The influence of data size on a high-performance computing memetic algorithm in fingerprint dataset

Priati Assiroj[1], Harco Leslie Hendric Spits Warnars[2], Edi Abdurachman[3], Achmad Imam Kistijantoro[4], Antoine Doucet[5]
[1,2,3]Computer Science Department, BINUS Graduate Program-Doctor of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia
[4]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, West Java 40132, Indonesia
[5]Laboratoire L3i-Université de La Rochelle, Avenue Michel Crépeau, F-17 042 La Rochelle Cedex 1, France

Article Info

Article history:

Received Dec 31, 2020
Revised Apr 29, 2021
Accepted Jun 1, 2021

Keywords:

Biometric recognition
Fingerprint identification
High performance computing
Memetic algorithm

ABSTRACT

The fingerprint is one kind of biometric. This biometric unique data have to be processed well and secure. The problem gets more complicated as data grows. This work is conducted to process image fingerprint data with a memetic algorithm, a simple and reliable algorithm. In order to achieve the best result, we run this algorithm in a parallel environment by utilizing a multi-thread feature of the processor. We propose a high-performance computing memetic algorithm (HPCMA) to process a 7200 image fingerprint dataset which is divided into fifteen specimens based on its characteristics based on the image specification to get the detail of each image. A combination of each specimen generates a new data variation. This algorithm runs in two different operating systems, Windows 7 and Windows 10 then we measure the influence of data size on processing time, speed up, and efficiency of HPCMA with simple linear regression. The result shows data size is very influencing to processing time more than 90%, to speed up more than 30%, and to efficiency more than 19%.

Corresponding Author:

Priati Assiroj
Computer Science Department, Binus Graduate Program-Doctor of Computer Science
Bina Nusantara University
Jl. Raya Kebon Jeruk No.27, DKI Jakarta 11480, Indonesia
Email: priati@binus.ac.id

1. INTRODUCTION

Nowadays, the growth of data and information cause scientists and researchers from various fields enter to an era that the requirement of computation resources and data storage capacity exceeds the available capacity. Scientists and researchers are more aware to utilize the computer system in their researches. This condition causes more effort to create the systems that available to run in large-scale computation to process the big data.

Fingerprint identification becomes an interesting research topic for two decades [1]. In this work, we use a memetic algorithm that runs in a parallel system to identify fingerprints. Parallel computation is a computation technique that runs by utilizing several computer resources simultaneously, actually caused by the required computation is very large such as to process big data or in a large computation process. In this computation model, the problem complexities are divided into smaller parts and run in a parallel environment.

The data that have a high complexity is fingerprint data and its problem is equal to the amount of fingerprint dataset, it needs a superfast process in identification. The memetic algorithm [2] is an improvement of the evolutionary algorithm with a separate local search [3]. A memetic algorithm is a simple algorithm with reliable performance [4], [5], generates high-quality solutions to solve problems in the real world [6]-[8].

The speed is a reason for the selected algorithm, the faster will be selected than the slower algorithm [9]. To process a high scale and big data in a reasonable time, we need a high-performance computation system. The effective and efficient time to simulate, compute and the process is a must, besides the quality and accuracy of the generated information must be maintained. The board of management in an organization needs fast and high-quality information to make a decision in the production process and to purchase raw materials for the next periods.

To generate high quality and fast information, it needs a system with specific hardware that supports the process of large scale data quickly and has a high performance, with the client-server based application and distributed database that accessible across the entire computers in the local or public computer network.

The advances in various fields of science require computer systems with high performance in speed and computing capacity. The implication is the technology of personal and supercomputer increases rapidly. The main obstacles of supercomputers are procurement cost, operation, and maintenance, and the alternative is parallel processing. A parallel distributes a work package that will be processed by all the entire computers in the system. With this parallel system, the investment cost can be reduced. Note that this system has high flexibility to adapt to the changes in computer technology. Users can customize the system based on their purposes. To get a fast computation process, it only needs to upgrade the processors and RAM without storage media in every computer, and for the application that produces a lot of data, it only needs to upgrade the storage media.

There are two ways to aim an efficient computation time in a high-performance computation (HPC) system, firstly is to produce a high-speed processor, and secondly is run the application in a parallel environment with multi-processors. For the first way, the processor manufacturer will meet a difficulty because the lithography technique is almost reaching the limit. The newest processor is made with 45nm fabrication technology and if it is reduced the processor's reliability will also reduce. Therefore, the big chance to improve the computation speed with a high possibility is a parallel computation technique [10].

HPC is a method to address the problem with high complexity related to workload and a large number of data [11]. One of the techniques in HPC is parallel computation [10]. A parallel processing system is a group of connected computers that working together as an integrated computer system to address the same problem with one goal [12].

