points respectively; actually they vary randomly with each PSO iteration in a range of 1 map state, in any direction respect to the first set of input points; this helps to make the obstacle avoidance easier, especially when one of the points is near an obstacle and it transform the problem from interpolation to function approximation.
What defines the efficiency of PSO is the fitness function

(aside of the parameters tuning). We must find a fitness function that allows to find a collision free path for the mobile robot to follow.
The global minimum should correspond whit a smooth and safe (collision free) trajectory, and an unsafe trajectory should be penalized by the fitness function.
For this approach, the following fitness function was designed:

$$f = RMSE + \frac{l}{sf} + c \tag{5}$$

Where $RMSE$ is the root mean square error (equation (6)), l is the length of the path (equation (8)), sf is a scaling factor (so that the length of the path could have more or less influence on the final result) and c (equation (10)) is the collision variable (it takes a higher value proportional to the number of states that a path crosses that are occupied by obstacles or near them).

$$RMSE(f) = \sqrt{\frac{\sum_{k=1}^{n}(e_k)^2}{n}} \tag{6}$$

$$e_k = y_k - f(x_k), \quad where\ f(x_k) \approx y_k. \tag{7}$$

$$l = \sum_{k=1}^{n-1} \sqrt{(x_{k+1} - x_k)^2 + (f(x_{k+1}) - f(x_k))^2} \tag{8}$$

$$S = \{(x, y) | (x, y) \in obstacles\, range\} \tag{9}$$

$$c = \sum f_m(s)\ where\ s \subseteq S \tag{10}$$

$$H = \begin{bmatrix} 0.0001 & 0.0006 & 0.0012 & 0.0006 & 0.0001 \\ 0.0006 & 0.0049 & 0.0099 & 0.0049 & 0.0006 \\ 0.0012 & 0.0099 & 0.0200 & 0.0099 & 0.0012 \\ 0.0006 & 0.0049 & 0.0099 & 0.0049 & 0.0006 \\ 0.0001 & 0.0006 & 0.0012 & 0.0006 & 0.0001 \end{bmatrix}$$

$$Max(f_m(s)) = 0.02 |\ it's\ an\ obstacle. \tag{11}$$

$$Min(f_m(s)) = 0.0001 |\ barely\ in\ the\ obstacle\ range. \tag{12}$$

The s value from equation (9) is the set of points in the range of an obstacle crossed by the path. As was mentioned before, the map is discretized, so a value is assigned to each state that is within a certain range from an object, like mentioned in (11) and (12), and zero if it's out of range; for this case, the gaussian mask H was used to define the values of the map f_m following the next pseudocode:

for $obstacle = 1 \rightarrow |S|$ **do**
  $(x, y) \leftarrow S(obstacle)$
  for $j = -2 \rightarrow 2$ **do**
    for $i = -2 \rightarrow 2$ **do**
      $f_m(x+i, y+j) \leftarrow MAX(f_m(x+i, y+j), H(i+3, j+3))$
    end for
  end for
end for

V. Simulation results

For all the tests a swarm with 40 particles was used, and the stop condition is that the number of iterations is t=100; each test was repeated 100 times and the average was obtained; the results can be seen in Table I. One first comparison was made between the MSE obtained from the approximated function obtained by this approach against the MSE obtained by MATLABs RBF neural network newrb; as seen in the table, the results obtained were favorable by a good margin.
Also the results illustrated in Figures from 3 to 7, show good results for the path planning problem, all the paths are collision free, pass through or acceptably near all the control points and it is obtained smooth paths for a mobile robot to follow.
It is important to note that approaches like Maklink-graph mutation PSO [12] can not deal with environments with concave obstacles, like the one showed in Fig. 4. If the navigation environment contains concave obstacles it would be necessary to compute something like convex hull of each one of them in order to produce the MAKLINK graph. This step will add complexity to the path planning system.

All the PSO adjustable parameters, where heuristically selected, of which were selected the next values as the set of parameters that gave the best results: $c_1 = 2$, $c_2 = 2.5$, $v_{max} = 0.15$, $\omega_{start} = 1$ and $\omega_{end} = 0.0005$.
The value of sf in the fitness function was selected also heuristically, for this case $sf = 37$; the value has to be high to prevent the dominating influence of the length in the fitness function, with smaller values the fitness function would favor the shorter paths instead of the paths that pass through the control points and avoid obstacles.
By giving c_2 a higher value than c_1 we are biasing the particles search ability towards P_{gbest}.
The inertia weight ω reduces the perturbations that make the particles walk away from promising positions, the best positions require strong attraction to refine the search results. A value bigger than 1 in $\omega_{start} = 1$ would make the particles spread more in the search space, making them to reach further positions, but it will take longer for the particles to converge in a promising region as ω_{end} tends to approximate ω_{start}. But also selecting a value higher than 1 form ω_{end} would make it difficult for the particles to converge faster.

Map showed in Fig. 8 was used to compare PSO-RBF approach against Maklink graph-Mutation PSO [12] for path planning. Map showed in Fig. 8 includes some coverage control points as blue squares. All the obstacles are modeled as convex polygons and therefore the Maklink graph-Mutation PSO can be applied to find optimal paths. As mentioned in Section I the strategy consists in three steps: First the MAKLINK graph is built to describe the working space of

TABLE I
MSE VALUES OBTAINED FROM AN RBF TRAINED WITH PSO AND MATLAB'S RBF (NEWRB)

	MSE	
	RBF trained with PSO	RBF (MATLAB newrb)
Map 1	1.4236e-04	0.0395139
Map 2	0.0053	0.08875
Map 3	8.5718e-005	0.0738776
Map 4	4.9906e-004	0.065
Map 5	0.0176	0.0726276

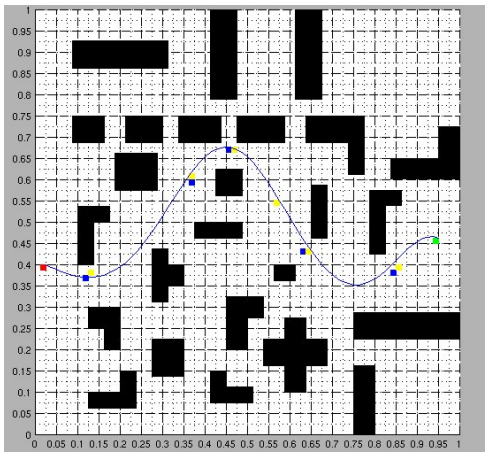


Fig. 3. Simulation Results from Map No. 1. Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

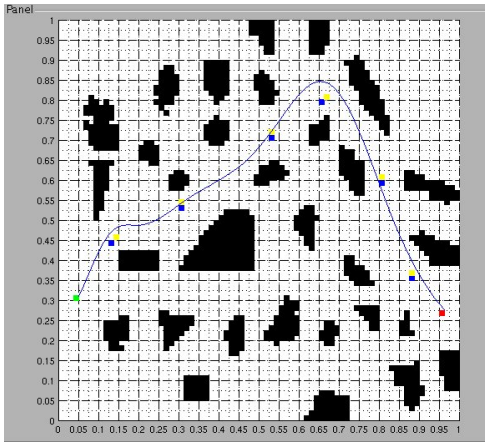


Fig. 4. Simulation Results from Map No. 2.Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

the mobile robot. Then, the Dijkstra's algorithm is used to obtain a sub-optimal path and finally PSO is adopted to get the optimal path. In order to generate enough particles in PSO, it has to be chosen a parameter $t_i \in [0, 1]$ for each free-

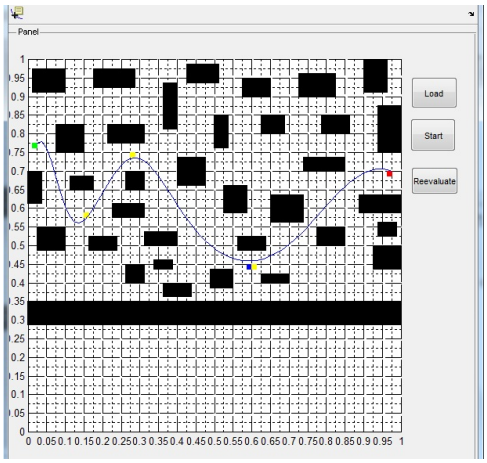


Fig. 5. Simulation Results from Map No. 3.Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

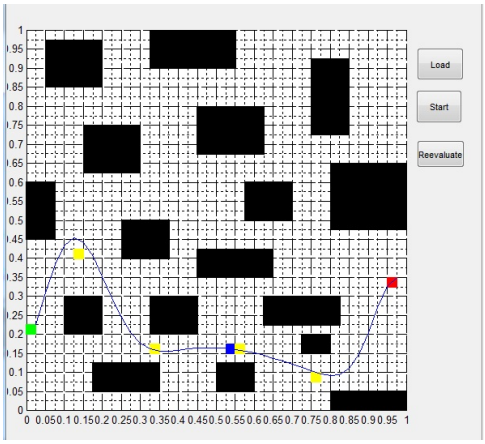


Fig. 6. Simulation Results from Map No. 4.Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

link in the MAKLINK graph. A free-link is defined as a line whose two ends are either corners of two obstacles or one of them is a corner and the other is a point on a working space boundary wall. Fig. 9 shows the MAKLINK graph generated from map in Fig. 8. Maklink graph-Mutation PSO was forced to take into account coverage control points by making that Dijkstra's algorithm looked for the sub-optimal path into the set of free links that pass on (or near of) these control points. The resulting sub-optimal path includes $D = 17$ free links and $t_i \in 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$. All the steps mentioned above make the computational complexity of the Makling graph-Mutation PSO algorithm grows with respect to the number of obstacles in the environment. Large number of obstacles generates MAKLINK graph with large number of nodes and edges and therefore it makes the Dijkstra's algorithm takes long time to run. Nevertheless, the path obtained with

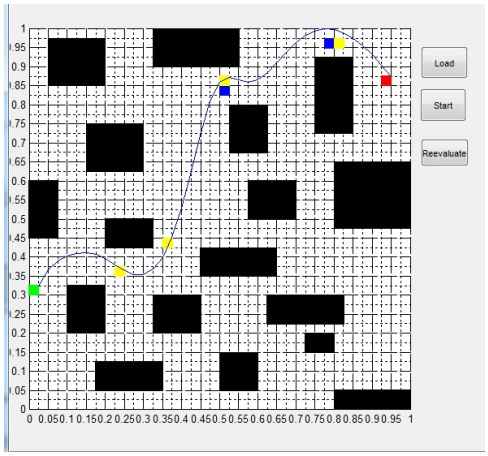


Fig. 7. Simulation Results from Map No. 5. Initial coverage control points of the trajectory are illustrated as blue squares, meanwhile final control points on which obtained trajectory passes on are shown as yellow squares. Smooth and collision free trajectory is obtained regardless of the concavity of the obstacles or the number of these.

this approach is the shortest one that covers (or pass near of) all the control points.

On the other hand, Fig.11 shows the path obtained with the path planner proposed in this paper. This path was obtained with a swarm with 40 particles, evolved by 100 iterations. Although this path is not the shortest one, it is a smooth path easy to be followed by a nonholonomic robot, and our approach needs fewer particles than Maklink graph approach which makes the last one of these, slower than PSO-RBF when the number of obstacles in the environment are numerous and the coverage control points are used.

All simulations were executed in a computer with Windows 7, Intel Core i5, 2.4 GHz, 4 GB RAM, Matlab 2009a. Maklink graph-Mutation PSO algorithm and PSO-RBF algorithm evolved 40 particles with 100 iterations. First approach mentioned took a total time for getting the final path showed in Fig. 10 of $40.56s$, considering all the steps of the approach, since the generation of MAKLINK graph to the training of PSO. Meanwhile our approach took $8.39s$ to get the path showed in Fig.11, although it is not the shortest path is a smooth path which is much easier to follow for a mobile nonholonomic robot than the one showed in Fig.10.

VI. Conclusions and Future Work

This paper presents an approach to solve the path planning problem as an optimization problem using the RBF networks trained with PSO algorithm. A set of trajectory constraints based on coverage control points as input pattern, which can be seen as places where is desirable for the robot to explore were used to approximate functions with PSO-RBF approach, in order to obtain smooth and collision-free paths. Furthermore, a combined fitness function is proposed with respect to three requirements: (i) achieve minimum mean square RBF- function approximation error ; (ii) avoid collisions and (iii) minimize the length of the obtained path .

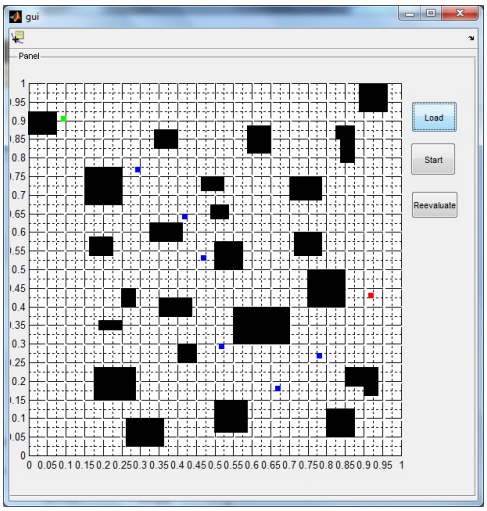


Fig. 8. Map with convex obstacles and coverage control points showed as blue squares.

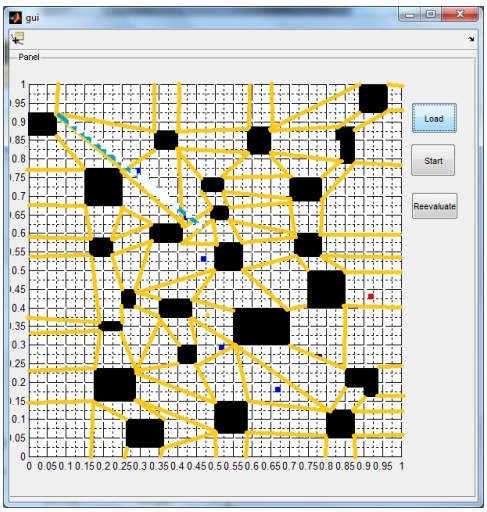


Fig. 9. Maklink graph generated from map showed in Fig.8

Results in simulation environments show that our approach obtains smooth and collision free trajectories regardless of the concavity of the obstacles or the number of these taking advantage of the using of coverage control points as trajectory constraints.

Future work includes the development of an hybrid algorithm which combines PSO-RBF with some PSO-based obstacle avoidance methodology in order to solve motion planning in dynamic environments

References

[1] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, 1992.

[2] Xin Chen and Yangmin Li. Smooth path planning of a mobile robot using stochastic particle swarm optimization. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1722–1727, Luoyang, China, 2006.

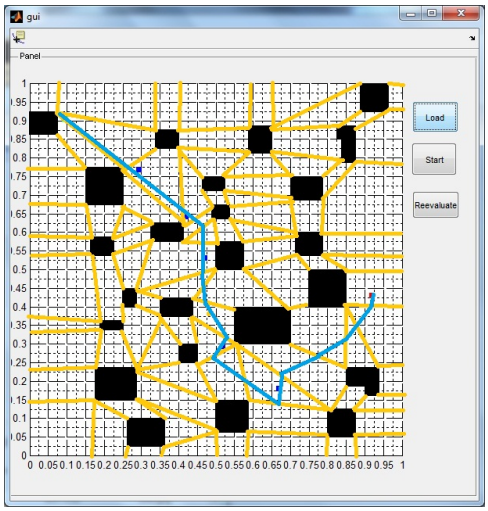


Fig. 10. Path obtained with Maklink-graph mutation PSO approach. This path was obtained with Maklink-graph, Dijkstra's algorithm and evolving 40 particles with PSO in a total time of $40.56s$. The path obtained with this approach is the shortest one that covers (or pass near of) all the control points.

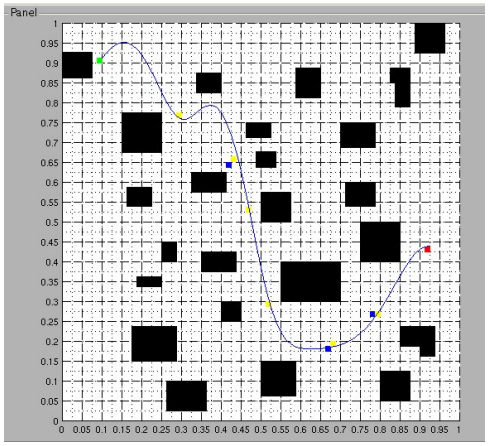


Fig. 11. Smooth Path obtained with PSO-RBF approach. It was obtained evolving 40 particles in a total time of $8.39s$.

[3] E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43–53, 2005.

[4] M. Gerke. Genetic path planning for mobile robots. In *Proceedings of the American Control Conference*, pages 2424–2429, San Diego California, 1999.

[5] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[6] S.S. Haykin. *Neural networks and learning machines*, volume 3. Prentice Hall, 2009.

[7] Y. Hu and S. X. Yang. A knowledge based genetic algorithm for path planning of a mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4350–4355, New Orleans, 2004.

[8] M. Hua-Qing, Z. Jin-Hui, and Z. Xi-Jing. Obstacle avoidance with multiobjective optimization by pso in dynamic environment. In *Proceedings of International Conference Machine Learning and Cybernetics*, volume 5, pages 2950–2956, Luoyang, China, 2005.

[9] James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Washington, DC, USA, November 1995. IEEE Computer Society.

[10] Wei Li and Gai-Yun Wang. Application of improved pso in mobile robotic path planning. In *Proceedings of the International Conference on Intelligent Computing and Integrated Systems (ICISS) 2010*, pages 45–48, Guilin, 2010.

[11] K.E. Parsopoulos and M.N. Vrahatis. *Particle swarm optimization and intelligence: advances and applications*. Information Science Reference, 2010.

[12] Yuan-Qing Qin, De-Bao Sun, Ning Li, and Yi-Gang Cen. Path planning for mobile robot using the particle swarm optimization with mutation operator. In *Proceedings of the Third International Conference on Machine Learning and Cybernetics 2004*, pages 2473–2478, Shanghai, 2004.

[13] E. N. Sánchez and A. Alanis. Redes neuronales: conceptos fundamentales y aplicaciones a control automático. *Cinvestav Unidad Guadalara. Editorial Prentice Hall*, 2006.

[14] M. Saska, M. MacaÌs, L. Preucil, and L. Lhotska. Robot path planning using particle swarm optimization of Ferguson splines. In *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*, pages 833–839. IEEE, 2006.

[15] V. Selvi and R. Umarani. Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications IJCA*, 5(4):1–6, 2010.

[16] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical report, DTIC Document, 1993.

[17] Li W., Yushu L., Hongbin D., and Yuanqing X. Obstacle-avoidance path planning for soccer robots using particle swarm optimization. In *Proceedings of IEEE International Conference on Robotics and Biomimetics, ROBIO 2006*, pages 1233–1238, 2006.

[18] Yunfeng Wang and G. S. Chirikjian. A new potential field method for robot path planning. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA '00*, volume 2, pages 977–982, 2000.

Call Admission Control Using Artificial Bee Colony Optimization

Pratyusha Rakshit[1], Pratyusha Das[2], Amit Konar[1], Atulya K. Nagar[3]
ETCE Department, Jadavpur University[1], Institute of Engineering & Management[2], Kolkata, India
[3]Department of Math & Computer Science, Liverpool Hope University, Liverpool, UK

Abstract- *The paper provides a new approach to Call Admission Control (CAC) problem using Artificial Bee Colony (ABC) optimization algorithm. This paper formulates CAC problem as a constrained optimization problem, where the primary objective is to minimize the call drop under dynamic condition of the mobile stations, satisfying the constraints to maximize the channel assignment and minimize the dynamic traffic load in the network. The performance of the ABC algorithm is compared to DE in connection with the CAC problem and the experimental results reveal that the ABC outperforms DE in all the experiments.*

Keywords- call admission control; feasibility; hotness; quality of service; artificial bee colony optimization.

1 INTRODUCTION

With the evolution of mobile communication, the types of services have been evolved from voice service to multimedia. As a large number of bandwidths are required for multimedia services for wireless cellular networks, the most important issue is to support Quality of Service (QoS) for subscribers. There has been a rapid development in wireless cellular communications, in which QoS guarantee remains one of the most challenging issues. One of the key elements in providing QoS guarantees is an effective Call Admission Control (CAC) policy, which not only has to ensure that the network meets the QoS of the newly arriving calls if accepted, but also guarantees that the QoS of the existing calls does not deteriorate. An efficient call management also aims at satisfying additional objective to assign appropriate channels to the incoming/handoff calls, so that the necessary soft constraints for channel assignment are maintained. Typically, soft constraints include co-channel, co-site, and adjacent channel constraints, all of which need to be satisfied to serve the secondary objective. CAC systems are mainly used to take decisions, whether a call should be serviced, blocked or dropped by a Base Station, and if serviced, it identifies the channel to be assigned to that call. At the time of taking such decision, the interference is considered the only factor in most of the current literature [1, 2].

In this paper, we propose a scheme that takes a more wide view of the CAC problem. We here use three parameters: i) speed, ii) direction, and iii) distance of the MS from the nearest base station to model the motion of the mobile station. The above three parameters play an important role at the time of soft handoff of a call from one cell to other. The importance can be explained with the help of Fig.1, where the central cell has six neighbors. The MS in such a cell while in service may move in various directions with different speed. If it moves toward cell 5 or 6 directly then the channel for handoff will be searched in those cells. When it moves slowly along the common boundary of cell 5 and 6, the cell with base station nearer to the current location of MS is considered. Again, if the above movement takes place with a very high speed, then the call may be dropped due to high interference. Suppose, it moves towards cell 2 very slowly, then there may not be a requirement for a soft handoff at all since it may never cross the existing cell boundary. Hence, we aim at searching an optimal set of assignment, which will take care of the quality of service, and the velocity aspect of the scheme.

In this paper, we propose a CAC scheme that employs Artificial Bee Colony (ABC) optimization algorithm. The ABC technique is a population based algorithm for numerical function optimization that draws inspiration from the stochastic behavior of foraging in bees [3, 6, 7]. We here apply the algorithm to the call admission control problem. Although any stochastic optimization algorithm, such as genetic algorithm (GA), particle swarm optimization (PSO), differential evolution (DE) and the like could have been used for the problem, we have selected ABC because of its faster convergence and qualitative time-optimal solution [8]. It has also been verified in the paper that ABC has outperformed DE in allocating channels dynamically in all the different types of settings of velocity.

Section 2 explains the formulation of CAC in light of the mobility of the MS and changing traffic load. Here in the beginning all the terms are defined and symbolized properly. Then in the next sub section the formulation for fitness evaluation is undertaken. Section 3 gives a detail outline of the artificial bee colony optimization algorithm. Section 4 states the algorithm together with all the operations done in it. Section 5 provides the simulation results with an explanation of them. Conclusions are listed at the end of section 6.

2 FORMULATION OF THE PROBLEM

2.1 Definitions

We consider a system of M hexagonal cells present in the network and each of them has N number of frequency channels. The maximum number of calls that can be serviced is given by Call. In CAC we need to find out the best allocation of calls in different cells which is usually represented by an allocation matrix. In this paper, we plan to select the appropriate allocation matrix online, so as to

satisfy given objective function and systems constraints to be introduced later. The knowledge of calls assigned to channels in any cell is a very important to measure the feasibility of assigning the new calls satisfying the soft constraints.

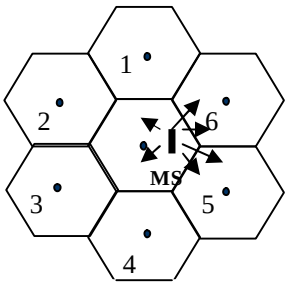


Fig.1. Illustrating the need to consider speed in a given mobile cellular network

Definition 1: Let F= $[f_{i,m}], \forall i \in M, m \in N$, be a binary matrix describing allocation of channels in given cells, where

$$F = [f_{i,m}] = \begin{cases} 1, & \text{if } m^{th} \text{ channel is allocated to serve a call in the } i^{th} \text{ cell} \\ 0, & \text{if the } m^{th} \text{ channel in the } i^{th} \text{ cell is free.} \end{cases}$$

Example: An example of a 4x3 allocation matrix where there are 4 cells and 3 channels is given as

$$\begin{matrix} & channel \rightarrow \\ Cell \downarrow & \end{matrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This implies that the 2nd channel of the 1st cell is serving a call and rest of the channels is free.

Definition 2: Let assignment matrix A= $\lfloor a_{i,m} \rfloor, \forall i \in M, m \in Call$, be a matrix describing the assignment of calls to the channels of a cell, where

$a_{i,m} = p$, *where p is the number of channel assigned to the* m^{th} *call in cell i*

Example: Let there be 10 calls in the network with 4 cells and 3 channels. Then

$$\begin{matrix} & calls \rightarrow \\ Cell \downarrow & \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 0 & 0 \end{bmatrix}$$

We can see that the 1st call is served by the 1st channel of the 1st cell, the 2nd call by the 2nd channel of 3rd cell, 3rd call by the 1st channel of 2nd cell and so on.

While allocating the calls to the channels we should maintain the QoS in terms soft constraints which ensure that the new call assignment do not have any interference with the existing calls assignment in the neighboring cells or in the same cell. In this paper, we measure the QoS as a function of three important network attributes: feasibility, hotness and motion of the MS. The feasibility of channel assignment is often expressed as linear combination of allocation and compatibility matrices.

Definition 3: The compatibility matrix *C* gives a measure of satisfaction of the soft constraints, attempted to minimize co-channel, co-site and the adjacency-channel interference, whose non-diagonal and diagonal elements are expressed by

$C_{j,k}$ = *minimum channel seperation required for a call assignment in cell j when there are calls assigned in cell k, and*

$C_{i,i}$ = *minimum channel seperation required to assign a call in cell i when there are other channels assigned in the same cell.*

Example: A compatibility matrix in a 4 cell network is as

$$\begin{bmatrix} 4 & 3 & 2 & 0 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 0 & 2 & 3 & 4 \end{bmatrix} \quad where,\ C_{i,i} = 4 \text{ and } C_{i,j} = \{0 \text{ to } 3\}$$

Definition 4: Speed V= $[v_{i,p}]$ in the present context refer to rate of position changing of a MS busy with a call utilizing a channel p in cell i.

Example:

$v_{i,p} = 0$ *when the MS is static*

$0 < v_{i,p} \leq 60$ *when inside the city*

$v_{i,p} > 60$ *in highways*

Definition 5: Distance of MS j from the BS i $\forall i \in M$ is denoted as Dis = $[dist_{i,j}]$.

Example: $dist_{i,j} = d$ where d is the distance of the MS from BS as in Fig. 2.

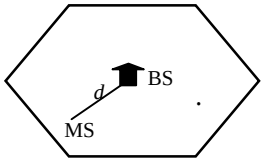


Fig. 2. Distance between BS and MS

Definition 6: Hotness of a cell is defined as the number of incoming calls per unit time and is denoted as H= [hi]. If the number of incoming calls becomes very high the limited resource of the system will not be able to handle the incoming calls and thus subsequent new calls will be blocked.

Example: Hotness in a 4-cell scenario where maximum number of incoming calls is 10 can be given as

$$H = [h_i] = \begin{bmatrix} 2 & 10 & 8 & 5 \end{bmatrix}$$

Definition 7: Angle of motion is the angle made by the direction of motion of a MS p with respect to the BS i and is denoted by Delt = $[\delta_{i,p}]$, and is illustrated in Fig. 3. It shows the direction in which the MS is moving and hence the search for cells with free channels becomes easier.

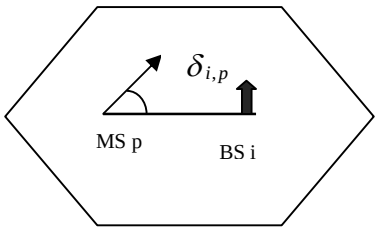


Fig. 3. Angle of motion

Time taken by each call is useful for finding out the calls going for a very long time. At the time of high congestion, these calls are dropped to free the channels for reuse.

2.2 Formulation

Let the time taken by each call in a given channel p, of each cell i is denoted by $T_{i,p}$. If $v_{i,p}$ be the speed with which an MS is moving in any direction, then, its velocity along that direction is $v_{i,p}\cos\delta_{i,p}$. Again if $dist_{p,j}$ be the distance traversed by the MS in time $\Delta T_{i,p}$ then average probability of capturing that MS is denoted as PCMS and is given by

$$P_{CMS} = \sum_{p} \frac{(v_{i,p}\cos\delta_{i,p})\Delta T_{i,p}}{dist_{p,j}} \quad (1)$$

An increase in the value of the above expression is created by a high range of velocity, which gives very little time to search a new channel and to go for a soft hand off (SHO). Hence it should be minimized.

The difference between the two calls in the two different channels should be at a minimum distance to avoid the interference described as soft constrains above. The feasibility of assignment of calls in a cell i can be checked by satisfying inequality (2).The condition demands that the channel separation should satisfy the bare minimum value obtained from compatibility matrix and is denoted by Feas.

$$\sum_{m}\sum_{n}(\sum_{i}(|a_{i,m}-a_{j,n}|)) < \sum_{i} C_{i,j} \Rightarrow Feas^{j} = \sum_{m}\sum_{n}(\sum_{i}(|a_{i,m}-a_{j,n}|-C_{i,j})) < 0 \quad (2)$$

It follows from the last inequality that smaller the value of the left hand side, lesser is the interference and thus better is the quality of service of the call assigned to the channel.

The traffic load in a cell is an important issue to determine the possible admission of incoming calls in a given cell. Traffic load in the cell i may be expressed as the ratio of incoming calls and the total free channels of the cell. We define a metric to measure the traffic load in a given cell i, denoted as Load and is given by

$$Load = \frac{h_i}{\sum_{m}(1-f_{i,m})} \quad (3)$$

Hence the traffic load is the more if the number of incoming calls exceeds considerably than the free cells in the system and starts affecting the overall system performance. Hence this is also to be minimized.

Since expressions (1), (2) and (3) all need to be minimized, a minimization of their linear combination offers an objective that jointly satisfies all the three basic objectives. The overall objective function, the minimization of which yields a possible solution to the call admission problem, is given by

$$Z = P_{CMS} + Feas + Load$$

$$= \sum_{i}\left(\sum_{p}\frac{(v_{i,p}\cos\delta_{i,p})\Delta T_{i,p}}{dist_{p,j}} + \sum_{m}\sum_{n}(\sum_{j}(|a_{i,m}-a_{j,n}|-C_{i,j})) + \frac{h_i}{\sum_{m}(1-f_{i,m})}\right) \quad (4)$$

The lower the value of the function better is the performance of CAC system. Hence, the fitness is defined as

$$fit = 1/(Z+1) \quad (5)$$

Here we also define the difference of fitness ΔZ as

$$\Delta Z = Z_{offspring} - Z_{parent} \quad (6)$$

3 Artificial Bee Colony Optimization Algorithm (ABC)

In ABC algorithm, the colony of artificial bees contains three groups of bees:

- Onlooker bee makes decision to choose a food source.
- Employed bee goes to the food visited by it previously.
- Scout bee carries out random search of food source.

Here, the position of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the fitness of the associated solution. The number of employed bees and onlooker bees is equal to the number of solutions in the population ABC consists of following steps:

3.1 Initialization

ABC generates a randomly distributed initial population P (g=0) of Np food source position where Np denotes the size of population. Each solution X_i (i=0, 1, 2,…, Np-1) is a D dimensional vector.

3.2 Placement of employed bees on the food sources

An employed bee produces a modification on the position in her memory depending on the local information (visual information) as stated by equation (8) and tests the nectar amount of the new source. Provided that the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one. Otherwise she keeps the position of the previous one in her memory.

3.3 Placement of onlooker bees on the food sources

An onlooker bee evaluates the nectar information from all employed bees and chooses a food source depending on the probability value associated with that food source, p_i, calculated by the following expression:

$$p_i = \frac{fit_i}{\sum_{j=0}^{Np-1} fit_j} \quad (7)$$

where fit_i is the fitness value of the solution i evaluated by its employed bee. After that, as in case of employed bee, onlooker bee produces a modification on the position in her memory and checks the nectar amount of the candidate source and memorizes the better position only.

In order to find a solution $X_i^{/}$ in the neighborhood of food source X_i, a solution parameter j and another solution X_k are selected on random basis. Except for the value of chosen parameter j, all other parameter values of $X_i^{/}$ are same as in the solution X_i, for example,

$X_i^{/} = (x_{i0}, x_{i1}, x_{i2}, \ldots, x_{i(j-1)}, x_{ij}^{/}, x_{i(j+1)}, \ldots, x_{i(D-1)})$.

The value of $x_{ij}^{/}$ parameter in $X_i^{/}$ solution is computed using the following expression:

$$x_{ij}^{/} = x_{ij} + u.(x_{ij} - x_{kj}) \quad (8)$$

where u is a uniform variable in [-1, 1] and k is any number between 0 to Np-1 but not equal to i.

3.4 Send scouts for discovering the new food sources

In the ABC algorithm, if a position cannot be improved further through a predefined number of cycles called 'limit',

the food source is abandoned. This abandoned food source is replaced by the scout by randomly producing a position.

After that again steps (B), (C) and (D) will be repeated until the stopping criteria is met.

4 CAC REALIZED WITH ABC

Here, we propose a CAC algorithm which ensures minimization of call drop and noise. We use ABC to ensure the optimization.

Pseudo Code:

Input: Compatibility matrix C and hotness H for a network of M cells, each with N channels and a threshold value ε .

Output: Allocation matrix F for the network.

Begin

Repeat Until $\Delta Z < \varepsilon$

Call ABC (C, H)

End.

Procedure ABC (C, H)

Begin

Initialize all the food sources X_i satisfying condition (2) with *trial[i]*=0 for *i = [0, Np-1]*, and problem parameters as well as algorithm parameters like "limit".

Evaluate the fitness *fit* (X_i), $\forall$ *i* using (4) and (5) .

For Iter=1 to Maxiter do

Begin

For each employed bee

Begin

Produce a new solution X_i' from (8);

Calculate its fitness value *fit* (X_i') using (4) and (5);

If *fit* (X_i')> *fit* (X_i) Then $X_i \leftarrow X_i'$; *trial[i]=0*;

Else *trial[i]= trial[i]*+1;

End If;

End For;

For each onlooker bee

Begin

Select the food source X_i depending on p_i as in (7);

Produce new solution X_i' using the same (8);

Calculate its fitness value *fit* (X_i') using (4) and (5);

If *fit* (X_i')> *fit* (X_i) Then $X_i \leftarrow X_i'$; *trial[i]=0*;

Else *trial[i]= trial[i]*+1;

End If;

End For;

Memorize the best solution X_{best} obtained so far;

Set *index* $\leftarrow$ *arg (max (trial[0], trial[1],…, trail[Np-1]))*;

If *trial[index]>limit* Then reinitialize X_{index} by scout bee;

End If;

End For;

Update: $F \leftarrow X_{best}$;

Return.

5 EXPERIMENTS AND RESULTS

5.1 Basic assumptions

In this experiment we have considered the following assumptions:

- The network has 21 hexagonal cells and 7 channels as shown in Fig. 4.
- The value of
  - Co-channel distance is 2
  - Adjacency channel distance is 3
  - Co-site distance is 4
- The number of incoming calls lies in the range 0 to 150 and changes dynamically.
- Initial population size was taken as 20.
- The velocity change was from 0 to 120 km/hr.
- The distance between two base stations is 2 km and remains unchanged.
- The calls are considered long if they go on more than 30 minutes.
- The direction of the MS, its distance from the base station and velocity changes dynamically.

In addition to these, we consider the random call hang-up in the system and express this phenomenon as Change. Suppose in a given cell r, a call served with channel c is disconnected by the caller. Then following the definition of allocation matrix F, we understand that the element $f_{m,l}$ = 0 for m= r and l=c, after disconnection of the call.

$$\text{i.e., } f_{m,l} = 0, \quad where \quad m = r, l = c \tag{9}$$

We have considered minimum two such hang-ups in the network. Accordingly random positions are generated where Change is done as described above. A call is forcefully terminated if the duration of the call is greater than Long call interval L. The initial conditions and necessary changes of the dynamic network are enforced obeying the above stated assumptions.

5.2 Results

The experiment was carried out on a simulated environment on Intel Core 2 Duo processor architecture with clock speed of 2GHz. Fig. 5 shows that the average cost function value gradually diminishes with iterations. Further, it is noted that the smaller the velocity settings of the mobile stations in program run, the faster is the fall off in the average cost function profile. An intuitive interpretation of this phenomenon is that with increase in velocity i.e., a high rate of change in position of the MS, more constraints are faced to allocate channels to the MS, thereby increasing average cost function.

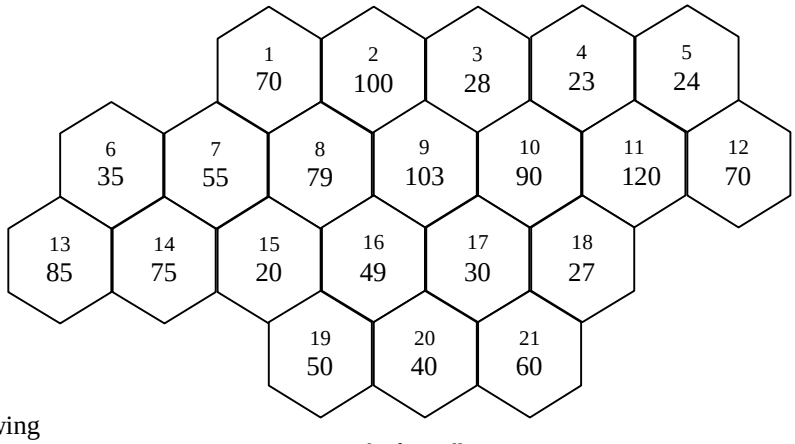


Fig.4. Network of 21 cells

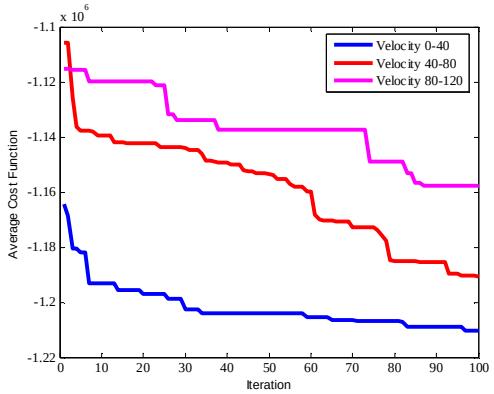


Fig.5. Average cost function vs. Iteration

The relative performance of DE and ABC can be studied through error estimation as indicated in Fig. 6-8. In these figures, we plotted the average cost function obtained from classical DE- and ABC - based experiments, corresponding to each value of iteration. We also evaluated the error in average cost function by taking the difference of the same obtained from DE and ABC as shown in Fig. 8(a) and (b). Let *Ei* be the error for the *i*-th sample data. Since the errors for different sample data are all positive, indicating a superiority of ABC over DE, a measure of the relative goodness of ABC over DE can be defined as the root mean square error *Er.m.s*= 29869 and *Er.m.s*= 29581 respectively. This shows ABC as having an advantage over DE for the call assignment control problem. Of course, the root mean square error (29869 and 29581 respectively) at the sample points being insignificantly less than the root mean square value (1162200 and 1159200 respectively) of the averaged average cost function profiles for DE- and ABC - based simulations.

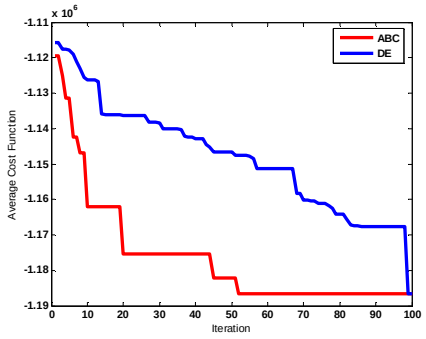


Fig.6. Average cost function vs. Iteration for velocity ∈ [0, 40]

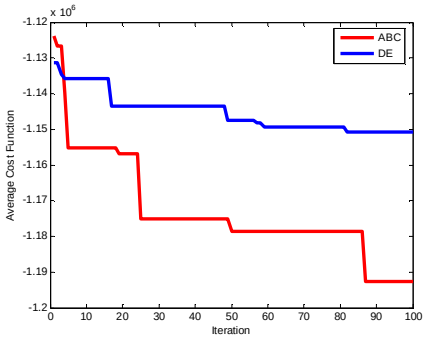


Fig.7. Average cost function vs. Iteration for velocity ∈ [40, 80]

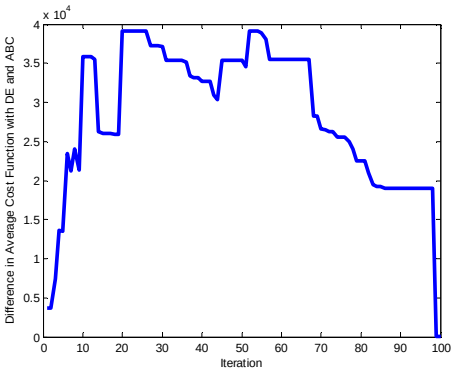


Fig. 8(a). Difference in Average cost function with DE and ABC vs. Iteration for velocity ∈ [0, 40]

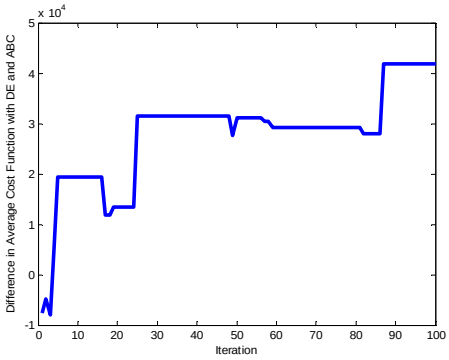


Fig. 8(b). Difference in Average cost function with DE and ABC vs. Iteration for velocity ∈ [40, 80]

The allocation matrices obtained using ABC and DE-based simulation with a velocity range of 0 to 40 and for a generation of 100 are given in TABLEs I-II.

TABLE-I

Allocation matrix obtained from ABC-based simulation with velocity ∈ **[0. 40]**

		Channel No.						
		1	2	3	4	5	6	7
Cell No.	1	0	1	0	0	0	0	0
	2	1	0	0	0	0	0	0
	3	1	0	1	1	1	0	0
	4	0	1	1	1	1	0	1
	5	0	0	0	1	1	1	0
	6	1	1	0	0	1	0	0
	7	1	0	1	1	0	1	0
	8	1	1	0	1	1	0	1
	9	1	1	0	0	1	0	0
	10	0	1	1	0	0	0	1
	11	1	0	1	1	0	1	1
	12	0	0	1	0	0	1	0
	13	0	0	1	1	1	0	1
	14	1	1	1	1	0	1	1
	15	1	0	0	1	0	1	0
	16	1	0	1	1	1	1	1
	17	1	0	0	0	0	0	0
	18	1	1	1	1	0	1	1
	19	1	1	0	1	1	1	0
	20	1	1	1	1	1	0	1
	21	0	1	1	0	1	1	1

For simulation purposes, we have considered two different classes of problems available in the literature:

1. The first class consists of data set, denoted as EX1 [4], as well as a slightly larger extension of EX1, denoted as EX2 [5].
2. The final set of problems is KUNZ1-KUNZ4 [5]. However, we have used only KUNZ1 in this paper.

Using the descriptive details of each problem, as tabulated in [5], we have obtained the following allocation matrices from ABC-based simulation as shown in TABLE-III.

