CUDA BASED MULTI OBJECTIVE PARALLEL GENETIC ALGORITHMS:

ADAPTING EVOLUTIONARY ALGORITHMS FOR DOCUMENT SEARCHES

Jason P. Duran, Sathish AP Kumar

Computer Science and Information Systems Department

National University, San Diego, CA 92123

Email: Jason_Duran@acm.org, sathish.ap@gmail.com

ABSTRACT:

This paper introduces a Multi Objective Parallel Genetic Algorithm (MOPGA) using the Compute Unified Device Architecture (CUDA) hardware for parallel processing. The algorithm demonstrates significant speed gains using affordable, scalable and commercially available hardware. The algorithm implements a document search using techniques such as Term Frequency Inverse Document Frequency (TF-IDF), Latent Semantic Analysis (LSA), Multi Objective Algorithms (MOA), Genetic Algorithm (GA), and Quad Tree Pareto Dominance techniques.

The objective of the proposed algorithm is to assemble an adaptable and scalable search mechanism to efficiently retrieve highly relevant document for a given search query. TFIDF and LSA vector space searches are two of the more common approaches to text mining. We have demonstrated that by combining results from both operations the number and quality of results could be improved. Evolutionary algorithms, specifically Genetic Algorithms have long been used to efficiently optimize multi-objective problems and so provide a natural starting point for our approach.

Keywords: Multi-Objective Parallel Genetic Algorithms, CUDA, TF-IDF, LSA, Pareto Quad Tree, Text Mining.

1 INTRODUCTION

The proposed algorithm searches through multiple documents looking for relevant matches to a given search string. The algorithm begins by converting the search string to a query vector in the TF-IDF, and LSA document search spaces. Chromosomes in this algorithm are composed of ten alleles that are each a direct encoding of a term. Using TFIDF domain knowledge a heap is constructed for each search term and any document that the term is relevant to. The algorithm exploits this domain knowledge to select unevaluated documents that have the strongest relation to the term. Iterative generations benefit from the principle of survival of the fittest in an attempt to discover the documents that are most closely related to the search query. The algorithm allocates parallel processes to CUDA enabled graphics devices to distribute the work across multiple processors, dramatically reducing the processes run time. Tabularized results for various search keys are presented along with corresponding execution times.

Multi-Objective Algorithms require ranking systems to properly evaluate tradeoffs between the search domains. Pareto Dominance ranking provides the ability to objectively analyze tradeoffs between both the different domain results that originate from different input. Genetic Algorithms represent an efficient heuristic search of some data set that can often retrieve near optimal results in fewer operations when compared to linear ranking methods. To further improve performance problems associated with the TFIDF and LSA search space models, the algorithm has been adapted to utilize massively parallel processors. The final result is a Multi-Objective Parallel Genetic (MOPGA) Algorithm that utilizes TFIDF, and LSA vector searches, implemented on the Compute Unified Device Architecture (CUDA) framework.

1.1 ALGORITHM ORGANIZATION

This algorithm runs on both the Linux PC host and on available CUDA devices. On the PC Side, some initial loading and search parameters are done in parallel where possible. This preloaded data is stored in a Read-Only database called data space. Before the search begins, data space is divided amongst the MOPGA Agents that are to be run. While running

each MOPGA Agent has its own, database local PKnown where Pareto Quad Tree and final document evaluations are stored. Mathematical operations take advantage of CUDA acceleration where possible. Finally, once a predetermined time or a number of cycles have elapsed the results are merged and sorted for display.

2.0 Domain Search Spaces

This implementation of MOPGA uses two vector search spaces for its Domain. The relevance of a given search vector is the angular difference between a search vector and a document vector. The two-domain spaces are "TFIDF" and "LSA" . The actual data that composes the matrices are book synopsis taken from user comments on Amazon.com. The mathematical explanations are as follows.

2.1.0 Term Frequency - Inverse Document Frequency (TF-IDF)

Term Frequency – Inverse Document Frequency provides the MOPGA a statistical method of determining how important a term is within a document with respect to a collection of documents. The MOPGA constructs a matrix where rows represent terms, columns represent documents, and individual cells record the number of occurrences of a given term in a document. Within this matrix often repeated terms are considered important terms in the document. This is not entirely true, as some words with such highly repeated frequency do not convey uniquely searchable concepts. Words such as "The" are an example. It will appear with high frequency in all documents and as such, it is not a relevant search term. Extending this concept if we have a set of documents all on a single subject such as databases, because all documents in the collection contain this word, a search for documents with that term would return the whole collection. For this reason, TF-IDF does not consider it a relevant search term. TF-IDF itself is a combination of two ratios, Term Frequency (TF), and Inverse Document Frequency (IDF).

