Document Clustering Using Multi-Objective Genetic Algorithms with Parallel Programming Based on CUDA

Jung Song Lee[1], Soon Cheol Park[1], Jong Joo Lee[2] and Han Heeh Ham[3]
[1]Division of Electronics and Information Engineering, Chonbuk National University, Jeonju-si, Republic of Korea
[2]Department of Korean Language and Literature, Chonbuk National University, Jeonju-si, Republic of Korea
[3]Department of Archeology and Cultural Anthropology, Chonbuk National University, Jeonju-si, Republic of Korea

Keywords: Document Clustering, Genetic Algorithms, Multi-Objective Genetic Algorithms, GPGPU, CUDA.

Abstract: In this paper, we propose a method of enhancing Multi-Objective Genetic Algorithms (MOGAs) for document clustering with parallel programming. The document clustering using MOGAs shows better performance than other clustering algorithms. However, the overall computation time of the MOGAs is considerably long as the number of documents increases. To effectively avoid this problem, we implement the MOGAs with General-Purpose computing on Graphics Processing Units (GPGPU) to compute the document similarities for the clustering. Furthermore, we introduce two thread architectures (Term-Threads and Document-Threads) in the CUDA (Compute Unified Device Architecture) language. The experimental results show that the parallel MOGAs with CUDA are tremendously faster than the general MOGAs.

1 INTRODUCTION

Clustering is an unsupervised classification technique that partitions the input space into K regions. The document clustering, which is one part of a clustering, is important in the text mining field (Croft et al., 2009).

Currently, Genetic Algorithms (GA), which is one of the artificial intelligence algorithms, is widely used in document clustering. GA is a randomized search and optimization technique guided by the principles of evolution and natural genetics, and can be used to handle large and complex landscapes. It provides near optimal solutions (Maulik and Bandyopadhyay, 2000). Document clustering based on GA can provide appropriate cluster solutions using the searching capability of GA. The performance of the document clustering based on GA is better than other clustering algorithms (Song and Park, 2009). However, it slows down the performance of clustering, so it is not used to prevent premature convergence. To effectively avoid a premature convergence, Fuzzy Logic based on GA (FLGA), which exerts several control parameters to manipulate crossover probability and the mutation probability of GA, has been proposed (Song and Park, 2010). When the best fitness iterations reach the consecutive maximum generation number without improvement, the diversity of the population is extended by increasing the crossover and mutation probability. Generally, it can effectively avoid trapping in a local optimum and also accelerate the evolving speed. However, it depends on several control parameters to manipulate the crossover probability and the mutation probability, such as parameter dependence. To solve these problems (premature convergence, parameter dependence), document clustering using Multi-Objective Genetic Algorithms (MOGAs) has been proposed (Lee et al., 2011), (Lee et al, 2011), (Lee and Park, 2013). MOGAs define the document clustering problem as a Multi-Objective Optimization Problems (MOP) having two cluster validity indices. It uses two of MOGAs, NSGA-II (Deb et al., 2002) and SPEA2 (Zitzler et al., 2002) to solve MOP. Document clustering using MOGAs shows a higher performance than the other clustering algorithms. However, when these algorithms which are implemented by serial computing are applied in document clustering, the computational complexity is increased by the high time complexities of NSGA-II and SPEA2. In order to solve this problem, the document clustering MOGAs on MATLAB Distributed Computing Server (MDCS) (Lee and Park, 2012) has been proposed. But this technique requires many computer servers called nodes, and each node must

Lee J., Park S., Lee J. and Ham H..
Document Clustering Using Multi-Objective Genetic Algorithms with Parallel Programming Based on CUDA.
DOI: 10.5220/0005057502800287
In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2014), pages 280-287
ISBN: 978-989-758-039-0

set up specific and precise configuration of the MDCS.