The average cost function profiles for three problem using ABC and DE are given in Fig. 9. A close observation of Fig. 9 indicates that ABC-based simulation has outperformed DE-based simulation in terms of convergence speed as well as quality of solution.

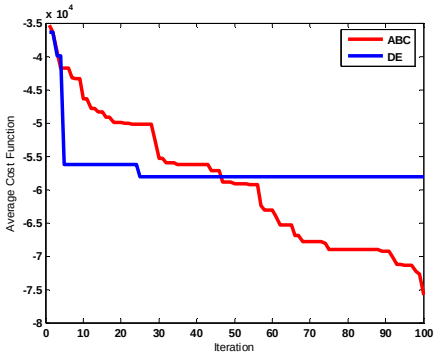
TABLE-II

Allocation matrix obtained from DE-based simulation with velocity ∈ **[0, 40]**

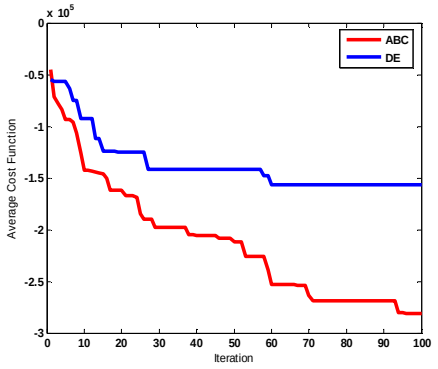
		Channel No.						
		1	2	3	4	5	6	7
Cell No.	1	0	1	0	0	0	0	0
	2	0	0	0	0	0	0	0
	3	1	1	1	1	1	1	0
	4	1	1	0	0	1	1	1
	5	0	1	1	1	1	1	0
	6	1	1	1	0	1	0	1
	7	1	0	0	1	0	1	1
	8	1	1	0	1	1	0	1
	9	1	0	0	1	1	0	1
	10	0	1	0	1	0	0	1
	11	1	0	0	1	1	1	1
	12	1	0	0	0	0	0	0
	13	1	0	1	0	0	0	1
	14	1	0	0	1	1	1	0
	15	1	1	0	1	1	0	0
	16	1	0	1	1	1	0	1
	17	1	1	0	0	0	0	1
	18	0	1	0	0	0	0	0
	19	1	1	1	1	0	1	0
	20	0	1	1	1	1	0	1
	21	1	1	1	1	0	1	1

6 CONCLUSION

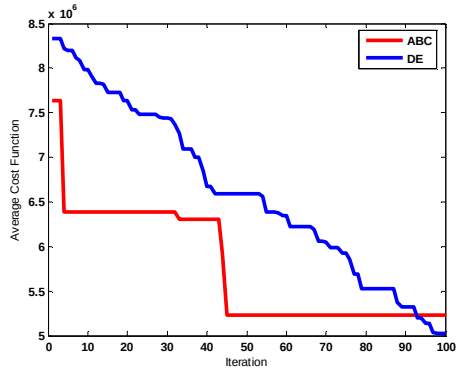
In the proposed work, we consider our cells to be hexagonal so as to easily track the movement of MS in the neighborhood cells. Instead of considering a single cell scenario, we have taken a small network to implement the algorithm to incorporate the intercellular communication efficiently. The decision of acceptance or rejection of a call dose not only depend on the feasibility and availability of the channel but also on the speed at which the MS moves and its direction of movement. Its geographical location with respect to the base station also has a significant importance. Moreover the traffic density on a cell is also considered to be a determining factor. The reuses of the channels also make the approach more effective. The CAC scheme using ABC technique has been compared with a similar scheme using DE. The results show that when ABC scheme is considered, a better optimum is obtained in comparison to the scheme using DE irrespective of the load value or the velocity of the MS as well as for different problem sets.



(a)



(b)



(c)

Fig.9. Average cost function profiles for (a) EX1 (b) EX2 (c) KUNZ-1

7 REFERENCES

[1] L. Wang, S. Arunkumaar, and W. Gu, "Genetic algorithms for optimal channel assignment inmobile communications", Proceedings of the 9th International Conference on Neural Information Processing ICONIP 2002, Vol. 3, pp. 1221-1225 Nov. 2002.

[2] J. Hou, and Y. Fang "Mobility-based call admission control schemes for wireless mobile networks" Wirel. Commun. Mob. Comput. 1:269–282 2001.

[3] B. Basturk, and Dervis Karaboga, "An artificial bee colony (ABC) algorithm for numeric function optimization", IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006, Indianapolis, Indiana, USA.

[4] K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum, "Channel assignment in cellular radio", Proc. 39th IEEE Veh. Technol. Soc. Conf., pp. 846-850, May, 1989.

[5] K. Smith, and M. Palaniswami, "Static and dynamic channel assignment using neural network", IEEE Journal on Selected Areas in Communications, vol. 15, no. 2, Feb 1997.

[6] P. Rakshit, A. K. Sadhu, P. Bhattacharjee, A. Konar, and R. Janarthanan, "Multi-robot box-pushing

using non-dominated sorting bee colony optimization algorithm", SEMCCO 2011: 601-609.

[7] P. Bhattacharjee, P. Rakshit, I. Goswami, A. Konar, and A. K. Nagar, "Multi-robot path-planning using artificial bee colony optimization algorithm", NaBIC 2011: 219-224.

[8] A. Halder, P. Rakshit, A. Chakraborty, A. Konar, and R. Janarthanan, "Emotion recognition from the lip-contour of a subject using artificial bee colony optimization algorithm", SEMCCO 2011: 610-617.

[9] S. Ghosh, A. Konar, and A. Nagar, "Dynamic channel assignment problem in mobile networks using particle swarm optimization", EMS 2008: 64-69.

[10] S. Bhattacharjee, A. Konar, and A. K. Nagar, "Channel allocation for a single cell cognitive radio network using genetic algorithm", IMIS 2011: 258-264.

TABLE-III

Allocation matrices obtained for EX1, EX2 and KUNZ1 using ABC-based simulation

Problem	Allocation Matrix																														
EX1	0	1	0	1	0	0	0	0	0	0	0																				
	0	1	0	0	0	0	0	0	0	0	1																				
	0	0	0	0	1	0	0	0	0	0	0																				
	0	0	0	0	0	0	0	0	0	0	0																				
EX2	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0														
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0														
	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0														
	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0														
	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0														
KUNZ1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	
	1	0	0	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	1	0	1	1	0	0	1	0	1	1	
	1	0	0	0	1	1	1	1	0	1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	
	0	1	0	0	1	0	1	0	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	
	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	0	1	1	0	1	0	0	0	0	1	1	
	0	0	1	0	1	0	0	1	1	0	1	0	0	1	1	1	1	1	0	1	0	1	1	1	1	0	1	1	0	0	
	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	0	1	0	0	1	0	0	1	
	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	
	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	0	
	0	1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	0	1	0	0	0	0	

SESSION

MEMETIC ALGORITHMS + FUZZY LOGIC + HEURISTIC METHODS

Chair(s)

TBA

An Efficient Direct Torque Control Based on Fuzzy Logic Technique

Jibo Zhao[1] **and Hong Wang**[2]
[1] Electrical Engineering and Computer Science Department, University of Toledo , Toledo, OH, USA
[2]Engineering Technique Department, University of Toledo, Toledo, OH, USA

Abstract *- Conventional Direct Torque Control (CDTC) system of Induction Motor (IM) faces the problem of high torque ripples, and has difficulty in improving the performance of dynamic torque response and controlling flux locus at very low speed. In this paper, a DTC control system for induction motor based on Fuzzy Logic technology (FLDTC) is proposed. The proposed system aims to make less torque ripples, faster dynamic response, and higher performance of flux control at very low speed by introducing some new fuzzy variables and rescheduling the fuzzy switcher rules. The model of the proposed system is built on simulink/matlab. Simulation results show that the proposed technique FLDTC is more efficient than CDTC.*

Keywords-Direct torque control, induction motor, fuzzy logic, and fuzzy switcher rules.

1 Introduction

Conventional direct torque control is a simple and efficient control technique to provide quick torque and flux control. The major advantages of direct torque control technique are its simple structure and robust control scheme without the complex mathematical transforms. However, CDTC also has some drawbacks like high electromagnetic torque ripple, high stator current distortion, relatively slow transient response to torque step changes of load and flux locus attenuation at very low speed [1-3].

To improve the performance of dynamic response of CDTC, some studies have been carried out in the past [4] [5] to increase the response speed of torque step change. One research has developed a methodology of optimizing the selection of the voltage vectors to give a maximum rate of torque increase or decrease to meet the torque step change [6], and dramatically improved the CDTC responding speed, but the expense is the low performance of flux locus.

By introducing CDTC technique to induction motor, the controller uses voltage vectors to control the flux or torque according to three elements: hysteresis of flux error, torque error and flux location. However, sometimes the flux locus or the torque only needs the voltage vectors to last for a very short time during a switching period in steady-state, or perhaps it needs the control signal to last for several switching period in dynamic-state. The hysteresis of CDTC system can only judge the situation by positive and negative error values, but it does not have the ability to adjust the flux and torque according to exact error values. For the purpose of handling this problem, a method to reduce torque ripple in DTC of induction motor by using fuzzy mode duty cycle [7] is applied to control the duty cycle of the switches according to the exact value of the torque and flux errors and has successfully decreased the torque ripples. On the other hand, lots of attempts based on fuzzy logic technique are shown to be efficient in many researches. For instance, in the fuzzy logic control method proposed in [8] and [9], the fuzzy logic controller can recognize how big the error is and makes an optimal adjustment; Moreover, a stator resistance estimator using fuzzy logic at low speed applied in [10], can help to improve the performance of torque ripple by making the mathematical model of CDTC more accurate. Some other researches [11-13] also provide several useful applications of fuzzy logic in DTC. All these methods have proven that fuzzy logic technique can make great contributions to DTC.

This paper aimed to take advantage of fuzzy logic technique to solve the problems mentioned above. A group of new FL switcher rules will be introduced. This FL controller can detect the steady and dynamic states of induction motor automatically, and control the flux and torque with optimal vectors according to fuzzy switcher rules. The FL controller has also solved the problem of flux attenuation at very low speed. Both the steady and dynamic performance of torque error and torque response to step changes can be improved by the proposed methodology. The simulation results of CDTC and FLDTC will be studied and compared.

2 Proposed technology with fuzzy logic

To improve the performance of CDTC, we apply a Mamdani-type fuzzy logic system based on DTC principles. The torque hysteresis in CDTC is substituted by this FL-controller. Different from commonly used controller, the proposed FL-controller has six input variables: Torque error (Te), flux error (Fe), flux position (SE), angle difference (A), rotor speed (SP) and working state (WS).

The membership function of “flux error” (Fe) has four fuzzy sets: negative (N), zero (Z), positive (P) and positive large (PL). The fuzzy variable “torque error” (Te) is

represented by five fuzzy sets: negative large (NL), negative (N), zero (Z), positive (P) and positive large (PL). The fuzzy membership function of "sector" (SE) which stands for the location of flux is represented by six fuzzy sets: sector (1-6) S1, S2, S3, S4, S5 and S6 as shown in Fig. 1. The membership functions of the three fuzzy variables are shown in Fig. 2 (a-c). The other three fuzzy variables will be introduced separately in the following paragraphs.

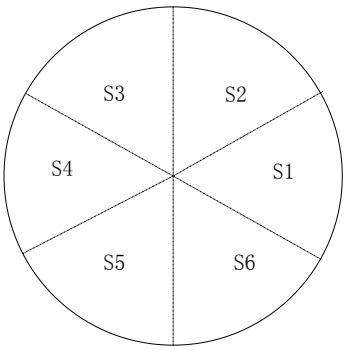
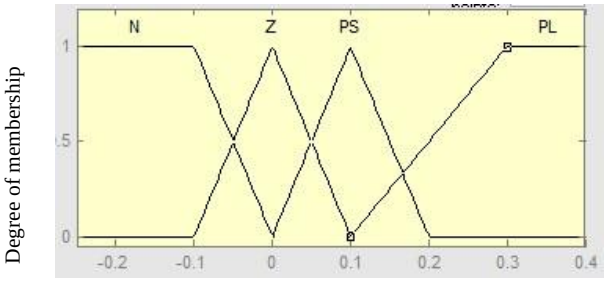
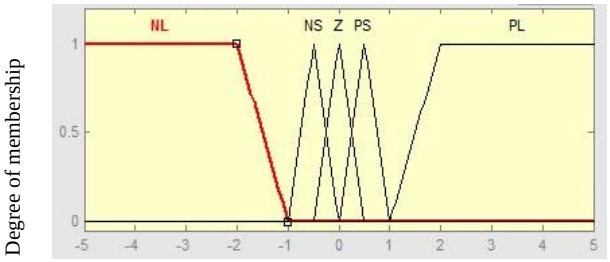


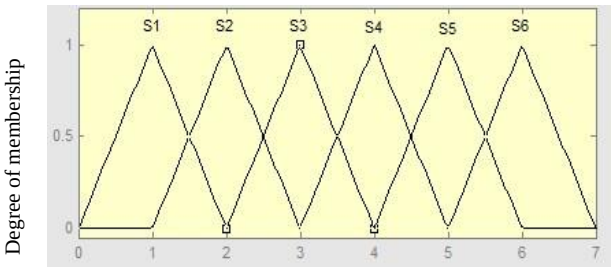
Figure 1. Spatial distribution of six sectors



Flux error
(a). Fuzzy membership functions for Fe.



Torque error (N.m.)
(b). Fuzzy membership functions for Te.



Sectors
(c). Fuzzy membership functions for SE.

Figure 2. Fuzzy membership functions for Fe, Te and SE.

The electromagnetic torque of IM can be expressed as follows,

$$T_e = \frac{3}{2}\frac{P_n}{L_\sigma}|\psi_s||\psi_r|\sin\delta \tag{1}$$

where $$L_\sigma = (L_s L_r - M^2)/M \tag{2}$$

$|\psi_s|$ and $|\psi_r|$ are the stator and rotor flux vectors, L_s is the stator inductance, L_r is the rotor inductance, M is the magnetizing inductance, δ is the angle between the stator and rotor fluxes and P_n is the number of the pole pairs.

When $\delta = 90^\circ$, $$T_e = \frac{3}{2}\frac{P_n}{L_\sigma}|\psi_s||\psi_r| \tag{3}$$

If the system is ideal no-load, then the average torque is zero. If assuming $|\psi_s| = |\psi_r|$, then we can get the maximum value of dynamic torque:

$$T_{e\max} = \frac{3}{2}\frac{P_n}{L_\sigma}|\psi_s|^2 \tag{4}$$

Thus, the electromagnetic torque can be written as

$$T_e = T_{e\max}\sin\delta \tag{5}$$

Because of the assumption that the IM is ideal no-load, the average torque is zero. The torque ripple can be written as:

$$\Delta T_e = T_{e\max}\sin\delta \tag{6}$$

Assuming the rotating speed of rotator flux is a short-term constant value, then,

$$\delta = (\omega_s - \omega_r)t \tag{7}$$

In condition that δ is very small,

$$\Delta T_e = T_{e\max}(\omega_s - \omega_r)t \tag{8}$$

The rotating speed of ω_s when the reference torque is ideal no-load can be expressed as:

$$\omega_a = \frac{2\pi}{T_a} \tag{9}$$

T_a is the time period for the stator flux. Merge (8) and (9), we can get the increasing time of torque:

$$t_i = \frac{\Delta T_e T_a}{T_{e\max}(1-\frac{\omega_r}{\omega_a})} \tag{10}$$

Similarly, we can get the torque decreasing time:

$$t_d = \frac{\Delta T_e T_a}{T_{e\max}\frac{\omega_r}{\omega_a}} \tag{11}$$

Formula (11) can be written as:

$$t_d = \frac{2\pi\Delta T_e}{T_{e\max}\omega_r} \tag{12}$$

From (12) we can get the conclusion that the time required to decrease the torque gets longer when the rotating speed is very low. It is clear that the torque decreases slower at low speed than that at high speed if the controller still uses zero voltage vectors. Thus, replacing zero voltage vectors with reversed voltage vectors may increase the response speed

because reversed voltage vectors can produce bigger negative torque change. Moreover, if we only use zero vectors to reduce the negative torque error without the usage of reversed vectors at very low speed, the flux locus would attenuate and even result in failure start. On the other hand, if using reversed vectors too often at very low speed may result in bigger toque ripple in steady state than zero vectors. To balance the problem, another fuzzy variable "SP" is used in the FL controller. The membership function of SP as indicated in Fig. 3 is divided into low speed mode (L) and high speed mode (H). When working at low speed (the speed less than 30% of rated speed is defined as low speed), the FL controller will test the flux error. If the flux error is N, Z or P, and the torque error is in the range of P, Z and N, the controller will work exactly in the same way as it does in high speed mode. However, if the flux error is PL, which means the flux locus is attenuating, the controller will enable the reversed voltage vectors to justify the flux locus. The switching rules at high speed and very low speed are shown in table 1 and table 2. The fuzzy variable "A" in the two tables will be introduced later. This method can take advantages of both zero vector and reversed vector.

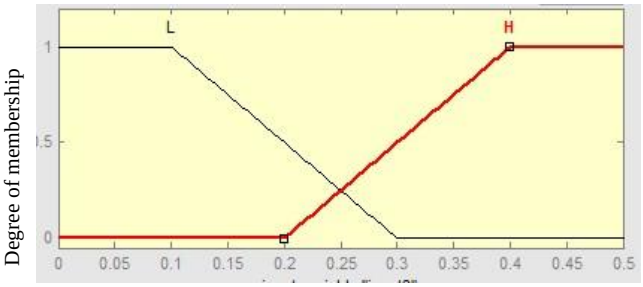


Figure 3. Memberships function for SP.

TABLE I. FUZZY CONTROL RULES OF FLDTC IN HIGH SPEED (K IS THE NUMBER OF SECTOR)

Te	Fe	A	V
NL/N/Z	N/Z/P/PL	L/S	V0/V7
P	N	L/S	V(k+2)
	Z/P/PL	L/S	V(k+1)
PL	N/Z/P/P	L	V(k+1)
		S	V(k+2)

In order to detect the working state, another variable "WS" is added to the FL controller. The variable "WS" carries the information of whether or not there is a toque step change in the load. The system can decide which working mode of the FL controller should be taken based on this information. The system automatically tests the change of the torque in the load and compares it with the output toque. Once the difference between them reaches the predefined threshold, the state of the system will change into dynamic working state. In this working state, the fuzzy rules will allow the controller temporarily neglects the regulation of flux locus. Because the dynamic state lasts only for a very short time, the transient change will not influence flux locus significantly, and the locus will recover as soon as the system turn back to steady state. The variable "WS" is composed of three fuzzy sets: dynamic work state whose step change is negative (DN), dynamic work state whose step change is positive (DP) and steady work state (S). The membership function is shown in Fig. 4.

TABLE II. FUZZY CONTROL RULES OF FLDTC IN LOW SPEED (K IS THE NUMBER OF SECTOR)

Te	Fe	A	V
NL	N/Z	L	V(k-2)
		S	V(k-1)
	PL	L/S	V(k-1)
N	N/Z	L/S	V0/V7
	P	L/S	V(k)
	PL	L/S	V(k-1)
Z	N/Z/P	L/S	V0/V7
	PL	L/S	V(k+1)
P	N	L/S	V(k+2)
	Z/P/PL	L/S	V(k+1)
PL	N/Z	L/S	V(k+2)
	P/PL	L	V(k+1)
		S	V(k+2)

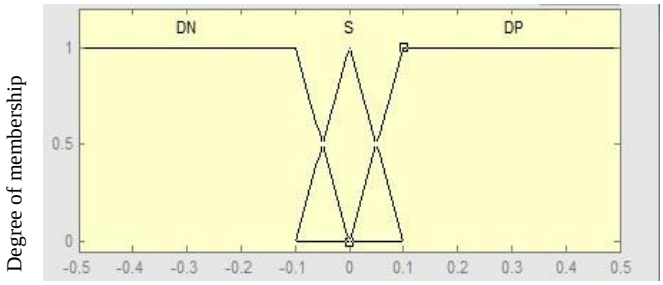


Figure 4. Memberships function for WS.

We can directly get the conclusion from [6] that the optimal voltage vectors giving the fastest response can be simplified as a problem of maximization:

$$fn = \max_{k}\{\sin(\theta_k - \theta_{ro})\} \quad (13)$$

where k is the order of voltage vector, θ_k and θ_{ro} are the stator voltage vector angle and initial rotor flux angle, respectively.

From (13) we know that the voltage vector which creates the largest sine value with rotor flux has the ability to produce the largest torque change. In order to take advantage of the conclusion, another fuzzy variable "A" is added to the fuzzy controller. The rotor flux can be approximately equivalent to stator flux because the slip angular velocity is actually very small. When torque needs to be increased, the fuzzy variable A is the angle between rotor flux and voltage vector V (k+1). When torque needs to be decreased, variable A becomes the angle between rotor flux and voltage vectors V (k-2).

For instance, as shown in figure 5, when the flux is at point P1, the angle between the vector V (k+1) and flux is π/2, which means that V (k+1) can create the biggest torque change according to (13), so A is L at this moment and V (k+1) is chosen to produce the biggest torque increase. As soon as the flux moves to point P2, the angle variable A decreases to π/3, which means that V (k+1) will not produce the fastest torque response in the following time and variable A becomes S at this moment. Hence V (k+2) will be taken instead of V (k+1). The way to produce the fastest torque decrease is similar. This conclusion also explains the reason to use the fuzzy variable A in table 1 and table 2.

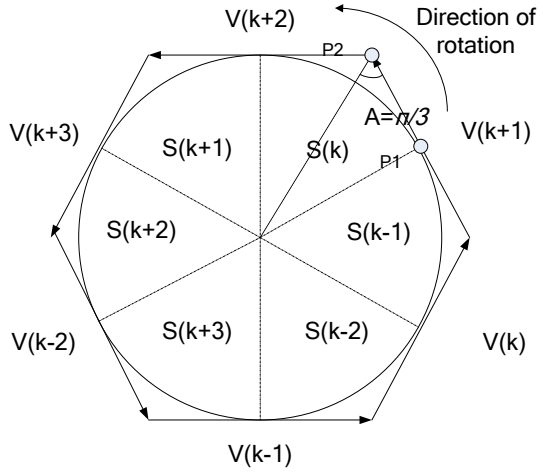


Figure 5. Rotor flux vectors and six voltage vectors.

Whether A is Large (L) or Small (S) is defined in the following way. When the torque needs to be increased quickly, if the difference between the angles of rotor flux and the voltage vector V (k+1) is in the range of (π/3, π/2), then A is large (L), otherwise A is small (S). Conversely, when the torque needs to be decreased quickly, if the difference between the angles of rotor flux and voltage vector V (k-2) is in the range of (π/2, 2π/3), then A is large (L), otherwise A is small (S).The results will be transferred to the FL controller which can analyze the composite conditions and give an optimal voltage vector selection according to the expert knowledge. The membership function of A shown in figure 6 is represented by two fuzzy sets: large (L) and small (S).

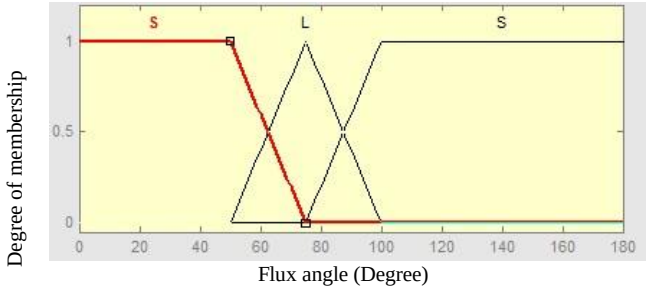


Figure 6. Membership function of A.

From the fuzzy variables and fuzzy rules introduced above, we can get the flow chart of the FL controller in Fig. 7.

To sum up, the fuzzy switching rules can be summarized to table 3. Each rule in table 3 can be written as: Ri: if WS is Ai, SP is Bi, Te is Ci, Fe is Di, SE is Ei and A is Fi, then V is Vi, where Ri is the ith fuzzy rule. Ai, Bi, Ci, Di, Ei and Fi are the values of fuzzy sets of the fuzzy variables WS, SP, Te, Fe, SE and A.

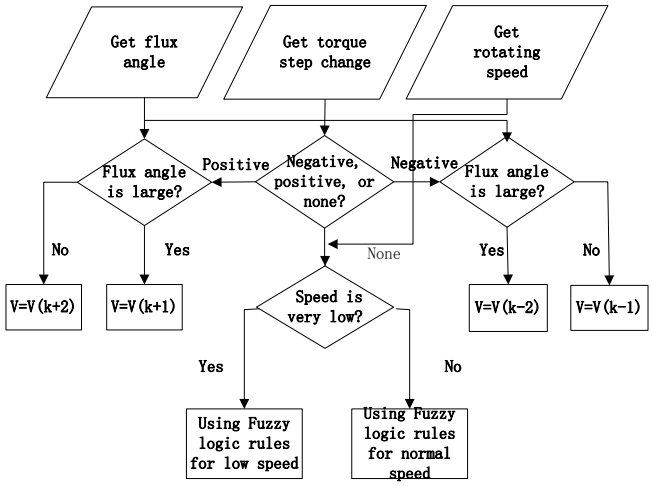


Figure 7. Flow chart of the FL controller

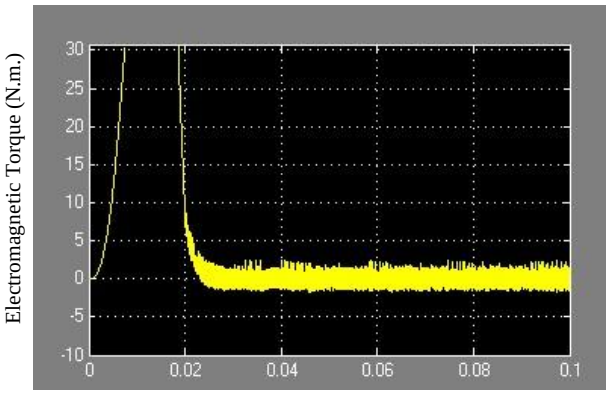
TABLE III. FUZZY CONTROL RULES OF FLDTC (K IS THE NUMBER OF SECTOR)

WS	SP	Te	Fe	A	V
S	L	NL	N/Z	L	V (k-2)
				S	V (k-1)
			PL	L/S	V (k-1)
		N	N/Z	L/S	V0/V7
			P	L/S	V(k)
			PL	L/S	V (k-1)
		Z	N/Z/P	L/S	V0/V7
			PL	L/S	V (k+1)
		P	N	L/S	V (k+2)
			Z/P/PL	L/S	V (k+1)
		PL	N/Z	L/S	V (k+2)
			P/PL	L	V (k+1)
				S	V (k+2)
	H	NL/N/Z	N/Z/P/PL	L/S	V0/V7
		P	N	L/S	V (k+2)
			Z/P/PL	L/S	V (k+1)
		PL	N/Z/P/P	L	V (k+1)
				S	V (k+2)
DP	L/H	N/P/Z	N/Z/P/PL	L	V (k+1)
				S	V (k+2)
DN	L/H	N/P/Z	N/Z/P/PL	L	V (k-2)
				S	V (k-1)

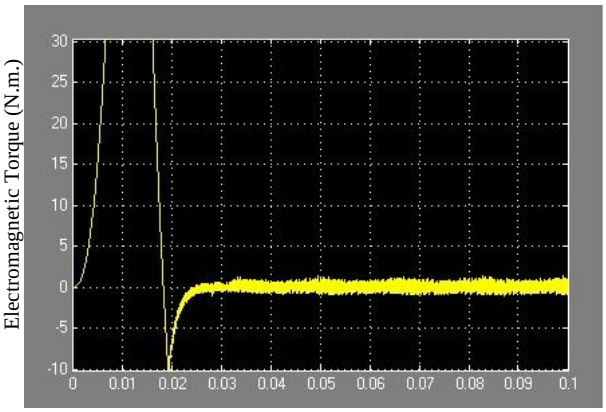
3 Simulation results

To verify the efficiency of the proposed system, the model is tested on matlab tool. The induction motor's parameters are given as follows:

Rated Voltage: 380 V
Pole pairs: 2
Stator Resistance 1.111 Ω
Rotor Resistance 1.083 Ω
Stator Inductance: 0.5974 H
Rotor Inductance: 0.5974 H
Mutual Inductance: 0.2037 H
Moment of inertia J: 0.02 kg.m^2
Friction factor: 0.0057 N.m.s
Sampling period of the system: 50 μs



(a). Electromagnetic torque for CDTC

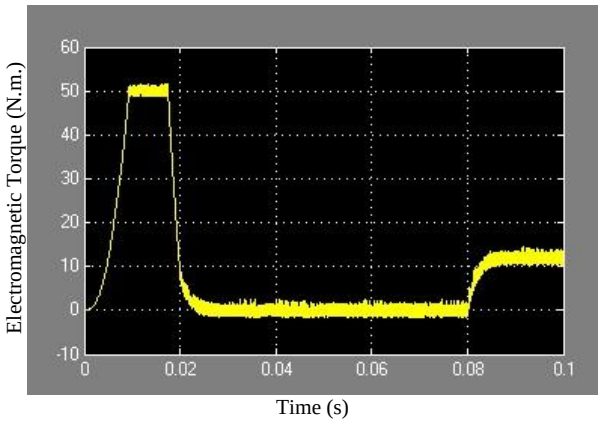


(b). Electromagnetic torque for FLDTC

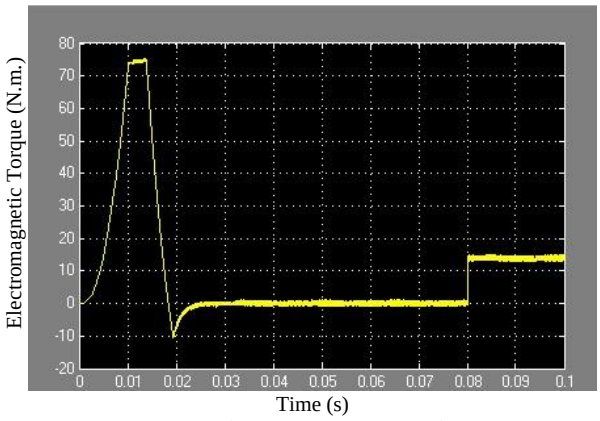
Figure 8. Electromagnetic torque

Fig. 8 (a) and (b) show the performances of the torque ripples of the motor at 300 rad/sec and no load for CDTC and FLDTC, respectively. It is clearly shown that the toque ripple in Fig. 8(b) is approximately 40% smaller than that in Fig. 8 (a). Hence we can conclude that FLDTC produce less torque ripple than CDTC in steady state.

Keeping the speeds unchanged, and adding a step torque change as big as 12N.m. at 0.08sec in the load, we can get the curves of torque response illustrated in Fig. 9. The torque response of FLDTC is significantly faster than that of CDTC.



(a). Response of the step torque change for CDTC

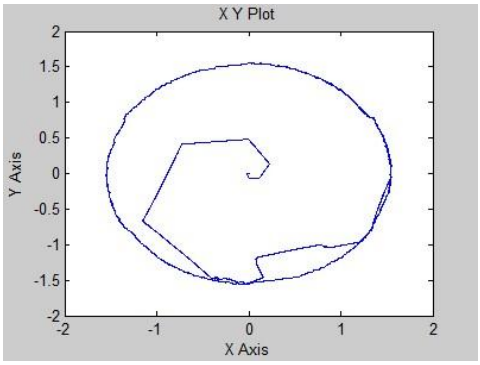


(b). Response of the step torque change for FLDTC

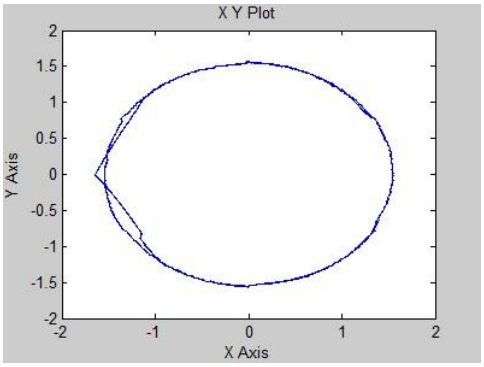
Figure 9. Electromagnetic torque responses with a step change of 12N.m at 0.08 sec for CDTC and FLDTC

The flux locus of FLDTC in both steady state and dynamic state with a torque step change of 12N.m at 0.08 sec are given by Fig. 10. From Fig. 10(a), we can see that the flux locus is not significantly different from the flux locus of CDTC in steady state. Nevertheless, the flux locus shown in Fig. 10(b) has a transient change when a step change is applied in the load. That is because the FL controller temporarily ignores the flux locus when working in dynamic state. In this state, the controller only imposes the voltage vectors producing the biggest torque change rate. Hence the flux locus moves toward the same direction as the voltages vector. This is why the flux locus rotates along a hexagon track at that moment, and then recovers as soon as the torque reaches the reference value.

Fig. 11 shows the flux locus of FLDTC at 20rad/sec and no load. We know that CDTC has the disadvantages such as flux locus distortion in very low speed. Simulation result proves that the flux locus can be improved by using the proposed controller. The success can be attributed to the rational selection between reversed voltage vectors and zero voltage vectors.



(a). Flux locus of FLDTC in steady state



(b). Flux locus of FLDTC in dynamic state

Figure 10. Flux locus of FLDTC

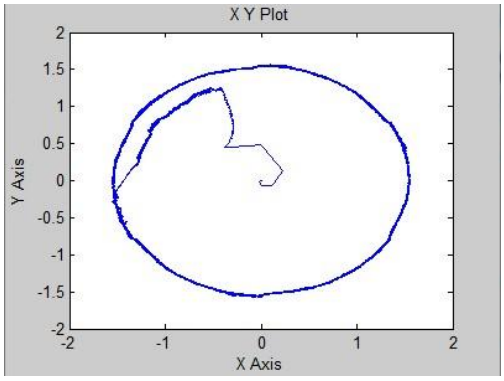


Figure 11. Flux locus of FLDTC at 20 rad/sec

4 Conclusion

A fuzzy logic based direct torque control system is implemented in this paper to improve the performance of conventional DTC system. The FL controller enables the system to choose optimal stator voltage vectors producing the most suitable rate of torque change according to the six fuzzy variables. Simulation results have shown the effectiveness of the proposed method. Through the comparison between CDTC and FLDTC, we have shown that the FLDTC design in this paper keeps all the advantages of CDTC, and makes some improvement in reducing torque ripples, faster torque response, and stability at very low speed.

5 References

[1] Chintan Patel, Rajeevan P. P, Anubrata Dey, Rijil Ramchand, K. Gopakumar, "Fast Direct Torque Control of Open-end Induction Motor Drive using 12-sided Polygonal Voltage Space Vectors," IEEE Transactions on Power Electronics, Vol. 27, Issue 1, 400 – 410, Jan. 2012.

[2] Eric Monmasson, Marcian N. Cirstea, "FPGA Design Methodology for Industrial Control Systems—A Review," IEEE transactions on industrial electronics, Vol. 54, Issue. 4, 1824-1842, Aug. 2007.

[3] Yongchang Zhang, Jianguo Zhu, Zhengming Zhao, Wei Xu, David G. Dorrell, "An Improved Direct Torque Control for Three-Level Inverter-Fed Induction Motor Sensorless Drive," IEEE transactions on power electronics, Vol. 27, Issue 3, 1502 – 1513, Mar. 2012.

[4] Narasimham PVRL, Sarma AVRS, Vargilkumar E, "A Sector Advanced Technique to improve dynamic response of a Direct Torque Controlled induction motor," 2010 IEEE International Conference on Power and Energy (PECon), 456-461, Dec. 2010.

[5] Auzani Jidin, Tole Sutikno, Aiman Z. Jidin, Nik Rumzi Nik Idris, Abdul Halim M. Yatim, "A Novel Dynamic Overmodulation Strategy of Direct Torque Control," Indonesian Journal of Electrical Engineering, Vol. 8, Issue 3, 275 – 284, Dec. 2010.

[6] S.A.Zaid, O.A.Mahgoub, K.A.El-Metwally, "Implement -ation of a new fast direct torque control algorithm for induction motor drives," IET Electr. Power Appl., 2010, Vol. 4, Issue 5, 305–313, May 2010.

[7] Turki Y. Abdalla, Haroution Antranik Hairik, Adel M. Dakhil, "Minimization of Torque Ripple in DTC of Induction Motor Using Fuzzy Mode Duty Cycle Controller," 2010 1st International Conference on Energy, Power and Control (EPC-IQ), 237-244, Dec. 2010.

[8] Soufien Gdaim, Abdellatif Mtibaa, Mohamed Faouzi Mimouni, "Direct Torque Control of Induction Machine based on Intelligent Techniques, " International Journal of Computer Applications, Vol. 10, Issue 8, 29-35, Nov. 2010.

[9] R.Toufouti S, Meziane, H. Benalla, "Direct Torque Control for Induction Motor Using Fuzzy Logic," ACSE Journal, Vol. 6, Issue 2, 19-26, June, 2006.

[10] F. Zidani, D. Diallo, M. E. H. Benbouzid, and R.Nait-Said, "Direct Torque Control of Induction Motor With Fuzzy Stator Resistance Adaptation," IEEE transactions on energy conversion, Vol. 21, Issue 2, 619-621, June 2006.

[11] YAN Wei-Sheng, LIN Hai, LI Hong, Yan Wei, "Sensorless Direct Torque Controlled Drive of Brushless DC Motor based on Fuzzy Logic," 4th IEEE Conference on Industrial Electronics and Applications, 2009. ICIEA 2009, 3411 – 3416, May 2009.

[12] Gao Sheng-wei, Wang You-Hua1, Cai Yan, Zhang Chuang, "Research on Reducing Torque Ripple of DTC Fuzzy Logic-based," 2010 2nd International Conference on Advanced Computer Control (ICACC), Vol. 2, 631 – 634, Mar. 2010.

[13] Yuedou Pan, Yihai Zhang. "Research on Direct Torque Control of Induction Motor Based on Dual-Fuzzy Space Vector Modulation Technology", Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09, Vol. 6, 383-388, Aug. 2009.

A Memetic Algorithm for Parallel Machine Scheduling

Serafettin Alpay
Eskişehir Osmangazi University, Industrial Engineering Department, Eskisehir, Turkiye

Abstract - *This paper focuses on the problem of scheduling a set of independent jobs with setup times and job splitting, on a set of identical parallel machines such that total tardiness is minimized. In this study, it is assumed that a job can be split into sub-jobs and these sub-jobs can be processed independently on parallel machines. Because the problem is NP-hard, a memetic algorithm (MA) is proposed. Several test problems are solved via MA and its performance is compared to solutions obtained via simulated annealing (SA) and tabu search (TS) approaches from the literature. Experimental results reveal that proposed MA produces better total tardiness performance than SA and TS.*

Keywords- Memetic Algorithms, Parallel Machines, Scheduling, Total Tardiness, Job Splitting

1 Introduction

The scheduling of jobs on parallel machines provides for interesting and challenging combinatorial problems that continue to interest researchers [1]. Parallel machine scheduling comes down to assigning each operation to one of the machines and sequencing the operations assigned to the same machine. We may have identical, uniform or unrelated parallel machines. If the machines are identical, then the processing time of each job is the same on all machines. Uniform machines work at different speeds, i.e., the processing time of each job differs by a constant factor for the individual machines. If the machines are unrelated, then there is no relation between the processing times of the jobs and the machines. [2]

Generally, there are two decisions to be made in parallel-machine scheduling problems. One is to assign jobs to the machines, and the other is to determine the sequence of the jobs on each machine [3]. Therefore, parallel machine problems generally seem to be harder than single machine problems [4].

There are various published papers in parallel machine scheduling problems. The common objectives studied in this area include the minimization of completion time, tardiness, and make-span [5]. As the single machine total tardiness problem is strongly NP-hard, it follows that the parallel-machine total tardiness problem is strongly NP-hard, too [4]. It was showed that even the minimization of total tardiness in two identical machine scheduling problem was NP-hard [6].

In this paper, an identical parallel machine scheduling problem with the objective of minimizing of total tardiness is addressed while considering a job splitting property of the jobs. The jobs can be split into a number of sub-jobs that can be processed independently on two or more parallel machines at the same time. Such a problem is called the parallel machine scheduling problem with splitting jobs [7]. Also in the paper, a job is assumed to be composed of a number of unit-jobs and unit-jobs from a job are identical. So, the processing times and the due dates of unit-jobs are the same. A set of unit-jobs from a job is defined as a sub-job that is processed on a machine consecutively and the sub-jobs from a job are processed on the parallel machines independently.

Although many studies have been published on parallel machine problems [8], there are very few research results on identical parallel machine scheduling problems with job splitting properties [7]. Serafini [19] studied identical parallel machine problem with a job-splitting property. He considered the objective of minimizing the maximum weighted tardiness for both uniform and unrelated parallel machine environments. He showed that minimizing maximum weighted tardiness can be done in polynomial time. Xing and Zhang [9] also consider the same problem with the objective of minimizing makespan and they proposed a heuristic algorithm to solve it. Kim et al. [10] proposed a two-phase heuristic algorithm for identical parallel machine scheduling problems with the objective of minimizing total tardiness. Logendran and Surbur [11] reported a methodology for minimizing the total weighted tardiness of all jobs intended to be processed on unrelated parallel machines while each job can only be split into two portions. Tahar et al. [12] studied the problem of scheduling a set of independent jobs with sequence-dependent setup times and job splitting, on a set of identical parallel machines such that maximum completion time (makespan) is minimized. They suggested a heuristic algorithm using a linear programming modeling with setup times and job splitting considerations. They tested the performance of their algorithm on large number of randomly generated instances. Shim and Kim [7] proposed a branch and bound

(B&B) algorithm for the identical parallel machine scheduling problem with the objective of minimizing total tardiness considering the job splitting property. They developed several dominance properties and lower bounds for the problem and incorporated them with their B&B algorithm. They concluded that the suggested algorithm solves problems of moderate sizes in a reasonable amount of computation time. Sarıçiçek and Çelik [13] dealt with the scheduling problem of identical parallel machines with splitting jobs. They formulated a mixed integer programming model for the problem and proposed two meta-heuristics: simulated annealing (SA) and tabu search (TS). Their computational results showed that SA has a better performance and consumes less time than TS, so they suggested SA as a better heuristic method than TS for considered problem.

In this paper, a memetic algorithm (MA) is proposed for parallel machine scheduling problem addressed above. Several test problems provided by Sarıçiçek and Çelik [13] are solved by MA and its performance is compared to their results obtained via SA and TS heuristics.

2 Problem Description

Some definitions used in this paper are given below.

Job : a production order unit to be composed of a number of unit-jobs

Unit-Job: A unit of a minimum process batch for a job. Unit-jobs from a job are identical so their processing times and their due dates are the same.

Sub-Jobs: A set of unit-jobs from a job that is processed on a machine consecutively. Sub-jobs from a job are processed on the parallel machines independently.

Tardiness of a job: $T_j = max\{0, C_j - d_j\}$, where C_j and d_j are the completion time and due date of job j, respectively

Completion time of a job: The time when all sub-jobs from the job are completed.

For the problem addressed here, it is assumed that:

- As the machines are identical, processing times of a unit-job on all the machines are equal,
- All jobs are available at time zero.
- Each machine can process only one sub-job at time.
- Each sub-job can be processed on only one machine.
- A setup operation is required before a sub-job is processed on a machine, if the job type of a sub-job to be processed is different from the job type of the sub-job just processed and setup times are independent of sequences of the sub-jobs.