2. PARALLEL COMPUTING ARCHITECTURE

Based on the instruction and data stream, the computer categorized into 4 groups, single instruction stream, single data stream (SISD), single instruction stream, multiple data streams (SIMD), multiple instruction streams, single data stream (MISD), and multiple instruction streams, multiple data stream (MIMD) [13]. There are several styles in parallel programming:

2.1. Single program, multiple data (SPMD)

Data and programs are distributed to each processor and the execution is scheduled. Each processor executes the same program but the processed data is different.

2.2. Master-slave

A processor as a master and several processors as slaves.

2.3. Multiple program, multiple data

Data and programs are distributed to each processor. Every processor executes a different program and data. The parallel computation system is included in the MIMD group, this group can be divided into a multi-processor system and multi-computer system. A multi-processor system is a parallel computing system that is based on the single memory utilization at the same time simultaneously. A multi-computer system is a parallel computer system with an independent processor and RAM in every computer. In this paper, we propose the high-performance computation using memetic algorithm (HPCMA) for fingerprint identification.

3. RESLATED RESEARCH

The related conducted researches are the researches about fingerprint identification that has been conducted by other researchers. [14] conducts research to identify fingerprints in the big data framework with a distributed model. [15] states that a memetic algorithm can improve efficiency, reduce memory consumption, and has a better ability to utilize the resource system. In the research [16], the memetic algorithm is used to do a feature selection in handwritten word recognition. Moscato *et al.* [17], explained that a memetic algorithm can outperform the proposed method even this algorithm needs more computation time and also generates a high-quality solution. Feng *et al.* [18], uses a memetic algorithm to do a treatment plan faster and [19] proposes a memetic fingerprint matching algorithm (MFMA) without local matching to do a fingerprint matching. The MFMA significantly reduces the generation that has to be identified [19]. To design a memetic algorithm, the considered problem is optimization as a specific problem [20]. Assiroj *et al.* [21], use the original memetic algorithm to process the fingerprint dataset and this algorithm works properly. This algorithm is also could be parallelized, Mirsoleimani *et al.* [22], implements parallel type on the graphics processing unit (GPU). This technique solves task scheduling problems for several multi-processing systems as also conducted by [23]. Island model of parallel memetic algorithms was proposed by [24]-[27] with dynamic local search.

4. METHOD

In this work, we propose a high-performance computing memetic algorithm (HPCMA) method. We run the original memetic algorithm in HPC mode. In Figure 1 is a framework of HPCMA. According to Figure 1, we modify the original memetic algorithm to run in HPC as a parallel condition. We use this HPCMA framework to process the image fingerprint dataset and here are the steps:

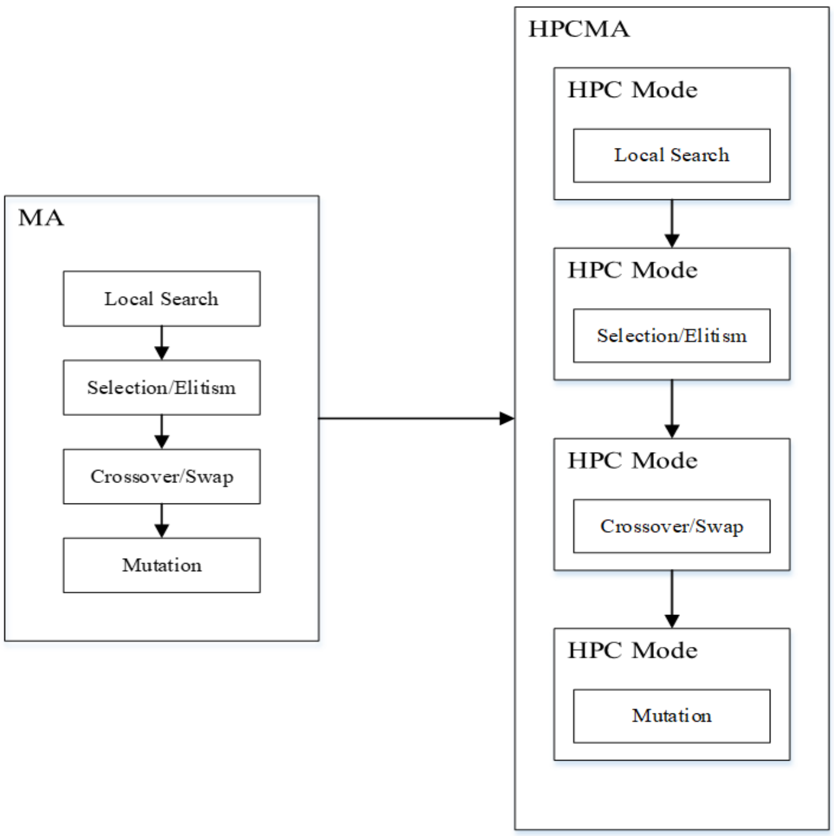

Figure 1. HPCMA framework

4.1. Local search in HPC mode

This process is to read all the entire file and folder image datasets that have been divided into four groups. After this reading process, the algorithm will convert all the image data. Firstly algorithm converts the image to an array string then secondly, the algorithm converts the string array to binary code. When this conversion is finished algorithm compares the number of converted data to all image fingerprint data and if it gets the same number process will be continued to the next selection, if not the process will wait until the local search process is complete.