Recently, parallel computing has been increasingly employed to increase the speed of computations. Parallel computing is the technique of using multiple compute resources simultaneously to solve a computational problem. So, we introduce the document clustering using MOGAs based on General-Purpose computing on Graphics Processing Units (GPGPU) to reduce the computational time in this paper. For this technique, we use CUDA (Compute Unified Device Architecture) language that is a platform developed by NVIDIA® for performing massively parallel computations on NVIDIA® Graphics Processing Unit (GPU) (The NVIDIA Corporation).

This paper is organized as follows. In the next section, we give a brief review of the document clustering using MOGAs. Details of the document clustering using MOGAs based on CUDA are described in Section 3. Section 4 shows the experimental results of the document clustering algorithms. Conclusions are given in Section 5.

2 DOCUMENT CLUSTERING USING MOGAS

2.1 Multi-Objective Optimization Problems

In the optimization problems, when there are several objective functions these problems are called Multi-Objective Optimization Problems (MOP). MOP has many solutions that optimize one objective function but does not optimize other objective functions (e.g. conflict among objectives). Therefore, it is almost impossible to simultaneously optimize all objective functions. The Pareto-based method is often used to solve MOP with this character. This method finds a set of solutions by the dominance relation between candidate solutions.

2.2 Multi-Objective Genetic Algorithms

Various algorithms have been suggested in order to solve the MOP. They are dependent on the initial search space and various solutions cannot be found. GA solves this disadvantage. GA for solving MOP is often called Multi-Objective Genetic Algorithms (MOGAs). Variations of MOGAs have been used in many applications and their performances were tested in several studies, i.e. PESA-II, SPEA2, NSGA-II, etc., representing leading research in this category. With these methods, NSGA-II and SPEA2 are easy to implement and do not have parameters for diversity in a population (Konak et al., 2006). So, we applied these algorithms to document clustering.

2.3 Document Representation

In order to perform document clustering, an important process is document representation. The general approach uses Vector Space Model (VSM) to represent documents. The document vector that represents the character of the document is formed by the weights of the terms indexed in a document (Choi et al, 2008). The following equation is the nth document vector whose size is 1 by t. t is the number of the total indexed terms in the corpus and W is the term weight.

$$d_n = \langle W_{n,1},\ W_{n,2},\ \cdots, W_{n,t} \rangle. \quad (1)$$

We extracted the indexed terms by using stop words and Porter's stemming, and calculated the term weight by Okapi's calculation (Salton and Buckley, 1988). In VSM we use cosine measure to compute the similarity between two documents (Xia et al., 2006). The cosine similarity between document d_1 and d_2 is defined by:

$$sim_{VSM}(d_1, d_2) = \frac{(\sum_{k=1}^{t} W_{1,k} \cdot W_{2,k})}{\left(\sqrt{\sum_{k=1}^{t} W_{1,k}^2} \cdot \sqrt{\sum_{k=1}^{t} W_{2,k}^2}\right)}. \quad (2)$$

2.4 MOP for Document Clustering

The document clustering using GA with a single objective function can be regarded as an optimization problem to optimize the cluster validity index. This view offers a chance to apply MOP on the clustering problem. Therefore, the document clustering was thought of as MOP optimizing two cluster validity indices through this trade-off relation as:

$$\arg\max_{C_i \in P} \left(F_{CH}(C_i) \wedge F_{DB}(C_i) \right), \quad (3)$$

where P is the population and $P = \{C_1, C_2, \ldots, C_i, \ldots, C_n\}$. C_i is a chromosome and $C_i = \{CN_1, CN_2, \ldots, CN_j, \ldots, CN_m\}$. CN_j is the cluster number assigned to a documents and $1 \leq CN_j \leq K$, n is the number of chromosomes in a population, m is the number of documents and K is the number of clusters. F_{CH} and F_{DB} are indicated as CH index (Calinski and Harabasz, 1974) and DB index (Davies and Bouldin, 1979) for the objective functions of MOGA. Objective function prescribes the optimality of a solution in GA. A clustering validity index was used as the objective function and determines the

optimal partition and the optimal number of clusters. If the cluster validity index has the minimum value or maximum value, optimal clustering will result.