Other notations:

m: number of parallel machines
n: number of jobs
i: index for machines, $i=1,\ldots,m$
j: index for jobs, $j=1,\ldots,n$
k: index for position of sub-job of job j on machine, $k=1,\ldots,n$
u_j: number of unit-jobs of job j
p_j: processing time of each unit-job of job j
s_j: sequence-independent setup time for job j
d_j: due date of job j
M: a large constant number which is at least as large as the sum of the processing times and setup times of all jobs

x_{ijk} : *1*; if the sub-job of j is processed on machine i in the kth position, *0*; otherwise:

y_{ijk} : The number of unit-jobs of the sub-job of job j which is processed on machine i in the kth position;

T_{ijk} : Tardiness of the sub-job of a job j processed on machine i in the kth position.

G_j : a decision variable that can have a value greater than T_{ijk} and is constrained to be non-negative.

Based on the definitions, notations and assumptions above, Sarıçiçek and Çelik [13] presented a mathematical model which is also considered in this study. The objective function of the model is minimizing the total tardiness and expressed as:

$$Minimize \sum_{j=1}^{n} \left\{ \underset{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant k \leqslant n}}{Maximize}\, T_{ijk} \right\} \quad (1)$$

Mathematical model:

$$\text{Minimize} \sum_{j=1}^{n} G_j \quad (2)$$

s.t

$$\sum_{j=1}^{n} x_{ijk} \leqslant 1 \quad \forall(i,k) \quad (3)$$

$$\sum_{k=1}^{n} x_{ijk} \leqslant 1 \quad \forall(i,j) \quad (4)$$

$$x_{ijk} \leqslant y_{ijk} \leqslant u_j x_{ijk} \quad \forall(i,j,k) \quad (5)$$

$$\sum_{i=1}^{m} \sum_{k=1}^{n} y_{ijk} = u_j \quad \forall(i) \quad (6)$$

$$T_{ijk} \geqslant \sum_{s=1}^{\kappa} \sum_{l=1}^{n} (p_l y_{ils} + s_l x_{ils}) - d_j - M(1 - x_{ijk}) \quad \forall(i,j,k) \quad (7)$$

$$T_{ijk} \geqslant 0 \quad \forall(i,j,k) \quad (8)$$

$$x_{ijk} \in \{0,1\} \quad \forall(i,j,k) \quad (9)$$

$$y_{ijk} \geqslant 0, \quad \text{integer} \quad \forall(i,j,k) \quad (10)$$

$$G_j \geqslant T_{ijk} \quad \forall(i,j,k) \quad (11)$$

$$G_j \geqslant 0 \quad \forall(j) \quad (12)$$

As the problem is NP-hard, finding an exact solution to the model in polynomial time is possible for only small sized problems in which *n* is relatively small, i.e. n<10. If *n* gets large, the time required to solve such problems increases exponentially with respect to *n* [13]. On the other hand, heuristic methods can produce good solutions (possibly even an optimal solution) quickly. In this paper, a memetic algorithm is proposed for the problem.

3 Proposed Memetic Algorithm

Memetic Algorithm also called hybrid genetic algorithm, represents a new meta-heuristic for combinatorial optimization problems **[**14]. The proposed MA integrates Variable Neighborhood Decent (VND) with the Genetic Algorithm (GA) to improve the solutions. During the processing of MA, VND is executed on each individual in the population to find the better solution. If it is found, it is replaced with the original individual.

The flowchart of the proposed MA is illustrated in Figure 1.

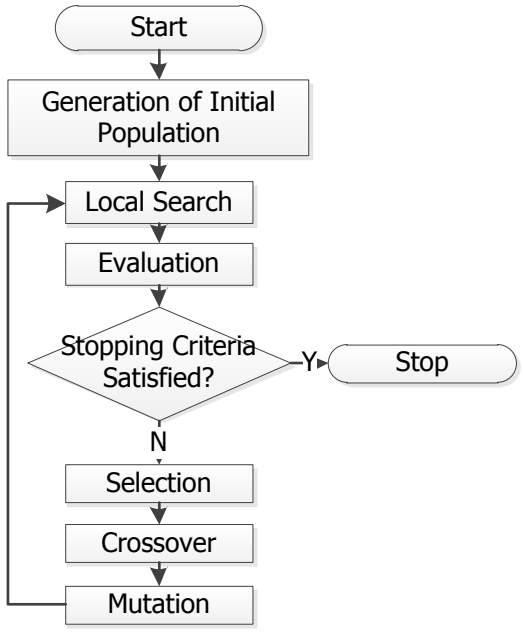


Fig. 1 Flowchart of the Proposed MA

Step 1: The initial population is generated by randomly. Each chromosome is encoded by using Earliest Due Date (EDD) dispatching rule on each initial solution. What that means is that randomly assigned unit jobs on each machine are sequenced by using EDD rule.

Step 2: To improve the solution quality, the local search based on VND is executed on each individual.

Step 3: Each individual is evaluated and obtained its fitness function value, that is, total tardiness. Then, all total tardiness values are compared with each other to find the local best solution. If the local best solution is better than the global best solution found so far in whole process, the global best solution is updated.

Step 4: If the stopping criteria is satisfied, the iterative process is stopped and the global best solution is returned. Otherwise, the process is continued with step 5.

Step 5: New population is generated by executing selection, crossover and mutation operators respectively. Then the process is continued with step 2.

3.1 Solution representation and selection mechanism

The most commonly used solution representation for the parallel machine scheduling problem is an array of jobs for each machine that represents the processing order of the jobs assigned to that machine [15]. As the jobs are considered to be consisted of unit jobs in this study, the arrays are based on unit jobs represented by a unique job numbers so the unit jobs of a job have the same job number. An example solution representation for 2 jobs, 2

machines, 3 unit jobs is given in Figure 2. The figure shows a single array composed by two groups of elements for two machines. The number of elements related to each machine is counted by multiplying the number of jobs and the number of unit jobs. For the example, the array consists of 6 (*2x3*) elements for each machine so there are total of 12 elements in the array.

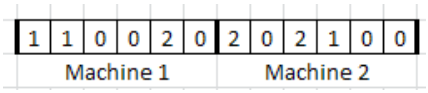


Fig. 2. Example solution encoding

The jobs given in the example are represented by the digits of "1" and "2" respectively in Figure 2. Each "1" denotes a unit job of job 1 assigned to the machines and each "2" shows a unit job of a job 2 assigned to the machines. "0" denotes no unit-jobs assigned to that position. Consequently, according to the figure 2, the processing order of the unit jobs is *1,1,2* for machine 1 and *2,2,1* for machine 2.

The selection is performed using a roulette wheel selection mechanism. The idea behind the roulette wheel selection is that each individual is given a chance to become a parent in proportion to its fitness.

3.2 Genetic operators

In genetic algorithms, genetic operators are used to combine existing solutions into new better solutions (crossover operator) and to generate diversity (mutation operator).

Crossover:

In general, the goal of the crossover operator is to generate two good individuals, called offspring, from the two selected progenitors. One of the most used crossover operators to the parallel machine case is the one-point-crossover [15]. So one-point-crossover is performed in this study. This type of crossover includes one point which is randomly selected for dividing first parent [16]. Once the parents have been selected, the crossover operator is applied according to the crossover probability that is 0.55 in the study. To prevent the generation of infeasible solutions after crossover operation, a repair operator is applied on the new offsprings. Repair operator can both repair infeasible total number of unit-jobs for each job by considering unchanged elements beyond the crossover point and infeasible solutions contains more than one sub-jobs of the same job on the same machine.

Mutation:

Once the offspring is obtained, the mutation operator is applied according to the mutation probability that is 0.05 in the study. The mutation operator is used to reduce the convergence rate [17]. In other words, the application of the mutation operator can increase the diversity of the population and avoid local optimization [16]. Two genes in the chromosome are randomly selected and exchanged their values. If a chromosome after mutation becomes infeasible according to the constraints 4 in the model then the repair operator is applied to maintain feasibility.

3.3 Local search

The local search procedure is employed to enrich and diversify the population produced by the genetic crossover operator without visiting other regions of the search space [16]. In this study, VND is used as local search procedure. VND is a variant of Variable Neighborhood Search (VNS) where the change of neighborhood is realized in a deterministic way during the local search phase [14]. Two different search neighborhoods [18] incorporated into the VND are defined below.

Forward insertion neighborhood

By starting from the leftmost unit job in an individual, all unit jobs are shifted forward to other positions respectively. After each shifting operation, a new solution is generated. From the generated solutions, non-feasible ones violate the constraints 4 in the model are eliminated. If an improvement occurs during the generating process, the original individual is updated. The process is stopped when no more improvement is found and the current solution is accepted as locally optimal.

Backward insertion neighborhood

This type of neighborhood is very similar to forward insertion neighborhood except all shifting operations are performed in reverse order by starting from the rightmost unit job. The process continues until no shifting operation gives a better result and the current solution is accepted as locally optimal.

All shifting operations for both search neighborhoods are performed without considering "0" values in the chromosomes to reduce the CPU times to complete local searches.

Based on the above neighborhood structures, the local search phase is described below:

Local search

0 Select first individual from the population.

1 Take the individual as the current solution and execute backward insertion neighborhood search. The local optimal solution → the current solution.

2 Take the current solution and execute forward insertion neighborhood search. The local optimal solution → the current solution.

3 If the local search was performed for all individuals then *Stop*, otherwise select the next individual on which local search has not been performed yet from the population and go to step *1*.

4 Computational Results

The proposed MA was coded in Borland Delphi 7 and run 10 times at each problem on Intel Core 2 Duo T6400 2.0 Ghz computer. The size of population varies according to the problem size. All other required parameter values for the problem are generated in a same way as Sarıçiçek and Çelik do: the processing time of a unit-job is integer and randomly generated from uniform distribution with range [5, 60]. Setup time for a job may be short, middle or long and it is generated from the discrete uniform distribution with range [5, 60] for short, [60, 120] for middle and [120, 180] for long. Due date of job j, d_j, is generated from the discrete uniform distribution with range $[\alpha\sum(s_j + u_j p_j)/m, \beta\sum(s_j + u_j p_j)/m]$, where s_j, p_j and u_j denote the setup time of job j, processing time of a unit-job of job j and the number of unit-jobs associated with job j respectively, α and β are parameters used to control tightness (and range) of due dates. A pair of values for (α, β) is used as (0, 1.2) in this study.

The performance related results of the proposed MA for average of 100 generations were compared with the results from the search heuristic approaches of SA and TS for the test problems and detailed in the Tables 1, 2, 3 respectively. The results for SA and TS on the test problems are reported by Sarıçiçek and Çelik [13]. The reported performance measures are based on *Total Tardiness (TT)* and *CPU time*. A comparison of the results of total tardiness of SA, TS and MA is reasonable but, because of the difference of the test platforms, the CPU time results of MA and other heuristics are not comparable. Nevertheless, the CPU time results of MA are given in this study for information purposes only. Also in the study, *the number of tardy jobs* (*n.of.TJ*) results obtained from MA runs are reported as another performance measure.

Table 1 – Test problems 1 and obtained results

Jobs	Machines	Unit-Jobs	Setup	n.of.TJ	Total Tardiness			CPU Time (s)		
					MA	TS	SA	MA	TS	SA
10	4	4	Short	2,6	68,90	282,00	280,00	0,24	2,671	0,188
			Medium	3	231,40	321,00	529,00	0,32	2,578	0,188
			Long	2	181,60	319,00	548,00	0,56	2,641	0,188
15	4	6	Short	2,5	155,10	1109,00	1320,00	1,91	7,625	0,5
			Medium	4,6	577,80	2507,00	2601,00	3,42	7,546	0,469
			Long	2,6	418,40	2039,00	2127,00	4,89	7,469	0,468
15	8	6	Short	4,7	265,10	959,00	1076,00	3,97	7,891	0,531
			Medium	5,4	502,90	1769,00	1657,00	5,59	7,781	0,5
			Long	5	371,00	2184,00	2194,00	7,06	7,625	0,5

Table 2 – Test problems 2 and obtained results

Jobs	Machines	Unit-Jobs	Setup	n.of.TJ	Total Tardiness			CPU Time (s)		
					MA	TS	SA	MA	TS	SA
20	8	8	Short	6,1	793,10	2696,00	2901,00	2,73	17,39	1,047
			Medium	7,8	1529,70	2341,00	2154,00	11,96	16,734	0,984
			Long	7,1	1330,20	2606,00	2571,00	4,68	16,641	1
20	12	8	Short	8	695,10	2556,00	2710,00	5,69	17,594	1,078
			Medium	8	1353,10	4940,00	4696,00	19,02	17,313	1,015
			Long	7,1	1158,60	3874,00	3623,00	6,68	16,718	1,015
25	12	10	Short	7,6	942,90	2246,00	1863,00	32,72	32,656	1,828
			Medium	9,7	2160,00	5513,00	5344,00	40,94	32,078	1,781
			Long	7	1637,30	4332,00	3942,00	24,17	31,812	1,75

Table 3 – Test problems 3 and obtained results

Jobs	Machines	Unit-Jobs	Setup	n.of.TJ	Total Tardiness			CPU Time (s)		
					MA	TS	SA	MA	TS	SA
30	15	10	Short	2,29	76,71	3037,00	2672,00	5,74	46,406	2,625
			Medium	3,29	154,43	4584,00	4315,00	9,64	45,281	2,453
			Long	6,14	605,14	6585,00	6169,00	9,93	45,281	2,485
40	20	12	Short	9,43	1038,57	8716,00	8901,00	116,44	92,921	5,313
			Medium	15,86	4145,30	5688,00	5174,00	143,88	96,062	5,125
			Long	17,14	4581,29	10549,00	10532,00	181,86	93,61	5,565
50	20	12	Short	2,00	85,57	7331,00	7524,00	50,18	146,95	7,797
			Medium	5,57	369,43	8037,00	7898,00	74,17	145,64	7,485
			Long	7,71	847,29	10752,00	10607,00	75,02	144,44	7,502

Tables show means of the three performance measures of interest: number of tardy jobs, total tardiness and CPU time. The tables are organized by the type of search procedure used: MA, TS and SA.

5 Conclusions

The results indicate that total tardiness performance of MA is superior to that of other heuristics for all test problems. As the problem size grows, MA gives much better performance in terms of total tardiness when

compared to TS and SA. Only for 40 jobs, 20 machines, 12 unit jobs and medium setup problem, the *TT* performance of MA is nearly same or similar to the performances of TS and SA. It should be noted that the local search presented here will probably need more CPU time to end up searching for the feasible search space than those of TS and SA heuristics as the problem size grows. So, one further investigation would be to enhance the computational efficiency of this approach especially for really large size scheduling problems.

6 References

[1] A. Cossari, J.C. Ho and G. Paletta, "A new heuristic for workload balancing on identical parallel machines and a statistical perspective on the workload balancing criteria", *Computers and Operations Research*, 39, 2012, pp. 1382-1393

[2] I. A. Chaudhry, S. Mahmood and R. Ahmad , "Minimizing Makespan for Machine Scheduling and Worker Assignment Problem in Identical Parallel Machine Models Using GA", *Proceedings of the World Congress on Engineering 2010*, Vol III, WCE 2010, June 30 - July 2, 2010, London, U.K.

[3] S. Shim and Y. Kim, "Scheduling on parallel identical machines to minimize total tardiness", *European Journal of Operation Research*, 177, 2007, pp. 135-146

[4] D. Biskup, J. Herrmann and J.N.D. Gupta, "Scheduling identical parallel machines to minimize total tardiness", *Int. J. Production Economics,* 115, 2008, pp. 134-142.

[5] D. Kim, K. Kim, W. Jang and F.F. Chen, "Unrelated parallel machine scheduling with setup times using simulated annealing", *Robotics and Computer Integrated Manufacturing*, 18, 2002, pp. 223–231

[6] R.M. Karp, "Reducibility among combinatorial problems: complexity of computer computations", *NewYork: Plenum Press*, 1972, p. 85–103.

[7] S. Shim and Y. Kim, "A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property", *Computers & Operations Research,* 35, 2008, 863 – 875

[8] M. Ranjbar, M. Davari and R. Leus, "Two branch-and-bound algorithms for the robust parallel machine scheduling problem", *Computers & Operations Research,* 39, 2012, pp. 1652–1660

[9] W. Xing, J. Zhang, "Parallel machine scheduling with splitting jobs", *Discr. Appl. Math*., 103, 2000, pp. 259–269

[10] Y.D. Kim, S.O. Shim, S.B. Kim, Y.C. Choi, H.M. Yoon, "Parallel machine scheduling considering a job splitting property", *Int. J. Prod. Res,.* 42, 2004, pp. 4531–4546.

[11] R. Logendran, F. Subur, "Unrelated parallel machine scheduling with job splitting", *IIE Transactions,* 36, 2004, pp. 359–372

[12] D.N. Tahar, F. Yalaoui, C. Chu and L. Amodeo, "A linear programming approach for identical parallel machine scheduling with job splitting and sequence dependent setup times", *Int. J. Prod. Econ*., 99, 2006, pp. 63–73.

[13] İ. Sarıçiçek and C. Çelik, "Two Meta-heuristics for Parallel Machine Scheduling with Job Splitting to Minimize Total Tardiness", *Applied Mathematical Modelling*, 35, 2011, pp. 4117-4126.

[14] B. Wang and G. Zhang, "Hybrid VNS and Memetic Algorithm for Solving the Job Shop Scheduling Problem", 18th International Conference on Industrial Engineering and Engineering Management (IE&EM 2011), 2011, China, pp. 924-927

[15] E. Vallada and R. Ruiz, "A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times", *European Journal of Operational Research*, 211, 2011, pp. 612-622.

[16] M. Souki, S.B. Youssef and A. Rebai, "Memetic Algorithm for Operating Room Admissions", *International Conference on Computers & Industrial Engineering, 2009. CIE 2009,* 519-524, 2009

[17] C.R.A., Reeves, "Genetic Algorithm for Flowshop Sequencing", *Computers and Operations Research,* 22, 1, 1995, pp. 5-13.

[18] Ş. Alpay, "GRASP with path relinking for a multiple objective sequencing problem for a mixed-model assembly line", *International Journal of Production Research*, Vol 47, No. 21, 2009, pp. 6001-6017

[19] P. Serafini, "Scheduling jobs on several machines with job splitting property", *Oper. Res.*, 44, 1996, pp. 617–628.

An Intelligent Invasive Weed Optimization: a Q-learning Approach

Abhronil Sengupta [1], **Tathagata Chakraborti** [1], **Amit Konar** [1], **and Atulya K. Nagar** [2]
[1] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India
[2] Department of Computer and Math Sciences, Liverpool Hope University, United Kingdom

Abstract - *Invasive Weed Optimization is a recently proposed population based meta-heuristic that mimics the colonizing action of weeds. In this article, an improvement to the classical algorithm has been proposed by introducing a constriction factor in the seed dispersal stage. Temporal Difference Q-Learning has been employed to adapt this parameter for different population members through the successive generations. The proposed memetic approach, named Intelligent Invasive Weed Optimization (IIWO) has been tested extensively on a set of 15 benchmark functions as well as the real world Circular Antenna Array Design problem. The results indicate the efficacy of our proposed approach.*

Keywords: Invasive Weed Optimization (IWO), Memetic Algorithms, Q-Learning

1 Introduction

Invasive Weed Optimization (IWO) [1] is a derivative-free optimization technique that mimics the ecological behavior of weeds. This meta-heuristic algorithm has attracted researchers because of its reduced computational cost and efficiency in tackling real world optimization problems. However, it is not free from the problems of stagnation and pre-convergence. We attempt to improve the performance of the traditional IWO algorithm by incorporating a learning strategy in the weed population to efficiently disperse seeds throughout the problem space during the reproduction phase. Such a memetic learning technique helps in balancing the exploration and exploitation capabilities of the weeds which is necessary for providing precise solutions to global optimization problems.

Coined by Dawkins[2] in 1976, the term "meme" refers to the basic unit of cultural transmission or imitation [1]. Memetic Algorithms (MAs) are population-based meta-heuristic search algorithms that combine the composite benefits of natural and cultural evolution. Natural evolution realized by Evolutionary Algorithm (EA) works on the Darwinian principle of the struggle for existence, and aims at determining the global optima in a given search landscape. Traditional EA usually takes an excessively large time to locate a precise enough solution because of its inability to exploit local information. Cultural evolution, on the other hand, is capable of local refinement. MA captures the power of global search by its evolutionary component and local search by its cultural component.

The early research on MA was confined in manual crafting of dedicated memes for a given problem. A paradigm shift in research to adaptively select a meme from a pool of memes for application to an individual member of the population has been observed during the new millennium. The class of algorithms incorporating the adaptive selection of memes is referred to as *Adaptive MA* (AMA). AMAs "promote both cooperation and competition among various problem-specific memes and favors neighborhood structures containing high quality solutions" to be attained at low computational costs. Usually, the selection of the meme for an individual member of the population is done based on its ability to perform local improvement.

Several variants of AMAs are found in the literature [4-5]. The one we would use in this paper is Roulette-Choice strategy based Hyperheuristic AMA [4]. In the Roulette-choice strategy, a meme M_e is selected with probability relative to the overall improvement. Given that $g(.)$ is a choice function, then the probability of selection of M_e is $g(M_e)/\sum_{i=1}^{n} g(M_i)$ where n is the total number of memes considered.

The AMA to be proposed, named Intelligent Invasive Weed Optimization (IIWO) includes an Invasive Weed Optimization (IWO) algorithm for global search and a Temporal Difference Q-Learning (TDQL) [6-7] for local refinement. A constriction factor has been included in the expression for standard deviation for dispersal of seeds. It is important to mention here that the constriction factors for all members of the population should not be equal for the best performance. A member with a good fitness should search in the local neighbourhood, whereas a poor performing member should participate in the global search. A good member thus should have small constriction factors, while worse members should have relatively large constriction factors. This is realized in the paper with the help of TDQL.

The TDQL works on the principle of reward and penalty. It employs a Q-table to store the reward/penalty given to an individual member of the population. Members are assigned suitable values of their constriction factors from a given meme pool before participation in the evolutionary process. After completion of the evolutionary process, members are rewarded based on their fitness, and the

reward/penalty given to the member depending on the improvement/deterioration in fitness measures of the trial solution is stored in the *Q*-table. The process of evolution and *Q*-table updating thus synergistically helps each other, resulting in an overall improvement in the performance of the AMA.

The rest of the paper is organized as follows. Sections 2 and 3 provide an overview of the Classical IWO algorithm and Differential *Q*-Learning. Our proposed approach has been described in Section 4. Extensive experimental results comparing the IIWO algorithm with IWO as well as other popular meta-heuristic algorithms namely Particle Swarm Optimization (PSO) [8] and Differential Evolution (DE) [9-10] have been presented in Section 5. Comparative results have been presented on a set of 15 benchmark functions as well as the Circular Antenna Array Design problem.

2 An Outline of Iwo Algorithm

2.1 Generation of Initial Population

IWO starts with a population of *NP* *D*-dimensional parameter vectors or weeds representing the candidate solutions. We shall denote subsequent generations in IWO by $G = 0, 1, \ldots, G_{max.}$ We represent the *i*-th vector of the population at the current generation as:

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}]$$

The initial population (at G = 0) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds:

$$\vec{X}_{min} = \{x_{1,min}, x_{2,min}, \ldots, x_{D,min}\}$$

and $\vec{X}_{max} = \{x_{1,max}, x_{2,max}, \ldots, x_{D,max}\}$

So we may initialize the *j*-th component of the *i*-th vector as

$$x_{j,i,0} = x_{j,min} + rand_{i,j}(0,1).\left(x_{j,max} - x_{j,min}\right) \quad (1)$$

where $rand_{i,j}(0,1)$ is a uniformly distributed random number lying between 0 and 1 and is instantiated independently for each component of the *i*-th vector.

2.2 Reproduction

The plants will produce seeds depending on their relative fitness which will be spread out over the problem space. Each seed, in turn, will grow into a flowering plant. Thus, if S_{max} and S_{min} denote the number of seeds produced by plants with best and worst fitness respectively then seed count of plants will increase linearly from S_{min} to S_{max} depending on their corresponding fitness values. The number of seeds produced by the *i*-th weed $\vec{X}_{i,G}$ is therefore given by,

$$s_{i,G} = \left\lfloor \frac{F_{max,G} - f\left(\vec{X}_{i,G}\right)}{F_{max,G} - F_{min,G}} . (S_{max} - S_{min}) \right\rfloor \quad (2)$$

where $F_{max,G}$ and $F_{min,G}$ are the maximum and minimum fitness values at the *G*-th generation of the weed colony.

2.3 Dispersal of Seeds through Search Space

The produced seeds are randomly distributed over the *D* dimensional search space by random numbers drawn from a normal distribution with zero mean but with a varying variance. However, the standard deviation (SD), σ, of the normal distribution decreases over the generations from an initial value, σ_{max}, to a value, σ_{min}, and is determined by the following equation,

$$\sigma = \left(\frac{G_{max} - G}{G_{max}}\right)^n . (\sigma_{max} - \sigma_{min}) + \sigma_{min} \quad (3)$$

where σ is the SD at the current generation and G_{max} is the maximum number of iterations while *n* is the non linear modulation index. This is the adaptation property of the algorithm.

2.4 Competitive Exclusion

If a plant does not reproduce it will become extinct. Hence this leads to the requirement of a competitive exclusion in order to eliminate plants with low fitness values. This is done to limit the maximum number of plants in the colony. Initially fast reproduction of plants take place and all the plants are included in the colony. The fitter plants reproduce more than the undesirable ones. The elimination mechanism is activated when the population exceeds a stipulated NP_{max}. The plants and produced seeds are ranked together as a colony and plants with lower fitness values are eliminated to limit the population count to NP_{max}. This is the selection property of the algorithm. The above steps are repeated until maximum number of iterations is reached..

3 Differential Q-Learning

In classical *Q*-learning, all possible states of an agent and its possible actions in a given state are deterministically known. In other words, for a given agent A, let $S_1, S_2, \ldots, S_n$, be *n*- possible states, where each state has *m* possible actions $a_1, a_2, \ldots, a_m$. At a particular state-action pair, the specific reward that the agent acquires is known as *immediate reward*. Let $r(S_i, a_j)$ be the immediate reward that the agent A acquires by executing an action a_j at state S_i. The agent selects its next state from its current states by using a policy. The policy attempts to maximize the cumulative reward that the agent could acquire in subsequent transition of states from its next state.

Step 1 For each state S and action a, initialize $Q(S,a) = 0$.
Step 2 Observe the current state S_i
Step 3 REPEAT
Select $a_j \in \{a_1, a_2, \dots, a_m\}$ and execute it.
Receive an immediate reward $r(S_i, a_j)$.
Observe the new state $S_k \leftarrow \delta(S_i, a_j)$.
Update the table entry $Q(S_i, a_j)$ by
$$Q(S_i, a_j) \leftarrow (1-\alpha).Q(S_i, a_j) + \alpha.(r(S_i, a_j) + \gamma \max_{a'} Q(\delta(S_i, a_j), a')).$$
$S_i \leftarrow S_k$
FOR EVER

Algorithm 1. Differential Q-Learning Algorithm.

Let the agent be in state S_i and is expecting to select the next best state. Then the Q-value at state S_i due to action of a_j is given by,

$$Q(S_i, a_j) = r(S_i, a_j) + \gamma \max_{a'} Q(\delta(S_i, a_j), a') \quad (4)$$

where $0<\gamma<1$ and $\delta(S_i,a_j)$ denotes the next state due to the selection of action a_j at state S_i. Let the next state selected be S_k. Then $Q(\delta(S_i,a_j), a') = Q(S_k, a')$. Consequently selection of a' that maximizes $Q(S_k, a')$ and in turn $Q(S_i, a_j)$ is an interesting problem.

The classical Q-learning algorithm for deterministic state transitions starts with a randomly selected initial state. An action 'a' from a list of actions $a_1, a_2, \dots, a_m$ is selected, and the agent because of this action receives an *immediate reward* r, and moves to the new state following the δ-transition rule given in a table. The Q-value of the previous state due to the action of the agent is updated following the Q-learning equation. Now, the next state is considered as the initial state and the steps of action selection, receiving immediate reward, transition to next state and Q-update are repeated forever.

Differential Q-learning is a modified version of Q learning. The Q-table update policy in Differential Q-learning is different from classical Q-learning. It has the ability to remember the effect of past Q value of a particular state-action pair while updating the corresponding Q value. The modified Q update equation is given by

$$Q(S_i, a_j) \leftarrow (1-\alpha).Q(S_i, a_j) + \alpha.(r(S_i, a_j) + \gamma \max_{a'} Q(\delta(S_k, a_j), a')) \quad (5)$$

The formula has the effect, that the Q-value $Q(S_i, a_j)$ is incremented, when the action a_j led to a state $\delta(S_i,a_j)$ in which there exists an action a', such that the best possible Q-value $Q(\delta(S_i,a_j), a')$ in the next time step plus the achieved reward $r(S_i,a_j)$ is greater than the current value of $Q(S_i, a_j)$. This is exactly the desired behaviour, because in such a situation, the old estimate of $Q(S_i, a_j)$ was too pessimistic. The learning rate α determines the extent to which the newly acquired information will override the old information. A setting of α= 0 makes the agent stop learning, while α=1 would make the agent consider only the most recent information.

The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent "opportunist" by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor is greater than or equal to 1, the Q values may diverge.

4 IIWO: The Proposed Approach

The modified algorithm is based on the concept that fitter individuals should be involved in local search while the remaining plants should search the problem space globally at a particular generation. The classical IWO algorithm neglects this fact by assuming the same standard deviation σ for all the weeds in the seed dispersal stage. Although σ is made to decay through the successive generations yet there is no provision for σ to attain low values for fitter individuals at a particular generation to enable the local search procedure. Local search is initiated only when the generation count has increased to a large value to ensure a low value of σ. Thus in classical IWO, all the weeds undergo a gradual behavioral transformation from an explorative to an exploitive one. In our proposed algorithm, we state that fitter individuals should behave in an exploitative manner through successive generations from the initialization of the weed colony and not wait for the standard deviation to reduce to low values. Following this concept we introduce a constriction factor, η in equation (3) as follows,

$$\sigma = \eta.\left(\left(\frac{G_{max} - G}{G_{max}}\right)^n.(\sigma_{max} - \sigma_{min}) + \sigma_{min}\right) \quad (6)$$

where $\eta \in (0,1]$. The proper choice of parameter η for different population members will help balance the explorative and exploitive capabilities of the individuals resulting in local refinement.

The proposed approach employs a synergy of IWO and TDQL to realize an Adaptive Memetic Algorithm for achieving superior performance in global optimization problems. After each evolutionary step, the performance of the members is evaluated based on their fitness. High performing members are rewarded with positive immediate reward, whereas low performing members are penalized. The reward/penalty given to a member is stored in the Q-table

using the TDQL learning rule. A meme pool for parameter η is maintained in order to select the control parameters for individual members of the population. The adaptive selection of memes is performed by a hyperheuristic choice-metric based selection from the meme pool. The process of selection of η from the meme pool, followed by one step of IWO and reward/penalty updating in the *Q*-table is continued until the condition for convergence of the AMA is satisfied.

The proposed AMA algorithm accesses the *Q*-table to select the appropriate constriction factors of the individual members before evolution, and updates the *Q*-table after one evolution. The row indices of the *Q*-table represent states of the population members obtained from the last iteration of the IWO algorithm, in order of their fitness. The column indices which represent the actions performed by the members at a particular state correspond to uniform quantized values of the control parameter in the range (0, 1]. For example, let the parameter under consideration be η with possible quantized values $\{\eta_1, \eta_2, \dots, \eta_{10}\}$. Then $Q(S_i, \eta_j)$ represents the total reward given to a member at state $S_i$for selecting $\eta = \eta_j$. The Roulette-Choice strategy is used to select a particular value of η from the meme pool $\{\eta_1, \eta_2, \dots, \eta_{10}\}$ using the $Q(S_i, \eta_j), j = 1,2, \dots,10$ for the individual member located at state S_i.

The adaptation of $Q(S_i, \eta_j)$ is done through a reward/penalty mechanism as used in classical TDQL. If a member of the population, residing at state S_i on selecting $\eta = \eta_j$ moves to a new state S_k by the evolutionary algorithm, and such state transition causes an improvement in fitness measure, then $Q(S_i, \eta_j)$ is given a positive reward following the TDQL algorithm. If the state transition results in no improvement in fitness measure, then a penalty is given to the selected $Q(S_i, \eta_j)$. The penalty is introduced by a decrease in *Q*-value. Principles used in designing the AMA are introduced below.

4.1 Initialization

The algorithm employs a population of *NP D*-dimensional parameter vectors representing the candidate solutions. The initial population (at $G = 0$) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds. Thus the *j*-th component of the *i*-th population member is initialized according to (1) as mentioned in section 2.

The entries for the *Q*-table are initialized as small values. If the maximum *Q*-value attainable is 100, then we initialize the *Q*-values of all cells in the *Q*-table as 1.

4.2 Adaptive Selection of Memes

We employ Fitness proportionate selection, also known as Roulette-Wheel selection, for the selection of potentially useful memes. A basic advantage of this selection mechanism is that diversity of the meme population can be maintained. Although fitter memes would enjoy much higher probability of selection, yet the memes with poorer fitness do manage to survive and may contribute some components as evolution continues. Mathematically, the selection commences by the selection of a random number in the range [0, 1] for each population member. Let us consider the selection from the η meme pool for a member of state S_i. The next step involves the selection of $\eta_j$such that the cummulative probability of selction of $\eta = \eta_1$ through η_{j-1} is greater than *r*. Symbolically,

$$\sum_{m=1}^{j-1} p(S_i, \eta = \eta_m) < r \leq \sum_{m=j}^{10} p(S_i, \eta = \eta_m) \qquad (7)$$

The probability of selection of $\eta = \eta_j$ from the meme pool $\{\eta_1, \eta_2, \dots, \eta_{10}\}$ is given by

$$p(S_i, \eta = \eta_j) = \frac{Q(S_i, \eta_j)}{\sum_{k=1}^{10} Q(S_i, \eta_k)} \qquad (8)$$

4.3 Invasive Weed Optimization

The IWO algorithm used here employs reproduction, seed dispersal and competitive exclusion as introduced in Section 3. The basic difference of the current realization is the selection of constriction factor η from the meme pool adaptively by step 4.2 before invoking the IWO process.

4.4 State Assignment

The population members are now ranked in increasing order of fitness and assigned corresponding states.

4.5 Updating the Q-table

Let a member at state S_i on selection of η_j moves to a new state S_k. The update equation for $Q(S_i, \eta_j)$ is given by,

$$Q(S_i, \eta_j) \leftarrow (1 - \alpha).\, Q(S_i, \eta_j) + \alpha.\, (r(S_i, \eta_j) + \gamma \max_{\eta'} Q(\delta(S_k, \eta_j), \eta')) \qquad (9)$$

The choice of the reward function is critical to the proper operation of the *Q*-learning mechanism. In case the seeds produced by a particular weed experience greater fitness in comparison to the parent weed then $r(S_i, \eta_j)$ is set equal to the absolute difference of fitness of the parent weed and the fittest seed. Otherwise a penalty of $-K$ is applied, however small.

Step 1 Set the generation number *G=0* and randomly initialize a population of *NP* individuals,
$P_G = \{\vec{X}_{1,G}, \vec{X}_{2,G}, \dots, \vec{X}_{NP,G}\}$ with $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \dots, x_{D,i,G}]$ with *i*= [*1, 2,……., NP*].
Initialize the *Q*-table: $Q(S_i, \eta_j) = 1 \; \forall i = [1, \dots, NP] and j = [1, \dots, 10]$
Step 2 Evaluate the population.
Step 3 WHILE stopping criterion is not reached, DO
Step 3.1 Initialize $r(S_i, \eta_j) = 0 \; \forall i = [1, \dots, NP] and j = [1, \dots, 10]$.
Step 3.2 /*Adaptive Selection of memes*/
FOR *i=1 to NP*
Select $\eta = \eta_k$ by Roulette-Wheel Selection.
END FOR
Step 3.3/*Reproduction*/
FOR *i=1 to NP*
Determine the number of seeds produced by the i-*th* population member at generation *G*,

$$s_{i,G} = \left\lfloor \frac{F_{max,G} - f(\vec{X}_{i,G})}{F_{max,G} - F_{min,G}} . (S_{max} - S_{min}) \right\rfloor$$

END FOR
Step 3.4/*Seed Dispersal*/
FOR *i=1 to NP*
FOR *j=1 to* $s_{i,G}$
Generate a population of $s_{i,G}$ seeds by,
$\vec{V}_{i,j,G} = [v_{1,i,j,G}, v_{2,i,j,G}, v_{3,i,j,G}, \dots, v_{D,i,j,G}]$
where $v_{k,i,j,G} = x_{k,i,G} + randn(0, \sigma)$ with *k*=[*1,2,…, D*]
and $\sigma = \eta . \left(\frac{G_{max} - G}{G_{max}}\right)^n (\sigma_{max} - \sigma_{min}) + \sigma_{min}$
END FOR
END FOR
Step 3.5/*Competitive Exclusion*/
Evaluate the colony of seeds and weeds.
FOR *i=1 to NP*
FOR *j=1 to* $s_{i,G}$
IF $f(\vec{V}_{i,j,G}) < f(\vec{X}_{i,G})$
$temp = | f(\vec{V}_{i,j,G}) - f(\vec{X}_{i,G}) |$
IF $temp > r(S_i, \eta_k)$
$r(S_i, \eta_k) = temp.$
END IF
ELSE $r(S_i, \eta_k) = -K.$
END IF
END FOR
END FOR
IF $NP + \sum_{i=1}^{i=NP} s_{i,G} > NP_{max}$
Form new weed colony with the first NP_{max}weeds arranged in order of fitness.
END IF
Step 3.6 /*State Assignment*/
Rank individuals in order of fitness and assign corresponding states.
Step 3.7/*Update of the *Q*-table*/
FOR *i=1 to NP*
FOR *j=1 to 10*
IF $r(S_i, \eta_j) \neq 0$
$Q(S_i, \eta_j) \leftarrow (1 - \alpha) . Q(S_i, \eta_j) + \alpha . (r(S_i, \eta_j) + \gamma \max_{\eta'} Q(\delta(S_k, \eta_j), \eta'))$
END IF
END FOR
END FOR
Step 3.8/*Increment the generation count*/ *G=G+1*
Step 4 END WHILE

Algorithm 2. The Proposed IIWO Algorithm.

The next step involves the determination of the factor $\max_{\eta'} Q(\delta(S_k, \eta_j), \eta'))$. A particular weed may enter the next generation along with multiple seeds or it may be completely eliminated. In case of multiple state acquisition in the next generation the factor is set equal to the maximum of $\max_{\eta'} Q(\delta(S_k, \eta_j), \eta'))$ for all S_k s. Otherwise it is set equal to 0 in case of plant exclusion.

The sections B-E are repeated till maximum number of iterations is reached.

5 Experiments and Results

5.1 Experimental Setup

We evaluate the performance of our proposed IIWO algorithm on a test-suite of 15 benchmark functions with varying degrees of complexity. The functions have been chosen from the benchmarks proposed in the CEC 2005 conference. Among them, the first five functions are unimodal while the remaining are multimodal. Due to lack of space we provide the results on the first 15 representative benchmarks. Details of the benchmark functions can be found in [12]. Results have been presented for 30 dimensions of all the benchmark functions. Each of the algorithms was run for a specified number of function evaluations: *D**1e+05 where *D* is the dimension of the problem. The mean value and standard deviation (within parenthesis) of the error in fitness value over 25 independent runs of each algorithm are presented in table 2.

Since all the algorithms start with the same initial population over each problem instance, we have used paired t-tests to compare the means of the results produced by the best and second-best algorithm (with respect to their final accuracies) for each benchmark. We have also reported the statistical significance level of the difference of means of the two algorithms in the respective columns of Table 2 and 3. The best performance has been highlighted in each row. The † sign indicates the t value of 49 degrees of freedom is significant at a 5% tolerance level of significance by 2 tailed test. The ‡ sign indicates that it is non-significant.

Comparisons have also been presented for the real world Circular Antenna Array Design problem [11]. The mean and standard deviation results have been presented after 1.5e+05 function evaluations. The optimization problem is briefly outlined below.

The array factor of a circular antenna array of N antenna elements placed on a circle of radius r in the x-y plane is given by:

$$AF(\phi) = \sum_{i=1}^{N} I_n \exp\left(jkr\left(\cos\left(\phi - \phi_{ang}^{n}\right) - \cos\left(\phi_0 - \phi_{ang}^{n}\right)\right) + \beta_n\right)$$

where $\phi_{ang}^{n} = 2\pi(n-1)/N$ is the angular position of the n[th] element in the x-y plane,

$kr = Nd$ where k is the wave-number, d is the angular spacing between elements and r is the radius of the circle defined by the antenna array,

ϕ_0 is the direction of maximum radiation,

ϕ is the angle of incidence of the plane wave,

I_n is the current excitation and

β_n is the phase excitation of the n[th] element.

Here we shall try to suppress side-lobes, minimize beamwidth and achieve null control at desired directions by varying the current and phase excitations of the antenna elements. For a symmetrical excitation of the circular antenna array objective function as:

$$OF = \left|AR(\varphi_{sll}, \vec{I}, \vec{\beta}, \varphi_0)\right| / \left|AR(\varphi_{max}, \vec{I}, \vec{\beta}, \varphi_0)\right| + 1/DIR(\varphi_0, \vec{I}, \vec{\beta}) + |\varphi_0 - \varphi_{des}| + \sum_{k=1}^{num} \left|AR(\varphi_k, \vec{I}, \vec{\beta}, \varphi_0)\right|$$

where φ_{sll} is the angle at which maximum sidelobe level is attained, φ_{des} is the desired maxima, *num* is the number of null control directions and φ_k specifies the k[th] null control direction.

The first component attempts to suppress the sidelobes. Nowadays directivity has become a very useful figure of merit for comparing array patterns. The second component attempts to maximize directivity of the array pattern and the third component strives to drive the maxima of the array pattern close to the desired maxima. The fourth component penalizes the objective function if sufficient null control is not achieved.

5.2 Other Competitive Algorithms

Differential Evolution and Particle Swarm Optimization has recently gained wide popularity as a fast and efficient optimization algorithm over continuous search spaces. We compare the performance of IIWO with classical IWO, DE and PSO. The parameter settings are given in the next page.

5.3 Simulation Results

The results obtained for the 15 benchmark problems as well as the real world optimization problem are tabulated below.

