Available online at http://scik.org
J. Math. Comput. Sci. 11 (2021), No. 4, 3813-3828
https://doi.org/10.28919/jmcs/5819
ISSN: 1927-5307

HIGH-PERFORMANCE COMPUTING MEMETIC ALGORITHM (HPCMA) MODEL TO PROCESS IMAGE FINGERPRINT DATASET

PRIATI ASSIROJ[1,2,*], HARCO LESLIE HENDRIC SPITS WARNARS[1], EDI ABDURACHMAN[1], ACHMAD IMAM KISTIJANTORO[3], ANTOINE DOUCET[4]

[1]Binus Graduate Program, Doctor of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia

[2]Manajemen Teknologi Keimigrasian, Politeknik Imigrasi, Depok 16514, Indonesia

[3]School of Electronics Engineering and Informatics, Institut Teknologi Bandung, Bandung 40132, Indonesia

[4]Laboratoire L3i - Université de La Rochelle, Avenue Michel Crépeau, F-17 042 La Rochelle Cedex 1, France

Abstract: The terminology of the memetic refers to “meme”. The meme is a part of natural information of the individual population and this information can be transmitted among the individual of the population. The memetic algorithm (MA) uses the evolutionary concept based on the Genetic Algorithm (GA) and is combined with a local search feature. Thus, as in GA, MA employs the basic steps such as selection, crossover, and mutation with additional components a local search to improve the solution of candidates. The major challenge of this algorithm is how to develop a good local search operator that can do a good exploration of the entire population. Several methods have been proposed and we have studied several parallel methods implemented on MA (PMA) to increase the efficiency of processing time. The parallel method is included in cluster computing, and that is a part of high-performance computing systems. In this work, we proposed the two new models of the memetic algorithm that run in a high-

[*]Corresponding author
E-mail address: priati@ binus.ac.id

Received April 5, 2021

performance computing system (HPCMA) environment by utilizing the multi-threads feature of processor, so-called HPCMA1 and HPCMA2 model, to process fingerprints image dataset. The result shows that the HPCMA1 model runs with high inconsistency data caused by its overlaps in the macro-process and to address the problem from the HPCMA1 model we use the HPCMA2 model.

Keywords: memetic algorithm; genetic algorithm; parallel ma; high-performance computing ma.

2010 AMS Subject Classification: 68W40.

1. INTRODUCTION

A memetic algorithm (MA) is an algorithm that is based on the natural concept, biology, and genetic such as in genetic algorithm (GA). The terminology of the memetic refers to “meme” [1]. The meme is a part of the natural information of the individual population. In other words, MA has both characteristics of culture and evolution. This algorithm is the so-called Local search genetic algorithm or hybrid genetic algorithm [2]. MA is one of the new improvement areas of evolutionary measurement [3]. MA is a meta-heuristic method population-based consists of the evolutionary core with a set of the local search algorithm that works side by side with the evolutionary core to find the best solution [4]. The original evolutionary algorithm is not efficient in the combinatorial environment but the hybridization technique can be developed to improve runtime efficiency [5]. MA can solve several optimization problems with a high complexity such as assembly in industrial applications [6], vehicle routes [7], real-time scheduling [8], wireless sensor networks [9].

The memetic algorithm (MA) uses the evolutionary concept based on the Genetic Algorithm (GA) and is combined with a local search feature. Thus, as in GA, MA employs the basic steps such as selection, crossover, and mutation with additional components a local search to improve the solution of candidates. A local search is performed on the environment of each newly created individual before being included in the population [10]. MA is also a synergy of the method that is based on the population within the separation of individual learning [11]. MA’s steps consist of an iterative process as described below [12]:

A. Population creation

The initial population is created randomly consists of chromosomes without any duplication.

B. Local search

The local search process is performed to find better chromosomes. Every offspring participate in the search process to maintain quality.

C. Fitness measurement

Measuring process conducted to decide the good of bad chromosomes.

D. Algorithm termination

The algorithm will stop if the number of iteration is equal to the number defined before.

E. Selection

The selection process is performed to obtain the fittest species then included as the candidates of the crossover process.

F. Crossover and mutation

Crossover is a crossbreeding process between individuals that performed randomly to obtain new offspring. The mutation is a swap process of the old chromosomes with the new crossover and mutated chromosomes.

G. Replacement

The bad performance individual will be replaced to obtain the best solutions.