2.5 Chromosome Encoding and Evolution Principles

In document clustering using MOGAs, data should be close to gene structure through the encoding process. The chromosome is encoded by a string of integers. Each chromosome in the population is initially encoded by a number of m genes with an integer value randomly in the range 1~K, where m is the number of documents and K is the number of clusters. Thus, each gene represents a document, and the value of a gene represents a cluster number.

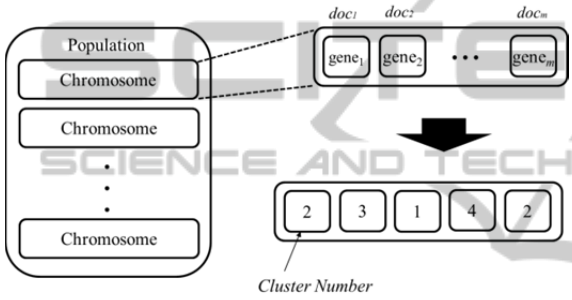

Figure 1: Chromosome encoding for the document clustering.

For example, in Figure 1, for m = 6 and K = 4, the encoding {2, 3, 1, 4, 2} allocates the first document to the second cluster, the second document to the third cluster and the third document to the first cluster, and so on. That is, the document clustering using MOGAs finds the optimal cluster group of each document. Also, Multi-Point Crossover and Uniform Mutation were adopted in the evolution operators.

3 DOCUMENT CLUSTERING USING MOGAS BASED ON CUDA

3.1 Basic CUDA

This section introduces the main concepts behind the CUDA (Compute Unified Device Architecture) programming model. CUDA is a platform developed by NVIDIA® for performing massively parallel computations on NVIDIA® Graphics Processing Unit (GPU). It enables dramatic increases in computing performance by the power of the GPU and provides the ability to use high-level languages such as C to develop the application. The basic architecture of CUDA is as Figure 2.

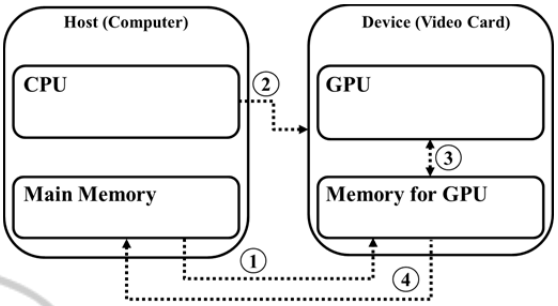

Figure 2: The basic architecture of CUDA.

(1) Copy input data from the Main Memory of Host to Memory for GPU in Device
(2) Instruct the processing (Load GPU program)
(3) Execute parallel in each Core of GPU
(4) Copy results from Main Memory in HOST to Memory for GPU in Device

As illustrated by Figure 3, parallel portions of an application execute on the device as kernels. A kernel function that is called by the host is the basic unit of work on GPU, and it is executed by an array of threads in parallel. All threads execute the same code and these are grouped into blocks. Blocks are organized into a grid and a kernel is executed as a grid of blocks of threads.

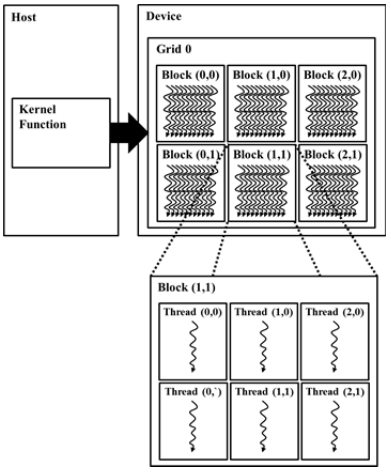

Figure 3: Thread hierarchy in CUDA.

Threads may access data from multiple memory spaces during their execution, as illustrated by Figure 4. Per-thread local memory is private to the thread. Per-block shared memory is shared by all threads of the block and same lifetime as the block. Threads can safely share data through this memory.