2.1.1 Term Frequency (TF)

Term Frequency is the statistical process of determining how important a term is in the context of a document. It does this by counting the occurrences of a particular term in the document, which divided by the overall count of terms in the document, which establishes a ratio of the terms statistical importance to the document [1].

2.1.2 Inverse Document Frequency (IDF)

Inverse Document Frequency is used to determine how unique a given term is within a collection of documents. By taking the ratio of the total number of document with respect to the number of documents a term appears in, the relative importance of a term as a unique identifier can be determined. By taking the log of this ratio, a dampened value that is suitable for combination with TF is derived [1].

2.1.3 Weighted Search Vectors

The MOPGA searches for documents in the TF-IDF search space by creating a query vector. A query vector is similar to a standard document vector in the search space, however, the TF portion is calculated slightly different. The MOPGA replaces the raw frequency of the term within the query with a user provided bias weight. In this manner, the MOPGA can create bias in favor of a particular term within a query. The MOPGA scores query vector to document vector comparisons by performing the following mathematical operations:

$$tf_{i,j} = \frac{|n_{i,j}|}{\sum_k n_{k,j}} \qquad (1)$$

$$idf_i = \log_2 \frac{|D|}{|\{j: t_i \in d_j\}|} \qquad (2)$$

$$tfidf_{i,j} = tf_{i,j} \times idf_i \qquad (3)$$

$$score = sim_{d,q} = \frac{D \cdot Q}{|D||Q|} \qquad (4)$$

2.1.4 Advantages and Limitations

TF-IDF search vectoring will never assign scores to documents that the exact search terms do not appear in, regardless of how informative a particular term may be. TFIDF does not connect words with their synonyms [2]. Further TF-IDF does not equate words to their conjugated forms, as an example a search for database would be unrelated to a search for databases. This lack of knowledge to make these connections makes the requirement of a strong lexical analyzer to process the information highly desirable, adding to the complexity of the task. The main advantage TFIDF gives is that it is simple to compute and easily parallelized.

2.2.0 Latent Semantic Analysis

Latent Semantic Analysis provides the MPOGA with the ability to find documents based on the relationship of words as they appear in context to one another. The MOPGA LSA function utilizes the same matrix of term relations to documents that are constructed for TF-IDF vector searches [3]. The MOPGA LSA function decomposes these relationships into different matrices that represent relations of terms and documents. Finally, the MOPGA LSA function filters noise in these relations to find better matches. These filtered relations do a good job at finding synonyms and other related words. As an example if a user was searching for "dog training" it would recognize that terms like leash, pet, and treat are related and possibly return documents with these terms in them. The individual steps used in the MOPGA LSA functions are presented next.

2.2.1 SVD DECOMPOSITION AND K REDUCTION

LSA depends on constructing a term by document rectangular matrix M x N, where cell values contain weighted TF-IDF values. This matrix is then decomposed via a Singular Value Decomposition into three different matrices where A=USVT. Where U is decomposed into an M x N matrix and S is an N x N diagonal matrix containing the ranked singular values, and V is an N x N matrix. Both U and V are unitary so that

$$U \cdot UT = I \qquad (5)$$

$$V \cdot VT = I \qquad (6)$$

K reduction is performed by selecting a K x K portion of the S matrix, M x K of the U matrix, and M x K of the V matrix such that [4].

$$A_k = U_k S_k V_k^t \qquad (7)$$

2.2.2 PSEUDO DOCUMENTS

With these, K reduced matrices the MOPGA LSA function can compare two terms to one another, or documents to documents. To compare a chromosome to a query the MOPGA LSA function first projects the query into the K dimensional space and treats it as if it were any other document in the collection. Because chromosomes themselves are M x 1 matrix representations of their contained terms they are easily converted into these pseudo documents. The MOPGA LSA function accomplished these projections into the K reduced space via the equation [4].

$$D_k = QtU_k S_k^{-1} \quad (8)$$

2.2.3 SIMILARITY

Given the resulting two pseudo documents, the MOPGA LSA function then compares the level of similarity between them by examining the angles between their vector representations with the following well-known equation

$$score = sim_{d,q} = \frac{D \cdot Q}{|D||Q|} \qquad (4)$$

This comparative process gives a range between 1 and -1 where 1 would be a very similar document and -1 as dissimilar as possible [2].