4.2. Selection in HPC mode

We use 2% of the population as the sample randomly. These 140 parents candidates will be divided into 2 groups, male and female then compare the number of the selected candidate to the number of data selection samples. If it gets the same number process will be continued to the next crossover and if not the process will wait until the selection process is complete.

4.3. Crossover in HPC mode

Crossover is a mating process for all the entire parent candidates to get new offspring. Each member of the male population will be crossed to all members of the female population. This crossover process will be looped until all the entire membership of both population, male and female, are well crossed then compare the number of crossed data to the number of multiplication of male and female, if it gets the same number process will be continued to the Next Mutation and if not the process will wait until crossover is complete.

4.4. Mutation in HPC mode

This is the final process of the memetic algorithm. A mutation is a process that reverses the value of the binary code of the generated offspring from the crossover process. The value 1 in binary code will be reversed into 0 and 0 will be reversed into 1. Therefore we will get the newest and highest quality offspring. When the mutation process is finished, the algorithm will measure the number of the mutated data and compare it to the generated offspring from a crossover, if it gets the same number process will be finished and if not the process will wait until the mutation is complete. Based on Figure 2, the left side, MA, is Memetic algorithm in original condition, and on the right side, HPCMA is a memetic algorithm that runs in HPC utilizes the threads feature of processors.

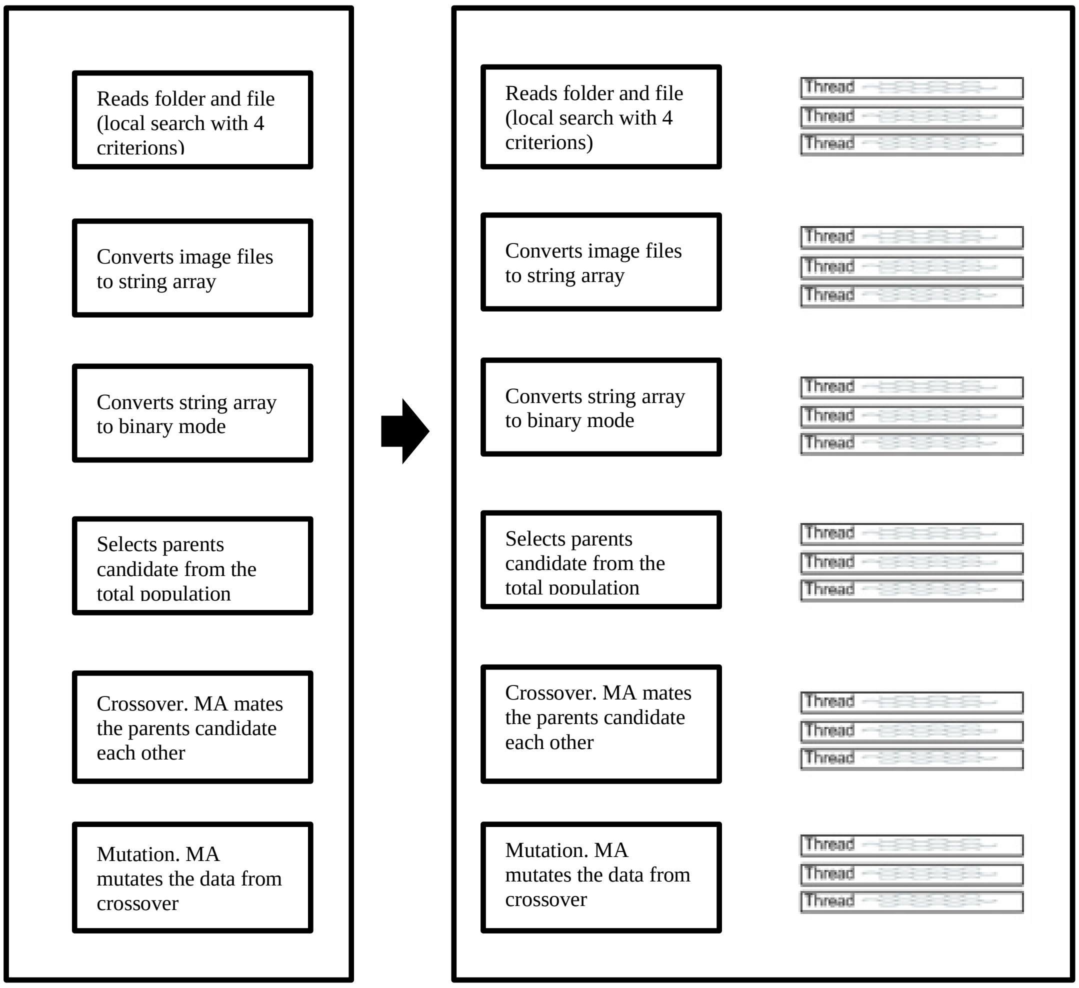
Reads folder and file (local search with 4 criterions)
Converts image files to string array
Converts string array to binary mode
Selects parents candidate from the total population
Crossover. MA mates the parents candidate each other
Mutation. MA mutates the data from crossover
Reads folder and file (local search with 4 criterions)
Thread
Thread
Thread
Converts image files to string array
Converts string array to binary mode
Selects parents candidate from the total population
Crossover. MA mates the parents candidate each other
Mutation. MA mutates the data from crossover