Global memory is device level memory that is shared by all threads.

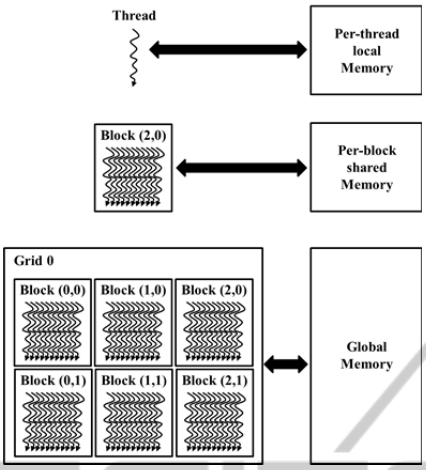

Figure 4: Memory hierarchy in CUDA.

3.2 Issue of Document Clustering using MOGAs based on CUDA

The procedure of document clustering using MOGAs is shown in Figure 5. First, documents are represented by using IR techniques (Stop Word Remove, Porter's Stemming). Second, the documents are clustered by using MOGAs, NSGA-II and SPEA2.

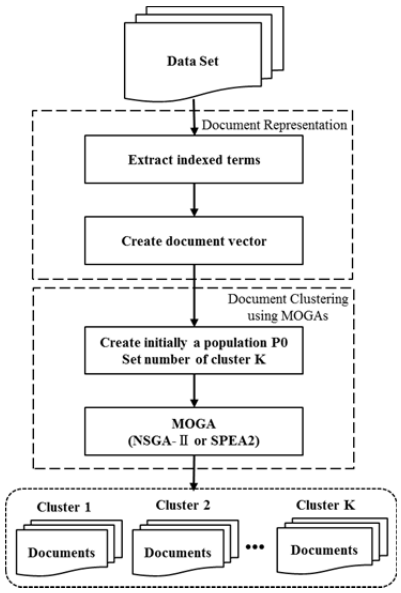

Figure 5: Procedure of document clustering using MOGAs.

The performance of the document clustering using NSGA-II and SPEA2 among MOGAs is better than the other clustering algorithms (*k*-means, conventional GA). The time complexities of NSGA-II and SPEA2 are $O(MN^2)$ and $O(MN^2 \log N)$ respectively, where M is the number of objective functions and N is the population size. MOGAs using the cluster validity indices as the objective functions require higher computational complexity. Moreover, in order to perform document clustering, an important process is document representation.

The general approach uses VSM to represent documents. VSM has many drawbacks; because each unique term in the vocabulary represents one dimension in feature space, VSM needs a large number of features to represent high dimensions, and it can easily cause the high cost of computational time. To overcome these problems, we adopted the data parallel computations on kernel function in CUDA to the calculating the objective function of the population.

In Figure 5, evolution operators (initial population, selection, crossover, mutation) of MOGAs are executed by serial code in host. The objective functions of each chromosome in population are calculated by parallel code with CUDA in device. We propose two architectures which are called **Term-Threads** and **Document-Threads**. We designed the implementation process described detailedly as following.

Term-Threads

Step 1 : i^{th} chromosome in a population at generation p, centroid vectors, and document vectors allocate to global memory in device.

Step 2 : Set block size n. n is the number of documents and each block indicates value of gene. Consequently, x^{th} block indicates n^{th} document vector (D_n). For example, first block allocates to the first document vector, second block allocates to the second document vector and so on.

Step 3 : Set threads size t in each block. Each thread indicates term weight ($W_{n,t}$) in document vector (D_n).

Step 4 : Start computation of each block; calculate cosine similarity between component of document vector ($W_{n,t}$) and component of centroid vector ($C_{k,t}$) in each thread.

Step 5 : Finally, the result of cosine similarity between document vector (D_n) and centroid vector (CV_k) allocates to the shred memory in each block. So, we use it for clustering validity index (CH index, DB index).