Table 1. Parameter Settings

PARAMETER	VALUE
Pop_size	50
Inertia weight	0.25-0.4
C_1, C_2	2
F	0.5
Cr	0.9
σ_{max}	10% of search range
σ_{min}	1% of search range

Table 2. Results for 30D Benchmark Problems

F.	IIWO	IWO	DE	PSO
1	**8.537e-01†** **(2.673e-01)**	4.977e+01 (9.453e-00)	7.229e+04 (2.599e+03)	1.523e+03 (3.743e+02)
2	**1.653e+00†** **(4.373e-01)**	8.251e+01 (9.487e+00)	7.176e+04 (6.289e+03)	8.578e+03 (2.788e+02)
3	**5.124e+05†** **(8.763e+04)**	2.899e+06 (5.11e+05)	6.286e+08 (5.271e+07)	8.176e+06 (2.389e+05)
4	**2.075e+00†** **(1.745e-02)**	1.667e+02 (1.013e+01)	4.389e+02 (1.865e+01)	4.391e+03 (5.283e+02)
5	2.481e+03 (8.351e+02)	**5.419e+01†** **(1.032e+01)**	2.747e+04 (2.577e+03)	1.011e+04 (3.733e+02)
6	**2.961e+02†** **(8.927e+01)**	3.766e+04 (1.198e+04)	3.281e+10 (2.744e+09)	5.789e+08 (7.639e+07)
7	**7.989e-02†** **(2.322e-03)**	1.836e+00 (1.921e-01)	2.836e+02 (4.899e+01)	3.137e+03 (5.533e+02)
8	**2.016e+01†** **(5.814e-05)**	2.094e+01 (1.344e-04)	2.115e+01 (4.436e-02)	2.291e+01 (2.487e-01)
9	1.185e+02 (4.013e+01)	**5.962e+01†** **(6.392e+01)**	7.321e+02 (2.987e+01)	7.491e+01 (3.987e+01)
10	1.173e+02 (1.332e+01)	**8.673e+01†** **(2.587e+01)**	5.287e+02 (4.731e+01)	1.928e+02 (2.677e+01)
11	**1.384e+01‡** **(6.037e+00)**	1.437e+01 (1.345e+00)	9.663e+01 (1.393e+00)	2.349e+01 (1.024e01)
12	**5.196e+04†** **(9.723e+03)**	9.148e+05 (7.285e+04)	9.825e+05 (1.281e+05)	1.064e+05 (3.112e+05)
13	**3.052e+00†** **(1.021e+00)**	1.265e+01 (1.626e+00)	5.973e+02 (1.385e+02)	6.979e+00 (2.562e+00)
14	**1.126e+01†** **(2.311e-01)**	1.135e+01 (3.156e-01)	1.453e+01 (1.121e-01)	1.217e+01 (1.452e+00)
15	**4.013e+02‡** **(6.724e+01)**	4.038e+02‡ (5.982e+01)	8.832e+02 (2.281e+01)	6.747e+02 (1.043e+02)

Table 3. Objective Function Values for the Circular Antenna Array Design Problem

IIWO	IWO	DE	PSO
-20.7013† **(1.312e-01)**	-16.4178 (4.293e+00)	-13.9306 (1.041e-01)	-5.4852 (3.543e-00)

6 Conclusions

In this paper we present a novel approach to improved global optimization by using a synergy of Invasive Weed Optimization and Temporal Difference *Q*-Learning to adaptively select memes (constriction factors) from the meme pool. To the best of our knowledge, such Machine Learning techniques have not been used previously to incorporate learning strategies in Evolutionary Algorithms. Experimental results conducted on a wide variety of benchmark functions as well as a real world optimization problem justifies our claim to the robustness and efficiency of the proposed approach.

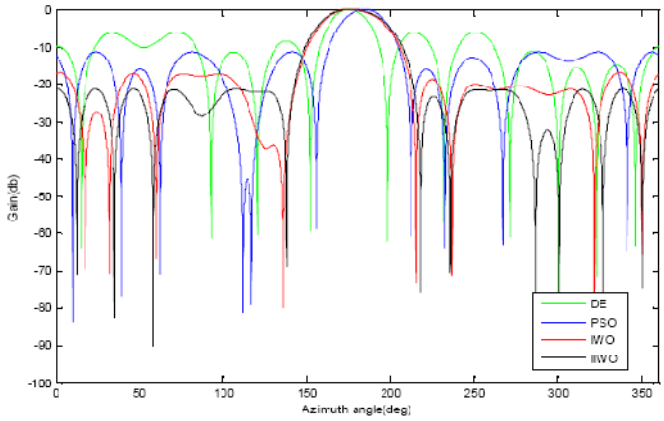


Figure 1. Power radiation pattern.

7 References

[1] Mehrabian, A. R. and Lucas, C. 2006. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics* 1 (2006), 355–366.

[2] Dawkins R. 1976. The Selfish Gene. *Oxford University Press* (1976).

[3] Moscato, P. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. In *Caltech Concurrent Computation Program* (report 826).

[4] Ong, Y.-S., Lim, M.H., Zhu, N. and Wong, K.-W. 2006. Classification of Adaptive Memetic Algorithms: A Comparative Study. In *IEEE Trans. on Systems, Man and Cybernetics* 36, 1 (Feb. 2006).

[5] Kendall, G., Cowling, P. and Soubeiga, E. 2002. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conferrence on simulated Evolution and Learning* (Singapore, Nov. 2002), 667-671.

[6] Watkins, C. 1989. Learning from delayed rewards. *PhD dissertation* (King's College, Cambridge, England, 1989).

[7] Watkins, C. and Dayan P. 1992. Q-learning. *Machine Learning,* 8, (1992), 279- 292.

[8] Kennedy, J. and Eberhart, R.C. 1995. Particle swarm optimization. In *Proceedings of IEEE International conference on Neural Networks* (1995), 1942-1948,

[9] Konar, A. and Das, S. 2006. Recent advances in evolutionary search and optimization algorithms. In *Proceedings of NGMS 2006* (BESU, Shibpur, Howrah, India, January 11-13, 2006).

[10] Storn, R. and Price, K. V. 1997. Differential Evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization* 11, 4 (1997), 341–359.

[11] Gurel, L. and Ergul, O. 2008. Design and simulation of circular arrays of trapezoidal-tooth logperiodic antennas via genetic optimization. *Progress In Electromagnetics Research* PIER 85 (2008), 243 - 260.

[12] P.N. Suganthan, N. Hansen , J.J. Liang, K. Deb, Y. P. Chen A. Auger and S. Tiwari, "Problem Definitions and Evalution Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," *Technical Report*, Nanyang Technological University, Singapore.

An Excel-based, Rotating Constellation Heuristic for Solving the Travelling Salesman Problem

Richard J. Perle
Department of Finance and Computer Information Systems,
Loyola Marymount University, Los Angeles, CA, USA

Abstract - *The Travelling Salesman Problem (TSP) is a well-known combinatorial problem that finds many applications in operational settings such as product distribution and manufacturing. Finding exact solutions to TSPs can be difficult, so heuristic methods are sometimes implemented. This paper develops and tests the performance of a new and novel, Excel-based heuristic algorithm for solving Euclidian plane, symmetric TSPs where the X-Y coordinates of all nodes are known. Test results show that the method works well for small problems. The method has an error function that varies linearly with the problem size. An advantage of the algorithm is that it is relatively easy to implement and so might be useful in smaller organizations which do not posses sophisticated mathematical or financial resources. It also might be used in an academic setting to demonstrate heuristic solution processes.*

Keywords: Traveling salesman problem, Heuristic, Algorithm

1 Introduction

The Travelling Salesman Problem (TSP) is a well-known and important example of combinatorial sequencing problems that find wide practical application in many fields. The classic case is where a salesperson wants to travel from his home base and visit clients in a number of other cities. The problem is to define a least-costly tour from city-to-city in which each city is visited only once and the salesperson ends up back at the home base. Interest in and application of the TSP began with the seminal paper by Dantzig, Fulkerson and Johnson [1] which found the shortest-route tour of 49 cities in the U.S. Other important, practical and useful examples of combinatorial sequencing problems with added refinements and extensions are; assigning airliners to routes [2], routing delivery trucks [3], drilling holes in printed circuit boards [4], order picking in a warehouse [5], and sequencing jobs on a machine [6].

A characteristic of any TSP is that it is easy to describe but difficult to solve to an exact, provable optimum. An exact solution to a TSP can be found by branch-and bound integer programming methods, described by Lawler, et. al. [7] and branch-and-cut methods, described by Junger, et al. [8]. An exact solution can also be found by explicit enumeration of all possible solutions ; however, this is not a viable methodology for larger problems as the number of possible solutions is a factorial of the number of nodes (cities). As a result of the difficulty in finding exact solutions, practical and useful solutions to TSPs can be obtained by the use of heuristic algorithms. A heuristic algorithm is a solution procedure that can lead to a good solution, but one that is not necessarily optimal. There are three general types of TSP heuristics; (1) construction methods, (2) improvement methods and (3) metaheuristic methods. Construction methods start at an arbitrary node and then select succeeding nodes according to a criterion such as cheapest or shortest distance. Well-known examples of construction methods are variations of the Nearest Neighbor Greedy (NNG) algorithm [9] which will be used for comparison purposes in this paper. Improvement methods start with a feasible tour and then make changes in a effort to find a shorter tour. The 2-Opt, 3-Opt and Lin-Kernighan algorithm [10] are examples of improvement methods. Metaheuristics such as simulated annealing, tabu search, genetic algorithms and artificial neural networks search neighborhoods for local optima and then use that information to search for better solutions without getting trapped in any one local neighborhood. A good description of basic metaheuristic methods can be found in [11]. A disadvantage of many of these methods is that they usually require specialized software that may be difficult or expensive to acquire and implement, especially for small companies that may find less than exact solutions to be an acceptable trade-off for a simpler solution methodology.

The Rotating Constellation Heuristic (RCH) algorithm described in this paper is a hybrid method. It starts with a feasible tour constructed by a simple node-to-node process, then systematically generates a subset of additional complete, feasible tours, ultimately selecting the best tour from the set of feasible tours. It has the advantage that the software may be developed and implemented by using the Excel sort function, and does not require extensive training in mathematics or expertise in a programming language. However, some basic skills in Excel Macros and VBA would be helpful to reduce the amount of time and effort required to find the best solution that the method is capable of delivering. The practical usefulness of any non-optimal heuristic solution, of course, would depend on its expected accuracy

which in this paper is measured as the expected percent over an optimal or benchmark tour.

The next two sections of this paper will present the RCH process logic by example, and then the mathematics of the general RCH model will be developed. Its robustness and accuracy will then be benchmarked against actual TSP data for which a very good or optimum solution is known.

2 The Rotating Constellation Heuristic: Description by Example

As a simple example consider a set of ordered pair X-Y values for a ten node TSP, shown in Table 1, where the pairs are sorted in ascending X-value order.

Table 1
X-Y values for a 10 node TSP.

Node	X-value	Y-value
1	7	53
2	15	81
3	20	26
4	26	67
5	32	39
6	54	57
7	61	80
8	68	37
9	87	49
10	93	72

The RCH algorithm is an eight step procedure, the first six of which are listed below. Steps 2, 4 and 5 are implemented by an Excel sort command. The algorithm is applied iteratively using Excel VBA in a search for a best solution tour distance value.

The RCH algorithm:
1. Assign a unique number to each node
2. Sort the complete set of nodes on the X-values, from low to high.
3. Based on the X-values, separate the nodes into two equal size sets.
   Left-most set = nodes with the smallest X-values.
   Right-most set = node with the largest X-values.
4. Sort the left-most set of nodes on the Y-values, from low to high.
5. Sort the right-most set of nodes on the Y-values, from high to low.
6. Connect the two sets to identify the complete tour and calculate the tour distance value.

Fig. 1 shows a graphic example of the first six steps of the algorithm applied to the data in Table 1. The vertical dashed line separates the 10 nodes into two equal size sets of five nodes each, designated as the left-most set and the right-most set (steps 2 and 3). The arrows show the tour path that the first six steps of the first iteration of the RCH algorithm would find from start to finish. Step 4 sorts the left-most set of nodes on the Y-values from low to high; step 5 sorts the right-most set on the Y-values, from high to low. The tour path by node number is 3-5-1-4-2-7-10-6-9-8-3. Its length is 314.20.

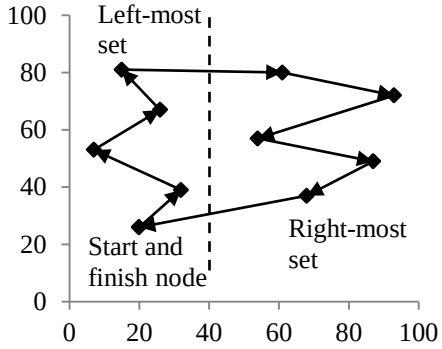


Fig. 1. Two-set Solution Path for Table 1 Data

It can also be inferred from Fig. 1 that an advantage of the RCH algorithm is that there will be no crossing-paths within either set. Crossing paths will add to the tour length, assuring that it is not optimal. The only time a crossing path could (but not necessarily) be generated is on the two paths that connect the sets together; the upper-most arrow and the lower-most arrow in Fig. 1.

The entire set of nodes is now treated as a constellation of ordered pair points that will be rotated about its geometric center in successive iterations of the RCH algorithm. The rotation does not change the distance between any pair of nodes. After each iterative rotation, steps 2 through 6 in the algorithm are repeated to generate a new solution. For example, the constellation of Fig. 1 is rotated 90 degrees clockwise and a new RCH solution is generated from the new, rotated X-Y coordinates shown in Table 2.

Table 2
Rotated X-Y values.

Node	New X-value	New Y-value
1	16.6	82.8
2	29.6	70.8
3	43.6	95.8
4	57.6	76.8
5	71.6	83.8
6	70.6	41.8
7	62.6	9.8
8	47.6	48.8
9	39.6	15.8
10	27.6	34.8

The new tour path by node number as shown in Fig. 2 is 9-10-2-1-3-5-4-8-6-7-9. The tour length is reduced to 262.83 which is a 31.69 percent improvement over the tour in Fig. 1. It is also optimal, proved by explicit enumeration.

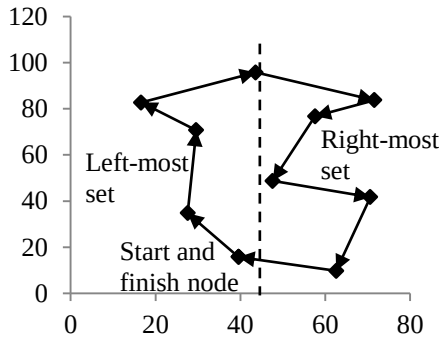


Fig. 2. Two-set Solution Path Rotated 90 Degrees, for Table 2 Data

Even though 90 degrees is the best rotation angle there is no way to know, a priori, that this is the case. So the objective in the RCH algorithm is to rotate the constellation incrementally, attempting to generate a new solution with each incremental rotation until a best, but not necessarily optimal solution is found. The condition that creates a new solution path occurs when a node passes from the right-most set to the left-most set and vice-versa due to rotation. In effect the two sets evolve by exchanging a pair of nodes at each incremental rotation. Consequently, the largest number of different solutions that the rotation process is capable of generating is n, the number of nodes. The set of n unique solutions, if they are all detected, is generated over a total rotation of only 180 degrees. Rotating the constellation beyond 180 and on to 360 degrees simply generates a repeat of the zero to 180 degree rotation solutions, except the tour path is in the opposite direction. But the tour length is obviously the same in either direction. There are a number of rules that could be used to determine the incremental angle of rotation, but the simplest rule, used here, is to rotate 180/n degrees in contiguous increments, starting from zero to (180 - 180/n) degrees. The zero degree position is the original configuration of the constellation. Two additional steps (7 and 8) are now added to complete the RCH algorithm.

7. Incrementally rotate the constellation 180/n degrees, then repeat steps 3 - 6 until the constellation has been rotated a total of (180 - 180/n) degrees.
8. Select the solution with the best tour length.

Another question to address is how to handle an instance where there is an odd number of nodes. Obviously the constellation cannot be divided into two sets, each with the same number of nodes. The easiest way to handle this is to arbitrarily assign one extra node to either set, which is done here. There are more sophisticated ways to solve the problem such as creating a "dummy" node as a duplicate of an existing node which will add nothing to any tour length because the sorting process should always connect the dummy node to its real node. However, this could perturb the geometric center of the constellation leading to unknown effects. The general RCH model will now be developed.

3 The Rotating Constellation Heuristic: General Model

Assume the Cartesian coordinates, x_iy_i (i = 1 to n) in a two-dimensional flat plane for a TSP with n nodes are known. The symmetric Euclidian distance, d_{ij} between any two nodes, i and j ($i \neq j$) is calculated as:

$$d_{ij} = SQRT[(x_i - x_j)^2 + (y_i + y_j)^2] \qquad (1)$$

The ordered pair, x_cy_c is defined as the geometric center of the constellation and is calculated as the mean of the X-values and the Y-values:

$$x_c = [\sum_{i=1}^{n} x_i]/n \qquad (2)$$

$$y_c = [\sum_{i=1}^{n} y_i]/n \qquad (3)$$

Consider now a translated coordinate system centered on x_cy_c in the original Euclidian plane which is divided into four quadrants (Q1 to Q4) as shown in Fig. 3. An arbitrary node, x_iy_i and its vector is shown in Quadrant 2 along with its angle, θ_i relative to the positive X-axis which is defined as zero degrees. The Euclidian distance from x_cy_c to x_iy_i is h_i and calculated as :

$$h_i = SQRT[(x_i - x_c)^2 + (y_i - y_c)^2] \qquad (4)$$

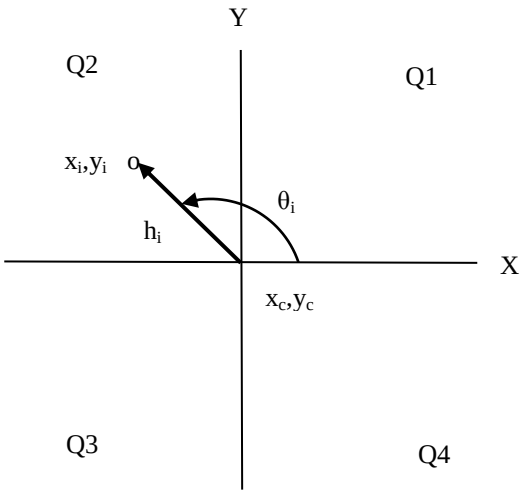


Fig. 3 Translated X-Y Coordinates

The calculation of any value of θ_i depends on which quadrant the node is in.

If: $x_i - x_c >= 0$ and $y_i - y_c >= 0$, the node is in Q1, and:

$$\theta_i = \sin^{-1}[(y_i - y_c)/h_i] \quad (5)$$

If: $x_i - x_c < 0$ and $y_i - y_c >= 0$, the node is in Q2, and:

$$\theta_i = 180 - \sin^{-1}[(y_i - y_c)/h_i] \quad (6)$$

If: $x_i - x_c < 0$ and $y_i - y_c < 0$, the node is in Q3, and:

$$\theta_i = 180 - \sin^{-1}[(y_i - y_c)/h_i] \quad (7)$$

If: $x_i - x_c >= 0$ and $y_i - y_c < 0$, the node is in Q4, and:

$$\theta_i = 360 + \sin^{-1}[(y_i - y_c)/h_i] \quad (8)$$

Once the set of θ_i have been calculated the entire constellation is incrementally rotated about the geometric center by angle, φ_k (k = 0 to n-1), where k is the kth iteration out of n total iterations.

$$\varphi_k = (180/n)k \quad (9)$$

If k = 0, then $\varphi_k = 0$, and the constellation is in its original, unrotated position. After each rotation, each node's new vector angle relative to the zero degree position in Fig. 3 is $(\theta_i - \varphi_k)$, assuming clockwise rotation. New coordinates, $x_i y_i$ in the original Euclidian plane can then be calculated for each rotated node:

$$x_i = h_i[\cos(\theta_i - \varphi_k)]x_c \quad (10)$$

$$y_i = h_i[\sin(\theta_i - \varphi_k)]y_c \quad (11)$$

Of course the distance, d_{ij} between any two rotated nodes is the same as the original, unrotated distances:

$$d_{ij} = d_{ij} \quad (12)$$

And the tour distance value, v_k for the kth contiguous rotational iteration is:

$$v_k = \sum d_{ijk} \quad (13)$$

where i and j are determined by the final sort sequence which becomes known after step 7 in the RCH algorithm. The best overall solution, v^* is then selected in step 8 of the RCH as:

$$v^* = \min\{v_k\} \quad (14)$$

which occurs at a rotation angle of φ^*.

4 Computational Test Results and Discussion

In this section results are presented on the performance of the RCH algorithm compared to the performance of the NNG algorithm using instances available in TSPLIB [12]. Instances of up to 76 nodes were selected from TSPLIB for which the X-Y coordinates and the optimal or best tour were known and available. All tour values are assumed to be correct as specified in TSPLIB. NNG and RCH solutions were generated for each instance and compared to the TSPLIB tour. Overall results are presented in Table 3.

It can be seen in Table 3 that the RCH algorithm at φ^* performed better than the NNG algorithm for instances of 38 nodes, or less. For these four instances, rotation of the constellation from $\varphi = 0$ to $\varphi = \varphi^*$ improved the solution tour from an average of 35.87 percent over TSPLIB to 14.96 percent over TSPLIB. As might be expected, it can be seen in Table 3 that the error increases with the number of nodes. The data in the second and last columns in Table 3 can be used to estimate the expected accuracy of the RCH algorithm, relative to TSPLIB, as a function of the number of nodes which have been divided into two sets, each of size n/2. Defining:

p = % over TSPLIB, RCH at $\varphi = \varphi^*$
n = number of nodes

Simple linear regression generates the following equation with adjusted $R^2 = 0.89$.

$$p = 0.93(n) - 10.56 \quad (15)$$

Obviously, the p value in (15) cannot be less than zero. The p-value is equal to zero when the n-value is 11.35. Given that the n-value must be integer, the error predictor equation is modified accordingly and approximated as:

$$p = \begin{cases} 0 & \text{for } n <= 11 \\ 0.93(n) - 10.56 & \text{for } n >= 12 \end{cases} \quad (16)$$

Table 3.
Selected instances from TSPLIB with known X-Y coordinates and optimal solutions

TSPLIB Instance	Number of nodes	% over TSPLIB, NNG	% over TSPLIB, RCH at $\varphi = 0$	φ^*, degrees	% over TSPLIB, RCH at $\varphi = \varphi^*$
ulysses16	16	52.70	9.84	0.00	9.84
ulysses22	22	27.82	12.84	163.64	10.69
wi29	29	31.83	53.69	43.45	5.15
dj38	38	46.47	67.12	123.16	34.16
att48	48	20.89	118.55	127.50	28.01
berlin52	52	19.08	86.02	72.69	38.26
st70	70	19.34	80.11	146.57	58.66
pr76	76	41.89	122.08	61.58	60.14

5 Conclusions

A new heuristic algorithm for solving the TSP has been developed and tested. The algorithm can be implemented with only a basic knowledge of trigonometry, Excel and Excel macros or VBA programming. The algorithm performs well for small problems with the error increasing linearly with the problem size. This decrease in accuracy is caused by the sort solution methodology which naturally searches for the outside perimeter defined on the graph of the nodes for all values of φ which can lead to excessive back and forth, zigzag travel in larger problems. Accuracy could likely be increased by dividing the complete set of nodes into more than two sets, allowing the sort procedure to delve more deeply into the interior region of the node graph with less overall travel within each set. It is expected that the error predictor equation (16) would hold within each pair of sets, even as the overall number of nodes in the instance increases. Connecting the pairs of sets into a complete tour would likely contribute additional error. This is a topic for future research. Another avenue for future research could address more thoroughly the question of how to handle an odd number of nodes, especially when more than two sets are defined. Additional rules for determining the incremental rotation angles could also be investigated. The general solution methodology might also be modified to work for three-dimensional or spherical coordinate problems as long as the node coordinates are specified.

6 References

[1] Dantzig, G. B., D. R. Fulkerson and S.M. Johnson, Solution of a Large-scale Traveling Salesman Problem. Journal of Operations Research 1954;2(4): 393-410.

[2] Clarke, Lloyd, Ellis Johnson, George Nemhauser and Zhongxi Zhu, The Aircraft Rotation Problem. Annals of Operations Research 1997; 69(0):33-46.

[3] Laporte, Gilbert, The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. European Journal of Operational Research 1992;59(3): 345-58.

[4] Grotschel, M., M. Junger and G. Reinelt, Optimal Control of Plotting and Drilling Machines: A Case Study. Mathematical Methods of Operations Research 1991;35(1): 61-84.

[5] Ratlif, H. Donald. and Arnon S. Rosenthal, Order-Picking in a Rectangular Warehouse; A Solvable case for the Traveling Salesman Problem. Operations Research 1983;31(3):507-21.

[6] Lenstra, JK, AHGR Kan and P Brucker, Complexity of Machine Scheduling Problems. Annals of Discrete mathematics. North Holland, Amsterdam, 1977:343-62.

[7] Lawler, E. L., J. K. Lenstra, A. H. J. Rinnooy Kan, and B. B. Shmoys (eds.). The Travelling Salesman: a Guided Tour of Combinatorial Optimization. J. Wiley and Sons, Chichester, England: 1985.

[8] Junger, M. G., G Reinelt, and G. Rinaldi, The Travelling Salesman Problem, in: Handbooks in Operations Research and management Science, edited by M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, North Holland, 1995, pp. 225-330.

[9] Rosenkrantz. D. J., R. E. Stearns and P. M. Lewis, An Analysis of several Heuristics for the Travelling salesman Problem. SIAM Journal of Computing 1977;6:563-581.

[10] Lin, S. and B. W. Kernighan, An Effective Heuristic Algorithm for the Travelling Salesman Problem. Operations Research 1973;21:498-516.

[11] Reeves, Collin, R. (ed.), Modern Heuristic Techniques for Combinatorial Problems, Halstad Press: an Imprint of J. Wiley and Sons, New York, 1993.

[12] TSPLIB. URL: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>Last access; December 7, 2011.

SESSION

APPLICATIONS

Chair(s)

TBA

Intelligent Players Competing in the Game of Qubic

James G. Schiiller, Greg L. Vitko, and Gautam B. Singh
Department of Computer Science, Oakland University, Rochester, MI, United States

Abstract - *The objective of this research is to develop two intelligent players to competitively play against each other in the game of Qubic. The two-player computerized board game environment possesses an excellent setting in which to conduct programming experiments. One player selected to compete in the contest is a deterministic, heuristic-based approach to artificial intelligence that plays well. The other player selected is a non-deterministic, genetic programming-based approach to machine learning. The immediate results clearly show the heuristic player as the dominant one. Two-pass evolution is applied to strengthen the genetic programmer well enough to win more than a few matches against the heuristic opponent. The analysis opened up areas of research into developing stronger intelligent heuristic and genetic programming players.*

Keywords: Artificial Intelligence in Games, Genetic Programming, Heuristics, Qubic

1 Introduction

The two-player computerized board game environment possesses an excellent setting in which to conduct programming experiments. The ultimate goal is to establish one winner for each game played. The outcome of the games, in their entirety, holds the key-point in which to compare intelligent players. Qubic, an advanced version of the traditional tic-tac-toe game, is the board game chosen for this study. The research trials were conducted with a two-player stance, competing over a 100 game set of Qubic. One of the competing players is the Open-123 heuristic; the other competing player is referred to as genetic programming (GP). The object of this research paper is to: develop two intelligent Qubic players; compete the players against each other; discover the final results of each game played; examine the moves; interpret the results of the analysis and state the final verdict of the project. The conclusion of the experiment will establish that the player with the most wins at the end of the tournament is the ultimate winner of the competition.

Keep in mind, there are more properties of the competition to consider than just the final outcome. These properties should not be dismissed as inconsequential. Although the outcome is decisive, many other factors are equally as important, since they may hold considerable bearing on the eventual outcome. For instance, the winner could move at an extremely slow pace while the loser moves more swiftly. Additionally, the loser may have an interesting playing style or possesses the potential to play at a higher-level, and so on. The ideas discovered while playing Qubic can be mapped to less trivial and more serious problems, e.g.) the medical field. Games could be the domain where "it" all starts).

2 Qubic

2.1 The Game of Qubic

Qubic is a board game originally sold by Parker Brothers, in the late 1960's. Qubic consists of four 4x4 tic-tac-toe boards. Qubic is easier to understand, visualize, and talk about in a two-dimensional representation as seen in Figure 1. Additionally, the boards can be combined together to form a three-dimensional cube as seen in the same Figure 1. [1]

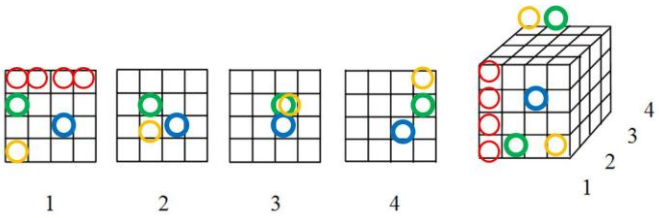


Figure 1: Two- and three-dimensional Qubic in layers.

Each board layer labeled 1, 2, 3, 4 are the same as the corresponding two-dimensional board representation. Sixteen squares on each of the four boards total the entire 64 square playing surface. The object of the game is to line four "X's" or "O's" in a row, along, for instance, the horizontal, vertical, or diagonal axes, on one square of each board, beginning with an outer board. X is the first player to move. Players then alternate moves. The first player to line four of their squares up in a row wins.

2.2 Qubic Mathematics

There are 76 winning lines in Qubic. Further details are contained within [2]-[4]. See Table 1 to view all of the 76 possible winning move sets. Qubic has an upper bound complexity level of 3^{64}. Each square can be filled by either X, O, or empty [5]. There are 64 total squares. That is, multiply three times three, 64 times. The result is an astronomical number. There are this many combinations (total number of

Table 1: The 76 winning line sets in Qubic

Vertical	Horizontal	Straight up	Vertical / Horizontal	Diagonal
{0,1,2,3}	{0,5,10,15}	{0,20,40,60}	{0,21,42,63}	{0,6,12,18}
{5,6,7,8}	{1,6,11,16}	{1,21,41,61}	{5,26,47,68}	{3,7,11,15}
{10,11,12,13}	{2,7,12,17}	{2,22,42,62}	{10,31,52,73}	{20,26,32,38}
{15,16,17,18}	{3,8,13,18}	{3,23,43,63}	{15,36,57,78}	{23,27,31,35}
{20,21,22,23}	{20,25,30,35}	{5,25,45,65}	{0,25,50,75}	{40,46,52,58}
{25,26,27,28}	{21,26,31,36}	{6,26,46,66}	{1,26,51,76}	{43,47,51,55}
{30,31,32,33}	{22,27,32,37}	{7,27,47,67}	{2,27,52,77}	{60,66,72,78}
{35,36,37,38}	{23,28,33,38}	{8,28,48,68}	{3,28,53,78}	{63,67,71,75}
{40,41,42,43}	{40,45,50,55}	{10,30,50,70}	{3,22,41,60}	{0,26,52,78}
{45,46,47,48}	{41,46,51,56}	{11,31,51,71}	{8,27,46,65}	{15,31,47,63}
{50,51,52,53}	{42,47,52,57}	{12,32,52,72}	{13,32,51,70}	{3,27,51,75}
{55,56,57,58}	{43,48,53,58}	{13,33,53,73}	{18,37,56,75}	{18,32,46,60}
{60,61,62,63}	{60,65,70,75}	{15,35,55,75}	{15,30,45,60}	
{65,66,67,68}	{61,66,71,76}	{16,36,56,76}	{16,31,46,61}	
{70,71,72,73}	{62,67,72,77}	{17,37,57,77}	{17,32,47,62}	
{75,76,77,78}	{63,68,73,78}	{18,38,58,78}	{18,33,48,63}	

positions) in Qubic. If you wanted to get an even more accurate picture of the complexity of Qubic, use 64 factorial, an even larger number, as the upper bound complexity level estimate of Qubic [6]. This estimate includes the 364 estimate plus it counts move order, a key game property to consider [6]. Counting move order is the difference between the two estimates. Either way, it means the Qubic search space is extremely large. This qualifies it as a computationally intractable game (NP-hard problem, concerning the time required to solve the game). We are forced to estimate solutions to the problem.

3 Intelligent Players

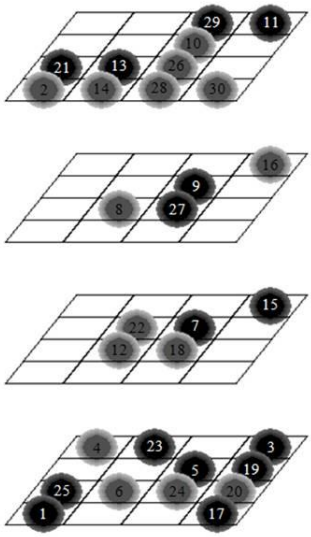
Processes found within the gaming communities are integral to the development of artificial intelligence (AI). Intelligent players are AI in the form of computer programs developed to play board games. Intelligent players have the notable ability to quickly solve complex problems. At the same time, intelligent players have trouble solving simple, very basic (common sense) types of problems. Two AI players can complete nearly 100,000 games of Qubic in a matter of hours. The same amount of games would take two human players, 100,000 consecutive minutes (or two months) of play to complete at the alarming rate of one minute per game.

3.1 Heuristics

Heuristics are deterministic, problem-dependent, and rule-based solutions to problems. Human programmers consult with experts in the field to take human experiences and advice, trial and error, educated guesses, or even brute force tactics, and transform the knowledge into computer procedures and programs. Improvement to the heuristic requires human programmer intervention. The problem with heuristics is the knowledge is considered shallow and the expertise is limited to the domain the system knows about. Heuristics do not generalize their knowledge. However, heuristics can provide valuable shortcuts that can reduce both time and cost. [7]

The heuristic player is one of the players to compete in the Qubic contest. The power of the heuristic player is contained within the evaluation function. The logic behind the evaluation function is called Open-123. The heuristic player is deterministic; it will make the same exact move each time it is given the same situation. The logic defining the evaluation function is simple. Higher values are assigned to moves containing a higher number of friendly squares, meaning squares filled by the player it is evaluating. Each line in each square is summed to give one total value to the square. Open 3 is assigned a higher value than Open 1. If there is a square filled by an enemy in the line being evaluated, the value is automatically returned as 0.0. The square containing the highest value is returned from the evaluation function. This is the square it will move to on the actual board. The Open-123 evaluation function can be used in opposites as the player's defense logic to block. Before evaluating itself, it can evaluate the opponent in order to detect an immediate win threat. Then, according to the killer heuristic philosophy, move to this very spot first before the opponent. During development, the heuristic player was benchmarked against the skill level of human players. It was found to be extremely difficult for humans to calculate what square Open-123 will play on next. The player was good enough at the moment in time when the

author of this work and other subjects were unable to defeat it. One person who was able to beat the player was Greg Vitko (Chess master). Vitko notably achieved the win in 30 moves on the first attempt, while playing second at a disadvantage. The other human players did not do as well. The final verdict was the Open-123 player plays well. There were some weaknesses exposed. All around, it evaluates well enough to serve its purpose. The move string and graphic game of the winning sequence can be seen in Figure 2. The graphical example game on the tilted boards below are the graphics developed by David Cao and reused to clearly demonstrate example games of Qubic.



0,60,18,03,12,6,32,46,52,72,78,26,66,65,38,58,15,31,17,16,61,27,8,11,1,71,51,70,73,75

Figure 2: Heuristic player defeated.

3.2 Genetic Programming

Genetic programming is a new, resurrected, exciting, and actively being researched area. It has been around since the 1960's and taken on and off of the shelf during the 1980's. It has been shelved in the past because of the limitations on computing power and heavy requirements placed on execution. Any modern personal computer satisfies the power requirements. Genetic programming is being revisited today.

Genetic programming is a machine learning technique inspired by the theory of biological evolution. Genetic programmers are algorithms that create algorithms. In tree-based GP, the generated computer programs are similar to decision trees and more precisely parse trees: decisions travel up the tree. Each node can be a complex mathematical formula (function) [8]. The general idea is to first try a few thousand random guesses to see which ones are best; then, take the random guesses and improve them. The basic work flow follows. First, generate the initial population of random programs built from the primitive function set. Then, rank the programs by evaluating them in some predefined way. The best programs are then selected. These are mutated and bred with crossover (recombination). The process is continued until the termination condition is satisfied.

GP programs in the Qubic implementation are stored as tree structures. The trees are initialized with a primitive set of five functions shown in the following list.

1) add(l):=return $l_0 + l_1$
2) multiply(l):=return $l_0 * l_1$
3) subtract(l):=return $l_0 - l_1$
4) if(l):=if $l_0 > 0$:return l_1 else:return l_2
5) is_greater(l):=if $l_0 > l_1$:return 1 else:return 0

Each function contains a set of simple operators. The GP recursively recombines and forms these functions into programs. The nodes on the trees are function, parameter, or constant nodes. For each function node there is a list l of other function, parameter, and constant nodes. The functions are represented in the list; the parameters are simple auto-generated lookup values determined at runtime; and the constants are hard coded integer values.

Trees are programs. They calculate their next move by propagating return values up the tree from the bottom nodes, towards the root of the tree, formulating to the output number in which the GP uses to make its next move (see Figure 3 for an example of a genetic program). Programs are designed by means of evolutionary pressure. The best player remains or is replaced, dependent on how well it plays against other competitors in its run. To be useful, the GP should be serialized to disk. The player can then be brought back to life and compete in more rounds of evolution and serialized back to disk again.

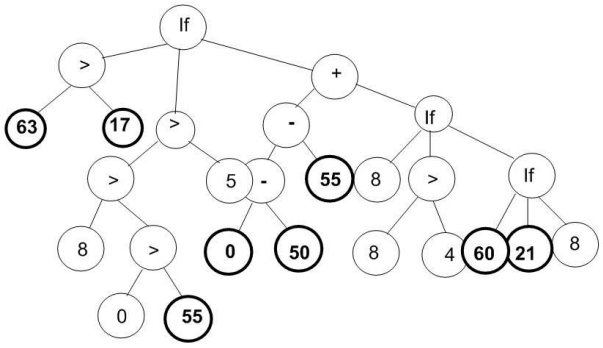


Figure 3: Example tree-based program.

4 Experiment

4.1 Hypothesis

The purpose of this experiment is to compete two AI players against each other in a 100 game contest of Qubic, to see who wins and to see what thoughts emerge. Solving the

game of Qubic is not the purpose. Qubic has already been weakly solved. Qubic is not being used for that purpose, but it supplies the domain which conducted the comparison results. It is the hope of this study to develop the GP player to win at least a few games against the heuristic player. If this experiment reveals the GP loses, it means there are open areas for future work.

4.2 Experiment Results

The heuristic player won every single game! The length of the games is an indicator of the equality of the players. This is true when humans compete against other human players in games such as chess. It is logical to deduce the same for AI players for Qubic. An average game of Qubic with roughly equal players lasts 20 moves. In this contest, an average game lasted only 10 moves. The length is not long enough to warrant equality of the players.

5 Opened Areas of Research

5.1 First Attempt Failed

The results of the experiment opened up areas of research into developing stronger intelligent GP players. The basic problem is the GP player lost all 100 games. This opened up an opportunity to improve the evolutionary process of the genetic programmer. There were three main problems with the earlier approach:

1) GP was not blocking immediate win threats (basic need).
2) The is_greater primitive function was returning 1/0.
3) The evolution process only involved self-competition.

5.2 Second Attempt Plan

In the first contest, the outcome of running the 100 game set of Qubic resulted in the heuristic player winning every single game. This means one pass of 100,000 evolutions of the GP player results in a poor playing program. Increasing the game experience would not improve the player. The genetic programmer is not executing basic needs by failing to block simple initial 'X' player moves 0, 18, 12 as seen in Figure 4. Computers are not skilled at handling common sense type of basic problems. The first problem to solve on the way to beating the heuristic player is to endow the GP player with basic logic to block. The GP needs to defend against immediate win threats. The second problem to address is to modify the is_greater primitive function to return moves, ranging from 0 - 78, including a padding of bits separating each 4x4 board, instead of returning binary numbers. Additionally, to address the third problem, instead of solely competing against itself, let it compete against a predator opponent.

.	.	.	X
.	.	X	.
.	.	.	.
X	.	.	.

3	8	13	18
2	7	12	17
1	6	11	16
0	5	10	15

Figure 4: Genetic programmer not blocking.

5.3 Solutions

The solution to the first main problem, to block, is apparent, and easy to correct. Use the same Open-123 logic as the heuristic player for defensive moves. The solution to the second problem requires more thought and includes the blocking solution for the first problem. After much experimentation, it was found that the set of initial programs to start the GP out with should be geared more towards Qubic. One specialized function was made to reference an in-memory lookup table into a small subset of winning lines. This idea fits into the way humans think. During a game of Qubic, human players follow six or seven solution lines they know will win. If one of the lines is blocked, they try another line. People also combine solutions to collect threats. The new function content pulls solutions from the 76 winning line sets of Qubic. Particularly, the new GP is made to pull from the fourteen handpicked solutions in the memory list of Table 2.

Table 2: Lookup Table of Solution Sets.

First Set of Solutions	Second Set of Solutions
{18,32,46,60}	{0,26,52,78}
{3,27,51,75}	{15,31,47,63}
{5,26,47,68}	{6,26,46,66}
{22,27,32,37}	{7,27,47,67}
{18,33,48,63}	{17,32,47,62}
{20,21,22,23}	{25,26,27,28}
{65,66,67,68}	{70,71,72,73}

The modified version of the is_greater primitive function takes the first or second element from the input list *l*, depending on the greater of the values, and applies the modulus operator to regularize the number as a valid solution in the lookup table:

- is_greater(*l*): if $l_0 > l_1$: return l_0 mod 7 from S_1 else: return l_1 mod 7 from S_2

The first two elements in the first and second set of solutions *S*1, *S*2 were chosen for the reason these lines contain four rich points (seven winning lines running through them). Two of which are the richest points on the board (i.e., the centers of the middle boards). Players past the beginning skill level of Qubic know perfect players open their games by playing on rich points. The rest of the solutions in the list are randomly

chosen. Comparable results would be generated using different sets of solutions. This solves the second problem. Lastly, to offset the domination of the heuristic player, the two-step evolution process is applied to solve the third main problem.

5.4 Two-Pass Evolution

Two-pass Evolution models the behavior of bucks (male deer) in the wild and may answer the third problem. Bucks compete by colliding antlers with other bucks where the act of competition decides dominance, and the winning males mate [9]. In the artificial model, colliding antlers is naturally equivalent to competing programs to decide dominance, and this is accomplished with first-pass evolution. The dominant program is now ready and allowed to mate. In the artificial world, mating is equivalent to throwing the best player, along with the heuristic player (predator), into the initial set of programs within second-pass evolution. In this second pass of evolution, the best GP player and predator mate. Hopefully, the mating produces better offspring through genetic evolution by applying genetic operators, recombination and mutation, to the best GP player with good players from the previous generation. By including the heuristic player as one of the players in the initial population of programs, the GP will eventually learn to block from the bad experience of losing to the predator, if reinforced with a win. This approach causes the evolving player to learn from others outside of its own class. Once the GP has evolved with the predator in the population, it is ready to compete in the real world. The aim is to evolve the GP into a stronger player capable to win more than a couple of games out of another 100 game contest of Qubic.

6 Next Experiment

6.1 Experiment Results: Second Attempt

The GP won 5 out of another 100 game contest of Qubic. Earlier, it was found, the GP was unable to solve the problem on its own. Now that the human programmer provided the right basic functions, modification of one of the primitive functions, and the addition of the equal blocking logic, it worked. The GP player defeated the heuristic player. The GP was able to find the right combination to pull the moves in by discovering the best way to arrange and the best order to call the solution sets. The most evolved player to date including second-pass evolution and selective termination achieved 25 wins and 12 draws, slowly approaching equality with the heuristic player. As a result of adding defense and second-pass evolution, the players start to become equal and longer games emerge.

6.2 Corrections to the Plan

Once the individual games start becoming long, the GP cannot find a valid move from the solutions (answers) list, all moves have already been taken, and the GP ends up timing out, or rather over-evaluating, and must give away the game to the opponent. There are only a maximum of 28 moves to choose from the list. If the opponent takes some of the moves away then there are less. Adding more solution lists to the specialized primitive function is not the answer. The set of solutions needs to be small enough to make sequential moves from within the same lists. Creating seven solution lists gives "optimal" success. For this reason, the GP had to generate random moves to complete close end games. The random move generator starts generating random moves until the move is allowed. For instance, if GP was the X player, it was X's move, the game was already 60 ply deep, and the GP's list of solutions were exhausted on the 61st ply, then the random move generator would kick in and return random moves for the GP to successfully complete the game.