MA has been implemented in various studies such as optimization, automatic programming, machine, and robotic programming, economic model optimization, immune system, ecology, genetic population, and social system interaction [13]. The memetic algorithm has also successfully optimized the control strategy parameter of the hybrid parallel electric vehicle [14]. The major challenge of MA is how to design a good local search operator to explore the large instances. However, in some cases, the ineffectiveness of exploration time still occurs, and due to this condition, the parallel computing model developed [15]. Below is the standard MA pseudocode:

```
Begin
Initialize: Create an initial population randomly;
Evaluate (population);
bestIndividual=best(population);
    While stopCriteria() not met do
        newPopulation=Ø;
        For j=1 to k do
                select c1 and c2 from population;
                {s1, s2}=crossover(c1, c2, Cc);
                mutate(s1, s2);
                localSearch(s1);
                        localSearch(s2);
                        add s1 and s2 to population;
        End For
            population=newPopulation;
            Evaluate (population);
            bestIndividual=best(population);
    End While
End
```

The Parallel Memetic Algorithm (PMA) is a modern parallel meta-heuristic class that combines the evolutionary algorithm, local search method, and parallel and distributed computing system. The goal is to obtain global optimization. The PMA generates a high-quality solution for large-scale combinatorial optimization faster than the original genetic and memetic algorithm [16]. The parallel technique is used to solve the bottleneck problem in the execution time of many meta-heuristics [17]. The performance of MA is confirmed for the quality of the solution, runtime, speedup, and efficiency [10]. The memetic algorithm is an algorithm that does not depend on the framework thus for each case will have a different treat in this algorithm [18].

High-Performance Computing (HPC) is a computer system that can process lots of data very fast, effectively, and efficiently. There are two methods to get high performance on the computer, firstly is make fast processors and secondly is make a parallel system with multi-processors.

Recently, semiconductor technology is almost reaching its lithography limit, the processor makers have to reduce some electronic components that make their processors are smaller with the shorter signals and this first method is not recommended because the product quality will decrease. Therefore, the most possible method is to make a parallel computation system [19]. The parallel computing system belongs to the cluster computer, and it is one of the High-Performance Computation (HPC) techniques, the others are super-computer, grid, and cloud computing [20].

2. RELATED WORKS

There are several types of research conducted to develop parallel MA (PMA). Mirsoleimani et.al [17] have proposed and implemented PMA on the Graphics Processing Unit (GPU). The proposed algorithm has been used to solve the task scheduling problems for heterogeneous multi-processing systems. This proposed PMA can increase the speeds up to 696x compared to the sequential method. Tang et.al [21] have proposed the island model parallel MA with dynamic local search (PMA-DLS), to control the local search frequency in PMA then the results compared to parallel with static local search (PMA-SLS). There are three advantages of PMA-DLS compared to PMA-SLS. First, the number of individuals for local searches is adaptive to the online fluctuation of diversity. Secondly, local search frequency is adjusted by PMA-DLS. Thirdly, PMA-DLS characteristics decide the frequency of applying local search. Armstrong et.al [10] have investigated circuit partitioning using PMA. The investigation has been done on the six parallel memetic algorithms and each uses a global shared-memory for information exchange then exploiting the multiple cores to reduce runtime. Blocho and Czech [22] have proposed PMA to solve the NP-hard vehicle routing problem with time windows (VRPTW). Their purpose is to design a PMA that can determine the highest quality solution of VRPTW. The process starts with the heuristic or hybrid algorithm of a GA with local refinement procedures. Process co-operate periodically to improve the results. Chyu and Chang [23] have proposed a competitive evolution strategy memetic algorithm (CESMA) to solve unrelated parallel machine scheduling. In the beginning, this method applied an efficient archive memetic algorithm (EAMA) and an ineffective archive memetic algorithm (IAMA) to produce the next high-quality offspring and with this good new offspring, CESMA has more opportunities to find quality solutions. Ding et.al [18] have proposed a hybrid memetic algorithm (HMA) to solve the machine scheduling problem. They make this method more efficient with a divide-and-conquer- based local search feature as recursive operator to solve the sub-problems, a compound neighborhood that combines insert and swap with the aim and generate search tracks to improve the solution quality, the recombination operator to generate a solution and maintain the population diversity by adopting the quality and distance-based population updating strategy. The experiment results show HMA can achieve a 100% success rate. Mendes et.al [2] have used PMA to solve the gene ordering problem in microarray by focuses on the efficiency of parallelization and the result shows that good speedup has been obtained with sustainable quality improvement of the solution. Nalepa and Blocho [24] have solved the Pickup and Delivery Problem with time windows (PDPTW) with PMA. Cheng and Gen [25] have implemented MA to solve machine scheduling problems. There are two important problems to be solved, the first is the partition of work among the machines and the second is the order of works on each machine. The permutation code schema is proposed as a representation and the crossover and mutation operators are defined to adjust the partition and permutation of works. Results show that MA is a promising method to solve machine scheduling problems.