2.2.4 ADVANTAGES AND LIMITATIONS

"Folding in" pseudo documents in this manner does not affect the underlying relationships, a new SVD decomposition is required to incorporate new relations into the process. LSA does not perform well with words that have different meanings based on their contextual usage. LSA is also conducted under the assumption that words and documents follow a Gaussian distribution when it is possible that a Poisson distribution exists [8]. Despite these limitations, the MOPGA LSA function is able to perform the SVD decompositions on the same underlying data that is constructed for the TF-IDF vector search. The ability to recognize relations of the terms to other terms is also crucial to returning relevant information. The actual generation of pseudo documents by the MOPGA requires only a few matrix multiplications once the SVD decompositions are complete.

3 MULTI OBJECTIVE ALGORITHMS

A Multi Objective algorithm seeks to maximize/minimize two or more distinct functions. As is often the case one set of input may increase the desirable results from one function while degrading the results from the other functions. To accomplish the goal of returning relevant information quickly the MOPGA must know how to balance trade offs between one resulting function score with respect to the other function. To accomplish this task the MOPGA uses Pareto Dominance [5].

3.1 QUAD TREE PARETO DOMINANCE BASED RANKING

A Quad Tree is a tree based data structure that stores different chromosomes. Each chromosomal node is a vector with two elements that reflect the TFIDF and LSA scores. All nodes residing in the tree are non-dominated. This is accomplished by assigning scores to the TFIDF and LSA

values of particular candidate. A 0 indicates that this chromosomes score is worse than the comparison node. A score of 1 indicates that the candidate's score is superior to the comparison's node. Each candidate thus scores one of four possible values. 0/0 indicates that it is completely dominated by the comparison node and thus subsequently discarded. A 0/1 or 1/0 indicates that one of the search values scores better than the comparison node and thus is not considered dominated by it. In this case if a child node exists further comparisons will be evaluated against that node. If there are no child nodes then the candidate is inserted into the tree at the appropriate position. In the final case 1/1 indicates that the candidate dominates the existing node. In this case the candidate will take the place of the existing node, and all child nodes of the existing node to be discarded will be evaluated for insertion. The Quad tree is thus guaranteed to contain only non-dominated nodes, and new candidates can be efficiently compared for insertion [6].

4 GENETIC ALGORITHMS

GAs has been successfully applied to a wide variety of problems. In particular, GAs excels at optimization problems where optimal solution execution times are exponentially dependent on the size of the data search space. GA's encode, directly or indirectly different input values for a chromosome. These chromosomes undergo genetic processes where the current best-known chromosomes contained in PKnown are used to generate new chromosomes, PCurrent. The best of these new PCurrent chromosomes replace less fit chromosomes in PKnown. Over many generations, chromosomes converge to optimal solutions [7]. MOPGAs offer a wide number of solutions to overcome problems associated with premature convergence.

4.1 ENCODING DOCUMENTS

Encoding is the process of mapping a problem and its possible solutions to a chromosome. These chromosomes can be a direct encoding of values, it can represent different states of a state machine, or even an order arrangement of items. Because of the number of possible terms that can be included in a relatively small set of documents it is not possible to represent all terms in a single chromosome. The MOPGA presented here uses a direct encoding scheme to limit the number of alleles in a chromosome to ten. Each individual allele is a unique term identifier that is assigned to terms as they are read into the dictionary.

The dictionary is an M x N matrix, where M is the number of searchable terms in all documents. The actual position number of a term in this dictionary is the real encoded value. As each document is read into, the dictionary the number of time a term appears in the document is recorded in the matrix. Once all documents are read into the dictionary the matrix is used to generate TFIDF values. This domain information is exploited later to intelligently select strong candidates for evaluation. This is accomplished by constructing a series of term heaps for each MOPGA Agent that include only documents within an agents assigned search space.

4.2 Selection

Each generation of the MOPGA generates a predefined number of chromosomes called PCurrent to evaluate for insertion into the Quad tree based set called PKnown. To generate the PCurrent set two PKnown nodes are selected for crossover, which generates four new chromosomes. The selection phase is the process of randomly picking two parents for the crossover phase. In terms of this process, the selected parent nodes represent the local space that will be further explored.