7 Future Work

The evaluation function, as well as it plays, has some flaws, and can be improved upon. First, the heuristic player is being overcautious, playing unnecessary defensive moves, before spotting a winning offense. Small corrections in the evaluation function will change the function's behavior. Greater improvements can be made in the area of inserting further expert knowledge into the evaluation function. For instance, it is safe to say that making three blindfolded corner moves on rich points at the beginning of the game is a strong play. Another possible improvement could be to implement the concept of collecting threats, where the number of threats is more important than the threat itself [6]. Significant improvements can be made to the GP. For instance, the list of solutions referenced by the primitive function can be rearranged. Moreover, an improved method could be implemented for the times when the GP runs out of solutions in the end game. After these changes are made, the challenge would of course be to further develop the Qubic GP to dominate the heuristic player. After this, the next challenger to try and beat would be Lutz Tautenhahn's Qubic AI at http://www.lutanho.net/stroke/play.html. It is beatable on expert level, although, it plays extremely well and plays stronger than the Open-123 heuristic player.

8 Final Conclusions

The original hypothesis of developing a strong GP player for competing in the Qubic competition failed in some respects. One major problem was discovered in the course of the experiment. The GP player lost every game. The corrections to the GP player resulted in more than a few defeats of the heuristic player. An intelligent Qubic player was developed to beat the heuristic in 25 out of 100 games of Qubic. This was achieved without adding considerable expert knowledge to the GP. The GP plays at a higher skill level than its author; however, the end result of the contest, in terms of the number of wins, finds the heuristic player as the better Qubic player. It was found, the human programmer needs to

supply basic needs, the right initial set of functions, and the right environment for the GP to evolve.

9 References

[1] B. Challenor. "Qubic2: 4x4x4 tictactoe, ti-83/84 plus assembly games (mirageos)"; [Online]. Available: http://www.ticalc.org/pub/83plus/asm/games/mirageos/ April 2012.

[2] J. Beck. "Combinatorial Games: Tic-Tac-Toe Theory". Cambridge University Press, 2008.

[3] N. Do, "Mathellaneous: How to win at tic-tac-toe"; Gazette of the Australian Mathematical Society, 32, 3, 159, July 2005.

[4] O. Patashnick, "Qubic: 4 x 4 x 4 tic-tac-toe"; Mathematics Magazine, 33, 4, 151—161, Sept 1980.

[5] L. V. Allis, "Searching for Solutions in Games and Artificial Intelligence," Phd dissertation, University of Limburg, 1994.

[6] G. Vitko, Oakland University Virtual Reality Lab, Rochester, MI, private communication, Feb 2009.

[7] J. Giarratano and G. Riley. "EXPERT SYSTEMS Principles and Programming". PWS Publishing Company, 1998, 8.

[8] T. Segeran. "Programming Collective Intelligence. 1st ed.". O'Reilly Media, Inc., 2007, 250–273.

[9] W. Embar, "Animal facts – deer"; [Online]. Available: http://www.veganpeace.com/animal_facts/Deer.htm July 2005.

Optimization of SMC parameters Using GA in a Full-Bridge Dc-Dc Converter

Amirhasan Shams Ansari
Department of Electrical Engineering, Tafresh University, Tafresh, Iran (email: ahsha123@gmail.com)
Farzad Razavi
Department of Electrical Engineering, University of Tafresh, Tafresh, Iran, P.O.BOX39518-79611 (Corresponding author phone: +98912177 3241; email: farzad.razavi@tafreshu.ac.ir)
Ahsan Ghoncheh
Department of Electrical Engineering, California State University, Fullerton, (email: a.ghoncheh@csu.fullerton.edu)
Hesamoddin Abdollahi
Department of Electrical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran (e-mail: hessam.abdollahi@gmail.com)
Ali Rahimi Mehrnia
Department of Electrical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran (e-mail: mehrnia.ali@gmail.com)

Abstract— Converters are one of the inseparable parts of Distributed Generation systems. They are variety of control methods implemented in converters in order to control their output quantities. One the robust methods which is considerably useful in such applications is Sliding Mode Control which is robust against inherent uncertainties and imprecisions of converters. The proposed method is explained and formulated using state space average model of a sample full-bridge converter in order to have a perfect current tracking of a full-bridge converter. Since the best output is obtained using the optimum design parameters, the parameters of the full-bridge converter are optimized using Genetic Algorithm (GA) in order to have the best response.

Keywords- Sliding Mode control, Genetic Algorithm, Full Bridge Converter, Current Tracking, Parameter Optimization

I. INTRODUCTION

Clean Power Generation (CPG) is at the center of the attention these days because it can protect environment from the problems of classical power generation such as greenhouse gas emission and pollution. Besides environmental repercussions, energy demand rise and depletion of energy resources are other reasons of CPG appearance [1-3]. There are different types of environmental-friendly generations systems such as wind turbines, hydro turbines, photovoltaic arrays, biomass and fuel cells. Advances in energy storage devices have greatly assisted the flourishing penetration of distributed generation (DG) into installation and operation of power generation plants [4-6]

Most of the DG generators do not have a considerable and well-regulated DC output. To tackle this problem, a dc-dc power electronic converter is used in order to regulate and boost the output to the desired value [7]. Power converters can be assigned to perform some functions such as output regulation under load variation or source change [1]. Several types of converters have been studied for such purpose. Buck converter, Boost Converter, Buck-Boost Converter, Push-Pull boost convert and full bridge converter are the most common converters which are used in DG applications [8-14]. To have the best performance of a converter, several control methods have been used [15-21]. The reason of this implementation is the generation systems limitations rooted in their physical characteristics. In order to have an acceptable control over the output of the DG system (DG) there are two ways exists. First, Control the rate of the physical and chemical quantities which take part in reactions to produce electricity. For example, output voltage of a fuel-cell is made of the reactions between hydrogen and oxygen. Thus if the rate of the Hydrogen and Oxygen is controlled, the output value is controllable [22]. Since controlling Chemical and physical quantities are not easy, a second way which is the control implementation in the connected converter is used. It is possible to regulate and change the output of a converter by changing its duty cycle [2].

PID controllers are the most common use controller in the case of converter control. They can improve system transient response during load step changes and source changes [23, 24] or using loop up tables instead of multiplier in order to minimize the energy consumption [25]. PID controllers are valid only around the nominal power point of the system and they cannot have an acceptable response in large scale disturbances and wide variety of load changes. Intelligent modern controllers such as fuzzy controller are alternatives controllers to obtain better performance of converters [8, 18] .

They are robust against such disturbances and uncertainties. In [26], PID method and fuzzy logic theorem are combined via a Linear Quadratic Regulator (LQR) to have a better performance. Some other combinations of fuzzy logic and PID are presented in [27, 28]. Besides Fuzzy and PID controllers, the most appropriate controller in order to use in converter control applications is nonlinear control methods. These controllers are the best options because of uncertainties in converters rooted in their elements and design. In addition to uncertainties, the behavior of controllers is nonlinear and the best option to use in their application is a nonlinear controller

to be valid in a large variety of values. This Basis is derived from the non-minimum phase nature of converters, variability in their structure and unpredictable nonlinear load changes [29, 30]. Sliding mode control is one of the well-fitted and useful methods converter studies especially in DG connected samples.[31] presents a simple unified approach to the design of fixed-frequency pulse width-modulation-based sliding-mode controllers for dc-dc converters operating in the continuous conduction mode. The problem of combining interleaved operation of several identical power converters with a hysteretic control is solved in [32] by inducing a sliding regime to all inductor currents in a ring configuration. In some references a combination of sliding mode with other methods is presented [11, 28, 33-36].The sliding mode fuzzy controller combines the advantages of both fuzzy controllers and sliding mode controllers. It also has advantages of its own that are well suited for digital control design and implementation [28]. In [37-39], investigations on Buck, Boost and Buck-Boost converters are carried out using SMCs in order to reach a desired voltage profile tracking during a load change, start-ups and transients. Full Bridge buck converter is studied in a part of the [40] in comparison with a cascaded connection boost converter for voltage, but not considering the transformer.

In this paper, a proposed sliding mode controller (SMC) is introduced and implemented in a full-bridge converter using its average model. The parameters of SMC are optimized using GA to have a proper response from two points of view; settling time and final value. The proposed method is confirmed using simulation in tracking performance and settling time.

II. FULL BRIDGE CONVERTER AND THE CONTROL SYSTEM

A. Full-Bridge Converter Performance and Average Modeling

Fig.1 depicts the general schematic of a full-bridge converter circuit. **Error! Reference source not found.**

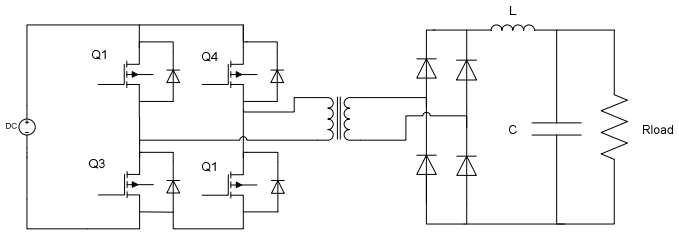


Fig.1. The full-bridge converter

The sample full bridge converter is made of a full bridge power converter ($Q_1 to Q_4$), a high turns-ratio transformer with ratio 1: n, a diode bridge rectifier and an output filter. The switches ($Q_1 to Q_4$) are located diagonally and turned on and off. The pulses are sent to the switches using a PWM pulse generator with the time duration of $D.T_S$. The PWM pulse generator has the value D as duty cycle input. The transformer eases boosting the output voltage.

The average model of a Full-Bridge DC-DC converter is derived in [1] and its state space model is shown in (1)-(5). In this model state variables are; inductor current (x_1) and capacitor voltage (x_2).

$$\dot{X} = AX + BV_d \tag{1}$$

$$V_O = CX \tag{2}$$

$$A = \begin{bmatrix} -\frac{2R_{th}d + r_D(1-2d)}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} \frac{2dn}{L} \\ 0 \end{bmatrix} \tag{4}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{5}$$

The average model is beneficial for simulation because of not using multiple switches and other electrical elements. Therefore, the time of simulation decreases. Also the state space model is suitable for controller design and stability analysis.

B. Implementation Of Sliding Mode Controller in a Full Bridge Converter

There are two stages exist in the design of a SMC controller; first, determining the sliding surface to provide sliding condition and second, reaching the sliding surface.

Consider the following single input dynamics.

$$x^{(n)} = f(x) + b(x)u(t) \tag{6}$$

$f(x)$ and $b(x)$ both are not exactly known but the former is upper bounded by a known continuous function of x named F where $\left|f - \hat{f}\right| \le F$ and the latter sign is known and it is also upper bounded by a continuous function of x. The objective is determination of u.

The tracking error vector is defined by the following equation;

$$\tilde{x} = x - x_d = \begin{bmatrix} \tilde{x} & \dot{\tilde{x}} & . & . & . & \tilde{x}^{(n-1)} \end{bmatrix} \tag{7}$$

$$s(x;t) = (\frac{d}{dt} + \lambda)^{n-1}\tilde{x} \tag{8}$$

Where $\lambda > 0$ and $n > 0$ are design parameter and system order respectively.

It is obvious that if the tracking error equals zero then (8) equals zero. It means:

$$\tilde{x} = x - x_d = 0 \Leftrightarrow s(x;t) = 0 \tag{9}$$

Where x_d is the desired output value.

Therefore, it is possible to keep s at zero with a proper choice of $u(t)$ at (1) .

A control law is introduced in order to keep the s at zero.

$$\frac{1}{2}\frac{d}{dt}s^2 \le -\eta|s| \tag{10}$$

where $\eta > 0$ is a design parameter.

State-space equations of the full-bridge converter can be obtained from (1) – (5) as

$$\dot{x}_1 = \frac{-R_{th}2d - r_d(1-2d)}{L}x_1 - \frac{1}{L}x_2 + \frac{2dn}{L}V_d \tag{11}$$

$$\dot{x}_2 = \frac{1}{c}x_1 - \frac{1}{RC}x_2 \quad (12)$$

$$V_O = x_2 \quad (13)$$

By changing (11) and (12) in the form of (6) we have:

$$\dot{x}_1 = \frac{-r_d}{L}x_1 - \frac{1}{L}x_2 + \left(\frac{2(r_d - R_{th})}{L}x_1 + \frac{2n}{L}V_d\right)d \quad (14)$$

$$\dot{x}_2 = \frac{1}{c}x_1 - \frac{1}{RC}x_2 \quad (15)$$

d is the controlling parameter of the system is the same as u in (6). Since there is no d available at (14) only the tracking of x_1 is accessible. Of course, it is highly likely that the tracking x_1 leads to the stability of x_2. Hence,

$$\hat{f}_1(x) = \frac{-R_d}{L}x_1 - \frac{1}{L}x_2 \quad (16)$$

$$\hat{b}_1(x) = \left(2\left(\frac{r_d - R_{th}}{L}\right)x_1 + \frac{2n}{L}V_d\right) \quad (17)$$

$\hat{f}$ is an estimation for f because of some available uncertainties in full-bridge converter. Another reason for this estimation is the semiconductor nonlinear characteristic. This explanation is also correct for b and $\hat{b}$.

Considering $n = 2$ in (8), the sliding surface is as follows

$$s = \dot{\tilde{x}} + \lambda\tilde{x} \quad (18)$$

The general form of (12) in comparison with (6) can be written as (20)

$$\dot{x}_1 = f(x) + b(x)u(t) \quad (19)$$

Where $f(x)$ and $b(x)$ can be the function of x_1 and x_2.

By replacing (20) in (19) we have

$$s = f + bu - \dot{x}_d + \lambda\tilde{x} \quad (20)$$

Now the effort is on the finding of the appropriate u in order to make $s = 0$.

by taking the estimations on f and b into account, the best u is found to satisfy $s = 0$

Firstly by considering $b = 1$, the following equation is derived

$$\hat{u} = -\hat{f} + \dot{x}_d - \lambda\tilde{x} \quad (21)$$

The above conditions in (21) and (22) are formulated with the assumption of $s = 0$, but there are some uncertainties in the parameters and the problem is out of the s plane. Therefore, u should be defined as

$$u = \hat{u} - k\,\mathrm{sgn}(s) \quad (22)$$

$$\text{Where } \mathrm{sgn}(s) = \begin{cases} 1 \ if \ s > 0 \\ -1 if \ s < 0 \\ 0 \ if \ s = 0 \end{cases}$$

Secondly, u is derived from (23) while $b \neq 1$

$$u = \hat{b}^{-1}\left(\hat{u} - k\,\mathrm{sgn}(s)\right) \quad (23)$$

By putting (24) in (20) and after some simplifications, the sliding surface could be achieved as

$$s = f - b\hat{b}^{-1}\hat{f} + b\hat{b}^{-1}\dot{x}_d - b\hat{b}^{-1}\lambda\tilde{x} - b\hat{b}^{-1}k\,\mathrm{sgn}(s) - \dot{x}_d + \lambda\tilde{x} \quad (24)$$

In order to calculate k, $s = 0$ must be solved, hence

$$b\hat{b}^{-1}k\,\mathrm{sgn}(s) = (f - b\hat{b}^{-1}\hat{f}) + (1 - \hat{b}b^{-1})(-\dot{x}_d + \lambda\tilde{x}) \quad (25)$$

Where $\hat{b} = (b_{\min}b_{\max})^{0.5}$

After some simplifications with regard to $f = \hat{f} + (f - \hat{f})$ k is represented as

$$k \geq \beta(F + \eta) + (\beta - 1)\left|\hat{u}\right| \quad (26)$$

Where

$$\beta = \left(\frac{b_{\max}}{b_{\min}}\right)^{0.5} \text{ and } 0 < b_{\min} \leq b \leq b_{\max}$$

After calculation of k, x is capable of tracking x_d and it can be controlled.

There are three design parameters. F is assumed as the radius of error between f and $\hat{f}$. Thus it is predicted that lower F results in better tracking accuracy and lowering the error. λ is another SMC parameter. λ is the coefficient of the non-derivative term of the sliding surface presented in (8). It affects the resolution of tracking the desired point in comparison with the reaching speed. Moreover it may influence the amount of the chattering. η is a positive constant design parameter and defines the sliding condition. It also implies that some of the system uncertainties and disturbances can be tolerated while still keeping the surface invariant set. Also, η can be influential on system dynamic especially in the system speed and changing in the proportion of chattering. General scheme of the proposed control system and its output and input is shown in Fig.2.

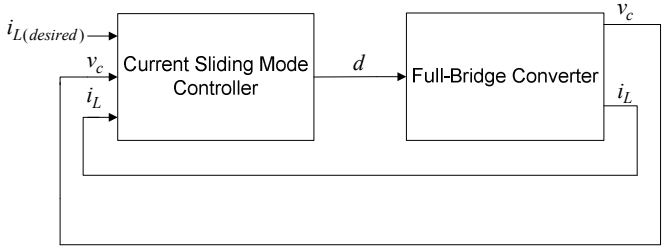


Fig.2. General Scheme of the Proposed SMC

III. Optimization of the Proposed SMC using Genetic Algorithm

In this paper, GA is used in order to optimize the three mentioned parameters of the proposed SMC to have the best output. In order to start the solution, first population size should be determined at the first step. This population is obtained from accidental quantization of chromosomes. These numbers are inserted to the function which is going to be optimized. The next step is injecting the generated population to objective function. The aim of this action is creating of fitness function which is derived from chromosomes. In this case proper answers are kept and others will be omitted. This circle will be continued until the size of population is reached.

the following chart illustrates the proposed GA.

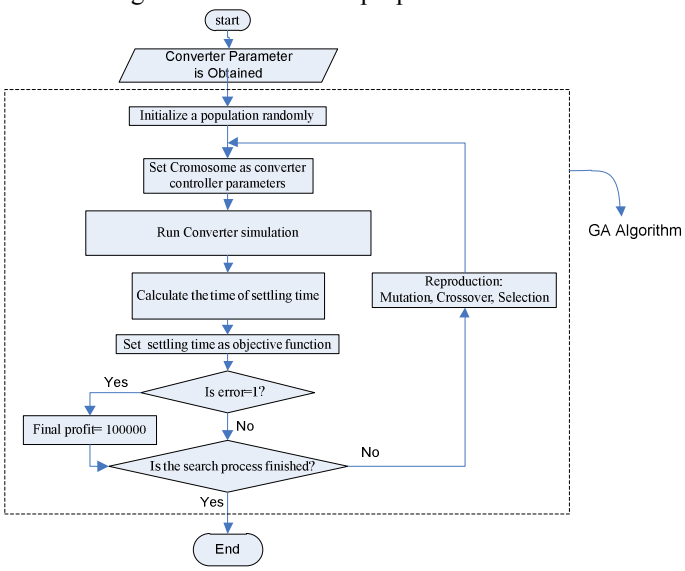


Fig.3. The proposed Genetic Algorithm chart

IV. Implementation of the Proposed Method

In order to verify the proposed method MATLAB/SIMULINK is used. The Converter model is modeled as follows:

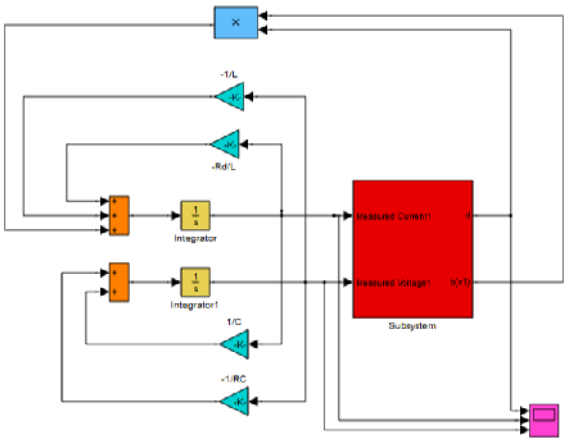


Fig.4. The Model of Dc-Dc Full-bridge Converter in MATLAB/SIMULINK

As it can be seen in above picture, because of some simulation problems such as increasing the time of the simulation and increasing the error message of the software during simulation with electrical parameters the full-bridge converter is simulated using block diagrams instead of resistors and capacitors.

As it can be understood from the simulated SMC, the inputs of the SMC are x_1 and x_2. The output of the system is the appropriate duty cycle which is sent to the converter switches.

For implementation of GA, MATLAB/GA toolbox is used.

The specification of the 5 kW sample Full-Bridge converter is presented in Table 1.

TABLE I
SPECIFICATION OF FULL-BRIDGE CONVERTER

Description	Parameter	Nominal Value
Input Voltage	V_d	50 V
Inductance	L	7 mH
Capacitance	C	330 µH
Diodes Resistance	R_d	1 mΩ
Switches On-Resistance	R_T	5 mΩ
Transformer Ratio	n	100
Switching Frequency	f_s	2000 Hz
Load Resistance	R_L	12.5 Ω
Equivalent Thevenin Resistance	R_{th}	100.002 Ω

V. Simulation Results and Discussion

In this part, the simulations and results using MATLAB/SIMULINK and MATLAB/GA toolbox is given and the appropriate explanations are also presented.

A. Design Parameter Study

First of all to have a better recognition of the design parameters their approximate effect on the tracking performance is discussed.

1) Effect of F

F may decrease the resolution of tracking. For example, $F=10$ expresses that the absolute deviation of the available values of the converter elements from original values are less than 10. The default values of other SMC parameters are as follows:

$$\begin{cases} \eta = 1 \\ \lambda = 10 \end{cases}$$

The optional value for x_1 to track is 22.5 A. The simulated waveforms for F=0.1, F=10, F=100, and F=1000 are shown in Fig.5.

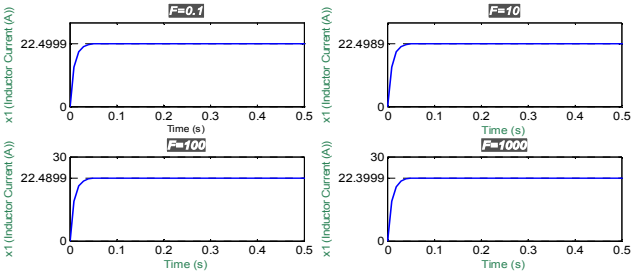


Fig.5. Simulated wave forms of Inductor Current for F=0.1, F=10, F=100, F=1000

2) Effect of "λ"

The simulation is performed for different values of λ using the following values for other parameters of the SMC.

$$\begin{cases} F = 0.1 \\ \eta = 1 \end{cases}$$

Also, in this study, the desired value of current is equal to 22.5 (A).

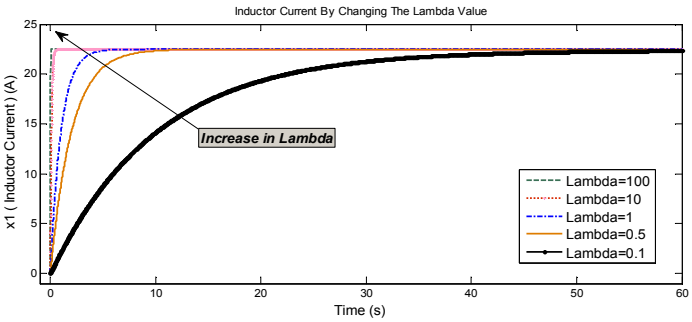


Fig.6. Simulated Waveforms of Inductor Current for $\lambda = 0.1, \lambda = 0.5,\ \lambda = 1, \lambda = 10, \lambda = 100$

As it can be understood from the above figure, by increasing λ, the slope of the current rises and the system reaches the steady-state condition in a shorter time.

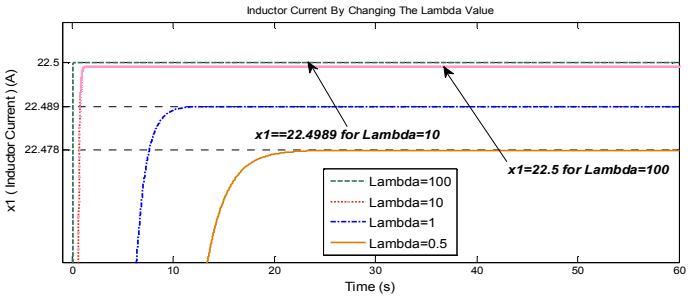


Fig.7. Tracking accuracy for $\lambda = 0.5,\ \lambda = 1, \lambda = 10, \lambda = 100$

Also it has got a significant effect on the tracking accuracy. For example the tracking error for F=10 is 0.0049%.

3) Effect of "η"

To study the effect of η, the default values for other parameters are

$$\begin{cases} F = 0.1 \\ \lambda = 10 \end{cases}$$

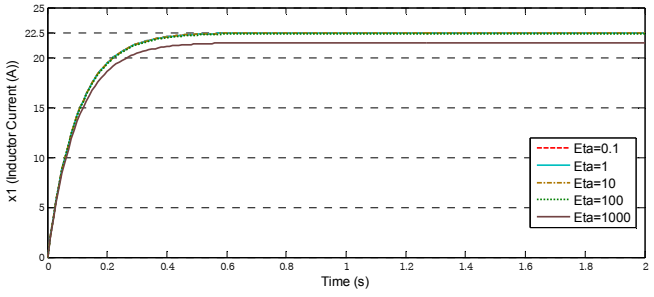


Fig.8. Simulated Waveforms for Inductor Current for $\eta = 0.1, \eta = 1, \eta = 10, \eta = 100, \eta = 1000$

The relative tracking error is around 4% for $\eta = 1000$, around 0.44% for $\eta = 100$ and approximately zero for other values.

B. Simulation of SMC Parameter Optimization Using GA

The estimated parameters to start simulation are:

$$\begin{cases} F = 0.1 \\ \eta = 1 \\ \lambda = 100 \end{cases}$$

And the result is presented in Fig.8. with a green dotted line.

Now the effort is to optimize the mentioned values in order to have the best response using GA. The simulation is run for the following values:

$$\begin{cases} Variable\ Names = F, \eta, \lambda \\ Population\ size = 10 \\ Generation = 50 \\ Absolute\ error = 0.001 \end{cases}$$

The optimized values for converter parameters are as follows:

$$\begin{cases} F = 0.051607 \\ \eta = 0.39232 \\ \lambda = 778.27 \end{cases}$$

And the specified point that the output has reached the best value is 14. It means that in 14 ms, the answer has got the best characteristics with the mentioned optimized values.

Now the effort is to run simulation for different values for absolute error to have a better understanding of the optimized points. The values for absolute error are 0.1, 0.01, 0.001 and 0.0001.

As it can be seen in Table II, the decrease in absolute error leads to the increase in reaching time.

TABLE II
GA RESULTS FOR DIFFERENT VALUES FOR ABSOLUTE ERROR

Absolute Error	0.1	0.01	0.001	0.0001
F	0.086175	0.038527	0.051607	0.37661
η	0.607	1.1747	0.39232	0.15359
λ	812.69	645.24	778.27	701.02
Reaching Time (ms)	8	13	14	19

The following figure depicts the inductor current simulated wave form for different values of absolute error.

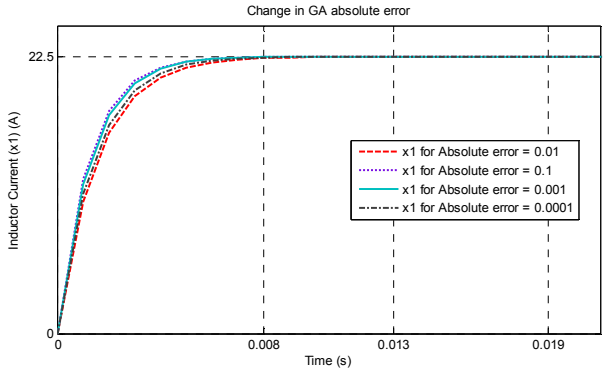


Fig.9. Simulated Waveforms for Inductor Current for different values for Absolute error

As it can be seen, the settling time is different for different values of absolute error. This method can be helpful during much sensitive cases like connection of a Dc-Dc converter to a distributed generator such as fuel cell and photovoltaic.

VI. CONCLUSION

Since most of the distributed generators does not have a considerable and regulated output, converters are implemented in DG systems in order to overcome the mentioned problems. Full-bridge converter is a useful converter which can perform both output regulation process and output boosting. SMC is a robust method which can be employed in order to force the variables of the converter to track a reference value. SMC has three design parameters. To obtain the best performance of the converter and controller this parameters should be optimized. In this paper GA is used in order to optimize the SMC parameters in order to have the best current output with the best tracking performance.

REFERENCES

[1] [1] H.R. Ali Asghar Ghadimi, Ali Keyhani, Journal of Iranian Association of Electrical and Electronics Engineers (IAEEE), 4 (2007) 52-59.

[2] [2] H.R. S.H Fathi, A.A Ghadimi, European Transactions on Electrical Power, 21 (2011) 801-823.

[3] [3] H. Kakigano, Y. Miura, T. Ise, Power Electronics, IEEE Transactions on, 25 (2010) 3066-3075.

[4] [4] M.N. Marwali, A. Keyhani, Power Electronics, IEEE Transactions on, 19 (2004) 1541-1550.

[5] [5] M.N. Marwali, J. Jin-Woo, A. Keyhani, Power Electronics, IEEE Transactions on, 19 (2004) 1551-1561.9

[6] [6] E. Ibarra, J. Andreu, I. Kortabarria, E. Ormaetxea, I.M. de Alegría, J.L. Martín, P. Ibañez, Electric Power Systems Research, 81 (2011) 538-552.

[7] [7] I.-Y. Chung, W. Liu, K. Schoder, D.A. Cartes, Electric Power Systems Research, 81 (2011) 1051-1059.

[8] [8] C. Elmas, O. Deperlioglu, H.H. Sayan, Expert Systems with Applications, 36 (2009) 1540-1548.

[9] [9] S. Bifaretti, P. Zanchetta, A. Watson, L. Tarisciotti, J.C. Clare, Smart Grid, IEEE Transactions on, 2 (2011) 231-243.

[10] [10] M. Brenna, G.C. Lazaroiu, G. Superti-Furga, E. Tironi, Power Delivery, IEEE Transactions on, 23 (2008) 907-914.

[11] [11] M. Phattanasak, R. Gavagsaz-Ghoachani, J.P. Martin, B. Nahid-Mobarakeh, S. Pierfederici, B. Davat, in, 2011.

[12] [12] H. Karimi, H. Nikkhajoei, R. Iravani, Power Delivery, IEEE Transactions on, 23 (2008) 493-501.

[13] [13] Y.Q. Zhan, Y.H. Li, S.S. Choi, S. Rajakaruna, D.M. Vilathgamuwa, Power Delivery, IEEE Transactions on, 21 (2006) 1421-1429.

[14] [14] H. Valderrama-Blavi, J.M. Bosque, J.A. Barrado, M. Munoz, J. Calvente, Power Electronics, IET, 1 (2008) 203-213.

[15] [15] C.-H. Cheng, P.-J. Cheng, M.-J. Xie, Expert Systems with Applications, 37 (2010) 733-740.

[16] [16] F.A. Olsen Berenguer, M.G. Molina, International Journal of Hydrogen Energy, 35 (2010) 5974-5980.

[17] [17] A. Hasanzadeh, C.S. Edrington, H. Mokhtari, Electric Power Systems Research, 81 (2011) 2188-2197.

[18] [18] L. Guo, J.Y. Hung, R.M. Nelms, Electric Power Systems Research, 83 (2011) 104-109.

[19] [19] T. Niknam, H.Z. Meymand, H.D. Mojarrad, Energy, 36 (2011) 119-132.

[20] [20] A. Sakhare, A. Davari, A. Feliachi, Journal of Power Sources, 135 (2004) 165-176.

[21] [21] C.N. Papadimitriou, N.A. Vovos, in: Power Electronics and Applications, 2009. EPE '09. 13th European Conference on, 2009, pp. 1-10.

[22] [22] P. Thounthong, S. Pierfederici, B. Davat, Energy Conversion, IEEE Transactions on, 25 (2010) 909-920.

[23] [23] F. Kurokawa, Y. Maeda, Y. Shibata, H. Maruta, T. Takahashi, K. Bansho, T. Tanaka, K. Hirose, in: TENCON 2010 - 2010 IEEE Region 10 Conference, 2010, pp. 2159-2164.

[24] [24] F. Kurokawa, Y. Maeda, Y. Shibata, H. Maruta, T. Takahashi, K. Bansho, T. Tanaka, K. Hirose, in: Power Electronics and Motion Control Conference (EPE/PEMC), 2010 14th International, 2010, pp. S13-15-S13-19.

[25] [25] A. Prodic, D. Maksimovic, in: Computers in Power Electronics, 2002. Proceedings. 2002 IEEE Workshop on, 2002, pp. 18-22.

[26] [26] W. Kexin, S. Qiang, L. Bin, D. Mingxing, in: Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on, 2008, pp. 139-143.

[27] [27] Y. Guo, C. Zhang, Z. Wang, L. Huang, in: Information Networking and Automation (ICINA), 2010 International Conference on, 2010, pp. V1-329-V321-332.

[28] [28] L. Guo, J.Y. Hung, R.M. Nelms, Electric Power Systems Research, 81 (2011) 99-106.

[29] [29] W.L. Jean-Jacques E.Slotine, Applied Nonlinear Control, 1 ed., Prentice Hall, New Jersey, 1991.

[30] [30] H. K.Khalil, Nonlinear Systems, 3rd ed., Prentice Hall, New Jersey, 2002.

[31] [31] T. Siew-Chong, Y.M. Lai, C.K. Tse, Circuits and Systems I: Regular Papers, IEEE Transactions on, 53 (2006) 1816-1827.

[32] [32] A. Cid-Pastor, R. Giral, J. Calvente, V.I. Utkin, L. Martinez-Salamero, Circuits and Systems I: Regular Papers, IEEE Transactions on, 58 (2011) 2566-2577.

[33] [33] W. Rong-Jong, S. Kuo-Ho, Industrial Electronics, IEEE Transactions on, 53 (2006) 569-580.

[34] [34] R. Melício, V.M.F. Mendes, J.P.S. Catalão, Energy, 36 (2011) 520-529.

[35] [35] C.P. Coleman, D. Godbole, in: Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on, 1994, pp. 1654-1659 vol.1653.

[36] [36] S.F. Pinto, J.F.A. Silva, Industrial Electronics, IEEE Transactions on, 46 (1999) 39-51.

[37] [37] J.F.A. da Silva, V. Fernão Pires, S.F. Pinto, J.D.s. Barros, Mathematics and Computers in Simulation, 63 (2003) 281-295.

[38] [38] E.M. Navarro-López, D. Cortés, C. Castro, Electric Power Systems Research, 79 (2009) 796-802.

[39] [39] Y.B. Shtessel, A.S.I. Zinober, I.A. Shkolnikov, Automatica, 39 (2003) 1061-1067.

[40] [40] D. Biel, F. Guinjoan, E. Fossas, J. Chavarria, Circuits and Systems I: Regular Papers, IEEE Transactions on, 51 (2004) 1539-1551.

Optimal Solution to the Problem of Balanced Academic Curriculum Problem Using Tabu Search

Lorna V. Rosas-Téllez[1], **José L. Martínez-Flores**[2], **and Vittorio Zanella-Palacios**[1]
[1]Engineering Department ,Universidad Popular Autónoma del Estado de Puebla UPAEP,21 sur 1103 Col. Santiago C.P. 72410,Puebla Pue., México
[2] Interdisciplinary Center for Postgraduate Studies, Research, and Consulting ,Universidad Popular Autónoma del Estado de Puebla UPAEP, 17 sur 901 Col. Santiago C.P. 72410,Puebla Pue., México

Abstract - *The Balanced Academic Curriculum Problem (BACP) is a constraint satisfaction problem classified as NP- Hard, this problem consists in the allocation of courses in the periods that are part of a curriculum such that the prerequisites are satisfied and the load of courses is balanced for the students. In this paper is presented the solution for a modified BACP where the loads may be the same or different for each one of the periods and is allowed to have some courses in a specific period. This problem is modeled as an integer programming problem and is proposed the use of tabu search for its solution because was not possible to find solutions for all the instances of this modified problem with formal methods.*

Keywords: Academic Curriculum Balanced Problem, Tabu Search

1 Introduction

A curriculum is formed by a set of courses and these courses have assigned a number of credits that represent the effort in hours per week that the student requires to follow the courses successfully. For parents or tutors and for the institution represents the economic cost of this course. The academic load is the sum of the credits of all the courses in a given period. Therefore, the correct planning of the curriculum results in benefit of the all the involved: For the institutions, favors the departmentalization and the resulting cost savings, for the students, one good load distribution represents the academic effort that they require invest, for the parents or tutors, a good distribution of the credits allow planning financial efforts. Balanced Academic Curriculum Problem (BACP) consists in the allocation of courses in the periods that are part of a curriculum such that the prerequisites are satisfied and the credits load is balanced. The BACP belongs to the class of problems CSP (Constraint Satisfaction Problems), and this is a decisional optimization problem classified as NP-Hard [1].

The BACP problem was introduced by Castro and Manzano [2] with three test cases called BACP8, BACP10 and BACP12 included in CSPLib [3] and these have been used to test models proposed by other researchers.

The model proposed in [2] uses the following integer programming model:

Parameters

m: Number of courses
n : Number of periods
α_i: Number of credits of course i; $\forall i$ =1..m
β : Minimum academic load allowed per period
γ : Maximum academic load allowed per period
δ : Minimum amount of courses per period
ε : Maximum amount of courses per period

Decision Variables

x_{ij} = { 1 if course i is assigned to period j ; 0 otherwise

c_i : academic load of period j, $\forall j$ =1, ..., n

$$c_j = \sum_{i=1}^{m} \alpha_i * x_{ij} \qquad \forall j = 1..n \tag{1}$$

Objective function

$$Min\ \ c = Max\{c_1, \ldots, c_n\} \tag{2}$$

Constraints
If the course b has the course a as prerequisite then:
$x_a < x_b$

$$\beta \le c_k \le \gamma \quad \forall j = 1\ldots, n \tag{3}$$

$$\delta \le \sum_{k=1}^{m} x_{ij} \le \varepsilon \quad \forall j = 1, \ldots, n \tag{4}$$

Recent works have tried to solve this problem using genetic algorithms and constraint propagation [4], local search techniques [5], formal methods (HyperLingo) for the integer programming problems [6] and multiple optimization, using genetic algorithm of local search [7]. All these approach have found the optimal for the three test cases included in CSPLib and in some cases also for the curriculums of their universities.

In [6] was proposed a modified BACP problem where are considered constraints of academic load and total of courses within a specific range per period, i.e., not necessarily all periods will have the same ranges for their academic loads and number of courses; also add the restriction of to locate a course in a given period. This problem was modeled as an integer programming problem and is reported to find optimum solutions using a formal method for some of its instances but not for all, and the solutions for the three instances included in CSPLib.

In this paper is solved the modified BACP using tabu search to find solutions to the instances that the formal method could not to solve

1.1 Formulation for Model BACP Modified.

In the model of interest proposed in [6] is considered to modify two constraints of the base formulation, the first one is to make flexible the course load per period and the second one is to make flexible the number of courses per period, i.e., that we can place different limits on course load and number of courses for each period. It also adds a restriction which allows the location of some of the courses in a specific period.

Parameters

Nta : Number of courses
Ntp : Number of academic periods
crd_i : Number of course credits i=1..*Nta*
mca_j: Minimum academic load allowed per period
Mca_j: Maximum academic load allowed per period
mna_j: Minimum number of courses per period
Mna_j: Maximum number of courses per period
c: Course it is desirable to locate between certain periods.
mpcc: Minimum period of location of the course
Mpcc: Maximum period of location of the course
C_j: Academic load

$$C_j = \sum_{i=1}^{Nta} crd_i * x_{ij} \qquad \forall j = 1..Ntp \tag{5}$$

Decision Variables

C_j: Academic load for the period j =1..*Ntp*
Cmx : Maximum course load

x_{ij} = { 1 if course *i* is assigned to period *j*; 0 otherwise

Objective Function

$$f_{objective} = Min\{Cmx\} \tag{6}$$

where $Cmx = Max\ \{ c_1, c_2, \ldots, c_{Ntp} \}$

Constraints

The load of the period *j* must be within the allowable range.

$$mca_i \le C_j \le Mca_j \qquad \forall j = 1..Ntp \tag{7}$$

The number of courses of the period must be within the allowable range.

$$mna_j \le \sum_{i=1}^{Nta} x_{ij} \le Mna_j \qquad \forall j = 1..Ntp \tag{8}$$

If the course *b* has the course *a* as prerequisite then

$$x_{bj} \le \sum_{r=1}^{j-1} x_{ar} \qquad \forall j = 2..Ntp \tag{9}$$

Convenient location for the course *c*

$$\sum_{j=mpv_c}^{Mpc_c} x_{cj} = 1 \tag{10}$$

2 Tabu Search

The proposal solution is based on use of heuristic in this case tabu search. The tabu search is a method used to solve combinatorial optimization problems. The main idea behind the tabu search method is that by using a "memory" forces the method to explore new areas in the search space. That is, it can "memorize" some solutions that have been examined recently and these points become forbidden (taboo) to make decisions about the following solution. To use this method is used the following structure to represent the different actors involved in the model. One element of the population is represented by a vector, where the position indicates the course and the content of each position indicates the period to which it was assigned, as shown in figure 1.

0	1	2	3	4	5		59	60	61
1	1	3	1	2	2		9	9	0

Figure 1 Element of the solution

In our case we use a short-term memory on the movements of courses by period. To keep track of movements that have been made, and will are prohibited, we use a two dimensional array, of size total number of periods (*ntp*), in which is stored the periods between which the movement took place as shown in figure 2.

	0	1		*ntp-1*
min	1	2	…	4
max	3	7	…	2

Figure 2 Short-term memory

We can consider that a balanced curriculum should have a uniform distribution of all the credits that make up the curriculum, so the fitness function used is the sum of the absolute error, which is calculated using the following function.

$$Fitness(h) = \sum_{k=1}^{Ntp} |C_k - P| \qquad (9)$$

Where C_k is the academic load of the period k calculated with the formula (4) and P is the average number of credits per period.

$$P = \sum_{i=1}^{Ntp} C_i / Ntp \qquad (10)$$

The initial population consists of the curriculum that we want to balance, this is a feasible solution. Once we have the first feasible solution is necessary to obtain the period with least load and period with more load. And these periods are stored in the array that will handle the short-term memory for the tabu search.

$Pmax = \max\{C_k\}$ to k=1, …, *ntp*
$Pmin = \min\{C_k\}$ to k=1, …, *ntp*

To generate a new possible solution is sought the first course in order of appearance, which belongs to the period with maximum load (*Pmax*). Given the course are validated the restrictions of prerequisites, load and preference of the period of a course, if they are satisfied the change is made, in another way another course is chosen in the same way and redo the validation of the new course. In case that none of the courses reassigned to *Pmax* period allow the change, the minimum period current is marked as ineligible and is calculate a new minimum period, with which is repeat the above process. At the time of finding a solution the periods marked as ineligible are marked as eligible. The two solutions, the current and the solution generated, are evaluated by the fitness function (formula 10) and the best is selected for the next generation. In each iteration is tested that the *Pmax* and *Pmin* periods are not in the tabu list. Once that the tabu list is filled, the counter of the tabu list is set at 0 to begin replacing first the old value like a circular list.

When is detected that a local optimum has been reached, a change in the process of generation of a new solution is made. Now the new solution is generated exchanging two courses, the first one in the period with more load and the second one in the period with less load. Having the two courses which will be exchanged, are evaluated the restrictions of prerequisite , load, course and period of preference, if the exchange can be given, the new individual is generated, in other way is chosen another course of the minimum period and the validation is made newly.