Consequently, in **Term-Threads** architecture as shown Figure 6, genes of i^{th} chromosome are assigned to each block. So, it simultaneously calculates n documents cosine similarity in an i^{th} chromosome and number of iteration is population size.

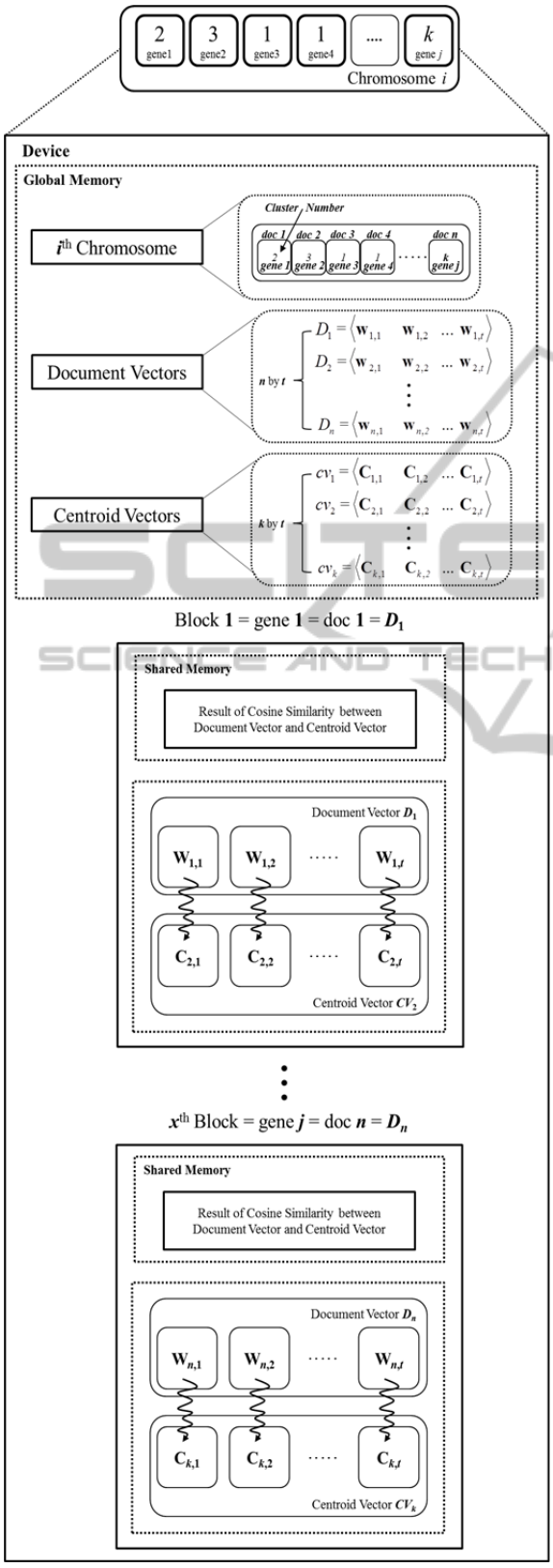

Figure 6: Term-Threads architecture.