4.3 SINGLE POINT CROSSOVER

The process by which the MOPGA generates new chromosomes from the Pknown set is called Single Point Crossover. Using two chromosomes A, and B from the Pknown set both chromosomes are split in halves resulting in A1,A2 and B1,B2. The second halves are then exchanged resulting in A1 B2, A1 B1, A2 B1, A2 B2, B1 A1, B1 A2, and B2 A1, B2 A2. The order of the alleles in the resulting chromosomes affects the local search space that is evaluated during the subsequent evaluation phase.

4.4 RANDOM MUTATION

One drawback to this type of search occurs when the MOPGA continually selects the same parents to generate the same results. When the MOPGA finds a series of solutions that are strong candidates many very similar results begin to be found in PKnown. From an evolutionary standpoint, these chromosomes represent the fit possibilities found so far, but they may in fact represent only local optima.

To force the MOPGA to explore chromosome that cannot be formed by the combination of PCurrent solutions, a pre-known percentage is used to determine if a random

mutation occurs in the PCurrent chromosomes generated by the MOPGA. A mutation is accomplished by selecting a random allele and then generating a random term value to encode. This mutation is similar to annealing concepts used in other algorithms. It forces the algorithm to explore other possible locations in the search space [8].

4.4 Evaluation

Evaluation is the process the MOPGA uses to determine if a given chromosome is fit for insertions into Pknown and ultimately presented to the user. The MOPGA selects a document and generates TFIDF and LSA vector similarity scores by comparing the appropriate document vectors to the search vectors. These scores are then used by the Pareto quad tree ranking mechanism to determine if the chromosome should be retained in the PKnown set.

The nature of a document search implies it is a finite search of available documents that are contained in a library. The MOPGA conforms to this constraint by selecting an allele in a chromosome and then using the term heaps to select a document. This process ensures that processing time can never exceed the processing time of a straight linear search of the same library. As documents are evaluated, they are inserted into a hash map that stores the evaluations and enable the evaluation to rapidly determine if an evaluation has already been performed against the selected document.

5 Parallelization

Many of the mathematical operations involved in TF-IDF and LSA lend themselves to parallelization, and in particular to CUDA based implementations [9]. When the algorithm first begins computing TF-IDF scores, it is beneficial to use reduction techniques when summing term counts. When computing a singular value decomposition there are several parallel techniques to apply. Beyond the mathematical applications, there is the process of exploring a Pareto front. By dividing the data space among the MOPGA Agents, different parts of the whole can be run in different threads. This organization conforms too many of the common parallel patterns in use today [10].

5.1 Databases

This MOPGA implementation uses two types of databases, data space and PKnown. Data space contains all of the "static" data that does not change while a search is underway. It contains all of the TFIDF and LSA domain information as well as document excerpts and indexed terms lists. This data space also contains a series of heaps for each MOPGA Agent that is scheduled to run. These heaps represent the assigned search area each MOPGA Agent will concern itself with. These heaps are copied into each MOPGA Agents PKnow database. There is only one copy of data space and it is read only once a search begins.

The second type of database is called PKnown. Each MOPGA Agent that is tasked to run has its own PKnown database that initially contains its assigned term heaps and an empty Pareto quad tree. As chromosomes are evaluated, they are translated into a document via the term heaps and the results are stored. Using these evaluation results a chromosome is then considered for insertion into the Pareto quad tree.

5.2 MOPGA Agent

The MOPGA Agent is tasked with attempting to find non-dominated chromosomes to insert into its own Pknown. When there are insufficient chromosomes in the Pareto quad tree the MOPGA Agent will generate random chromosome to help explore the Pareto front. After two chromosomes are generated or selected, single point crossover and mutations are finished a random allele is selected. Using the term heaps assigned to this agent a document in its assigned sections of data space is then evaluated. In the case that a chromosome has an allele that has no more documents in its heap then the allele is mutated and a different existing allele is selected. If the resulting document evaluation is non-dominated then the chromosome is inserted into Pknown for the MOPGA Agent to exploit.

5.3 Limitations

While the CUDA devices have the ability to greatly speed up some of the more complex mathematical operations, there are only a limited number of these devices on a given machine. This limitation means that access to the devices has to be shared with semaphore type locks. This algorithm has the ability to find the Maximum optima for a given search there is no guarantee that it will be found if the search constrained by time or cycles.