Figure 2. Illustration MA to HPCMA

This work implements the FVC2006 with data 7200 fingerprint data then categorized into 4 characteristics. Firstly is a full-sized image, secondly is 60% with dark color boundaries, thirdly is 60% with bright color boundaries, and fourthly is 80% with bright color boundaries and unclear image. Then we make 15 specimens from the combination of data. 1st specimen consists of 7200 fingerprint data, 2nd specimen consists of 1800 fingerprint data, 3rd specimen consists of 1800 fingerprint data, 4th specimen consists of 1800 fingerprint data, 5th specimen consists of 1800 fingerprint data, 6th specimen consists of 3600 fingerprint data, 7th specimen consists of 3600 fingerprint data, 8th specimen consists of 3600 fingerprint data, 9th specimen consists of 3600 fingerprint data, 10th specimen consists of 3600 fingerprint data, 11th specimen consists of 3600 fingerprint data, 12th specimen consists of 5400 fingerprint data, 13th specimen consists of 5400 fingerprint data, 14th specimen consists of 5400 fingerprint data, and the last specimen, 15th, consists of 5400 fingerprint data.

5. RESULT AND DISCUSSION

This work uses a 7200 synthetic fingerprint dataset from FVC2006 and runs in the computer system with Intel i5 2540M 2.6GHz 4 core and 16GB RAM, 500GB SSD as HPCMA machine and computer system with Intel i5 2430M 2.4GHz 4 core and 8GB RAM, 250GB SSD as database machine. Testing begins with data mapping and thread creation in each computer with different numbers of data. With more data to be processed and more created threads, the mapping time is also longer.

In this work, we compare the test in two environments of operating systems. The first is the Windows 7 operating system and the second is Windows 10 operating system. Data are divided into fifteen specimens with each character to see the data holistically then we measure the size of each specimen and measure the speed up and efficiency. Below are the results of the experiment from each operating system. Table 1 and Table 2 are a list of data size for each specimen, speed up, and efficiency of HPCMA on Windows 7 and Windows 10. Figure 3 is the speed-up visualization of each specimen in Windows 7, and Figure 4 is the speed-up visualization for each specimen in Windows 10.

Table 1. Experiment result in Windows 7

Specimen	Data Size	Speed up (ms)	Efficiency
1	22.8GB	249.0038057	10.37067904
2	0.237 GB	34.55627211	2.053812416
3	4.8 GB	294.3173384	13.56162175
4	4.3 GB	269.0168797	12.45388033
5	2.4 GB	160.8715959	7.943400165
6	8.8 GB	288.9740842	12.50843364
7	7.8 GB	266.42708	11.6195177
8	4.5 GB	162.3858895	7.275164187
9	11.4 GB	301.0793305	13.35765794
10	8 GB	200.1693822	9.272002684
11	7.6 GB	191.844378	8.976055725
12	17.2 GB	306.8103217	12.95318989
13	13 GB	241.1332293	10.30241264
14	15.7 GB	249.0009356	10.99598499
15	12.2 GB	224.7810952	9.629383921

Table 2. Experiment result in Windows 10

Specimen	Data size	Speed Up (ms)	Efficiency
1	22.8 GB	241.3684533	9.843870006
2	0.237 GB	31.73842967	1.917808586
3	4.8 GB	274.3291009	12.76242446
4	4.3 GB	237.2079802	11.17275106
5	2.4 GB	152.7474748	7.587524869
6	8.8 GB	268.1426055	11.68635096
7	7.8 GB	238.2088843	10.37998272
8	4.5 GB	156.0196491	6.972254909
9	11.4 GB	271.8752708	12.19454521
10	8 GB	187.7135075	8.90476869
11	7.6 GB	149.3226829	7.231500221
12	17.2 GB	275.2537613	11.52186279
13	13 GB	218.8803572	9.308268217
14	15.7 GB	236.1300983	10.44332603
15	12.2 GB	206.8560388	8.757247622