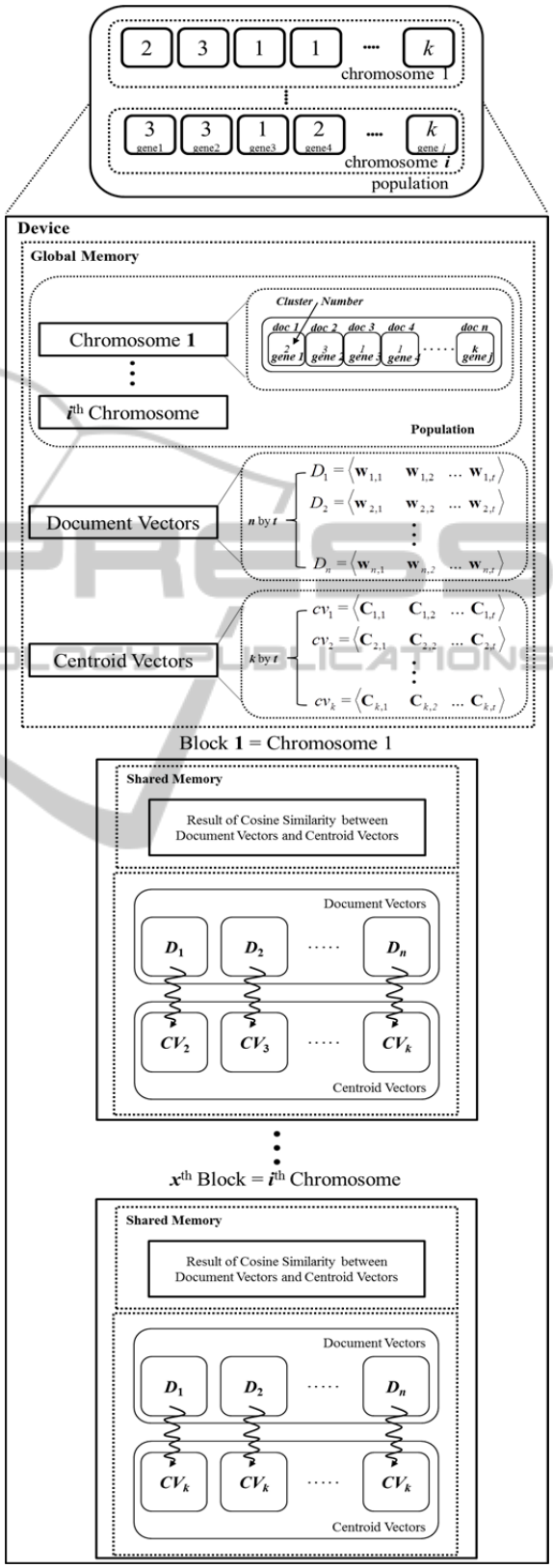

Figure 7: Document-Threads architecture.

Document-Threads

Step 1 : population at generation p, centroid vectors, and document vectors allocate to global memory.

Step 2 : Set block size i. i is the number of chromosome in population at generation p and x^{th} block indicates i^{th} chromosome. For example, first block allocates to the first chromosome, second block allocates to the second chromosome and so on.

Step 3 : Set threads size n in each block. Each thread indicates value of gene in i^{th} chromosome. Consequently, n^{th} thread indicates n^{th} document vector (D_n).

Step 4 : Start computation of each block; calculate cosine similarity between document vector (D_n) and centroid vector (CV_k) in each thread.

Step 5 : Finally, the result of cosine similarity between document and centroid vector allocates to the shred memory in each block. So, we use it for clustering validity index (CH index, DB index).

While in **Document-Threads** architecture as shown Figure 7, chromosomes of population at generation p are assigned to each block. So, it simultaneously calculates $n \times i$ documents cosine similarity in a generation p.

4 EXPERIMENT RESULTS

In this section, we implement our method of MOGAs on CUDA for document clustering on the Reuters-21578 collection, which is one of the most widely adopted benchmark datasets in the text mining field, and compare and discuss the performance of MOGAs on CUDA with conventional GA. Also, we use F-measure (Fragoudis et al., 2005) to evaluate the performances of these clustering algorithms. The population number in our conventional GA and MOGAs is 300. These algorithms are terminated when the number of generations reaches 1,000, or when the iterations without improvement reach consecutive 20. In the current test data set 1, containing 100 documents from three topics (*acq*(30), *crude*(30), *trade*(40)), data set 2 containing 200 documents from four topics (*coffee*(50), *trade*(50), *crude*(50), *sugar*(50)), and data set 3 containing 300 documents from six topics (*coffee*(50), *trade*(50), *crude*(50), *sugar*(50), *grain*(50), *ship*(50)) are selected. After being processed by word extraction, stop word removal, and Porter's stemming, there are 1019, 3436 and 4210 index terms, respectively. Also, the index term weight that is extracted by using the Okapi's calculation was determined.