6 Algorithms

6.1 MOPGA Algorithm (Steps 2 – 5 in are in parallel and agents run in parallel)

1. Read in Data, build term dictionary and document library matrices.
2. Perform TFIDF / LSA
3. Divide Search Spaces
4. Build term heaps per MOPGA Agent
5. Gather search terms
6. Launch MOPGA agents.
7. Wait for time, cycle, or generation end condition
8. Present results from agent PKnown.

6.2 MOPGA AGENT

1. Select parent chromosomes from Pknown or generate random chromosome if insufficient chromosomes.
2. Perform single point crossover.
3. Mutate chromosomes
4. Determine document – term heaps.
5. Evaluate – perform TFIDF/LSA.
6. Insert chromosomes into Pknown.

7 EXPERIMENTS

A series of experiments were conducted to demonstrate that combined TFIDF/LSA ranking returns both more and superior results than using either of the technique alone would generate. Experimentations also includes execution times of the different techniques. All experiments were conducted on a intel i7-970 and 12 GB of ram and 4 cpu cores. Additionally the test machine has two GTX480 Nvidia video cards, each having 1.5 GB of memory and 480 CUDA cores. The code was written in C++ and C for CUDA version 2.5. Mathematical libraries provided by NVIDA and EM Photonics CULA were used were possible. Serial tests relied on TNT JAMA linear algebra libraries and templates.

7.2 EXPERIMENT 1

The first experiment was conducted to gather data that could be used to determine if combining TFIDF and LSA would yield better results. The MOPGA algorithm was allowed to iterate over all data as well as by cyclic restriction. The data set size was also altered. LSA K reductions were held constant at 100 for both the MOPGA and the LSA serial version. It should be noted that the TNT JAMA linear algebra libraries could not process a matrix larger than 10,000 X 300 so the final two runs have no results recorded.

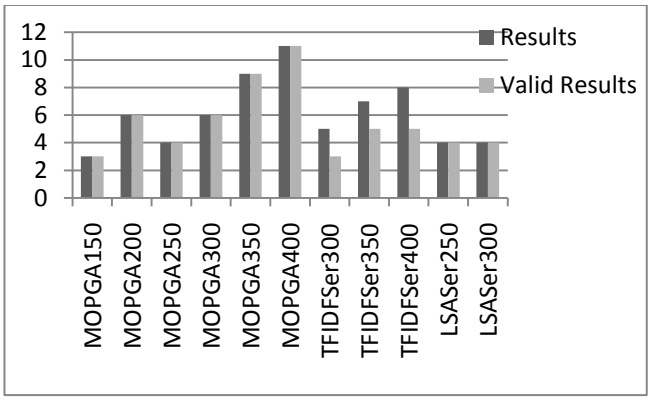

Fig 1.

Figure 1 shown here demonstrates that both the LSA and the MOPGA algorithm returned only highly relevant data, while TFIDF seemed to return some items that were not truly related to the search but contained a term in the search vector. Both the MOPGA algorithm and the TFIDF returned more results than did the LSA. The MOPGA algorithm did return the most relevant documents.

7.2 EXPERIMENT 2

The second experiment is very similar to the first experiment, except that execution times recorded in an attempt to show how well the proposed MOPGA algorithm performed against the other serial implementations. The time shown in these charts is both the initial time spent processing a data set before a query and the search time spent finding the results. It should be noted that while the data set size was increased from 8860 X 150 to 11882 X 400 for the two serial versions, only the larger dataset was used for the recorded MOPGA time shown here.

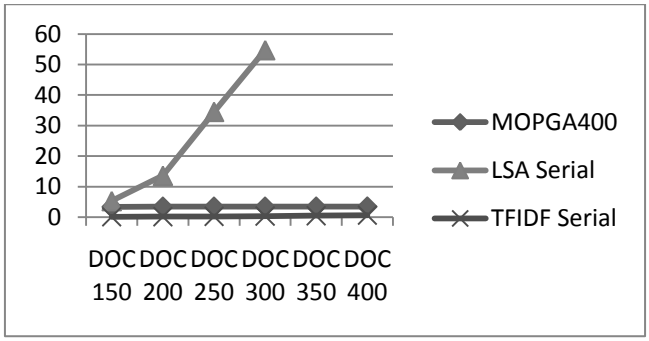

Fig 2.