3 Results

The tests were carried out for the three base cases included in CSPLib and the cases proposed by [5] for which no solution could be found.

3.1 Base Cases

The base cases included in CSPLib are: BACP8, BACP10 and BACP12, whose features are shown in table 1.

Table 1. General features of curriculums

Code	BACP8	BACP10	BACP12
# Total Courses	46	42	66
# Total credits	133	134	204
#Total Academic period	8	10	12
#Relation Prerequisite	33	34	65
Min. Courses /period	2	2	2
Max. Courses / period	10	10	10
Min Load/ period	10	10	10
Max Load/ period	24	24	24
#Courses with location	0	0	0

Table 2 shows the result obtained with the proposed algorithm, in all cases the optimum is reached.

Table 2. Results summary

Code	Optimum	Average Iterations	Average time (min.)
BACP 8	17	57.6	1.5
BACP 10	14	87.7	1.7
BACP 12	17	162.0	2.5

The academic load per period obtained by the algorithm is shown in table 3.

Table 3. Solution found for BACP 8

Period	Load	Courses
1	17	7
2	17	5
3	17	5
4	17	6
5	17	6
6	17	6
7	15	5
8	16	6

3.2 **Proposed Cases**

The cases not included in library CSPLib used to test this algorithm are taken from [5], the first is one for which could not always find the optimal and the second is where the optimum never was found. The features of these two problems are shown in table 4.

Table 4. General features of curriculums

Code	Ici-06	Ind-06
# Total Courses	61	61
# Total credits	488	376
#Total Academic period	9	9
#Relation Prerequisite	48	47
Min. Courses /period	5	4, 4, 4, 4, 4, 4, 4, 4, 2
Max. Courses/ period	8	9, 9, 9, 9, 9, 9, 9, 9, 4
Min Load/ period	20	20, 20, 20, 20, 20, 20, 20, 20, 15
Max Load/ period	60	60, 60, 60, 60, 60, 60, 60, 60, 40
#Courses with location	15	21

In tables 5 and 6 is showing the courses that have preference of location in each of the curriculums, Ici-06 and Ind-06 respectively.

Table 5. Preference of location Ici-06

Course Code	Minimum Period	Maximum Period
C07001	7	9
C07002	7	9
C07003	7	9
CIV200	1	2
CIV400	6	9
CIV401	8	9
CIV403	6	9
MAT005	1	5
MAT006	1	5
MAT008	1	5
MAT009	1	5
OI103101	1	4
OI103102	1	4
OI103103	1	4
OI103104	1	4

Table 6. Preference of location Ind-06

Course Code	Minimum Period	Maximum Period
C12001	7	9
C12002	7	9
C12003	7	9
C12004	8	9
FHU001	1	6
FHU002	1	6
FHU003	1	6
IND100	1	2
IND208	4	6
IND212	4	6
IND214	6	8
IND400	7	9
LPCI	1	6
LPCII	1	6
OH25001	1	6
OI103101	1	6
OI103102	1	6
OI103103	1	6
OI103104	1	6
SSC001	5	9
SSP002	5	9

Table 7 shows the results obtained with the algorithm; in all cases the optimum was reached.

Table 7. Results summary

Code	Optimum	Average Iterations	Average time (min.)
Ici-06	55	57.6	16
Ind-06	44	87.7	19.7

The academic load per period obtained by the algorithm is shown in table 8.

Table 8. Solution found for Ind-06

Period	Load	Courses
1	54	7
2	54	6
3	54	6
4	54	7
5	55	6
6	55	6
7	54	8
8	54	7
9	54	8

4 Conclusions

In this paper we present the solution, using tabu search, for a modified Balanced Academic Curriculum Problem, where the load for each period can be equal or different and is allowed to have some courses in a specific period. In a previous work is showed that is possible to find solutions with HyperLingo for some of the instances of the problem, but not for all of them. However by the results obtained was proved that the use of tabu search helps to find solutions to the problems that could not be resolved with the formal method.

5 References

[1] J. Salazar, "Programación matemática", Madrid: Diaz de Santos, 2001.

[2] C. Castro and S. Manzano, "Variable and value ordering when solving balanced academic curriculum problem", Proc. of the ERCIM Working Group on Constraints, 2001.

[3] T. Lambert, C. Castro, E. Monfroy, F. Saubion, "Solving the Balanced Academic Curriculum Problem with an Hybridization of Genetic Algorithm an Constraint Propagation", In Proceedings of ICAISC, pages 410-419, Berlin: Springer-Verlag, 2006.

[4] Luca Di Gaspero, Andrea Schaerf, "Hybrid Local Search Techniques for the Generalized Balanced Academic Curriculum", In Proceedings of HM, pages 146-157, Berlin: Springer-Verlag, 2008.

[5] José Antonio Aguilar Solís, "Un modelo basado en optimización para balancear planes de estudio en Instituciones de Educación Superior", Phd.Tesis, Puebla: UPAEP, 2008

[6] Carlos Castro, Broderick Crawford, Eric Monfroy, "A Genetic Local Search Algorithm for the Multiple Optimisation of the Balanced Academic Curriculum Problem", In Proceedings of MCDM, pages 824-832, Berlin: Springer-Verlag, 2009.

SESSION

GENETIC AND EVOLUTIONAY ALGORITHMS + NOVEL APPLICATIONS

Chair(s)

Prof. Hamid Arabnia

Application of Genetic Algorithm to Sequential Irrigation/Single Machine Scheduling Problem

Zia Ul Haq[1] and Arif A. Anwar[2]
[1]Assistant Professor, Department of Agricultural Engineering, University of Engineering and Technology, Peshawar, KPK, Pakistan.
[2]Senior Researcher-Irrigation and Head, International Water Management Institute (IWMI) Pakistan 12km Multan Road Chowk Thokar Niaz Baig, Lahore, Pakistan

Abstract- *In sequential/rotation irrigation scheduling farmers are supplied water sequentially. This problem has an analogy with the classical single machine earliness/tardiness scheduling problem in operations research. Such scheduling problems belong to a class of combinatorial optimization problems known to be computationally demanding (NP-hard) where computation time increases exponentially with problem size. Hence exact algorithms like integer program can only be used to solve relatively small problems. For practical applications (having large problem size), metaheuristics such as genetic algorithms, simulated annealing, or tabu search methods need to be used. This paper explores the potential of genetic algorithm (GA) to solve the sequential irrigation scheduling problems.*

Keywords: Sequential irrigation scheduling, single machine earliness/tardiness, genetic algorithm.

1 Introduction

Several delivery methods are in practice in irrigated agriculture throughout the world and a variety of classifications have been suggested by different researchers. Demand, arranged, and rotation are the three main types of irrigation schedules/delivery methods. The rate, frequency, and duration are all fixed and remain fixed for the entire irrigation season in rotation schedules and each farmer is supplied water sequentially for a specified period of time. [1] described rotation schedules (sequential irrigation) known as *warabandi* in the Indian subcontinent, as the most widely used of the modern irrigation delivery methods. The supply of irrigation water to farmers in an irrigation scheme under rotation or *warabandi* can be described as a single machine scheduling problem [2]. There is a single resource/machine i.e. water and a number of jobs to be processed, i.e. farmers supplied with water. The duration of water required by any farmer is comparable to the processing time of any job. In irrigation scheduling the term irrigation interval is used to describe the time period within which all farmers have to be supplied with water. A comparable term for irrigation interval in OR is the deadline which is different than the due dates mostly used in OR for the desirable completion time of jobs. [3] argued that deadlines must be met and cannot be violated in contrast to due dates which may be violated. This is also compatible with irrigation systems where the irrigation interval is never violated but the supply of water i.e. jobs may be either early or tardy. [4] used the term preferred starting time at which it would be desirable to start processing a job and preferred completion time as an equivalent term for due dates. Although there is no such term as due date used in irrigation, however it may be conveniently replaced by target start time. Target start time plus duration of a certain job becomes the due date for that job. However it is more convenient to use the target start time in irrigation, as the farmers usually place their orders in terms of the start time of their irrigation not completion time.

There are different ways whereby the distribution of irrigation water can be managed. The better among them are the flexible distribution systems, where an effort is made by the supplier to match the scheduled irrigation start times to the target start times requested by the farmers. It will be more intuitive to judge the suitability of any such schedule by the determination of how close the scheduled start times are to the target start times. This constitutes a typical OR scheduling problem, i.e. sequencing with earliness and tardiness (ET), distinct due dates and a common deadline. [3] described one such problem briefly while citing an earlier reference. [5] described distinct due dates as one of the classes of single machine ET problems and argued that the ET model with the distinct due date assumption intentionally minimizes the sum of jobôs earliness and tardiness, and facilitates a feasible delivery schedule. [4] also described the problem of minimizing the total discrepancy from preferred starting times over a single machine. [6] described the single machine job scheduling problem with distinct due dates and general penalty weights for early and tardy jobs. They presented a two-step solution to the problem, i.e. a timing algorithm first to determine the optimal start for each job and then a genetic algorithm for determining near optimal sequences with idle time inserted between blocks with contiguous jobs. However, [7] demonstrated how this two-step solution i.e. sequencing and scheduling separately, could lead to deterioration in solution. [8] described considering sequencing and scheduling simultaneously with inserted idle time as a new area of research and recommended further research on it. This wealth of information may be very effectively applied to irrigation scheduling which is as

demonstrated, not any different from the classical scheduling problems found in OR.

[5] described the just-in-time (JIT) philosophy to be a popular management concept since its introduction in 1970s. Earliness and tardiness problem where both early and late jobs are undesirable is one of the key problem areas in JIT implementation. [9] observed that JIT production philosophy led to a growing interest in scheduling problems considering both earliness and tardiness penalties and that a majority of which are devoted to single machine problems. Similarly [5] also concluded that the vast majority of published ET research dealt with sequencing jobs on a single machine.

In machine scheduling problems idle time insertion can result in a better schedule. [7] defined inserted idle time (IIT) schedules as a feasible schedule in which a machine is kept idle when it could begin processing an operation. [8] described idle time insertion necessary in just-in-time (JIT) environments where costs associated with even early completion of jobs are relevant i.e. the performance measure is nonregular. Similarly [3] considered the assumption of no inserted idle time to be inconsistent with JIT philosophy or earliness and tardiness criteria where jobs are neither allowed to be early nor tardy. [5] cautioned that an ET solution procedure may fail to find a true solution if idle time is not treated properly.

From irrigation scheduling perspective, idle time can only be inserted in a sequential irrigation system when the sum of all the individual farmersô irrigation (or jobsô) durations called makespan in OR is less than the irrigation interval. However if simultaneous application to several farmers is allowed this becomes irrelevant. If idle time has to be inserted, two scenarios could be imagined. One is that the supply channel is continuously flowing; farmers abstract water as scheduled; and when water is not being used i.e. idle time inserted, it is either drained and/or if possible reused. The other scenario could be to shut the channel each time idle time is inserted or water is not being used. The former may result in wastage of water while the latter would result in an excessive number of gate operations. An alternative solution would be to schedule the irrigation water supply contiguously, i.e. when one farmer finishes his turn of irrigation the supply is diverted to the next adjacent farmer and so on [2]. There are some implicit assumptions here that either no time is taken by water to travel from one farmerôs outlet to another or the travel time is part of each farmerôs irrigation duration or is very small and hence negligible. If all jobs are scheduled contiguously the gates are only needed to be opened at the beginning of the first farmer irrigation and closed when the last farmer has finished his turn. However the idle time insertion still needs to be addressed. There are three options: either to insert all the idle time in the beginning of the schedule, or to insert all the idle time at the end of the schedule or both, though corresponding parallel with OR literature could not be drawn for either of these options. However, examples of jobs scheduled contiguously in blocks or groups and then inserting idle time between different blocks could be found in OR literature e.g. [6]. The three options discussed in the preceding lines for idle time insertion are also considered by [2] in their contiguous sequential irrigation models. The decision to insert idle time or not, or schedule jobs contiguously is dictated by several factors. These include the type of distribution system in vogue, the level of service provided, the total amount of water available, and canal capacities and automation etc.

[2] were the first to demonstrate sequential irrigation analogous to single machine scheduling problem and presented an integer programming solution. Machine scheduling problems belong to a class of combinatorial optimization problems known to be computationally demanding (NP-hard). A problem is termed NP-hard if its solution time increases exponentially with the problem size. As described by [10], for NP-hard optimization problems like the machine scheduling problems, larger problems may require excessive solution times using exact algorithms e.g. the integer program. Approximate algorithms or heuristics like GA, are considered the appropriate choice for such problems. Similar views are shared in OR literature (e.g. [5]).

This paper explores the potential of GA for sequential irrigation scheduling problem (single machine scheduling in OR). Single machine irrigation scheduling models are developed using GA. These GA models are applied to the practical application by [11], for validation and evaluation purposes.

2 Mathematical formulation

2.1 Model 1

Model 1 refers to the non-contiguous single machine ET model that allows idle time to be inserted between jobs i.e. there will be times within the irrigation interval where water will not be used by any farmer. This arrangement may require an excessive number of gate opening and closing operations, depending on the number of times idle time is inserted between jobs. Alternatively, a continuous flow system may be adopted and water allowed draining when not in use. A detailed description of the decision variables, objective function and the constraints is given below.

2.1.1 Decision variables

There are two decisions to be made: which outlet to receive water and at what time, i.e. the sequencing and scheduling. Thus the genes of a chromosome representing solution to this problem must have answers to these questions. The answers to these questions are incorporated into a single decision variable. This decision variable is represented by a scheduled start time row vector. Each element in the vector is a positive integer representing the point in time at which an outlet

is scheduled to start receiving water and is expressed as follows.

S_j = an element of the scheduled start time row vector (schedule start time of outlet *j*) (1)

where subscript *j* represents the outlet index i.e. the position of job in the chromosome and hence the sequence.

2.1.2 Objective function

The objective of the model is to find a sequence of jobs and the scheduled start times for all jobs with a minimum difference between the scheduled start time and the target start time. This is achieved by penalizing both early and tardy jobs. Some farmers may have higher priority for getting water supply earlier than others for a variety of reasons. For example, his/her crops have more value than others or more sensitive to water stress or perhaps for social/political reasons, etc. By using different unit costs for either earliness or tardiness, jobs may be prioritized. The objective function can be expressed as

$$\text{Minimize } [\sum_{j=1}^{J} (\alpha_j E_j + \beta_j T_j)] \quad \forall\ j = 1, 2, \ldots J \tag{2}$$

where E_j = earliness of job *j* (the difference of the target start time and the scheduled start time of outlet *j*); T_j = tardiness of job *j* (the difference of the scheduled start time and the target start time of outlet *j*); α_j = cost of earliness per unit of time for job *j*; β_j = cost of tardiness per unit of time for job *j*; *j* = job/outlet index = 1, 2, … *J*; and *J*= total number of jobs/outlets.

2.1.3 Constraints

Any constraint violation causes a schedule to become infeasible. There are different techniques available to control infeasibility in genetic algorithm. Based on the literature reviewed, penalty strategy that turns out to be the most simple and widely practiced technique for controlling infeasibility, is adopted for the present model. In the penalty function technique each instance of infeasibility is appropriately penalized and (constraint violations expressed as) penalties are then added to the objective function. The resulting objective function may then be termed as fitness function. There are two constraints in the current model. The first constraint is the irrigation interval constraint and the second is the overlap constraint. The penalties for constraint violations in the present formulation are as follows.

2.1.3.1 Irrigation interval constraint

Each outlet is to be scheduled within the specified irrigation period. Any outlet scheduled outside this period will result in infeasible schedule. The penalty for this constraint violation may be mathematically expressed as:

$$P_I = \sum_{j=1}^{J} [(S_j + D_j - G)\delta_j + (S_{int} - S_j)\lambda_j] \quad \forall\ j = 1, 2, \ldots J \tag{3}$$

where P_I = penalty for irrigation interval violation; G = total irrigation time available; S_{int} = start time of the irrigation interval; S_j = scheduled start time of outlet *j*; and D_j = duration of outlet *j*.

$$\delta_j = 1 \quad \text{if} \quad S_j + D_j > G \quad \forall\ j$$
$$= 0 \quad \text{otherwise} \tag{4}$$

$$\lambda_j = 1 \quad \text{if} \quad S_j < S_{int} \quad \forall\ j$$
$$= 0 \quad \text{otherwise.} \tag{5}$$

2.1.3.2 Overlap constraint

Only one outlet is to be served at a time. The penalty for violation of this constraint is determined by summation of the number of times overlap occurs in all time blocks and is expressed mathematically as follows.

$$P_O = \sum_{t=1}^{T} (\sum_{j=1}^{J} \psi_{tj}) \tag{6}$$

$$\text{if } (\sum_{j=1}^{J} \psi_{tj}) > 1 \text{ then } (\sum_{j=1}^{J} \psi_{tj}) = 0$$

$$\forall\ t = 1, 2, \ldots T \tag{7}$$

where P_O = penalty for overlap of jobs; *t* = time block index = 1, 2, … *T*; and, *T* = total number of time blocks.

$$\psi_{tj} = 1 \quad \text{if } S_j \le t < S_j + D_j; \ \forall\ t, \forall\ j$$
$$= 0 \quad \text{otherwise} \tag{8}$$

By adding these penalties for constraint violations to the objective the resultant fitness function is expressed as follows.

Minimize

$$F = [\sum_{j=1}^{J} (\alpha_j E_j + \beta_j T_j) + R_I P_I + R_O P_O] \tag{9}$$

where F = fitness function; R_I = penalty weight for P_I ; and R_O = penalty weight for P_O .

2.2 Model 2

Model 2 refers to a series of single machine ET contiguous irrigation scheduling models. There are three variations of Model 2 i.e. 2a, 2b, and 2c. In Model 2a jobs are scheduled contiguously and idle time inserted at the end of the last job. In Model 2b all the jobs are scheduled contiguously and idle time inserted before the start of the first job. In Model 2c the jobs are scheduled contiguously and idle is inserted preceding the start of first job and /or proceeding the end of the last job.

2.2.1 Decision variables

The only decision to be made in these permutation models is the sequence of jobs. The chromosome is a permutation of jobs where each gene represents job *j*. Once the sequence of jobs is decided, the scheduled start time of each job can then be calculated. For Model 2a the scheduled start time of the first job in the sequence is ñ0ò or the time the irrigation interval starts. The scheduled start time for the rest of the jobs can be calculated as follows.

$$\check{S}_i = \check{S}_{i-1} + \check{D}_{i-1} \qquad \forall\ i = 2, 3\acute{e}\ J \tag{10}$$

where $\check{S}_i$ = scheduled start time of the job at the i^{th} position in the jobs sequence (chromosome); $\check{S}_{i-1}$ = scheduled start time of the job preceding the job at the i^{th} position in the jobs sequence; $\check{D}_{i-1}$ = duration of the job preceding the job at the i^{th} position in the jobs sequence; and i = position of the job in the jobs sequence, so that $i = 2$ represents the second job in the sequence whereas $i = J$ the last job in the sequence. For Model 2b the scheduled start time of the first job is the end of the idle time and can be expressed as:

$$\check{S}_1 = G - \sum_{j=1}^{J} D_j \tag{11}$$

where $\check{S}_1$ = the scheduled start time of the first job in the sequence. For the remaining jobs the scheduled start time can be calculated as in (10) after $\check{S}_1$ has been calculated. For Model 2c the scheduled start time of the first job is the end of the idle time inserted before the start of first job. Idle time in this case has a value equal to a random integer number in the range of irrigation interval minus the makespan. For the remaining jobs the scheduled start time can be calculated as in (10) after $\check{S}_1$ has been calculated as follows.

$\check{S}_1$ = an integer randomly selected in the range between

$$0 \text{ and } (G - \sum_{j=1}^{J} D_j) \tag{12}$$

2.2.2 Objective function

There is no change in the objective function and is similar to that of Model 1 (2) for all contiguous models i.e. 2a, 2b, and 2c. The objective is to find a sequence or permutation of jobs that best matches the scheduled start times with the target start times.

2.2.3 Constraints

Since in single machine contiguous models the population consists of a permutation of jobs, no irrigation interval constraint violation occurs. Other infeasibility problems are controlled via modified genetic operators. The objective function for all models in this category is the fitness function for all the individuals of the population.

3 GA implementation

The GA, for all the models described in the preceding sections, was implemented using JGA, a java genetic algorithms library ([12]). Some of the built-in classes were modified and some additional new classes were added to develop a complete GA implementation. The logic for this implementation of the genetic algorithm is presented in Figure 1; where, *t* is the generation counter; *T* is the maximum number of generations; *P(t)* is the population at generation *t*; *Cm(t)* and *Cc(t)* are the children populations obtained by the mutation and crossover operators, respectively; *C(t)* is the children population; and *E(t)* is the expanded population formed by the current population and their children.

1: $t \leftarrow 1$
2: initialize $\mathcal{P}(t)$
3: evaluate $\mathcal{P}(t)$
4: **while** $t \leq T$ **do**
5: mutate $\mathcal{P}(t)$ and generate $\mathcal{C}_m(t)$
6: cross $\mathcal{P}(t)$ and generate $\mathcal{C}_c(t)$
7: $\mathcal{C}(t) \leftarrow \mathcal{C}_m(t) \cup \mathcal{C}_c(t)$
8: evaluate $\mathcal{C}(t)$
9: $\mathcal{E}(t) \leftarrow \mathcal{P}(t) \cup \mathcal{C}(t)$
10: select $\mathcal{P}(t+1)$ from $\mathcal{E}(t)$
11: $t \leftarrow t + 1$
12: **end while**

Figure 1 The logic behind GA implementation ([12]).

3.1 Initial Population

Initial population for Model 1 is randomly generated. The chromosome consists of positive integers, randomly generated within the range of irrigation interval. The length of the chromosome is equal to the number of jobs considered. The population size of 100, that proved sufficient during preliminary experiments, is used for Model 1. For all versions of Model 2 the chromosome consists of non-repeated integer valued genes, of length equal to the number of jobs. Each gene is an integer in the range between 1 and the total

number of jobs. The population consists of randomly generated permutations of job sequences because permutations are considered natural representation for sequences. Initial population size of 200 is used that proved satisfactory during preliminary experiments.

3.2 Selection

The best individual selection is used for the present models. Best individual selection is described by [13] as ñ*elitism*ò, where the elite member is not only selected but a copy of it is also preserved and becomes a part of the next generation without any perturbation by crossover or mutation operators. In the best individual selection used here, best individuals are selected from an enlarged population. The enlarged population is formed by offspring produced from crossover and mutation of parents as well as by the individuals from the current population ([14]).

3.3 Crossover

For Model 1, the most simple and widely used crossover operator, the single point crossover is used. For all versions of Model 2, order-based crossover (OX) is used ([15]). The OX operator selects at random two cut points along the strings. The substrings between the two cut points of both parents are exchanged. Starting from the second (right) cut point of both parents the remaining positions for each chromosome is completed by omitting the duplicated genes. When the end position of the string is reached, it continues from the first position till the chromosome is completed. In this way the OX operator avoid any infeasibility due to repeated genes in a sequence. A crossover probability of 0.8 was used for all models after satisfactory initial experimentation, and which is also used by [14] in some of their JGA application.

3.4 Mutation

For non-contiguous model, random assignment mutation is implemented which is a simple and widely used operator. In random assignment mutation each position in the chromosome is randomly reassigned with a certain probability. For all contiguous models, inversion mutation is used as a mutation operator. In inversion mutation the order of a randomly picked permutation section is inverted. For instance, if 1-2-3-4-5-6 is a sequence, and 3-4-5 is the randomly picked section; the mutated permutation is 1-2-5-4-3-6. The purpose is to maintain diversity as well as feasibility. A mutation rate of 0.2, which was found satisfactory during preliminary experimentation, is used in all models. The mutation rate is interpreted as the chance of mutation of a given genotype. The same has been used in some of the application of JGA by [14] and several other applications in literature.

3.5 Termination

An early auto-stopping criteria is used for all models. The improvement in the fitness function is monitored over 1000 generations (iterations). If the improvement is less than or equal to 0.001%, the iterations are terminated otherwise it continues until the given maximum number of generation is reached. The maximum number of generations used is 10,000 generations.

4 Practical Application

The GA models developed in this paper are applied to the tertiary unit taken from [1] and used by [11] with some modification for the purpose of demonstrating application of their models. The data for the practical application used by [11] and for the application in this paper is presented in Table 1. Table 2 presents a comparison of the GA models with the integer programming (IP) and heuristics (H) by [11] when applied to the data in Table 1.

Table 2 presents only the objective function values which is the earliness/tardiness (in minutes) of the schedules developed. The contiguous GA models have performed very well, with 1% or less than 1 % error relative to IP and outperforming the heuristics by a significant margin. The non-contiguous model did not perform as well against IP as it did against H.

Table 1. Data for Practical Application (Tertiary unit, Bula Project, Philiphine) [11]

Lot	Area (ha)	Target start time	Duration (h:min)
24.1	1.05	Mo 08:19	8:57
25.1	3.00	Mo 20:51	8:57
22.1	3.20	Tu 00:30	9:32
23.2	1.00	Tu 09:03	3:13
24.2	3.00	Tu 06:45	8:57
26.3	2.56	Tu 19:40	7:41
22.2	3.20	Tu 04:06	9:32
24.4	1.05	We 16:02	3:22
25.2	3.00	Th 16:41	8:57
26.1	3.00	Sa 21:29	8:57
25. 3	2.13	Su 12:09	6:27
23.4	1.97	Fr 02:08	6:00
24. 3	1.00	Fr 05:55	3:13
23.3	1.00	Fr 06:59	3:13
23.1	2.50	Su 22:35	7:31
26.2	2.50	Mo 03:03	7:31

Note: Mo = Monday; Tu = Tuesday; We =Wednesday; Th =Thursday; Fr = Friday; Sa = Saturday; and Su = Sunday.

Table 2. Results for Practical Application (Objective function values)

Statistics	Model			
	1	2a	2b	2c
Optimum value (IP)	3,822	19,667	31,952	19,001
Heuristics (H)	15,830	19,987	33,821	25,635
Genetic Algorithms (GA)	6,578	19,667	32280	19,146
Relative error of GA with IP (%)	72.0	0.0	1.0	0.8
Relative error of GA with H (%)	-58.0	-1.6	-4.6	-25.3

5 Conclusions

The GA models have performed very well and completely outperformed the heuristics by [11]. The difference in performance of the three contiguous GA models shows the sensitivity of the models to the insertion of the idle time. Inserting idle time on both sides of a schedule (before the start of irrigation and after the irrigation is complete), has been found useful. This is indicated by the fact that Model 2c performed better than Model 2a and 2b in terms of solution quality and computational efficiency. Having some spare time before the start of irrigation and after the irrigation is complete, may also provide some convenience to irrigation managers in managing the water supply.

The potential of the GA, to model the *warabandi* type (sequential irrigation) of irrigation systems has been demonstrated. The GA has proved to be an efficient optimization tool for the contiguous irrigation scheduling problems in particular. The contiguous scheduling models presented in this paper models the type of *warabandi* widely practiced in the sub-continent with the additional advantage of incorporating farmers desires as regard to their requested irrigation times, thus providing a better level of service. The models also have the ability to prioritize the irrigation turns, based on crops value, or sensitivity to water stress, or social/political reasons, etc.

6 Notation

The following symbols are used in this paper:

$\check{D}_{i-1}$ = duration of the job preceding the job at the i^{th} position in the jobs sequence;
D_j = duration of outlet j;
E_j = earliness of outlet j;
F = fitness function;
G = irrigation interval;
i = position of the job in the jobs sequence;
j = represents the outlet index;
J = total number of outlets;
P_I = penalty for irrigation interval violation;
P_O = penalty for overlap of jobs;
R_I = penalty weight for P_I;
R_O = penalty weight for P_O;
$\check{S}_1$ = the scheduled start time of the first job in the sequence;
$\check{S}_i$ = scheduled start time of the job at the i^{th} position in the jobs sequence;
$\check{S}_{i-1}$ = scheduled start time of the job preceding the job at the i^{th} position in the jobs sequence;
S_{int} = start time of the irrigation interval;
S_j = an element of the scheduled start time row vector;
t = time block index = 1, 2é T;
T = total number of time blocks;
T_j = tardiness of outlet j;
α_j = cost of earliness per unit of time for job j;
β_j = cost of tardiness per unit of time for job j;
δ_j = binary variable;
λ_j = binary variable;
ψ_{tj} = binary variable;

7 References

[1] Bishop, A. A., and Long, A. K. óóIrrigation water delivery for equity between users.ôô J. Irrig. Drain. Eng., 109(4), 349ï 356. (1983).

[2] De Vries, T.T., and Anwar, A.A. ñIrrigation scheduling I: Integer programming approach.ò J. Irrig. Drain. Eng., 130(1), 9-16. (2004).

[3] Baker K.R. and Scudder G.D. ñSequencing with earliness and tardiness penalties: A reviewò. Operations Research, Operations Research Society of America, 38(1), 22-36. (1990).

[4] Garey, M.R., Tarjan, R.E., and Wilfong, G.T. ñOne-processor scheduling with symmetric earliness and tardiness penaltiesò. Mathematics of Operations Research, 13(2), 330-348. (1988).

[5] Heady, R. B., Zhu, Z. ñMinimizing the sum of job earliness and tardiness in a multimachine systemò. Int. J. Prod. Res. 36, 1619-1632. (1998).

[6] Lee, C.Y. and Choi, J.Y. ñA genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weightsò. Computers Ops Res., 22(8), 857-869. (1995).

[7] Kanet, J.J and Sridharan, V. ñScheduling with inserted idle time: problem taxonomy and literature reviewò. Operations Research, 48(1), 99-110. (2000).

[8] Colin, E.C. and Quinino, R.C. ñAn algorithm for insertion of idle time in the single machine scheduling problem with convex cost functionsò. Computer & Operations Research, 32, 2285-2296. (2005).

[9] Lauff V. and Werner F. ñScheduling with common due date, earliness and tardiness penalties for multimachine problems: A surveyò, Mathematical and Computer Modelling .40, 637-655. (2004).

[10] De Vries, T.T., and Anwar, A.A.). ñIrrigation scheduling with travel times.ò J. Irrig. Drain. Eng.,132(3), 220-227. (2006

[11] Anwar, A.A., and De Vries, T.T. ñIrrigation scheduling II: Heuristics approachò. J. Irrig. Drain Eng., 130(1),17-25. (2004)

[12] Medaglia, A. L., and Gutiérrez, E. JGA: An Object-Oriented Framework for Rapid Development of Genetic Algorithms. IN: Rennard, J.P., ed. Handbook of Research on Nature Inspired Computing for Economics and Management. (2006a).

[13] Coley D.A., An Introduction to Genetic Algorithms for Scientists and Engineers, World Scientific Publishing Co. Pte. Ltd. (1999)

[14] Medaglia, A. L., and Gutiérrez, E. Applications of JGA to Operations Management and Vehicle Routing. IN: Rennard, J.P., ed. Handbook of Research on Nature Inspired Computing for Economics and Management. (2006b).

[15] Davis, L. A handbook of genetic algorithms, Van Nostrad Reinhold, New York. (1991)

New Parameters for the Evaluation of Benchmarks for Fast Evolutionary Scheduling of Workflows

Sylvia Strack[1], Wilfried Jakob[2], Günther Bengel[1], Alexander Quinte[2], Karl-Uwe Stucky[2] and Wolfgang Süß[2]

[1] University of Applied Sciences, Department of Computer Science, Paul-Wittsack-Str. 10, 68163 Mannheim, Germany

[2] Karlsruhe Institute of Technology (KIT), Institute of Applied Comp. Sc. (IAI), P.O. Box 3640, 76021 Karlsruhe, Germany

Abstract - *The basic task is the scheduling of* application jobs *consisting of a workflow of elementary* grid jobs *to a set of heterogeneous resources forming a computational grid. Application jobs have due dates and cost budgets, which must be adhered to resulting in a multi-objective optimisation problem. It is also a highly dynamic task as e.g. new application jobs may be added or resources may be switched off. Due to this dynamic nature of grid scheduling a permanent replanning process is required that adjusts the actual plan to the changed conditions. For the assessment of scheduling procedures benchmarks are needed that represent different load conditions as best as possible. Based on our benchmark parameters introduced in 2007 additional new ones were defined to cover the tightness of time and cost limits of application jobs with respect to the available resources.*

Keywords: Scheduling, benchmark parameter, workflow, grid computing, Evolutionary Algorithms

1 Introduction

This paper continues our work on benchmarks for the assessment of scheduling procedures for jobs on a computational grid [16]. Our investigation is motivated by the task of scheduling several *application jobs* organised as workflows consisting of elementary *grid jobs*. For each grid job users state which resources like software, storage or computing power are needed and how long the job will run on a standard hardware. Users also specify due dates, cost budgets and a preference for cheap or fast execution for their application jobs. Resources have different abilities and pricing schemes according to day time or day of the week and the costs may reflect their performances like CPU speed or bandwidth in case of computing or storage hardware. A formal definition of the basic scheduling problem can be found in [8] or [13], where it is also shown that the task is NP-complete. To meet the different requirements of resource owners and users the optimisation is based on the following main criteria: *costs* and *execution times* of the application jobs measured as the usage of user given budgets and time frames and then averaged. Additionally and more provider oriented, the *rate of utilisation* and the total *makespan* of all application jobs are weighted together with the two other criteria and summed up for the *raw fitness*. If application jobs are late or too costly penalty functions are applied, yielding a value between zero and one, with which the raw fitness is multiplied resulting in the *end fitness*. If no violations occur, both fitness values are equal. For more details see [7] and [8].

The practice of grid scheduling is characterised by a permanent replanning process as the following events may occur long before the actual plan is completed: new application jobs may introduce the system, waiting or running ones may be cancelled, new resources may become serviceable or others unavailable, to name only the most important inducements. The resulting benchmarks for replanning are based on the ideas and characteristics of those for the investigation of the "planning from scratch"-scenario as described by Süß et al. [16]. They are enhanced by new parameters, which express the tightness of budgets and time frames. Additionally, they are constructed in a more systematic way to allow a better scaling of the amounts of grid jobs and resources while keeping the other properties as far as possible unchanged. The time for planning is limited to three minutes as we can assume that this is much shorter than the processing times of most grid jobs. Due to this limit and the NP-completeness of the task only approximate solutions can be expected.

The new benchmarks described in detail in Sect. 3 are used firstly to find out the maximum possible scheduling load which can be processed by our *Global Optimising Resource Broker and Allocator* GORBA and secondly to assess the newly introduced parameters. Sect. 2 gives a brief overview about related work. In Sect. 4 the heuristic and the evolutionary planning phases of GORBA are explained together with the algorithms used. The two questions raised above are answered by the experiments reported about in Sect. 5 and the paper concludes with Sect. 6.

2 Related work

When we started our work on grid job scheduling in 2006 little literature about a comparable problem could be found only, see e.g. Brucker [3] or Brucker and Knust [4]. The literature review from Setämaa-Kärkkäinen et al. [15] came to

the same result. But then more papers have been published that dealt with multi-objective optimisation in the field of grid job scheduling. Wieczorek et al. [17] reported in a review article that frequently just two criteria are used, from which one often serves as a constraint so that only one is really optimised, like in the work of Sakellariou et al. [14] or Yu and Buyya [19]. An early real multi-objective optimisation seeking a compromise between resource owners and users was investigated by Kurowski et al. [10]. They used a modified weighted sum for contradicting criteria like costs and several time-related objectives. But their work was based on single grid jobs instead of workflows. Later they extended their work to a workload of up to 1500 jobs, which were grouped into 60 sets with different optimisation preferences [11]. They used advanced reservations like we do and extended their scheduler by the standard multi-objective genetic algorithm NSGA-II. The work of Mika et al. [12] is based on workflows and constrained resources like ours but based on an algorithm for finding feasible solutions. They discuss similarities with the resource-constrained project scheduling problem (RCPSP), see also Brucker and Knust [4].

The work of Xhafa et al. [18] will be discussed in more detail as their work comes closest to ours. The following features are in common: usage of an Evolutionary Algorithm, the concept of fast and permanent replanning, multi-criteria optimisation, and experiments based on large workloads and heterogeneous resource pools (up to half the size of ours). The main differences concern the following three issues. They schedule single jobs rather than multiple workflows and use only two criteria, *makespan* and *average flow time*, which can be estimated as less conflicting as *time*, *costs*, and *utilisation rate*. Thirdly, we use more heuristics for seeding the initial population and for resource allocation, as we have found that heuristic resource selection outperforms its evolutionary counterpart, if the runtime for planning is strictly limited [7]. But despite these differences both approaches can be regarded as successful solutions to the problem of fast rescheduling of grid jobs.

3 Benchmarks

Benchmarks are widely used for the assessment of scheduling procedures. They are either derived from real applications or constructed synthetically. For our purpose they should cover a wide range of possible scheduling loads and they should be scalable. As it is easier to steer dissimilarities and to ensure desired properties, we decided for synthetic benchmarks. For a more detailed discussion of that topic, see [16].

Our benchmarks are constructed according to a given number of grid jobs and resources, and a small or large degree of resource alternatives *R* and of mutual dependencies of the grid jobs *D*. Due to the lack of space we give here the formulas in Eq. 1 only and refer to the detailed description given by Süß et al. [16].

All four combinations of small or large degrees result in four basic *benchmark classes* denoted by *sRsD*, *sRlD*, *lRsD*, and *lRlD*, where *s* stands for *small* and *l* for *large*.

$$D = \frac{2 \cdot \sum_{i=1}^{n} p_i}{n \cdot (n-1)} \qquad R = \frac{\sum_{i=1}^{n} m_i}{n \cdot m}$$

with

- p_i number of all direct or indirect predecessors of grid job *i*
- n number of all grid jobs
- m number of all resources
- m_i number of all alternatively usable resources of grid job *i*

(1)

The first experiments based on 50, 100, and 200 grid jobs and 10 or 20 resources indicated that the two benchmark classes *sRlD* and *lRlD* are the most hardest ones [7]. Thus, they were also used for first experiments for replanning [8]. In [9] these experiments were widened to the two remaining classes, but unfortunately the two new benchmark series were faulty due to a software error of the benchmark generator. Therefore, and for the reasons given in the introduction, we constructed complete new benchmarks for the experiments reported here.

3.1 New benchmark parameters

The new parameters are aimed at an evaluation of the average *time* and *cost reserves* of the application jobs with respect to their user given limits. These reserves have an important impact on the complexity of the scheduling task. They are based on the average performance, costs, and off times of the resources and on the critical path of a workflow.

The *average performance* $\overline{f}$ of all hardware resources, the *average costs* $\overline{c}$, and the *off time rate* $\overline{R_{off}}$ of all resources are defined as

$$\overline{f} = \frac{\sum_{j=1}^{m_{hw}} f_j}{m_{hw}} \qquad \overline{c} = \frac{\sum_{j=1}^{m} \frac{c_j}{f_j}}{m} \qquad \overline{R_{off}} = \frac{\sum_{j=1}^{m} R_{off,j}}{m}$$

with

- m_{hw} number of hardware resources
- f_j speed factor of resource *j* w.r.t. a standard CPU
- c_j costs per time unit of resource *j*
- m number of all resources
- $R_{off,j}$ ratio between off-time of resource *j* and the latest due date of all application jobs minus the earliest start time of all application jobs

(2)

The *average processing time* t_k of application job *k* is based on the grid jobs of the critical path of its workflow, while the *average cost* c_k has to take all grid jobs into account. They are calculated as follows.

$$t_k = \sum_{i=1}^{n_{krit,k}} \left\lceil \frac{t_i}{f} \right\rceil \qquad c_k = \bar{c} \sum_{i=1}^{n_k} \left\lceil \frac{t_i}{f} \right\rceil \tag{3}$$

with

$n_{krit,k}$ number of grid jobs of the critical path of application job k

t_i processing time of grid job i on a standard CPU

n_k number of grid jobs of application job k

Now the *average time* and *cost reserves* $t_{res,k}$ and $c_{res,k}$ of application job k can be calculated. They are given as relative values with respect to the available time frame and budget respectively.

$$t_{res,k} = \frac{t_{fin,k} - t_{st,k} - \left\lceil t_k \cdot (1 + \overline{R_{off}}) \right\rceil}{t_{fin,k} - t_{st,k}}$$

$$c_{res,k} = \frac{c_{BUD,k} - c_k}{c_{BUD,k}} \tag{4}$$

with

$t_{fin,k}$ latest finishing time of application job k

$t_{st,k}$ earliest start time of application job k

$c_{BUD,k}$ budget of application job k

They are averaged for all application jobs of a benchmark *bm* yielding the *time* and *cost benchmark reserve* $\overline{t_{res,bm}}$ and $\overline{c_{res,bm}}$. Their use will be described at the end of the next section.

3.2 Benchmarks for rescheduling

As we consider the arrival of new grid jobs as one of the replanning events with the greatest impact, we chose it as the basic scenario of our replanning benchmarks. They are created in the following two steps. Firstly *basic benchmarks* consisting of 200 grid jobs and 20 resources are constructed for each benchmark class. For this purpose a set of application jobs with short sequences of grid jobs and therefore more parallel branches is manually created for benchmarks with small dependencies. For those with large dependencies application jobs with long sequences are made. A basic benchmark is assembled randomly from these application jobs of the actual class in such a way that the given number of grid jobs is adhered to and that the application jobs have ten grid jobs on average.

Small groups of resources with the same ability but different cost performance ratios are formed. If a large degree of alternatives is required more groups of the same ability are merged than in case of a low degree. The requests of the grid jobs are then distributed evenly among the available resource abilities. Next the due dates and budgets of the application jobs of each basic benchmark are adjusted in such a way that the heuristics of the first planning phase of GORBA (see also Sect. 4) just cannot produce violation free schedules. This means that there are a few application jobs which are a little bit too late or too costly or both. The resulting question for the experiments is, whether the subsequent GLEAM run can overcome these violations or not as explained in Sect. 5.

The basic units of resources and application jobs are dimensioned in such a way that they can be scaled easily in the steps of factors of 1.5, 2, 2.5, 3, and so on resulting in the 23 scheduling loads per benchmark class as shown in figures 2 and 3 or 4. All benchmarks have a resource set of the size of 10% of the number of grid jobs.

If the scaling of each basic benchmark is done well all benchmarks of a class should have almost the same amount of *time* and *cost benchmark reserves*. The mean values of these reserves per benchmark class are shown in Table 1 together with their minima and maxima. As the values deviate per class only slightly, we can conclude that the benchmarks of a class differ mostly in size and not in the two key properties *budget* and *time reserves* as desired. The small differences between the classes can be explained by the stochastic elements of the construction of the basic benchmarks and the varying flexibility for scheduling when using small or large values for *R* and *D*.

In the first step described so far a total of 92 benchmarks for all four classes is constructed on the basis of an empty grid. In the second step GORBA is run three times for every benchmark and the best schedule is taken. This is done to avoid distortions by accidental poor schedules, as they can occur in a stochastic scheduler (see also Sect. 4.2). The best schedule is processed until 10% of the grid jobs are finished

Table 1: Averages $\overline{t_{res}}$ and $\overline{c_{res}}$ of the time and cost benchmark reserves $\overline{t_{res,bm}}$ and $\overline{c_{res,bm}}$ together with their minima and maxima per benchmark class. The reserve values are given as percentages of the corresponding time frames and budgets.

benchmark class	time benchmark reserve [%]			cost benchmark reserve [%]		
	$\min(\overline{t_{res,bm}})$	$\overline{t_{res}}$	$\max(\overline{t_{res,bm}})$	$\min(\overline{c_{res,bm}})$	$\overline{c_{res}}$	$\max(\overline{c_{res.bm}})$
lRlD	47.2	48.9	50.1	27.3	27.7	28.1
sRlD	53.7	55.9	58.3	25.2	25.9	26.7
lRsD	57.2	59.2	62.1	22.3	23.3	24.1
sRsD	63.4	64.4	65.5	20.8	21.5	22.0

and then the same amount of new grid jobs is added to the scheduling task. Together with the active allocations of the old schedule this creates a rescheduling benchmark.