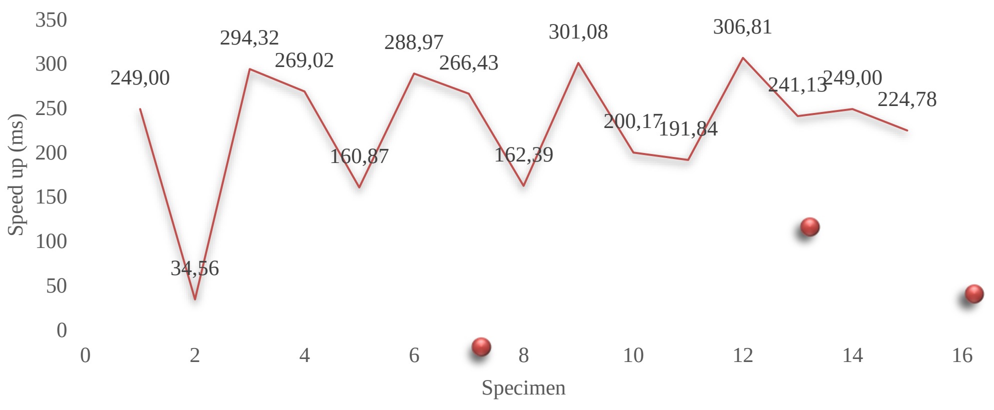

Figure 3. Speed up of HPCMA on Windows 7

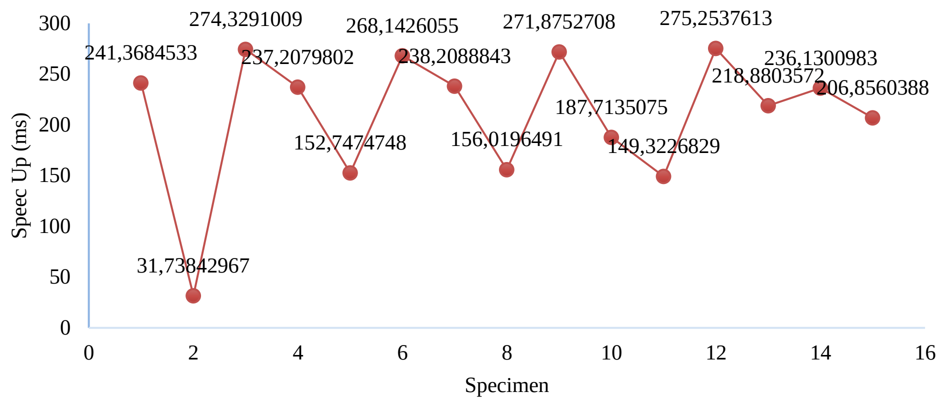

Figure 4. Speed up of HPCMA on Windows 10

Figure 5 is a visualization of HPCMA efficiency for each specimen in Windows 7. The efficiency of HPCMA in specimen 1 is 10.37067904, and in specimen 2 is 2.053812418. The efficiency of HPCMA in specimen 3 to specimen 15 is also displayed in Figure 5.

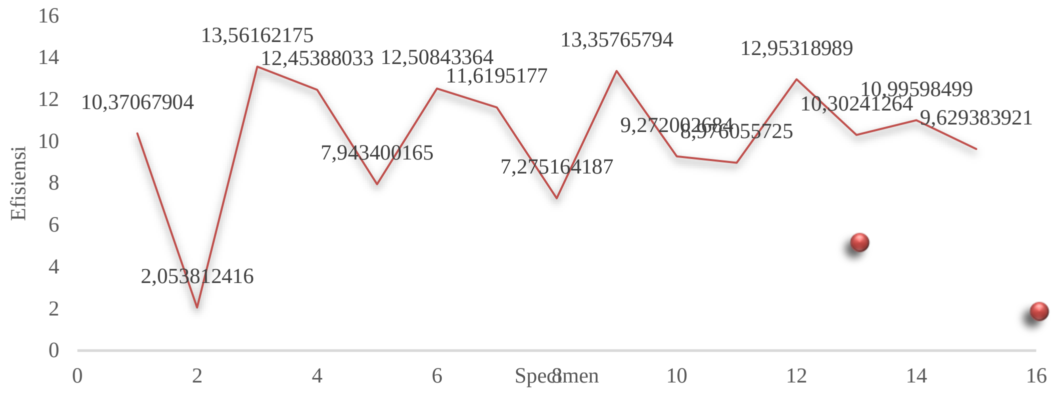

Figure 5. Efficiency on Windows 7

Figure 6 is a visualization of HPCMA efficiency for each specimen in Windows 10. The efficiency of HPCMA in specimen 1 is 9.84387006, and in specimen 2 is 1.917808586. The efficiency of HPCMA for specimen 3 to specimen 15 is also displayed in Figure 6.