Figure 2 shows that as data is added, the serial linear algebra libraries in the LSA Algorithm scales very poorly and even fails to execute with matrices that exceed some 10,000

X 300 cells. The MOPGA algorithm also has some initial load time due to the SVD decomposition in LSA but scales more like the TFIDF algorithm as the number of cycles increases.

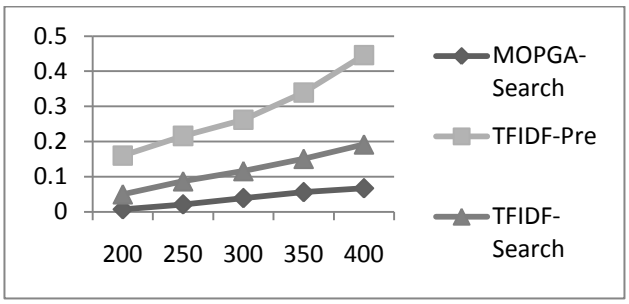

Fig 3.

Taking into consideration the total processing time and breaking up the processing times into pre-search and vector similarity computations demonstrate that LSA pre-compute times are several orders of magnitude greater than the search times so those values do not appear..

8 Conclusion

By combining TFIDF and LSA the MOPGA algorithm demonstrates that both the relevance and number of results returned to a user submitted search are improved and the scaling implications are much more favorable for the MOPGA algorithm in comparison to the LAS and TFIDF. The greatly reduced SVD decomposition time means that if necessary MOPGA could be re-indexed frequently lending itself to much more volatile databases. The MOPGA shows promise with much larger datasets where iterating over all possible results may not be possible. The Pareto Quad Tree is easily adapted to include other search domains lending itself to further refinement and experimentation.

Throughout the testing of this algorithm a constant K dimensional reduction of 100 was applied to the results. Many experiments show that altering this value can have a great effect on the quality and quantity of LSA searched. Some experimentation should also be conducted with the frequency with which genetic mutations are introduced for evaluation. Currently this is a constant 5% and this may not be optimal.

When the initial data set size grows beyond what fits on a single CUDA device, the data dpace database will need to be split before genetic operations are conducted. MOPGA Agents currently deal with an assigned piece of the data space database so further scaling experiments should be conducted to determine what the scaling limits are.

References

[1] Manning, C. D., Raghavan, P., & Schutze, H. (2008). *Term frequency and weighting*. from http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html Retrieved March 23, 2010.

[2] Ramos, J. *Using TF-IDF to Determine Word Relevance in Document Queries.* Piscataway, NJ: Rutgers University, 2001.

[3] Landauer, T., Foltz, P., & Laham, D. An Introduction to Latent Semantic Analysis. *Discourse Processes , 25* (2 & 3), pp. 259 – 284, 1998.

[4] Papdimitriou, C., Tamaki, H., Raghavan, P., & Vempala, S. "Latent semantic Indexing: a probabilistic analysis", In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* (pp. 159-168). Seattle, Washington, United States: ACM, 1998.

[5] Coello, C. A., Lamont, G. B., & Van Veldhuizen, D. A. "*Evolutionary Algorithms for Solving Multi-Objective Problems 2nd Ed.* New York: Springer Science + Buisness Media., 2007.

[6] Mostaghim, S., Teich, J., & Tyagi, A., "Comparison of Data Structures for Storing Pareto-sets in MOEAs" In *Proceedings of the 2002 World on Congress on Computational Intelligence. 1*, pp. 843-848. Honolulu, HI, USA: WCCI, 2002.

[7] Bhattacharya, M "Exploiting Landscape Information to Avoid Premature Convergence in Evolutionary Search" In *Proceedings of the 2006 IEE Congress on Evolutionary Computation* (pp. 560 - 564). Vancouver, BC: IEEE, 2006.

[8] Xu, Y., Deli, Y., & Yu, L., "Efficient Annealing - Inspired Genetic Algorithm for Information Retrieval from Web-Document", In Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, 2009.

[9] Mattson, T. G., Sanders, B. A., & Massingill, B. L. *Patters for Parallel Programming.* Boston: Addison-Wesley, 2005.

[10] NVIDIA Corporation. *CUDA Zone - Documentation.* Retrieved November 24, 2009, from CUDA Zone: http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf, accessed 2009, August 26.