4 Scheduling algorithms

Scheduling is done in GORBA in two stages. In the first stage two sets of heuristics produce schedules, which are used in the second stage to seed the initial population of a subsequent run of the Evolutionary Algorithm GLEAM (General Learning Evolutionary Algorithm and Method) [1], [2]. The second stage is described in Sect. 4.2.

4.1 Heuristic scheduling and rescheduling

The first set of heuristics is aimed at scheduling jobs to an empty grid while the second one is designed for replanning. The first set works with the following heuristic rules:

earliest due time: grid jobs of the application job with the earliest due time first,

shortest working time of grid job: grid jobs with the shortest working time first,

shortest working time of application job: grid jobs of the application job with the shortest working time first.

Resources are allocated to the resulting three grid job sequences by applying each of the following three resource allocation strategies (RAS) yielding nine schedules in total.

RAS-1: Use the fastest of the earliest available resources for all grid jobs.

RAS-2: Use the cheapest of the earliest available resources for all grid jobs.

RAS-3: Use RAS-1 or RAS-2 for all grid jobs of an application job according to its time/cost preference given by the user.

All grid jobs which have not been started at the replanning event or will not be started within three minutes after it are subject to replanning. They are put into the same sequence as they have been in the old schedule. All other grid jobs affect the scheduling as already existing resource allocations, which must be observed by the scheduler. If the replanning event is the arrival of new grid jobs, as it is the case for our investigation, the new grid jobs are added to the sequence of old ones by applying each of the three heuristics for sequence generation described above. This yields three sequences, while for other replanning events only the one of the old schedule is produced. Again, the three RAS are applied, resulting in either nine or three replanning schedules. The best of them is regarded as the preliminary result, which is hopefully improved by the subsequent GLEAM run.

The effect of the heuristics for scheduling was investigated in [7], while those for replanning were assessed in [8].

4.2 Evolutionary scheduling

Evolutionary Algorithms (EA) mimic the principles of biological evolution like *heredity*, or *survival of the fittest* to improve already found solutions or to generate new ones. EAs are stochastic algorithms by nature and do not guarantee to find the optimum in reasonable time. On the other hand they are well known to come up with good or at least feasible solutions comparatively fast. Our GLEAM based scheduler is described in detail in [7] and [9]. We give here only a summary of the features important to the task at hand. As GLEAM is an elitist EA the best individual will always survive. Hence, the best heuristic result can only be improved.

A chromosome consists of genes, each of which represents a grid job and the gene sequence determines the scheduling sequence of the corresponding grid jobs. Resource assignments are made by one of the three RAS (see Sect. 4.1), which is selected by an additional special gene. This coding restricts the search space to some extent, which is accepted because it yields better improvements within the given time frame of three minutes for planning [7]. Possible precedence violations of the grid jobs caused by mutations or crossover are cured by a phenotypic repair. For more details the interested reader is referred to the given literature.

5 Experimental results

Due to the stochastic nature of EAs several runs have to be accomplished to compare e.g. different settings of an EA. The comparison is then based on the averages of e.g. the achieved fitness values. To obtain sound results with reasonable effort we set the number of GLEAM runs to 50 per benchmark and parameter setting of the EA as described later.

We compare the heuristic results with those from the subsequent GLEAM run by their *success rate* and the *fitness gain* attained from the GLEAM run. A run is considered successful if the generated schedule is free of budget or due date violations (cf. Sect. 3.2). The *success rate* is the percentage of successful runs in relation to all 50 runs. The *fitness gain* is the difference between the best heuristic result and the average of all GLEAM runs. A fitness gain can be regarded as significant, if the best heuristic result is outside of the confidence interval of the corresponding set of GLEAM runs. This is always the case for our experiments, even if a confidence value as high as 99.9% is taken as the basis.

The course of the success rate with growing numbers of grid jobs and resources is shown in Figure 1 for all scheduling loads, where at least one benchmark yields a success rate greater than zero. It is not surprising that the rates drop differently depending on the benchmark class. But within a class a more or less continuous decline was expected from the homogeneous values of the two benchmark reserves. This is obviously not the case as e.g. the results of *sRsD-400*, *lRsD-500*, *lRlD-700* or *lRlD-900* show. In these examples we have success rates below 100% followed by benchmark sizes, which can be scheduled always successfully. Of course, the tendency of a decreasing success rate with growing loads exists, but it is much more disrupted as expected. The reason for these distortions may be the stochastic elements of the GLEAM run that produced the first schedule, which was interrupted by the rescheduling event to create a replanning benchmark (cf. Sect.

Figure 1: Success rates for all grid job loads yielding successful runs.

3.2). Or the two benchmark reserves do not have the significance as expected since they are based on the assumption of an empty grid. As explained in Sect.3.1 they were calculated per application job without taking the competition from other application jobs for the resources into account. But this is true for all other characteristic numbers calculated prior to scheduling too. The analysis of the fitness gain will give some reasons that the influence of the stochastic GLEAM run for preparing the benchmarks has a greater impact than originally expected.

The results of the fitness gain are shown in two diagrams for better representation. Figure 2 starts with the results for up to 2000 grid jobs. Large improvements in the magnitude of 2500% contrast to small ones of 11.6% or 69% at minimum. Big enhancements are possible if the heuristics performed poorly. An example for this is the *sRlD-200* benchmark, for which the best heuristic yielded a normalised fitness value of 2.35. Normalisation is done in the range of 0 to 100. The average GLEAM result is 61.0, which is a growth of 2496%. On the other hand *lRlD-500* has a heuristic result of 37.0 which is improved to 62.5, an increase of 69%. Thus, GLEAM and the heuristics frequently complement each other. The *lRlD-200* benchmark with its improvement rate of 11.6% is an exceptional case because the replanning task meets the cost and time constraints, while its basic plan did not. Thus only comparable small improvements of the fitness value are possible. This is the only benchmark with such an exception and all others started with solutions below the level of violation free schedules as desired. Both figures also show that successful runs have small confidence intervals, which means that their fitness values deviate only slightly from each other. The much larger confidence intervals of only partially successful benchmarks are based mostly on the high fitness devaluations due to the penalty functions activated by violations of time and/or cost limits.

The example of benchmark *lRlD-200* demonstrates the possible impact of the stochastic improvement of the basic schedule by GLEAM, followed by an execution of some grid jobs and the arrival of new ones. It may change the tightness of cost and time limits drastically. This is a clear indication that the observed inhomogeneous decline of the success rate is due to the creation of the rescheduling benchmarks and less because of a lack of significance of the *cost* and *time benchmark reserves*.

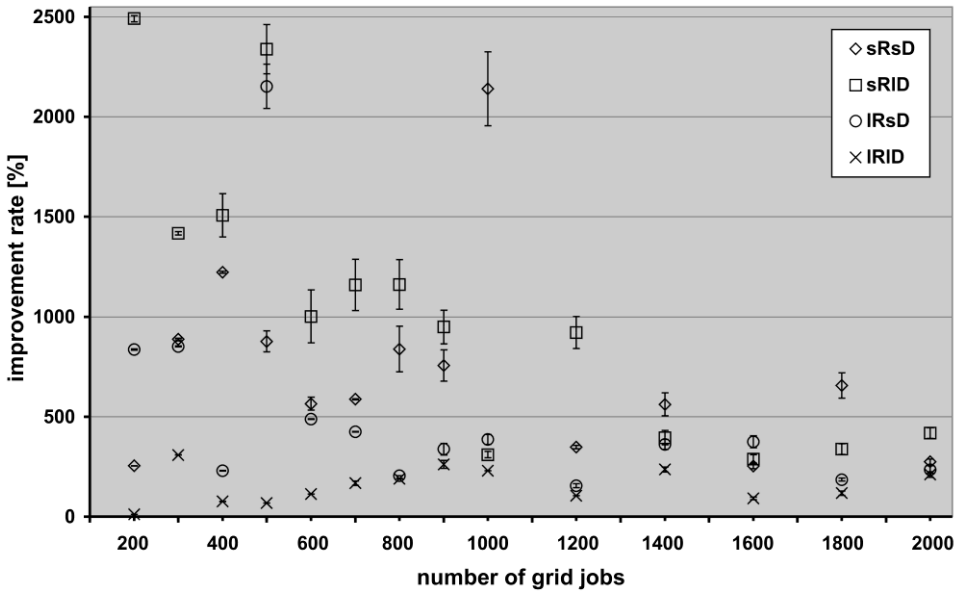


Figure 2: Improvement rates and confidence intervals (95% confidence) for all grid job loads of up to 2000 jobs for all four benchmark classes.

Figure 3 shows the improvement rates with increasing scheduling loads beyond the achievability of schedules free of budget or due date violations. The tendency of decreasing improvement rates with growing loads continues as expected. In the end, rates between 33% and 133% can still be achieved as the results for bench-

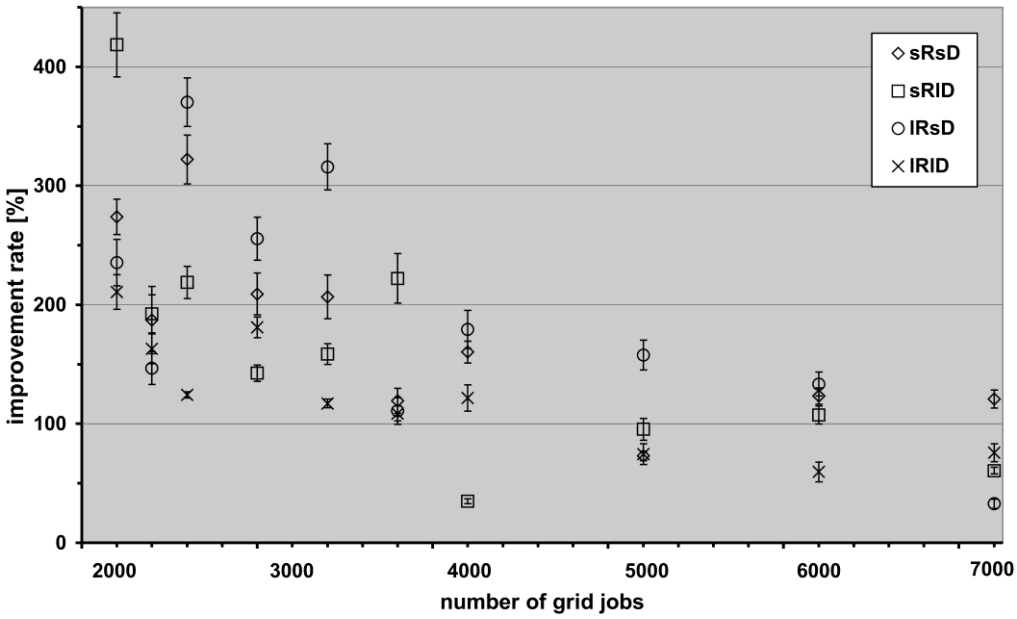


Figure 3: Continuation of Fig.2 for all loads from 2000 grid jobs and 200 resources to 7000 and 700 respectively.

mark sizes of 6000 or 7000 grid jobs show. As these runs still reduce the amount of accumulated budget or due date overruns as well as the number of affected application jobs, it is useful to maintain the EA run.

This raises the question whether further improvements for more than 7000 grid jobs and 700 resources are possible with the given hard- and software. This issue is close related to the number of evaluations which can be performed within the three minutes time frame available for EA improvement. The time required to assess a schedule depends on the size of the allocation matrix, on the number of grid jobs to be processed and on the number of alternatively available resources, among which a selection must be made. As the matrix lists the time of allocation, suspension and availability for all resources, its size depends on the number of resources as well as on the time scale of possible makespans. Thus, the processing time of a schedule increases with

- the number of all resources
- the average number of resource alternatives per grid job (i.e. parameter *R* of a benchmark)
- the range of possible makespans
- the number of grid jobs

Consequently, the number of processible evaluations drops with an increasing number of grid jobs as Figure 4 shows. And the impact of the benchmark parameter *R* can be seen clearly: small values of *R* correspond with less resource alternatives which can be checked faster than many and therefore, more evaluations can be performed as with large values of *R*.

Regardless of the degree of resource alternatives *R* the number of evaluations per GLEAM run drops below 5000 for scheduling loads of more than 5000 grid jobs

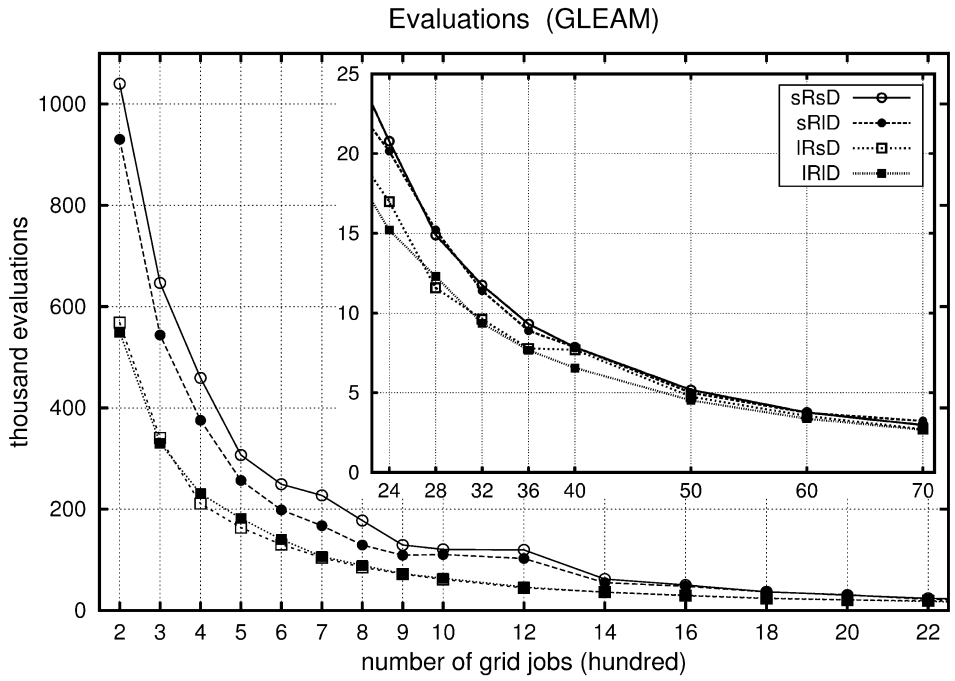


Figure 4: Number of evaluations achievable in three minutes for increasing numbers of grid jobs and resources and for all four benchmark classes.

and 500 resources. But since EAs need an application-dependent number of trials to the search space of at least some thousands to gain a relevant improvement, we cannot expect further advancements of the heuristic schedules for considerably greater loads. To overcome this, either a tuning of the scheduling and assessment software as well as faster hardware might help to some extend or a parallelisation of the EA itself, see e.g. [5] or [2].

6 Conclusion and outlook

We have introduced two new benchmark parameters, the *time* and *cost benchmark reserves*, for the problem of rescheduling large quantities of grid jobs based on advanced reservations. The parameters are aimed at determining the tightness of user given due dates and budgets, two properties which have a great impact on the complexity of a scheduling task. New benchmarks for rescheduling were constructed so that only very small deviations of these figures could be observed while scaling them from 200 grid jobs to 7000 and 20 resources to 700. The experiments showed a greater disruption of quality decrease with growing loads as expected from the homogeneity of the two benchmark parameters. This is probably due to some stochastic elements in the construction process of the rescheduling benchmarks as described. The experiments also showed the limits of EA based scheduling as a supplement of heuristic planning when the load exceeds 5000 grid jobs and 500 resources and the processing time is strictly limited. Especially the population size has to be adjusted well for loads greater than about 600 grid jobs. As shown by Jakob [6], the extension of an EA to a Memetic Algorithm by adding local search to the offspring creation process can significantly reduce the range of well performing population sizes as well as improve the performance considerably.

7 References

[1] Blume, C. and Jakob, W., 2002. GLEAM – An Evolutionary Algorithm for Planning and Control Based on Evolution Strategy. In Cantú-Paz, E. (ed), *Conf. Proc. GECCO-2002*, vol. LBP, p. 31-38.

[2] Blume, C. and Jakob, W., 2009. *GLEAM – General Learning Evolutionary Algorithm and Method: ein Evolutionärer Algorithmus und seine Anwendungen*. In German. Series of the Inst. for Appl. Comp. Science / Automation Technology, Karlsruhe Institute of Technology, vol 32, KIT Scientific Publishing. Karlsruhe.

[3] Brucker, P., 2004. *Scheduling Algorithms*. Springer. Berlin.

[4] Brucker, P and Knust, S., 2006. *Complex Scheduling*. Springer. Berlin.

[5] Gorges-Schleuter, M., 1990. *Genetic Algorithms and Population Structures - A Massively Parallel Algorithm*. (PhD Thesis). Dept. of Comp. Science, University of Dortmund, Germany.

[6] Jakob, W., 2010. A general cost-benefit-based adaptation framework for multimeme algorithms. *Memetic Computing*, (2)3: 201-218.

[7] Jakob, W., Quinte, A., Stucky, K.-U. and Süß, W., 2008. Fast Multi-objective Scheduling of Jobs to Constrained Resources Using a Hybrid Evolutionary Algorithm. In Rudolph, G. et al. (eds.) *Conf. Proc. PPSN X*, LNCS 5199, pages 1031-1040. Springer. Berlin.

[8] Jakob, W., Quinte, A., Stucky, K.-U. and Süß, W., 2010a. Fast Multi-objective Rescheduling of Grid Jobs by Heuristics and Evolution. In Wyrzykowski, R. et al. (eds.), Par. Proc. and Appl. Math., 8th Int. Conf. PPAM 2009, LNCS 6068, pages 21-30. Springer. Berlin.

[9] Jakob, W., Möser, F., Quinte, A., Stucky, K.-U. and Süß, W., 2010b. Fast Multi-objective Rescheduling of Workflows Using Heuristics and Memetic Evolution. Scalable Computing: Practice and Experience, 11(2):173-188.

[10] Kurowski, K., Nabrzyski, J., Oleksiak, A. and Węglarz, J., 2006. Scheduling Jobs on the Grid - Multicriteria Approach. *Computational Methods in Science and Technology* 12(2): 123-138.

[11] Kurowski, K., Oleksiak, A. and Węglarz, J., 2010. Multicriteria, multi-user scheduling in grids with advanced reservation. *Journal of Scheduling*, 13(5):493-508.

[12] Mika, M., Waligóra, G., and Węglarz, J., 2011. Modelling and solving grid allocation problem with network resources for workflow applications. *Journal of Scheduling*, 14(3):291-306.

[13] Möser, F., Jakob, W., Quinte, A., Stucky, K.-U. and Süß, W., 2010. An Assessment of Heuristics for Fast Scheduling of Grid Jobs. In Cordeiro, J. Virvou, M. and Shishkov, B. (eds.), *Proceedings of the Fifth Int. Conf. on Software and Data Technologies (ICSOFT 2010)*, vol. 1, pages 184-191. SciTePress.

[14] Sakellariou, S., Zhao, H., Tsiakkouri, E. and Dikaiakos, M.D., 2007. Scheduling Workflows with Budget Constraints. In Gorlatch, S. and Danelutto, M. (eds.), *Integrated Research in GRID Computing (CoreGRID Integration Workshop 2005, Selected Papers)*, CoreGRID Series, pages 189-202. Springer. Berlin.

[15] Setämaa-Kärkkäinen, A., Miettinen, K. and Vuori, J., 2006. Best Compromise Solution for a New Multiobjective Scheduling Problem. *Computers & OR*, 33(8):2353-2368.

[16] Süß, W., Quinte, A., Jakob, W. and Stucky, K.-U., 2007 Construction of Benchmarks for Comparison of Grid Resource Planning Algorithms. In Filipe, J. et al. (eds.): *ICSOFT 2007, Proc. of the 2nd ICSOFT*. Vol. PL/DPS/ KE/WsMUSE, pages 80-87. Inst.f. Systems and Techn. of Inf., Control and Com., INSTICC Press.

[17] Wieczorek, M., Hoheisel, A. and Prodan, R., 2008. Taxonomy of the Multi-criteria Grid Workflow Scheduling Problem. In Talia, D., Yahyapour, R. and Ziegler, W. (eds.), *Grid Middleware and Services - Challenges and Solutions*, p. 237-264. Springer. NY.

[18] Xhafa, F., Alba, E., Dorronsoro, B., Duran, B. and Abraham, A., 2008. Efficient Batch Job Scheduling in Grids Using Cellular Memetic Algorithms. In Xhafa, F. and Abraham, A. (eds.), *Metaheuristics for Scheduling in Distributed Computing Environments*, pages 273-299. Springer. Berlin.

[19] Yu, J. and Buyya, R. 2006. A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms. In *Workshop on Workflows in Support of Large-Scale Science, Proc. of the 15th IEEE Int. Symp. on High Performance Distributed Computing (HPDC 2006)*, IEEE CS Press.

A Novel Cooperation Strategy in Artificial Immune Network for Multimodal Optimization

Ehsan Biria[1] **and Kamran Zamanifar**[2]
Department of Computer Engineering, University of Isfahan, Isfahan, Iran
[1]ehsanbiria@gmail.com, [2]zamanifar@eng.ui.ac.ir

Abstract - *In recent years, among many branches of artificial immune systems (AIS), artificial immune networks could attract many researchers' interest in solving the engineering problems. Considering a noticeable drawback in the original optimization version of artificial immune network (opt-aiNet) that is the lack of any cooperative interaction between cells, several methods are proposed to introduce cooperation in the population for better multimodal optimization. Most of these methods emphasize on moving new cells towards the best experience of population. On the contrary, for a better exploration, a new strategy, named Reverse Cooperation, is introduced in this paper which applies a new Repulse operator to repel new cells from the discovered peaks. This strategy is accompanied with two modifications in hypermutation operator and diverse introduction process in order to increase the number of optima that is found and accelerate the convergence of population. The new algorithm, as compared with the others, could optimize multimodal functions more effectively and efficiently.*

Keywords: Artificial immune network, Multimodal function optimization, Reverse cooperation, Repulse operator

1 Introduction

Multimodal functions are problems that may have several global optima, i.e. several points where the values of the objective function are equal to the global optimum value. Furthermore, there exist some local optima in the function's landscape. These local optima may not be undesirable necessarily. Since algorithmic modeling of a real-world problem often involves some abstractions, finding all global optima and the utmost number of local optima would supply multiple choices for decision makers and engineers [1].

Traditional optimization methods, like the quasi-Newton method, emphasize on accurate and exact computation, so they cannot optimize multimodal functions efficiently. During recent years, several bio-inspired techniques have been proposed to solve multimodal optimization problems. These techniques provide more robust and efficient solutions for multimodal optimization. However, they may fail to find all global and local optima. Some bio-inspired techniques which are engaged in multimodal optimization are the Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Artificial Immune Systems (AIS) [2].

Artificial Immune System, also called Immunological Computation, is a field of research that mimics the biological immune system behavior in human and animal for developing computational models. Biological immune system is a complex, adaptive and distributed learning system with several mechanisms as a defensive means against pathogens [3]. Hence AIS attracts the researcher's attention to apply to various engineering application areas, including optimization problems [4].

The remainder of the paper is organized as follows. In section 2, some of the related works that have applied AIS in multimodal optimization are reviewed and the authors' opinions on a better algorithm are presented. In section 3, a reverse cooperative immune network is proposed in detail. In section 4, several experimental results are discussed and the performances of the new algorithm with the two others are compared. Finally, section 5 concludes the paper.

2 Related works

The clonal selection principle [5] is applied to explain the basic features of an adaptive immune response to an antigenic stimulation. This principle establishes the idea that only those cells that recognize the antigens are selected to be proliferated. The selected cells are subject to an affinity maturation process where their affinity to the selective antigens is improved. The CLONALG algorithm is primarily derived from the clonal selection principle with the theory of evolution in order to perform pattern recognition tasks and then it is adapted to solve optimization tasks, emphasizing multimodal and combinatorial optimization problems [6].

An artificial immune network, called aiNet, is presented to accompany the clonal selection and immune network theories [7]. The aiNet algorithm introduced for data compression and data clustering first and then was further extended to solve multimodal optimization problems. The optimization version of aiNet, called opt-aiNet, is a well-known immune network with some intriguing features for optimization, such as dynamic adjustable population size, exploration and exploitation of the search space and capability of locating and maintaining multiple optima solutions [8]. The opt-aiNet consists of a network of cells which evolves in two phases: local search phase, to exploit better optima and global search phase, to explore the function

landscape. At local exploitation the cells continuously pass through clonal expansion, fitness maturation and elitist selection steps while reaching a stable state. Afterwards, at global exploration the network cells undergo suppression operation that removes cells with lower fitness values and similar to those with higher fitness values. Fundamentally, the network suppression describes interactions within network cells as competition correlation to prevent their clustering on a single optima. After suppression, a number of new random cells are generated to introduce more diversity in the network population and then, the local exploitation process restarts [9].

There are some drawbacks in opt-aiNet, such as lack of cooperation between network cells and unused cells' experiences in previous iterations. Moreover, new random cells can land in already exploited landscape which makes these cells inefficient and leads in the waste of resources. Several other algorithms are proposed on the base of opt-aiNet in order to reduce its drawbacks or improve the capabilities [2].

A Hybrid Artificial Immune Network (HaiNet) is proposed in [10], inspired not only from biological immune system but also from the group behaviors of particle swarm optimization. The swarm learning technique in PSO is applied in the hypermutation operation of the artificial immune network, in a manner where each cell can use its own best encountered position and the best encountered position among all the cells, in order to speed up the optimization process. The addition of swarm learning aims at faster convergence to global optima, but reduces the number of known local optima.

In [9] another Cooperative Artificial Immune Network (CoAIN) model is proposed for multimodal function optimization. The CoAIN uses cooperative strategy inspired by particle swarm behavior to explore and exploit searching space efficiently and effectively. In this model, a new distinct cooperative step is introduced after the suppression step. This step enables each network cell to cooperate with the others as a particle flying around in a searching space for adjusting its position according to its own experience and the experience of other particles.

Another algorithm is a Predication-based Immune Network (PiNet) proposed in [2]. To explore and exploit the searching space efficiently and effectively, the PiNet uses a cooperative strategy between homologous antibodies, that is to say, the next position of a cell is related not only to its position but also its parent antibody. Moreover, the selection probability of a new random cell changes dynamically according to the sum of affinities between memory cells and itself [2].

In this paper, a Reverse Cooperative Artificial Immune Network called RC-aiNet, is proposed to improve both search ability as well as search speed. The RC-aiNet introduces a new strategy for inter-cellular cooperation, which is called reverse cooperation. Here, younger cells are repulsed by memory cells instead of inducing younger cells to move towards the best encountered position. The reverse cooperation increases the chance for the new cells to investigate unexplored sections of solution space.

Here, some modifications are proposed in the hypermutation operation of the affinity maturation process, which establishes acceleration in local exploitation. In this manner, the position of a mutated cell, in addition to its current position, is related to the previous movements at prior iterations. It should be added that the RC-aiNet proposes a mechanism in the diverse introduction step (update step in PiNet) which changes the probability of different zones in solution space to embrace new cells. This new mechanism increases the probability of placing the new cells in unexplored landscapes, thereupon causes better exploration.

The experiments show that compared to PiNet and opt-aiNet, the suggested mechanisms in new algorithm lead to optimizing multimodal functions more effectively and efficiently.

3 Artificial Immune Network with Reverse Cooperation

In order to present RC-aiNet algorithm, the following list of terminology is assumed:

- Antibody cell: an individual of the network population which indicates a point in function landscape; each cell is presented as a real-valued vector in an Euclidean shape-space
- Antigen: an objective multimodal function to be optimized
- Fitness: interaction between antibody cells and antigen; the cell fitness is the value of objective function when evaluated for the given cell
- Affinity: interaction between two antibody cells that will be calculated as Euclidean space between them
- Clone: the offspring that are exact copies of a cell; the offspring cells will further suffer hypermutation for become variations of their original cell

3.1 Procedure of the RC-aiNet algorithm

The main process of the new algorithm can be described through the following 9 steps:

Step 1. *Initialization*: Randomly initialize a population of N_I cells.

Step 2. *Reverse cooperation*: The memory cells of the last global exploration phase repulse younger cells in their neighborhood. In other words, the new cells are repelled by the neighbor memory cells. The neighbor cell is one that its affinity is less than repulse threshold σ_{rep}.

Step 3. *Clone expansion*: Generate N_c clones for each network cell.

Step 4. *Affinity maturation*: Compute a mutation center for each clone according to its prior movement. A mutation center is a point that mutation performs in the centrality of it instead of the original cell's coordinates. Afterwards mutate the clones in proportion to the fitness of their original cell.

Step 5. *Elitist selection*: Determine the fitness of all mutated clones and in each subpopulation of clones which are copied from an original cell, select one with the highest fitness.

Step 6. *Exploitation convergence criteria:* Calculate the average fitness of the network population. If the average fitness does not change significantly in comparison to the previous iteration, then continue, otherwise return to step 2.

Step 7. *Suppression*: Compute the affinity between all cells in the network, next eliminate all but the highest fitness of the cells whose affinities are less than the suppression threshold σ_{sup}, then determine the number of remaining cells as memory cells.

Step 8. *Termination criteria*: If the number of memory cells does not vary in comparison to the prior global exploration phase, the optimization process is completed, otherwise continue. Other stopping criterion is to meet the maximum number of evaluated points in search space.

Step 9. *Diverse introduction*: To generate *d%* new cells, first assume that the objective function's landscape as a crisscrossed grid where the number of squares in the grid is twice the number of the memory cells. Second, rank the squares in inverse proportion to the number of the memory cells in each square and its adjacent squares and then create a roulette wheel according to these ranks. Third, select a square from the wheel and randomly generate a new cell inside the bounds of selected square followed by uniform distribution. Fourth, reduce the rank of the selected square and repeat creating roulette wheel, selecting a square and generating a new cell inside again. And finally, add the new cells to current population and return to step 2 to restart local exploitation.

Since RC-aiNet is extended from opt-aiNet, accordingly, the new algorithm consists of two phases: local search, to exploit feasible solutions and global search, to explore solution space. Steps 2 to 6 above constitute the local exploitation where at each iteration a population of cells is optimized locally through the repetition of clone expansion, fitness hypermutation and elitist selection. Steps 7 to 9 are involved in the global exploration process by introducing a number of new cells probably in unexplored zones first and then the local search restarts. Although the reverse cooperation step is involved in local exploitation phase, it really complements the global exploration process to examine unsought zones.

3.2 Repulse operator for reverse cooperation

The reverse cooperation subjective is using the memory cells' experience in order to avoid locating the peaks which are found in prior iterations. Therefore, a new operator is introduced called the repulse operator. This operator repels the new generated cells from old memory cells, which increases the chance for the new cells to find some unprecedented optima. To perform repulsing operation for a population cell denoted as a_i, the set of its neighbor memory cells denoted as R_i should be determined as equation (1), where a_i is a population cell, F is the set of all memory cells, σ_{rep} is repulse threshold and $\| a_i - f_j \|$ is the affinity of two cells calculated as Euclidean distance between them.

$$R_i = \{ f_j \mid f_j \in F, \| a_i - f_j \| < \sigma_{rep} \} \quad (1)$$

Next, for each neighbor memory cell f_i, a partial repulse vector $\vec{V}_j$ is defined as equation (2).

$$\vec{V}_j = \frac{\overrightarrow{a_i - f_j}}{\| a_i - f_j \|} \times (\sigma_{rep} - \| a_i - f_j \|) \quad (2)$$

It is clear that $\frac{\overrightarrow{a_i - f_j}}{\| a_i - f_j \|}$ is a unit vector in the direction from f_i to a_i and the phrase $(\sigma_{rep} - \| a_i - f_j \|)$ determines the length of the partial repulse vector, in inverse proportion to affinity of f_i and a_i. Afterward, a repulse resultant vector for a_i, named $\overrightarrow{Rv_i}$, is created from these partial vectors and then a_i moves based on the repulse resultant vector, according to equation (3).

$$a_i' = a_i + (\overrightarrow{Rv_i} \times 2^{-It(a_i)}) \quad (3)$$

In equation (3), a_i' is repulsed cell of a_i and $It(a_i)$ is the age of a_i that represents the number of iterations that a_i has been through. The phrase $2^{-It(a_i)}$ decays repulsion, so that the older the cell a_i, the weaker the repulsing. Consider that memory cells will not be repulsed at all.

A repulsed cell should not exceed the bounds of the objective function. Also, a factor named rw_i computes which displays the weakness of repulsing for a_i, to use in mutation operator.

3.3 Accelerating in hypermutation operation

To increase the speed of the local exploitation, a technique is suggested in mutation operator which uses the experience of a cell in prior iterations to accelerate its movement towards an optimum. For this purpose, a mutation center is defined for each cell as a point that mutation performs in the centrality of it, instead of original cell's coordinates. One of the factors involved in the mutation center computation is the cell's last movement.

The mutation operator is defined by the following equation (4), where c_i is a mutated cell of the cell b_i, p_i is the mutation center calculated as equation (5), parameter β is to

control the decay of mutation, b_i' is the parent of b_i in the last iteration, $f^*(b_i)$ is the fitness of b_i normalized in the interval [0,1], $\Delta f^*(b_i)$ is the fitness variation between the parent and the son cell in two continuous populations which is restrained up to 1 as equation 6, rw_i is the repulse weakness factor computed in reverse cooperation step, c_1 is a constant to regulate tendency to previous fitness variation, $N(0,1)$ is a Gaussian random variable of zero mean and standard deviation $\sigma=1$ and pf_i is a factor defined to regulate the consideration of previous experiment for the mutation center calculation.

A mutation is only accepted if the mutated cell c_i is within the objective function bounds.

$$c_i = p_i + \left(\frac{1}{\beta}\times\exp(rw_i\times c_1\times\Delta f^*(b_i) - f^*(b_i))\right)\times N(0,1) \quad (4)$$

$$, \quad p_i = b_i + \left(rw_i \times pf_i \times (\overrightarrow{b_i - b_i'})\right) \quad (5)$$

$$\Delta f^*(b_i) = \frac{f(b_i) - f(b_i')}{\underset{1\le j\le N}{Max}\{f(b_j) - f(b_j')\}} \quad (6)$$

The pf_i factor is defined as equation (7):

$$pf_i = \begin{cases} \circ & if \quad It(b_i)=\circ \\ \frac{1}{2} & if \quad It(b_i)=1 \\ 1-\left(\frac{angle_i}{180}\right)^2 & if \quad It(b_i)>1 \end{cases} \quad (7)$$

where, $angle_i = Angle(\overrightarrow{p_i' - b_i'}, \overrightarrow{b_i - p_i'})$ (8)

and p_i' is the mutation center of b_i' in last iteration. The phrase $angle_i$ refers to the angle between two vectors, where $\overrightarrow{p_i' - b_i'}$ determines the algorithmic desired direction in the last iteration and $\overrightarrow{b_i - p_i'}$ determines the direction of real movement in the last iteration. Therefore, a straight movement in prior iteration will increase the speed and any deviation leads to reduction in speed.

3.4 Purposeful mechanism for generating cells

In the diverse introduction step, situating the new random cells at unsought zones of landscape causes the power of exploration to increase. To generate new random cells followed by non-uniform distribution, first the objective function's landscape is assumed as a crisscrossed grid with the number of squares twice the number of memory cells. In this grid, every memory cell contained in a square scores a constant ns_1 for that square and scores a constant ns_2 for the adjacent squares; hence, the more optima within or close to a square result in more scores for that square. Next, create a roulette wheel in inverse proportion to the scores. Afterwards, algorithm repeats the following cycle: first, it selects a square from the wheel; second it generates a new uniform random cell in the bounds of the selected square and adds constant score ns_3 to the selected cell and finally it creates the roulette wheel using modified score again and the cycle repeats while d% new cells are generated. Here, the constants $ns_1=2$, $ns_2=1$ and $ns_3=4$ are assumed.

4 Experiments and Results

4.1 Experimental protocols

The proposed RC-aiNet is coded in Matlab 2008 and is compared to opt-aiNet and PiNet. To examine the search ability and the performance of the proposed RC-aiNet, 9 benchmark functions with different complexities are used [2]. The benchmark functions are listed in Table 1.

Table 1. Benchmark functions for multimodal optimization

Notation	Name	Function	Interval	Number of local optima
F_1	Multi	$F_1 = x_1\sin(4\pi x_1) - x_2\sin(4\pi x_2+\pi)+1$	$x_1, x_2 \in [-2, 2]$	100
F_2	Schaffer	$F_2 = 0.5 - \frac{\sin^2(\sqrt{x_1^2+x_2^2}) - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$	$x_1, x_2 \in [-10,10]$	infinite
F_3	Roots	$F_3 = \frac{1}{1+\lvert z^6-1\rvert}$, $z = x_1 + ix_2$	$x_1, x_2 \in [-2,2]$	6
F_4	Sinc	$F_4 = \frac{\sin(\lvert x_1-10\rvert + \lvert x_2-10\rvert)}{\lvert x_1-10\rvert + \lvert x_2-10\rvert}$	$x_1, x_2 \in [-20,20]$	infinite
F_5	Alex	$F_5 = -(x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7) - \sin(x_1^2 + x_2^2)$	$x_1, x_2 \in [-4,4]$	4
F_6	Rastrigin	$F_6 = 20 + x_1^2 + x_2^2 - 10(\cos(2\pi x_1) + \cos(2\pi x_2))$	$x_1, x_2 \in [-2.5,2.5]$	36
F_7	Shubert	$F_7 = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \times \sum_{i=1}^{5} j\cos[(j+1)x_2 + j]$	$x_1, x_2 \in [-10,10]$	761
F_8	Camel	$F_8 = -4x_1^2 + 2.1x_1^4 - \frac{1}{3}x_1^6 - x_1x_2 + 4x_2^2 - 4x_2^4$	$x_1, x_2 \in [-3,3]$	6
F_9	Rastrigin2	$F_9 = \cos(18x_1) + \cos(18x_2) - x_1^2 - x_2^2$	$x_1, x_2 \in [-2,2]$	169

Table 2. Specialized parameters for optimization of benchmark functions

Functions	Mutation strength β	Affinity suppression threshold σ_{sup}	Fitness improvement threshold λ_f (only for PiNet)	Repulse threshold σ_{rep} (only for RC-aiNet)	Search space S
F1	100	0.2	0.1	0.8	550,000
F2	100	0.2	0.1	0.8	550,000
F3	100	0.2	0.01	0.8	550,000
F4	100	0.2	0.1	0.8	1,000,000
F5	100	0.2	0.1	0.8	200,000
F6	200	0.1	0.1	0.4	400,000
F7	100	0.2	0.1	0.8	800,000
F8	100	0.5	0.1	0.8	100,000
F9	100	0.2	0.3	0.8	800,000

All of common parameters for RC-aiNet are set exactly the same as the values used in [2] for testing opt-aiNet and PiNet. So in all benchmarks, initial population size (N_I) is 20, cell cloning scale (N_c) is 10 and diverse introduction rate (d) is 40%. Some other parameters including mutation decay (β), affinity suppression threshold (σ_{sup}), maximum search space (S), fitness improvement threshold (λ_f only for PiNet) and repulse threshold (σ_{rep} only for RC-aiNet), according to [2], are specialized for each function as Table 2 shows. Each function test is repeated 50 times and the average and the standard deviation of results are calculated. The results of benchmarks optimization using RC-aiNet, opt-aiNet and PiNet are presented in Table 3. The results of the latter two algorithms are extracted from [2].

4.2 Comparison of the proposed algorithm with opt-aiNet and PiNet

The comparative factors used in evaluating the algorithm's efficiency include: success rate, the number of discovered optima, the generations of convergence and the number of function evaluations.

Success rate is the probability of finding a stable state of memory cells in limited number of function evaluations which is named as maximum search space. The RC-aiNet could record good success rates as well as PiNet and do much better than that of the opt-aiNet.

As described before, finding all global optima and the utmost number of local optima is an important aspect of the performance of multimodal optimization. With the help of reverse cooperation and purposeful diverse introduction mechanism, RC-aiNet performs better a global exploration in landscape which assures an increase in the number of discovered optima. According to Table 3, when compared with opt-aiNet and PiNet, the new algorithm on average finds 36% more optima than opt-aiNet and 16% more than PiNet. The number of generations to convergence has an effective reflection on algorithm speed. Using the accelerating mechanism in hypermutation causes a decrease in the number of iterations required for local exploitation.

Table 3. Optimization results of benchmark functions using the three algorithms

Function	Algorithm	Success Rate (%)	Local optima number		Generations of convergence		Function evaluation number	
			Avg	SD	Avg	SD	Avg	SD
F_1	opt-aiNet	96	91.1*	5.8*	441*	72*	370,657*	122,270*
	PiNet	**100**	94.9	3.4	219	27	187,374	34,881
	RC-aiNet	**100**	**99.5**	**0.7**	**154**	**15**	**140,038**	**21,565**
F_2	opt-aiNet	0	219.3	10.3	682	37	+550,000	**580**
	PiNet	0	279.5	9.5	317	17	+550,000	1320
	RC-aiNet	0	**394.3**	**9.0**	**243**	**14**	+550,000	36,581
F_3	opt-aiNet	100	6.1	0.54	542	326	79,705	43,179
	PiNet	100	**6**	**0**	191	136	35,051	20,512
	RC-aiNet	100	**6**	**0**	**72**	**32**	**11,645**	**4,197**
F_4	opt-aiNet	0	287.2	38.3	1040	62	+1,000,000	**947**
	PiNet	0	444.9	**27.3**	458	42	+1,000,000	1992
	RC-aiNet	0	**730.8**	34.0	**325**	**18**	+1,000,000	106,415
F_5	opt-aiNet	100	4	0	620	70	99,990	16,326
	PiNet	100	4	0	175	31	28,517	4659
	RC-aiNet	100	4	0	**77**	**5.4**	**12,488**	**851**
F_6	opt-aiNet	66	**35.97**	3.522	674	156	229,247	88,901
	PiNet	**100**	34.76	1.836	208	52	77,396	28,126
	RC-aiNet	**100**	34.90	1.200	144	29	53,325	13,869
F_7	opt-aiNet	0	538.4	32.480	485	20	+800,000	**1832**
	PiNet	0	596.94	17.875	239	12	+800,000	2364
	RC-aiNet	0	**753.4**	**3.618**	**214**	**6**	+800,000	68,524
F_8	opt-aiNet	100	5.94	0.240	491	68	82,933	8074
	PiNet	100	**5.98**	**0.141**	194	111	31,955	16,225
	RC-aiNet	100	**5.98**	**0.141**	**66**	**10**	**10,932**	**1293**
F_9	opt-aiNet	100	120.14	2.382	492	54	561,508	101,582
	PiNet	100	157.42	4.343	253	37	383,938	81,899
	RC-aiNet	100	**167.56**	**1.013**	**198**	**18**	**275,729**	**42,941**

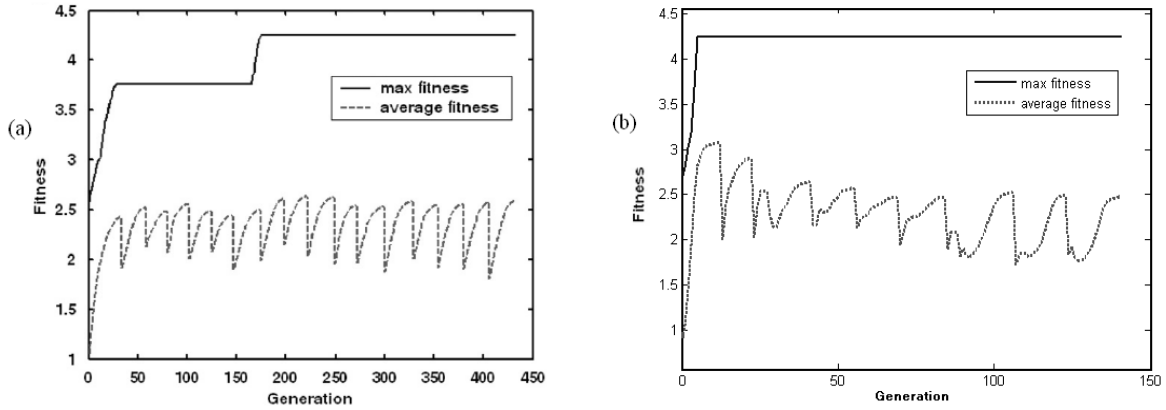


Figure 1. Comparison of maximum and average fitness of population on Multi Function. (a) opt-aiNet. (b) RC-aiNet.