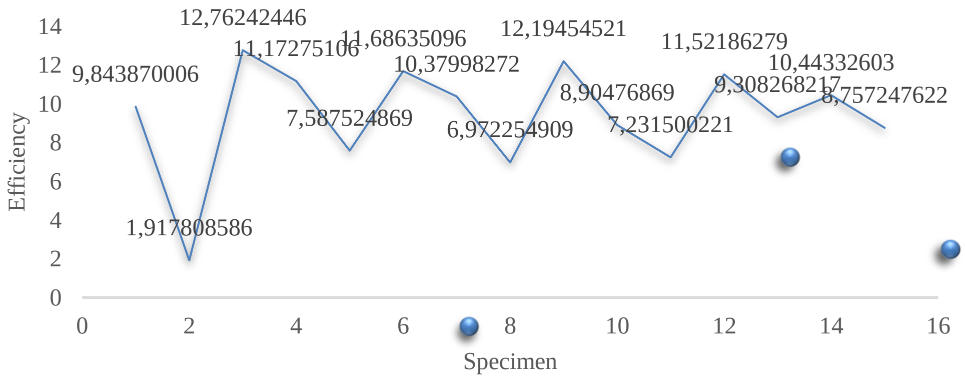

Figure 6. Efficiency on Windows 10

Visualization of the influence of data size with processing time. The bigger data size needs a longer processing time and the smaller data size is faster to be processed. From figure 4 above, specimen 1 with 22.8GB data size needs 72.904 seconds, and specimen 2 with 0.237GB data size only needs 8.347 seconds. The performance of HPCMA in Windows 7 and Windows 10 is almost similar. For example, HPCMA processed specimen 1 in 72.904 seconds in Windows 7 and 80.982 seconds in Windows 10 shown in Figure 7.

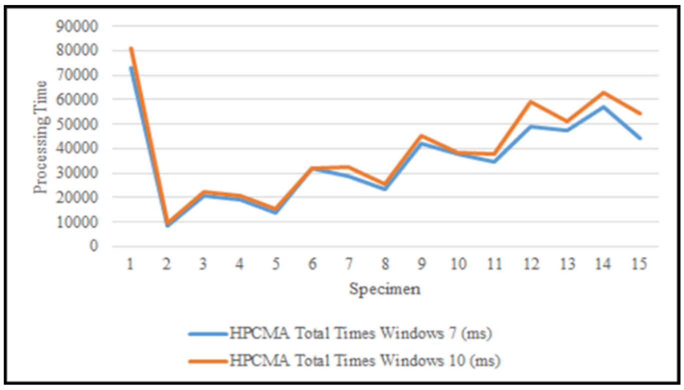

Figure 7. Processing time of each Specimen

6. CONCLUSION

In the simple linear regression, the experiment result of data size influence to HPCMA's processing time in Windows 10 is 0.937 or 93.7%. It means data size is very influential to HPCMA's processing time in Windows 10 for 97% and 6.3% depends on other variables. For Windows 7, data size is very influential to HPCMA's processing time for 95.9% and 4.1% depends on other variables. The experiment result of data size influence to HPCMA's efficiency in Windows 10 is 0.195 or 19.5%. It means data size is only influencing efficiency for 19.5%, and 80.5% depends on other variables. For Windows 7, data size is influencing efficiency for 19.3%, and 80.7% depends on other variables. The experiment result of data size influence to HPCMA's speed up on Windows 7 is 0.286 or 28.6%. It means data size is only influencing speed up for 28.6%, and 71.4% depends on other variables. For Windows 10, data size in influencing speed up for 31.7%, and 68.3% depends on other variables. On the other hand, data size is very influential to HPCMA's processing time in Windows 7 and Windows 10 about 90%. It influences about 30% on speed up and not for efficiency in Windows 7 or Windows 10.

ACKNOWLEDGEMENTS

This work is supported by Research and Technology Transfer Office, Bina Nusantara University as a part of Bina Nusantara University's International Research Grant entitled MEMETIC ALGORITHM IN HIGH-PERFORMANCE COMPUTATION with contract number: No.026/VR.RTT/IV/2020 and contract date: 6 April 2020.