3. RESEARCH METHOD

In this work, we propose two models based on High-performance computation using the memetic algorithm (HPCMA) to process image fingerprints data, the so-called HPCMA1 model, and HPCMA2 model.

A. HPCMA1 MODEL

Figure 1 is the HPCMA1 model process. The Blue signs are the macro processes of the algorithm and the red signs are the micro-processes of the algorithm.

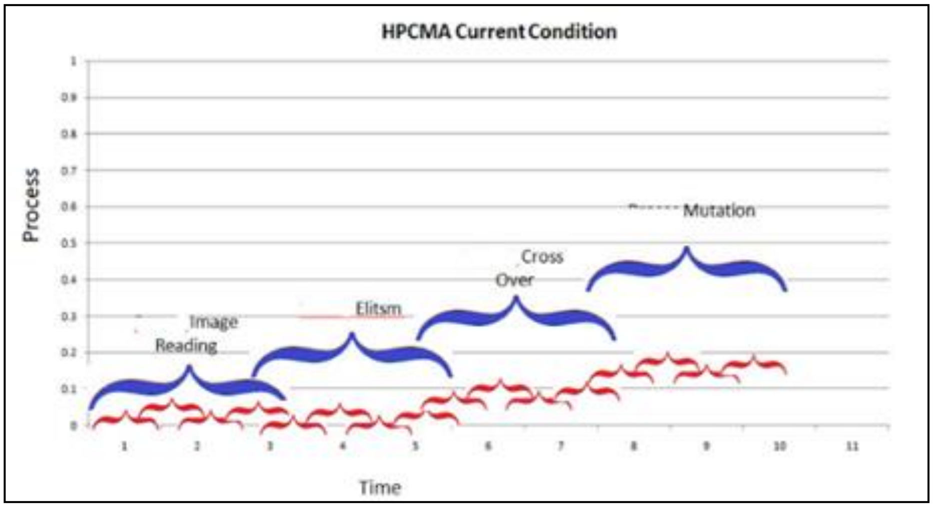

Figure 1. HPCMA1 Model Process

Based on Figure 1, the HPCMA1 model processes are divided into 4 macro processes as described below:

1. *The first macro process is data reading and conversion.*

Image fingerprint data are converted to an array and binary code. Firstly the algorithm reads the folder and sub-folder that contain fingerprint image data then saved it to the data class. The conversion process is run in HPC mode. The process then saves the binary data to the database server, and if the conversion is finished process will be continued to the second macro process.

2. *The second macro process is the selection or elitism process, using 2% of the population.*

In this macro process, we decide elitism data with a 2% data sample of total binary data. Note that we can change the amount of sample as we need. This is a random process in HPC mode to generate the parent candidates, then save the selected candidates to the database server. The algorithm will ensure that this selection process is finished before continuing to the third macro process.

3. *The third macro process is crossover.*

This is a crossbreeding process to obtain the new offspring. The algorithm selects parent candidates from the previous process then mutates each other. This process runs in HPC mode then saves all new offspring to the database server and if this process is finished the process will be continued to the last macro process.

4. *The fourth macro process is mutation.*

In this macro process, the algorithm mutates several bytes of binary data from the crossover to obtain new offspring. Mutate all the candidates by exchanging the byte from binary data to obtain the new offspring in HPC mode then save all the newest offspring to the database server, if the mutation process is finished algorithm will end, and if not mutation process will be repeated.

Figure 2 below is the flowchart of the HPCMA1 model.