The better exploration techniques lead to reduce the repetitions of global search, therefore the number of generations to convergence which consists of local and global search iterations in RC-aiNet is significantly less than opt-aiNet and PiNet. Figure 1 illustrates the average fitness and maximum fitness of cells population on F1 for opt-aiNet and RC-aiNet.

The number of function evaluation, called search space, is another criterion to estimate convergence speed. The new algorithm uses the information of prior movements of a cell to generate acceleration. This technique results in decreasing the required number of points to evaluate for each cell to reach to an optima. So the number of evaluations in RC-aiNet on average is 75% less than opt-aiNet and 49% less than PiNet. Therefore, the RC-aiNet could improve both the number of discovered optima and convergence speed.

It is clear in Table 3 that standard deviation in every criteria obtained by RC-aiNet usually is much less than that of the other two algorithms, especially in the number of optima and the generations of convergence. Figure 2 contains two box plots which compare the distribution of results in opt-aiNet, PiNet and RC-aiNet on F1. The closeness of the results indicates that RC-aiNet is more reliable and robust and is far from being affected by several executions with different initial populations.

5 Conclusion

In this paper, an improved algorithm named RC-aiNet is presented to optimize multimodal functions. For a better exploration in RC-aiNet, memory cells repel new generated cells. Also, the information of cells in current population and their parents in previous iterations has been utilized to affect exploitation speed. Compared with opt-aiNet and PiNet, the newly proposed algorithm is capable of improving algorithm performance in convergence speed and quality of solutions. Also, notable decrement in standard deviation of criteria is a reason for this newly proposed algorithm robustness.

For further studies, finding another dynamic diverse introduction technique based on a non-uniform probability distribution function may lead to an improved exploration. Some other techniques could introduce to reduce the influence of function shape in algorithm performance. Also algorithm's parameters could be adjusted more accurately to increase their efficiency.

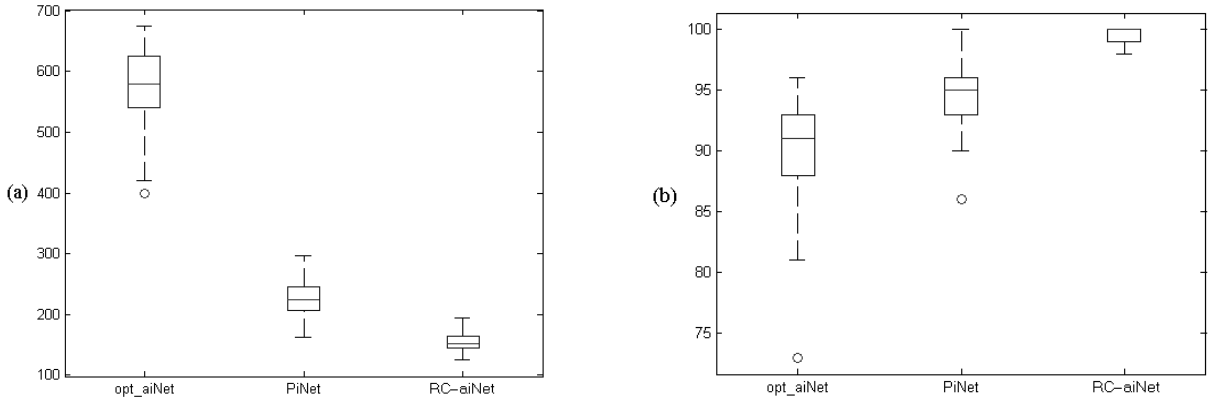


Figure 2. Two box plots to compare the three algorithms.
(a) The generations to convergence. (b) The number of discovered optima.

6 References

[1] A. Ahrari, M. Shariat Panahi and A.A. Atai, “GEM: a novel evolutionary optimization method with improved neighborhood search.” Applied Mathematics and Computation, vol. 210, issue 2, pp. 376–386, 2009.

[2] Q. Xu, L. Wanga and J. Si, “Predication based immune network for multimodal function optimization.” Engineering Applications of Artificial Intelligence, vol. 23, pp. 495-504, 2010.

[3] D. Dasgupta and L.F. Nino, Immunological Computation; theories and applications, New York: Auerbach, 2009.

[4] E. Hart and J. Timmis, “Application areas of AIS: the past, the present and the future. ” Applied Soft Computing, vol. 8, issue 1, pp. 191–201, 2008.

[5] F.M. Burnet, The Clonal Selection Theory of Acquired Immunity, Cambridge University Press, 1959.

[6] L.N. De Castro and F.J. Von Zuben, “Learning and optimization using the clonal selection principle.” IEEE Trans. on Evolutionary Computation, vol. 6, pp. 239–251, 2002.

[7] L.N. De Castro and F.J. Von Zuben, “aiNet: An artificial immune network for data analysis”, In Data Mining: A Heuristic Approach, by H.A. Charles, R.A.S. Newton and S. Abbas (Eds.), Hershey, PA: Idea Group Publishing, pp. 231–259, 2001.

[8] L.N. De Castro and J. Timmis, “An artificial immune network for multimodal function optimization. ”, Proc. IEEE Congresson Evolutionary Computation, 2002.

[9] L. Liu and W.B. Xu, “A cooperative artificial immune network with particle swarm behavior for multimodal function optimization.” Proc. IEEE Congress on Evolutionary Computation, pp. 1550–1555, 2008.

[10] J. Timmis, C. Edmonds and J. Kelsey, “Assessing the Performance of Two Immune Inspired Algorithms and a Hybrid Genetic Algorithm for Function Optimisation”, Proc. IEEE Conference od Evolutionary Computations (CEC2004), pp. 1044-1051, 2004.

Community Detection in Complex Networks based on Multiobjective Honey Bee Mating Optimization

Babak Amiri, Liaquat Hossain and John W Crawford
The University of Sydney, Sydney, NSW, Australia

Abstract— *Detecting community structure is crucial for uncovering the links between structures and functions in complex networks. Most of contemporary community detection algorithms employ single optimization criteria (e.g., modularity), which may have fundamental disadvantages. This paper considers the community detection process as a Multi-Objective optimization Problem (MOP). To solve the community detection problem this study used improved honey bee mating optimization (HBMO) algorithm. In the proposed algorithm, an external repository is considered to save non-dominated (Pareto) solutions found during the search process. The efficiency of the proposed algorithm is studied by testing on several data sets. Numerical results show that the proposed evolutionary optimization algorithm is robust and suitable for community detection problem.*

Keywords- complex network; community; multi-objective; honey bee mating optimization

I. INTRODUCTION

Most of real word networks possess inherent community structure, such as biological networks, web graphs and social networks. In network a community is a group of nodes with high dens connection within groups and sparse connection between groups. Communities in networks provide us information about how network function and topology affect each other[1].

In complex networks the number of communities is typically unknown and the communities are often of unequal size or density, and it has been shown that in complex networks communities have a hierarchical structure so we can say that finding communities in complex networks is a non trivial task [2]. Community detection problem has been introduced formally by Gervan and Newman in 2002 [3] and recently has been attracted the attention of researches in deferent areas. The community detection problem can be considered almost like an optimization problem [4] and lots of studies have been done based on evolutionary methods like GA [5-9], SA [10] and collaborative evolutionary algorithms[11] and [12] to solve it. To consider community detection problem as an optimization problem we need an objective function to be improved like modularity Q that is used as the stopping criterion in GN [3]. Most of community detection problems are based on the single objective optimization and their differences are based on their objective functions. The single objective based community detection algorithms have some shortcomings such as: single-objective optimization algorithms attempt to optimize just one criterion, they may fail when the optimization criteria are inappropriate and also most of them require prior information like the number of communities, which is usually unknown for real networks.

To overcome these shortcomings the community detection problem can be considered as a multi-objective problem, so multiple objective functions can be considered to obtain more accurate and comprehensive community structure.

In this paper we considered community detection problem as a multi-objective optimization problem and introduced an improved multi-objective algorithm based on honey bee mating optimization algorithm.

One of the recently proposed evolutionary algorithms that have shown great potential and good perspective for the solution of various optimization problems is honey bee mating optimization (HBMO). The HBMO algorithm has remarkable accuracy and calculation speed to deal with the optimization problem. Advantages of the HBMO algorithm are presented in [13, 14]. Refs. [15] and [14] have used the HBMO algorithm for solving optimization problems on two separate applications. In this paper, a multiobjective optimization is used for the placement and sizing of REGs by the improved HBMO algorithm. Original HBMO often converges to local optima. In order to avoid this shortcoming, in this paper a new mating process is proposed for rising accuracy of the algorithm. The proposed algorithm optimized two objective functions, the community score that measures the density of the clusters obtained and community fitness that minimizes the external links. A prior knowledge of a number of communities was not needed because this method returns a set of solutions where each of them correspond to different trade- offs between the two objectives, and gives a great chance to analyze the hierarchy of communities. The rest of the paper is organized as follows. Section 2 introduced the problem of community detection. The concept of a multi-objective optimization problem is reviewed in section 3. The original honey bee mating optimization algorithm is explained in section 4. In section 5 the proposed multiobjective algorithm used to detect community is presented, and then in section 6 the experimental results of the proposed algorithm in comparison with other approaches are shown.

II. COMMUNITY DETECTION

Community detection has been studied in many fields for many years such as computer science, physics and biology and lots of methods have been introduced in this field. In this study an undirected network G = (V, E) defined by a set of nodes [13] or vertices, and a set of links [16] connect two elements of V. A community consists of vertices and an edge between these nodes, where the nodes often cluster into tightly knit groups with a high density and a lower density of between the group connections[17].

A network can be represented mathematically by an adjacency matrix A, if there is an edge from v_i to v_j, $A_{ij} = 1$ and $A_{ij} = 0$ otherwise. The degree k_i of a node i, defined as $k_i = \sum_j A_{ij}$. Let $C \subset G$ the sub-graph where node i belongs to, the degree of i with respect to C can be split as $k_i(C) = k_i^{in}(C) + k_i^{out}(C)$, where $k_i^{in}(C) = \sum_{j \in C} A_{ij}$ is the number of edges connect the i to the other nodes in C, and $k_i^{out}(C) = \sum_{j \notin C} A_{ij}$ is the number of edges connecting i to the rest of the network. A sub-graph C is a community in a strong sense if $k_i^{in}(C) > k_i^{out}(C), \ \forall i \in C$. A sub-graph C is a community if $\sum_{i \in C} k_i^{in}(C) > \sum_{i \in C} k_i^{out}(C)$.

The quality measure of a community C that maximizes the in-degree of the nodes belonging to C has been introduced in [8]. On the other hand, in [18] a criterion that minimizes the out-degree of a community is defined. We now recall the definitions of these measures first, and then we show how they can be exploited in a multi-objective approach to find communities. In the following, without losing its generality, the network graph is assumed to be undirected.

Let μ_i denote the fraction of edges connecting node i to the other nodes in C. More formally, $\mu_i = \frac{1}{|C|} k_i^{in}(C)$ where $|C|$ is the cardinality of C.

The power mean of C of order r, denoted as $M(C)$ is defined as

$$M(C) = \frac{\sum_{i \in C} (\mu_i)^r}{|C|} \qquad (1)$$

Notice that in the calculation of $M(C)$, since $0 \leq \mu \leq 1$, the exponent r increases the weight of nodes having many connections with other nodes belonging to the same module, and diminishes the weight of those nodes having few connections inside C.

The volume v_C of a community C is defined as the number of edges connecting vertices inside C, i.e the number of 1 entries in the adjacency sub-matrix of A corresponding to C, $v_C = \sum_{i,j \in C} A_{ij}$.

The score of C is defined as $score\ (C) = M(C) \times v_C$. Thus, the score takes into account both the fraction of interconnections among the nodes (through the power mean) and the number of interconnections contained in the module C (through the volume). The community score of a clustering $\{C_1, \ldots C_k\}$ of a network is defined as

$$CS = \sum_{i=1}^{k} score(C_i) \qquad (2)$$

The community score gives a global measure of the network division in communities by summing up the local score of each module found. The problem of community detection has been formulated in [8] as the problem of maximizing CS.

In [18] the concept of community fitness of a module C is defined as

$$P(C) = \sum_{i \in S} \frac{k_i^{in}(C)}{(k_i^{in}(C) + k_i^{out}(C))^{\propto}} \qquad (3)$$

where $k_i^{in}(C)$ and $k_i^{out}(C)$ are the internal and external degrees of the nodes belonging to the community C, and α is a positive real-valued parameter controlling the size of the communities. The community fitness has been used by [18] to find communities.

III. MULTI-OBJECTIVE OPTIMISATION

In a multi-objective optimization problem, the purpose is to optimize several conflicting objectives simultaneously while still meeting some constraints. The community detection problem algorithm can be formulized as a multi-objective problem because two objective functions are competing with each other. The first is maximizing the inter-connecting links and the second is minimizing connections between communities. The multi-objective problem can be described as follows[19]:

$$min\ F = [f_1(X), f_2(X), \ldots, f_n(X)] \qquad (4)$$

where $f_i(X)$ is the i^{th} objective function and X is the vector of the optimization variables, n is the number of objective functions.

The solution to the multi-objective optimisation problem is a set of Pareto points. In the multi-objective optimization problem, a solution $X^* \in \Omega$ is a Pareto optimal if there is no solution (X) in Ω such that X dominates X^*. Ω is the set of all feasible values of X. The solution X_1 is said to dominate the solution X_2 if

$$\forall\, j \in \{1, 2, \ldots, n\}, f_j(X_1) \leq f_j(X_2)$$
$$\exists\, k \in \{1, 2, \ldots, n\}, f_k(X_1) \leq f_k(X_2) \qquad (5)$$

Solutions which dominate others but not themselves, are called non-dominated solutions.

IV. HONEY BEE MAITING OPTIMIZATION ALGORITHM

In order to deal with multiobjective problems, some modifications in the HBMO algorithm should be made. After the generation of the initial population and respective evaluation of objective functions, the selection of the ''best solutions'' (queens) should be made, but no longer based only on the comparison of single objective function values. Under a multiobjective approach, a new concept, such as the Pareto dominance concept, is needed for dealing with different solutions, i.e. classifying them as dominated or non-dominated solutions. The ''best solutions'' (queens) selected from the initial population are the non-dominated solutions.

Once identified the non-dominated solutions (queens), the iterative process is initiated in the same way as in the single objective case (mating flights, generation of new queens, improvement of the queens and of the new generation and selection of new queens). Each non-dominated solution will generate a certain number of solutions after each iteration. The criteria for generation

and improvement of the solutions specially best solution (queen) are the same as employed in the uni-objective version. With the new generated solutions and the non-dominated solutions from the previous iteration, the new set of non-dominated solutions is identified, which forms the Pareto front. These new solutions are saved in the repository and will generate the new solutions in the next iteration. The process is repeated until the stop criterion is satisfied. Frequently, the number of solutions that belong to the Pareto front increases as the algorithm evolves, thus each non-dominated solution is a potential generator (queen) of new solutions in the next iteration of the algorithm. It is noted that these non-dominated solutions saved in the repository are not the final non-dominated solutions because the repository will be updated after generation of broods in the next iterations. Besides, our experiences in the implementation of the proposed algorithm shows that it is safe to say the repository of the non-dominated solutions will be significantly updated in each new iteration with respect to the previous ones in the initial iterations. However, after some iteration the results of the repository may be saturated. That is the non-dominated solutions may be remained unchanged. Indeed the new solutions in the higher number of the iterations may be equal to the same of the repository solutions or they will be dominated by the repository ones. Therefore, it can be concluded from this manner that the non-dominated solutions of the repository after some iteration are trustworthy.

V. ROPOSED ALGORITHM

Previously in this work the search capability of the honey bee mating optimization algorithm was specifically used to find communities in complex networks. The steps of the proposed community detection algorithm as shown in figure 1 are as follows:

Step 1: Initializing the problem and algorithm parameters

In this phase, as described above, we are interested in identifying a partitioning $\{C_1, \ldots C_k\}$ that maximizes the number of connections inside each community and minimizes the number of links between the modules. The first objective was fulfilled by the community score. The first objective function is therefore $CS = \sum_{i=1}^{k} score\,(C_i)$. The second objective was carried out by the community fitness by summing up the fitness of all the C_i modules. The parameter α, that tunes the size of the communities, has been set to 1 because in most cases the partitioning found for this value are relevant [18]. The second objective is therefore $\sum_{i=1}^{k} P\,(C_i)$.

Our partitioning algorithm uses the locus-based adjacency representation proposed in [20] and used by [20] for multi-objective clustering. In this graphic representation, an individual of the population consists of N variable $x_1, \ldots, x_N$ and for each variable there is a set of possible range of values based on the adjacency matrix. For example, if node 1 has a connection with nodes 3, 5, and 6, the possible range of values for x_1 will be {3, 5, and 6}. For the isolated node k in the network, the possible range of values can be *{1, 2, …, k-1, k+1,…N}*.

Variables and values represent nodes of the graph $G = (V, E)$ modeling a network N, and a value j assigned to the i^{th} variable is interpreted as a link between the nodes i and j of V. This means that in the clustering solution found, i and j will be in the same cluster. However, a decoding step is needed to identify all the components of the corresponding graph. The nodes participating with the same component are assigned to one cluster. As observed in [20], the decoding step can be done in linear time. The main advantage of this representation is that the number of communities will be automatically determined by the number of components contained in an individual, and will be determined by the decoding step.

Step 2: Generate initial population

Step 3: Begin with i = 1.

Step 2: Pick randomly two candidates for selection X1 and X2.

Step 3: Pick randomly a comparison set of individuals from the population.

Step 4: Compare each candidate, X1 and X2, against each individual in the comparison set for domination using the conditions for domination given in Eqs. (4) and (5).

Step 5: If one candidate is dominated by the comparison set while the other is not, then select the later for reproduction and go to Step 7, else proceed to step 6.

Step 6: If neither or both candidates are dominated by the comparison set, then use sharing to choose winner.

Step 7: If the criteria i = N is reached, stop selection procedure, else set i = i + 1 and go to Step 2.

VI. Experimental results

In this section effectiveness of the proposed multi-objective honey bee mating optimization algorithm (MHBMO) has been compared with Clauset, Newman and Moore (CNM) [21] and MOGA-Net [8] using some real world datasets and synthetic benchmark datasets.

The effectiveness of stochastic algorithms is greatly dependent on the generation of initial solutions and therefore, for every dataset, algorithms have individually performed 100 times to test their own effectiveness, and each time with randomly generated initial solutions. Our algorithm was implemented into Matlab 7.1. All the experiments were conducted on a computer with Intel Core 2 Duo, 2.66 GHz, 4 GB RAM.

A. Evaluation Criteria

To evaluate the quality of the proposed community detection method we used Normalized Mutual Information (NMI) and Modularity (Q). The Normalized Mutual Information (NMI) is a similarity measure proven by Danon et al [22] to be reliable. Given two partitions A and B of a network in communities, let C be the confusion matrix whose element C_{ij} is the number of nodes of community i of the partition A that are also in the community j of the partition B. The normalized mutual information $I(A, B)$ is defined as:

$$I(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} C_{ij}\, log(\frac{C_{ij}N}{C_{i.}C_{.j}})}{\sum_{i=1}^{c_A} C_{i.}\, log(\frac{C_{i.}}{N}) + \sum_{j=1}^{c_B} C_{.j}\, log(\frac{C_{.j}}{N})} \quad (6)$$

where c_A (c_B) is the number of groups in the partition A (B), $C_{i.}$ ($C_{.j}$) is the sum of the elements of C in row i (column j), and N is the number of nodes. If $A=B$, $I(A, B)=1$. If A and B are completely different then $I(A, B)=0$.

The modularity of Newman and Girvan[23] is a well known quality function used to evaluate the goodness of a partition. Let k be the number of modules found inside a network, the modularity is defined as:

$$Q = \sum_{s=1}^{k}\left[\frac{l_s}{m} - \left(\frac{d_s}{2m}\right)^2\right] \quad (7)$$

where l_s is the total number of edges joining vertices inside the module s, and d_s is the sum of the degrees of the nodes of s. The first term of each summand of the modularity Q is the fraction of edges inside a community and the second one is the expected value of the fraction of edges that would be in the network if they fell at random without regard to the community structure. Values approaching 1 indicate a strong community structure.

B. Real World Networks

The Zackary's Karate Club network was generated by Zachary, who studied the friendship of 34 members of a karate club over a period of two years [24]. During this period, because of disagreements, the club divided in two groups almost of the same size.

The Bottlenose Dolphins network: A network of 62 bottlenose dolphins living in Doubtful Sound, New Zealand, was compiled by Lusseau after studying their behavior for seven years. A tie between two dolphins was established by their statistically significant frequent association. The network split naturally into two large groups where the number of ties was 159[25].

The American College Football network: comes from the United States college football. The network represents the schedule of Division I games during the 2000 season. Nodes in the graph represent teams and edges represent the regular season games between the two teams they connect. The teams are divided into conferences. On average the teams played 4 inter-conference matches and 7 intra-conference matches, thus they tend to play between members of the same conference. The network consists of 115 nodes and 616 edges grouped in 12 teams [3].

The political books compiled by V. Krebs: The nodes represent 105 books on American politics brought from Amazon.com, and the edges join pairs of books frequently purchased by the same buyer. Books were divided by Newman [28] according to their political alignment (conservative or liberal), except for a small number (13) having no clear affiliation.

The e-print Arxiv: initiated in Aug 1991, has become the primary mode of research communication in multiple fields of physics, and some related disciplines. It is a network of 9000 scientific paper and their citations (9000 nodes and 24000 links) [29].

The webpage network: were obtained from the complete map of the nd.edu domain, which contains 325,729 documents and 1,469,680 links [26].

C. Results

A comparison of results for running different algorithms on each real world dataset that mentioned in last section is illustrated in Tables 1–8.

Table 1. Modularity result obtained by the three algorithms on Zackary's Karate Club data

Method	Modularity			Standard deviation
	Best	Average	Worst	
MHBMO	0.4161	0.4161	0.4161	0.0000
CNM	0.3811	0.3708	0.3621	0.0100
MOGA-Net	0.4151	0.4149	0.4148	0.0010

Table 2. Modularity result obtained by the three algorithms on Bottlenose Dolphins data

Method	Modularity			Standard deviation
	Best	Average	Worst	
MHBMO	0.5233	0.5233	0.5233	0.0000
CNM	0.4605	0.4486	0.4367	0.0110
MOGA-Net	0.5048	0.5038	0.5029	0.0090

Table 3. Modularity result obtained by the three algorithms on American College Football data

Method	Modularity			Standard deviation
	Best	Average	Worst	
MHBMO	0.5812	0.5804	0.5793	0.0010
CNM	0.5433	0.5188	0.5046	0.0237
MOGA-Net	0.5148	0.4978	0.4784	0.0158

Table 4. Modularity result obtained by the three algorithms on Krebs' books on American politics data

Method	Modularity			Standard deviation
	Best	Average	Worst	
MHBMO	0.5162	0.5162	0.5162	0.0000
CNM	0.4934	0.4715	0.4522	0.0186
MOGA-Net	0.5176	0.5136	0.5075	0.0039

Table 5. Modularity result obtained by the three algorithms on Arxiv data

Method	Modularity			Standard deviation
	Best	Average	Worst	
MHBMO	0.7854	0.7811	0.7776	0.0042
CNM	0.7721	0.7415	0.7112	0.0304
MOGA-Net	0.7911	0.7226	0.7743	0.0083

Table 6. Modularity result obtained by the three algorithms on Web nd.edu data

Method	Modularity			Standard deviation
	Best	Average	Worst	
MHBMO	0.9286	0.9274	0.9260	0.0011
CNM	0.9274	0.8852	0.8501	0.0411
MOGA-Net	0.9304	0.9187	0.9073	0.0116

For Zachary's Karate Club data the MHBMO algorithm provided a value of 0.4161 in all runs, but on the other hand the CNM and MOGA-Net algorithms attained 0.3811 and 0.4151 respectively, as shown in Table 1. The MHBMO found 4 communities for this dataset in all runs. For Bottlenose Dolphins data (Table 2) the

MHBMO algorithm attained the values of 0.5233 for modularity in all runs and four communities were detected by the MHBMO. The best modularity values provided by MOGA-Net and CNM were 0.4605 and 0.5048. The MHBMO algorithm detected 11 communities for the American College Football data (Table 3) attained the best value of 0.5812 for modularity. The CNM and MOGA-Net algorithms provided the best values of 0.5433 and 0.5148 in terms of modularity. For the community detection problem, Krebs' books on the results of the American Politics data given in Table 4 shows that the MHBMO provided an optimum value of 0.5162 for modularity, and it found 4 communities in every solution. The proposed multi-objective algorithm works better than CNM and MOGA-Net in large datasets as shown in tables 5 and 6. For Arxiv and Web nd.edu datasets the MHBMO found 0.7854 and 0.9286 for modularity. The MHBMO algorithm was much more stable than the other algorithms, as can be observed from Tables 1 to 6. The results illustrate that the proposed MHBMO community detection approach can be considered as a viable and an efficient heuristic to find optimal or near optimal solutions to the problem of community detection in networks

VII. CONCLUSION

This paper presents a multiobjective community detection algorithm based on the improved honey bee mating optimization. The proposed algorithm has several advantages compared to other optimization techniques in that it does not require a complex calculus, thus it is free from divergence and there is no need to set initial values for the decision variables. The proposed algorithm for community detection can be used when the number of clusters is unknown a priori. To evaluate the performance of the proposed algorithm, it was compared with the MOGA-Net and CNM algorithms. The algorithm was implemented and tested on several real world datasets, and showed that it was quite efficient at discovering the community structure of complex networks. Thus, this proposed algorithm can be considered as a viable and an efficient heuristic to find the optimal or near optimal solutions to clustering problems.

REFERENCES

[1] S. Fortunato, "Community detection in graphs," *Physics Reports,* vol. 486, pp. 75-174, 2010.

[2] T. B. S. de Oliveira and L. Zhao, "Complex Network Community Detection Based on Swarm Aggregation," 2008, pp. 604-608.

[3] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences,* vol. 99, p. 7821, 2002.

[4] A. Ferligoj and V. Batagelj, "Direct multicriteria clustering algorithms," *Journal of Classification,* vol. 9, pp. 43-61, 1992.

[5] M. Tasgin, A. Herdagdelen, and H. Bingol, "Community detection in complex networks using genetic algorithms," *Arxiv preprint arXiv:0711.0491,* 2007.

[6] X. Liu, D. Li, S. Wang, and Z. Tao, "Effective algorithm for detecting community structure in complex networks based on GA and clustering," *Computational Science–ICCS 2007,* pp. 657-664, 2007.

[7] C. Pizzuti, "Community detection in social networks with genetic algorithms," 2008, pp. 1137-1138.

[8] C. Pizzuti, "GA-Net: A genetic algorithm for community detection in social networks," *Parallel Problem Solving from Nature–PPSN X,* pp. 1081-1090, 2008.

[9] C. Pizzuti, "A multi-objective genetic algorithm for community detection in networks," 2009, pp. 379-386.

[10] J. Liu and T. Liu, "Detecting community structure in complex networks using simulated annealing with k-means algorithms," *Physica A: Statistical Mechanics and its Applications,* vol. 389, pp. 2300-2309, 2010.

[11] A. Gog, D. Dumitrescu, and B. Hirsbrunner, "Community detection in complex networks using collaborative evolutionary algorithms," *Advances in Artificial Life,* pp. 886-894, 2007.

[12] J. Liu, W. Zhong, H. A. Abbass, and D. G. Green, "Separated and overlapping community detection in complex networks using multiobjective Evolutionary Algorithms," 2010, pp. 1-7.

[13] A. Afshar, O. Bozorg Haddad, M. A. Mariño, and B. Adams, "Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation," *Journal of the Franklin Institute,* vol. 344, pp. 452-462, 2007.

[14] M. Fathian, B. Amiri, and A. Maroosi, "Application of honey-bee mating optimization algorithm on clustering," *Applied Mathematics and Computation,* vol. 190, pp. 1502-1513, 2007.

[15] O. B. Haddad, A. Afshar, and M. A. Marino, "Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization," *Water Resources Management,* vol. 20, pp. 661-680, 2006.

[16] Available: http://www.cs.cornell.edu/projects/kddcup

[17] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 101, p. 2658, 2004.

[18] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics,* vol. 11, p. 033015, 2009.

[19] C. M. Lin and M. Gen, "Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using

multiobjective hybrid genetic algorithm," *Expert Systems with Applications,* vol. 34, pp. 2480-2490, 2008.

[20] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *Evolutionary Computation, IEEE Transactions on,* vol. 11, pp. 56-76, 2007.

[21] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E,* vol. 70, p. 066111, 2004.

[22] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment,* vol. 2005, p. P09008, 2005.

[23] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E,* vol. 69, p. 026113, 2004.

[24] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research,* pp. 452-473, 1977.

[25] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology,* vol. 54, pp. 396-405, 2003.

[26] R. Albert, H. Jeong, and A. L. Barabási, "The diameter of the world wide web," *Arxiv preprint cond-mat/9907038,* 1999.

INVESTOR PROTECTION AND REGULATIONS IN DERIVATIVE MARKET

Dr. Rajiv Kumar Agarwal
Assistant Professor, Faculty of Commerce
S.S.(PG) College, MJP Rohilkhand University
Mumukshu Ashram, Shahjahanpur(UP)-India
Mobile no. +919415060433
Email : dr.rajivkagarwal@gmail.com

ABSTRACT

With the opening of the economy to multinationals and the adoption of the liberalized economic policies the economy is driven more towards the free market economy. The complex nature of the financial structuring is self involves the utilization of multicurrency transaction. It exposes the clients to various risks such as exchange rate risk, interest risk, economic risk & political risk .In the present state of the economy there is an imperative need for the clients to protect their operating profits by shifting some of the uncontrollable financial risk to those who are able bear and manage them. Thus, risk management becomes a must for survival since there is a high volatility in the present's financial markets In the context, derivatives occupy an important place as a risk reducing machinery. Derivatives are useful to reduce many of the risks. Importance of Derivatives led to high growth of Derivatives trading and development of Derivatives market in India. Therefore there is a great need of investors protection and regulatory mechanisms for such markets. SEBI plays an important role in this process. This paper tends to highlights some regulatory and investors protection measures related to Derivatives Market in India.

1. Introduction:

Derivative trading in India takes can place either on a separate and independent Derivative Exchange or on a separate segment of an existing Stock Exchange. Derivative Exchange/Segment function as a Self-Regulatory Organisation (SRO) and SEBI acts as the oversight regulator. The clearing & settlement of all trades on the Derivative Exchange/Segment would have to be through a Clearing Corporation/House, which is independent in governance and membership from the Derivative Exchange/Segment.

2. Regulatory Objectives:

The LCGC outlined the goals of regulation admirably well in this report which believes that "The regulation should be designed to achieve specific, well-defined goals. It is inclined towards positive regulation designed to encourage healthy activity and behavior. It has been guided by the following objectives:

2.1. **Investor Protection:** Attention needs to be given to the following four aspects:

2.1.1. **Fairness & Transparency:** The trading rules should ensure that trading is conducted in a fair and transparent manner. Experience in other countries shows that in many cases, derivatives-brokers / dealers failed to disclose potential risk to the clients. In this context, sales practices adopted by dealers for derivatives would require specific regulation. In some of the most widely reported mishaps in the derivatives market elsewhere, the underlying reason was inadequate internal control system at the user-firm itself so that overall exposure was not controlled and the use of derivatives was for speculation rather than for risk hedging. These experiences provide useful lessons for us for designing regulations.

2.1.2. **Safeguard for clients' moneys:** Moneys and securities deposited by clients with the trading members should not only be kept in a separate clients' account but should also not be attachable for meeting the broker's own debts. It should be ensured that trading by dealers on own account is totally segregated from that for clients.

2.1.3. **Competent and honest service**: The eligibility criteria for trading members should be designed to encourage competent and qualified personnel so that investors/clients are served well. This makes it necessary to prescribe qualification for derivatives brokers/dealers and the sales persons appointed by them in terms of a knowledge base.

2.1.4. **Market integrity**: The trading system should ensure that the market's integrity is

safeguarded by minimizing the possibility of defaults. This requires framing appropriate rules about capital adequacy, margins, clearing corporation, etc.

2.2. Quality Markets:

The concept of "Quality of Markets" goes well beyond market integrity and aims at enhancing important market qualities, such as cost-efficiency, price-continuity, and price-discovery. This is a much broader objective than market integrity.

2.3 Innovation:

While curbing any undesirable tendencies, the regulatory framework should not stifle innovation which is the source of all economic progress, more so because financial derivatives represent a new rapidly developing area, aided by advancements in information technology."

3. Derivatives Trading – Regulatory Framework :

With the amendment in the definition of 'securities' under SC(R)A (to include derivative contracts in the definition of securities), derivatives trading takes place under the provisions of the Securities Contracts (Reg Contracts) (Regulation) Act, 1956 and the Securities and Exchange Board of India Act, 1992. Dr. L.C Gupta Committee constituted by SEBI had laid down the regulatory framework for derivative trading in India. SEBI has also framed suggestive bye-law for Derivative Exchanges/Segments and their Clearing Corporation/House which lay's down the provisions for trading and settlement of derivative contracts. The Rules, Bye-laws & Regulations of the Derivative Segment of the Exchanges and their Clearing Corporation/House have to be framed in line with the suggestive Bye-laws. SEBI has also laid the eligibility conditions for Derivative Exchange/Segment and its Clearing Corporation/House. The eligibility conditions have been framed to ensure that Derivative Exchange/Segment & Clearing Corporation/House provide a transparent trading environment, safety & integrity and provide facilities for redressal of investor grievances.

4. Market Regulation and Investor Protection:

We have seen that pursuant to the recommendations of JR Verma Committee SEBI formulated and approved guidelines to the stock exchanges (NSE/BSE) and permitted trading in Derivatives. We will now discuss the regulatory measures as envisaged by SEBI.

4.1 Futures/ Options contracts in both index as well as stocks can be bought and sold through the trading members of National Stock Exchange, or the BSE Mumbai Stock Exchange. Some of the trading members also provide the internet facility to trade in the futures and options market.

4.2 The investor is required to open an account with one of the trading members and complete the related formalities which include signing of member-constituent agreement, constituent registration form and risk disclosure document.

4.3 The trading member will allot the investor an unique client identification number.

4.4 To begin trading, the investor must deposit cash and/or other collaterals with his trading member as may be stipulated by him. SEBI has issued detailed guidelines for the benefit of the investor trading in the derivatives exchanges. These may be viewed and studied.

4.5 Margins are computed and collected on-line, real time on a portfolio basis at the client level. Members are required to collect the margin upfront from the client & report the same to the Exchange.

4.6 All the Futures and Options contracts are settled in cash at the expiry or exercise of the respective contracts as the case may, be. Members are not required to hold any stock of the underlying for dealing in the Futures / Options market.

5 Important Regulatory Conditions Specified by SEBI:

5.1 Derivative trading to take place through an on-line screen based Trading System.

5.2 The Derivatives Exchange/Segment shall have on-line surveillance capability to monitor positions, prices, and volumes on a real time basis so as to deter market manipulation.

5.3 The Derivatives Exchange/ Segment should have arrangements for dissemination of information about trades, quantities and quotes on a real time basis through at least two information vending networks, which are easily accessible to investors across the country.

5.4 The Derivatives Exchange/Segment should have arbitration and investor grievances redressal mechanism operative from all the four areas / regions of the country.

5.5 The Derivatives Exchange/Segment should have satisfactory system of monitoring investor complaints and preventing irregularities in trading.

5.6 The Derivative Segment of the Exchange would have a separate Investor Protection Fund.

5.7 The Clearing Corporation/House shall perform full novation, i.e., the Clearing Corporation/House shall interpose itself between both legs of every trade, becoming the legal counterparty to both or alternatively should provide an unconditional guarantee for settlement of all trades.

5.8 The Clearing Corporation/House shall have the capacity to monitor the overall position of Members across both derivatives market and the underlying securities market for those Members who are participating in both.

5.9 The level of initial margin on Index Futures Contracts shall be related to the risk of loss on the position. The concept of value-at-risk shall be used in calculating required level of initial margins. The initial margins should be large enough to cover the one-day loss that can be encountered on the position on 99% of the days.

5.10 The Clearing Corporation/House shall establish facilities for electronic funds transfer (EFT) for swift movement of margin payments.

5.11 In the event of a Member defaulting in meeting its liabilities, the Clearing Corporation/House shall transfer client positions and assets to another solvent Member or close-out all open positions.

5.12 The Clearing Corporation/House should have capabilities to segregate initial margins deposited by Clearing Members for trades on their own account and on account of his client. The Clearing Corporation/House shall hold the clients' margin money in trust for the client purposes only and should not allow its diversion for any other purpose.

5.13 The Clearing Corporation/House shall have a separate Trade Guarantee Fund for the trades executed on Derivative Exchange / Segment.

6 Measures Specified by SEBI to Ensure Investor Protection in Derivative Market:

SEBI has also specified measures to ensure protection of the rights of investors. These measures are as follows:

6.1 Investor's money has to be kept separate at all levels and is permitted to be used only against the liability of the Investor and is not available to the trading member or clearing member or even any other investor.

6.2 The Trading Member is required to provide every investor with a risk disclosure document which will disclose the risks associated with the derivatives trading so that investors can take a conscious decision to trade in derivatives.

6.3 Investor would get the contract note duly time stamped for receipt of the order and execution of the order. The order will be executed with the identity of the client and without client ID order will not be accepted by the system. The investor could also demand the trade confirmation slip with his ID in support of the contract note. This will protect him from the risk of price favor, if any, extended by the Member.

6.4 In the derivative markets all money paid by the Investor towards margins on all open positions is kept in trust with the Clearing House /Clearing Corporation and in the event of default of the Trading or Clearing Member the amounts paid by the client towards margins are segregated and not utilized towards the default of the member. However, in the event of a default of a member, losses suffered by the Investor, if any, on settled / closed out position are compensated from the Investor Protection Fund, as per the rules, bye-laws and regulations of the derivative segment of the exchanges.

6.5 Presently, SEBI has permitted Derivative Trading on the Derivative Segment of BSE and the F&O Segment of NSE. Derivative products have been introduced in a phased manner starting with Index Futures Contracts in June 2000, Index Options and Stock Options introduced in June 2001 and July 2001 followed by Stock Futures in November 2001.

7 Types of Derivative Contracts Permitted by SEBI

Derivative products have been introduced in a phased manner starting with Index Futures Contracts in June 2000. Index Options and Stock Options were introduced in June 2001 and July 2001 followed by Stock Futures in November 2001.

7.1 Minimum Contract Size

The Standing Committee on Finance, a Parliamentary Committee, at the time of recommending amendment to Securities Contract (Regulation) Act, 1956 had recommended that the minimum contract size of derivative contracts traded in the Indian Markets should be pegged not below Rs. 2 Lakhs. Based on this recommendation SEBI has specified that the value of a derivative contract should not be less than Rs. 2 Lakh at the time of introducing the contract in the market.

7.2 The Lot Size of a Contract

Lot size refers to number of underlying securities in one contract. Additionally, for stock specific derivative contracts SEBI has specified that the lot size of the underlying individual security should be in multiples of 100 and fractions, if any, should be rounded of to the next higher multiple of 100. This requirement of SEBI coupled with the requirement of minimum contract size forms the basis of arriving at the lot size of a contract

For example, if shares of XYZ Ltd are quoted at Rs.1000 each and the minimum contract size is Rs.2 lacs, then the lot size for that particular scrips stands to be 200000/1000 = 200 shares i.e. one contract in XYZ Ltd. covers 200 shares.

7.3 SEBI Amendment to Stipulations on Lot Size

While the Legislative body stipulated the minimum contract size in terms of value (Rs.2 Lacs), the system of standardizing securities trade in Lots, had a multiplying effect, on the minimum value of a contract, when the prices of the premium Scrips started appreciating over time. BSE Sensix Index which was less than 3000 at that time swelled to nearly 6000 presently. As the value of individual scrips increased, smaller number of such scrips would be sufficient to cover the minimum contract value of Rs.2.00 Lacs prescribed by the Standing Committee of the Parliament. But stipulating a fixed number of shares as the lot in many cases swelled the value of the contract to Rs.5 Lacs and even more in many cases. This brought derivatives trading beyond he scope of the small investor.

Considering the fact SEBI revised its stipulations regarding Lot size, but retaining the minimum contract value at Rs.2 Lacs and issued a press release on 07.01.2004 stating:

It has been noticed that in several derivative contracts the value has exceeded Rs. 2 lakh. In such cases it has been decided to reduce the value of the contract to close to but not less than Rs. 2 lakh by using an appropriate lot size / multiplier which could be half or 50%. The exchanges could determine any other lot size / multipliers to keep the contract size of derivatives close to Rs. 2 lakh, but in any case not less than Rs. 2 lakh. The exchanges would be able to reduce the contract size of a derivative contract by submitting a detailed proposal to SEBI and after giving at least two weeks prior notice to the market.

8. CONCLUSION:

Though Financial markets experts and regulatory authorities specially SEBI has successfully implemented various strategic measures to control, regulate and strengthen the Derivatives markets as far as investors protection is concerned but even then a lot more has to be achieved in this regard. This volume driven market is still lacking on parameters like more transparency , fairness in dealing , investor knowledge , prohibition of insider trading , adaptability of processes , accessibility of information , determination of brokerage rates and marketing strategies of derivatives trading companies.

9. Refernces:

9.1 Bhalla, V.K. , "*Investment Management - Security Analysis and Portfolio management",* 15th Edition(2009) ISBN 81-219-1248-2

9.2 Dash , A.P. , *"Security Analysis and Portfolio management" Second Edition (2009*), ISBN 9789380026107.

9.3 *"Derivatives and Alternative Investments"* CFA curriculum, Volume 6, Level 1 (2009),ISBN 978-0-536-53708-9

9.4 Rustagi , R.P., "*Financial Management*" , Third edition (2006) ISBN 81-85989-28-1

9.5 Singh , Rohini , "Security Analysis and Portfolio management" , First edition (2009) ,ISBN 9788174467485

9.6 Varma , Jayanth Rama, *"Derivatives and Risk Management*", Fourth edition (2010) ISBN 978-0-07-060430-8