REFERENCES

[1] A. K. Jain and J. Feng, "Latent Fingerprint Matching," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 88-100, Jan. 2011, doi: 10.1109/TPAMI.2010.59.
[2] P. Moscato, "Memetic Algorithms: A Short Introduction," *New ideas in optimization*, pp. 219-234, 1999.
[3] J. Lin and Y. Chen, "Analysis on the Collaboration Between Global Search and Local Search in Memetic Computation," in *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 608-623, Oct. 2011, doi: 10.1109/TEVC.2011.2150754.
[4] P. Merz and B. Freisleben, "Fitness Landscapes and Memetic Algorithm Design," *Electrical Engineering*, pp. 1-19, 1999.
[5] Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu and Kok-Wai Wong, "Classification of adaptive memetic algorithms: a comparative study," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 1, pp. 141-152, Feb. 2006, doi: 10.1109/TSMCB.2005.856143.
[6] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore and M. Sumner, "A Fast Adaptive Memetic Algorithm for Online and Offline Control Design of PMSM Drives," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 28-41, Feb. 2007, doi: 10.1109/TSMCB.2006.883271.

[7] M. Gong, Z. Peng, L. Ma and J. Huang, "Global Biological Network Alignment by Using Efficient Memetic Algorithm," in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 6, pp. 1117-1129, 1 November 2016, doi: 10.1109/TCBB.2015.2511741.
[8] M. Urselmann, S. Barkmann, G. Sand and S. Engell, "A Memetic Algorithm for Global Optimization in Chemical Process Synthesis Problems," in *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 659-683, Oct. 2011, doi: 10.1109/TEVC.2011.2150753.
[9] V. Pachori, G. Ansari, and N. Chaudhary, "Improved performance of advance encryption standard using parallel computing," *International Journal of Engineering Research and Applications*, vol. 2, no. 1, pp. 967–971, 2012.
[10] P. Assiroj, A. L. Hananto, A. Fauzi and H. L. Hendric Spits Warnars, "High Performance Computing (HPC) Implementation: A Survey," *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, 2018, pp. 213-217, doi: 10.1109/INAPR.2018.8627040.
[11] M. Abd Rahman and A. Mamat, "A Study of Image Processing in Agriculture Application under High Performance Computing Environment," *International Journal of Computer Science and Telecommunications*, vol. 3, no. 8, pp. 16-24, 2012.
[12] P. Assiroj *et al.*, "The Form of High-Performance Computing: A Survey," *IOP Conference Series: Materials Science and Engineering*, vol. 662, no. 5, p. 052002, 2019.
[13] J. L. Hennessy and D. a Patterson, "Computer Architecture," *Fourth Edition: A Quantitative Approach*. 2006.
[14] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, and J. M. Benitez, "Fast fingerprint identification for large databases," *Pattern Recognition*, vol. 47, no. 2, pp. 588-602, 2014, doi: 10.1016/j.patcog.2013.08.002.
[15] R. Welekar and N. V Thakur, "An Enhanced Approach to Memetic Algorithm Used for Character Recognition," *Springer Singapore*, vol. 768, pp. 593-602, 2019, doi: 10.1007/978-981-13-0617-4_57.
[16] M. Ghosh, S. Malakar, S. Bhowmik, R. Sarkar, and M. Nasipuri, "Memetic Algorithm Based Feature Selection for Handwritten City Name Recognition," *Springer*, vol. 775, pp. 599-613, 2017, doi: 10.1007/978-981-10-6430-2_47.
[17] P. Moscato, A. Mendes, and R. Berretta, "Benchmarking a memetic algorithm for ordering microarray data," *BioSystems*, vol. 88, no. 1-2, pp. 56-75, 2007, doi: 10.1016/j.biosystems.2006.04.005.
[18] L. Feng, A. H. Tan, M. H. Lim, and S. W. Jiang, "Band selection for hyperspectral images using probabilistic memetic algorithm," *Soft Computing*, vol. 20, no. 12, pp. 4685-4693, 2016, doi: 10.1007/s00500-014-1508-1.
[19] W. Sheng, G. Howells, M. Fairhurst, and F. Deravi, "A memetic fingerprint matching algorithm," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 402–411, 2007.
[20] W. Sheng, G. Howells, M. Fairhurst and F. Deravi, "A Memetic Fingerprint Matching Algorithm," in *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 402-412, Sept. 2007, doi: 10.1109/TIFS.2007.902681.
[21] P. Assiroj, H. L. H. S. Warnars, E. Abdurrachman, A. I. Kistijantoro, and A. Doucet, "Measuring memetic algorithm performance on image fingerprints dataset," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 19, no. 1, pp. 96-104, 2021, doi: 10.12928/telkomnika.v19i1.16418.
[22] S. A. Mirsoleimani, A. Karami, and F. Khunjush, "A parallel memetic algorithm on GPU to solve the task scheduling problem in heterogeneous environments," *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 2013, pp. 1181–1188, doi: 10.1145/2463372.2463518.
[23] R. Cheng and M. Gen, "Parallel machine scheduling problems using memetic algorithms," *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929)*, 1996, pp. 2665-2670 vol.4, doi: 10.1109/ICSMC.1996.561355.
[24] J. Tang, M. H. Lim, and Y. S. Ong, "Adaptation for parallel memetic algorithm based on population entropy," *GECCO 2006 - Genetic and Evolutionary Computation Conference*, vol. 1, pp. 575-582, 2006, doi: 10.1145/1143997.1144100.
[25] M. Blocho and Z. J. Czech, "A Parallel Memetic Algorithm for the Vehicle Routing Problem with Time Windows," *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2013, pp. 144-151, doi: 10.1109/3PGCIC.2013.28.
[26] A. Mendes, C. Cotta, V. Garcia, P. Franca and P. Moscato, "Gene ordering in microarray data using parallel memetic algorithms," *2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, 2005, pp. 604-611, doi: 10.1109/ICPPW.2005.34.
[27] E. Armstrong, G. Grewal, S. Areibi and G. Darlington, "An investigation of parallel memetic algorithms for VLSI circuit partitioning on multi-core computers," *CCECE 2010*, 2010, pp. 1-6, doi: 10.1109/CCECE.2010.5575207.