We implement our experiments in two steps: Firstly, by comparing the performances of the MOGAs to those of the other clustering algorithms for the different data sets in serial code. Secondly, by comparing the computational times of these algorithms for the same data sets then we design GPU parallel code and compare the computational times with CPU serial code.

Figure 8 shows the performances of clustering algorithms with the different data sets. GA indicates the conventional GA with single objective function. NSGA-II and SPEA2 indicate the MOGAs with two objective functions. DB and CH indicate objective functions applied in the clustering algorithms. DB stands for DB index and CH for CH index. In summary, the clustering performances in all data sets are highest F-measure when using NSGA-II and next, SPEA 2. With each data set, the average F-measure value of MOGAs using the NSGA-II and SPEA2 is **0.79**, **0.75** and **0.80**. Consequently, the document clustering applying the MOGAs shows the performance about **21%** than conventional GA.

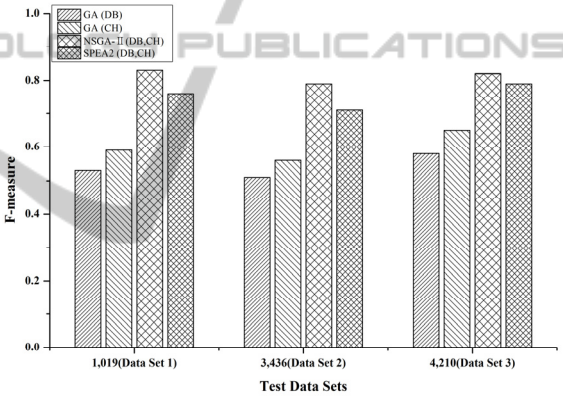

Figure 8: The clustering performance for each data set.

However, as a result of Table 1 the computational time of MOGAs is greater than GAs. So, we adopted CUDA for reduce computational times in document clustering using MOGAs. We evaluate performance of our MOGAs base on CUDA implementation using GeForce GTX 550Ti video cards with 192 cores and 1 GB of RAM DDR3. GeForce GTX 550 Ti consists of four Streaming Multiprocessors (SM). Each SM consists of eight processor cores called Streaming Processors (SP). In effect, GeForce GTX 550 Ti is a massively-parallel multi-core processor embodying 32 SPs in total and maximum number of active threads on each block is 1,024, in total more than 130,000 threads. The computer used for experiments was a desktop PC with Intel Core i7-2600 to 3.4 GHz, 8 GB of RAM, operating system 64-bit Microsoft Windows 7 sp1.

Table 1: The computational time of MOGAs and general GA (minutes).

	Test Data Sets		
	Data Set 1	Data Set 2	Data Set 3
GA(DB)	34	110	237
GA(CH)	52	130	253
NSGA-II	84	162	334
SPEA 2	93	178	350

Table 2 shows the computational times of each data set of MOGAs using serial code and MOGAs using parallel code based on CUDA with two thread architecture, **Term-Threads** and **Document-Threads**.

Table 2: The computational time of MOGAs based on CUDA, general MOGAs (seconds).

			Test Data Sets		
			Data Set 1	Data Set 2	Data Set 3
Serial Code	NSGA-II		5,040	9,720	20,040
	SPEA 2		5,580	10,680	21,000
Parallel Code	NSGA-II	Term-Threads	120	–	–
		Document-Threads	7	24	220
	SPEA 2	Term-Threads	122	–	–
		Document-Threads	8	25	220

From Table 2, we can see that the **Term-Threads** and **Document-Threads** architecture are faster than serial code. Also, CUDA helps to reduce computational times substantially. Especially, **Document-Threads** architecture can get **600x**, **400x**, and **90x** speed-up in each data sets respectively. However, in Table 2, **Term-Threads** architecture is impossible to implement over 1,024 threads because maximum number of active threads per block is 1,024 on GeForce GTX 550 Ti. That is, **Term-Threads** architecture is impossible to execute when number of term is over 1,024.