BIOGRAPHIES OF AUTHORS

Priati Assiroj was born in Cirebon, Jawa Barat, Indonesia. She has Bachelor and Master's in Computer Science. She received the Bachelor from STMIK Bani Saleh Bekasi, in 2011and received her Master from STMIK LIKMI, Bandung, Indonesia, in 2016. From 2014 to 2016, she was a lecturer in Universitas Singaperbangsa Karawang, Indonesia, and from 2016 to 2019 she was a lecturer in Universitas Buana Perjuangan Karawang in Information System Dept. Since January 2019 she is a lecturer in Politeknik Imigrasi, Ministry of Law and Human Rights, Republic of Indonesia. She is a doctoral student in Computer Science since March 2018 at Bina Nusantara Graduate Program, Doctor of Computer Science, Bina Nusantara University Jakarta, Indonesia. Her research fields are data mining, high-performance computing, and evolutionary algorithm.

Harco Leslie Hendric Spits Warnars received a Ph.D. degree in Computer Science from Manchester Metropolitan University. Since September 2015 he is a Head of Information Systems concentration at department Doctor of Computer Science Bina Nusantara University, works some project research with my doctoral computer Science students in research area such as Game, Artificial Intelligence including Data Mining, Machine Learning and Decision Support System application such as DSS, BI, Dashboard, Data Warehouse, and so on

Edi Abdurrachman, received B.Sc and Master of Statistics in Applied Statistics from Bogor Agricultural University then received M.Sc and Ph.D. in survey statistics and statistics from IOWA State University, USA. He is currently a professor and dean of the Binus Graduate Program, Doctor of Computer Science, Bina Nusantara University Jakarta. His research interest includes statistics, survey statistics, and applied statistics and management information systems. Mr. Abdurrachman’s awards and honors include the MU SIGMA RHO Society (1985) and Best Lecturer Binus University (2012). He is also a member of the American Statistical Association, International Association of Engineers (IAENG), Gamma Sigma Beta, and as a Vice President of the Asian Federation for Information Technology in Agriculture. From 1980-2015 actives in the ministry of agriculture in many positions of the director. He is also active as a public speaker in national and international seminars.

Achmad I Kistijantoro, received the B.Eng. degree in informatics from the Institute of Technology Bandung, (ITB), Bandung, Indonesia, the masters’ degree from TU Delft, Delft, The Netherlands, and the Ph.D. degree from the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K., His current research interests includes distributed systems, parallel computation, and high-performance computation.

Antoine Doucet is a Full Professor in computer science at the L3i laboratory of the University of La Rochelle since 2014. He leads the research group in document analysis, digital contents, and images (about 40 people) and is additionally the director of the ICT department of the Vietnam-France University of Science and Technology of Hanoi. Additionally, he is the principal investigator of the H2020 project NewsEye, running until 2021 and focusing on augmenting access to historical newspapers, across domains and languages. He further leads the effort on semantic enrichment for low-resourced languages in the context of the H2020 project Embeddia. His main research interests lie in the fields of information retrieval, natural language processing, and (text) data mining. The central focus of his work is on the development of methods that scale to very large document collections and that do not require prior knowledge of the data, hence that are robust to noise (e.g stemming from OCR) and language-independent. Antoine Doucet holds a Ph.D. in computer science from the University in Helsinki (Finland) since 2005, and a French research supervision habilitation (HDR) since 2012.