5 CONCLUSIONS

We have presented document clustering using MOGAs based on GPGPU to solve the problem of long computational time in the general MOGAs. The document similarity of the objective functions in MOGAs is computed with GPGPU using the parallel computing platform of CUDA.

The results show that these MOGAs enhanced the clustering performance about 12% higher than those of both the general GA. Furthermore, two architectures, **Term-Threads** and **Document-Threads**, show faster execution times than general MOGAs. Especially, **Document-Threads** architecture gains over **600x** speed-up and handles more than 1,000 documents simultaneously with **Term-Threads** architecture.

In future, we will do more work on some optimization techniques of the document clustering in CUDA, seeking the best of threads architectures for document clustering.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea Grant funded (NRF) by the Korean Government (NRF-2013S1A5B8A01072201 & NRF-2013R1A1A2063572) and the Brain Korea 21 PLUS Project, National Research Foundation of Korea.

REFERENCES

Croft, W. B., Metzler, D., and Strohman, T., 2009. *The book*, Search engines information retrieval in practice. Addison Wesley.

Maulik, U., and Bandyopadhyay, S., 2000. Genetic algorithm-based clustering technique. Pattern Recognition, 33(9):1455-1465.

Song, W., and Park, S.C., 2009. Genetic algorithm for text clustering based on latent semantic indexing. *Computers and Mathematics with Applications*. 57:1901-1907.

Song W., and Park, S.C., 2010. Latent semantic analysis for vector space expansion and fuzzy logic-based genetic clustering. Knowledge and Information Systems. 22:347-369.

Lee, J. S., Choi, L. C., and Park, S. C. 2011. Document clustering using multi-objective genetic algorithm with different feature selection methods. 1st International Workshop on Semantic Interoperability.

Lee, J. S., Choi, L. C., and Park, S. C., 2011. Multiobjective genetic algorithms, NSGA-II and SPEA2, for document clustering. *Communications in Computer and Information Science*. 257:219-227.

Lee, J., S. and Park, S., C. 2013. Generation of Non-redundant Summary based on Sentence Clustering Algorithms of NSGA-II and SPEA2. *The 4th International Conference on Evolutionary Computation Theory and Applications*. 176-182.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., 2002. A fast elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*. 6(2):182-197.

Zitzler, E., Laumanns, M., and Thiele, L., 2002. SPEA2: Improving the strength pareto evolutionary algorithm.

Proceedings of the EROGEN.

Lee, J., S. and Park, S., C. 2012. Document Clustering Using Multi-Objective Genetic Algorithms on MATLAB Distributed Computing. *The 3rd International Conference on Information Science and Applications*.

The NVIDIA Corporation. 2012. *The book*, NVIDIA C.U.D.A. Programming guide.

Konak, A., Coit, D. W and Smith, A. E., 2006. Multi-objective optimization using genetic algorithms : A tutorial. *Reliability Engineering and System Safety*. 91:992-1007.

Choi, L.C., Choi, K.U., and Park, S.C., 2008. An automatic semantic term-network construction system. In *International Symposium on computer Science and its Applications*.

Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Information Processing & Management.

Xia, H., Wang, S., and Yoshida, T., 2006. A modified ant-based text clustering algorithm with semantic similarity measure. *Journal of Systems Science & Systems Engineering*. 15(4):474-492.

Calinski, T., Harabasz, J., 1974. A dendrite method for cluster analysis. *Communucations in Statistics*.

Davies, D.L., and Bouldin, D.W., 1979. A cluster separation measure. *IEEE transactions on Pattern analysis and Machine Intelligene*.

Fragoudis, D., Meretakis, D., and Likothanassis, S., 2005. Best terms:an efficient feature-selection algorithm for text categorization. *Knowledge and Information Systems*.