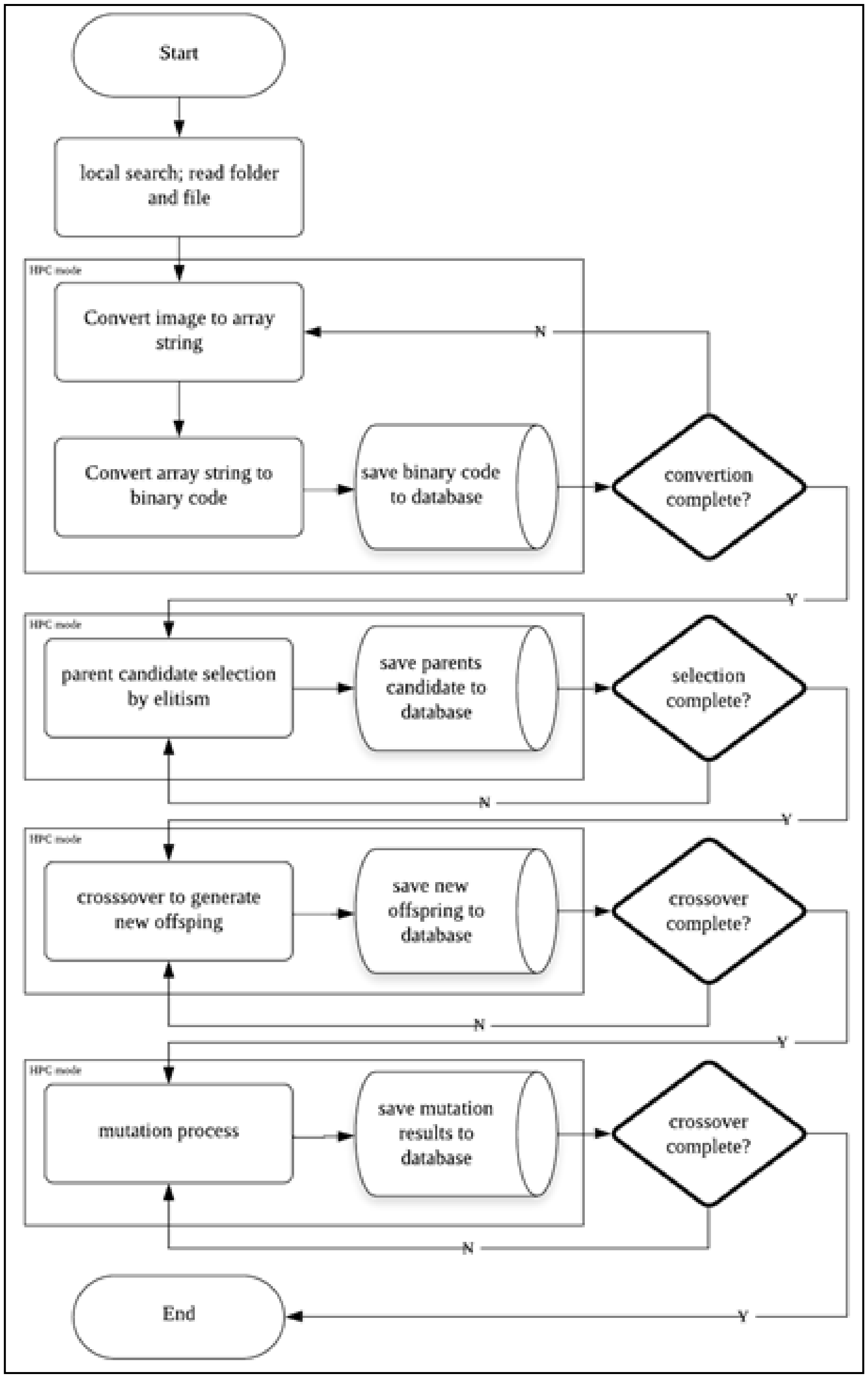
Start
local search; read folder and file
HPC mode
Convert image to array string
N
Convert array string to binary code
save binary code to database
convertion complete?
Y
HPC mode
parent candidate selection by elitism
save parents candidate to database
selection complete?
N
Y
HPC mode
crosssover to generate new offsping
save new offspring to database
crossover complete?
N
Y
HPC mode
mutation process
save mutation results to database
crossover complete?
N
End
Y

Figure 2. HCPMA1 model Flowchart

B. HPCMA2 MODEL

Figure 3 below is the HPCMA2 model process. The blue signs are macro processes of the algorithm and the red signs are micro-processes of the algorithm.

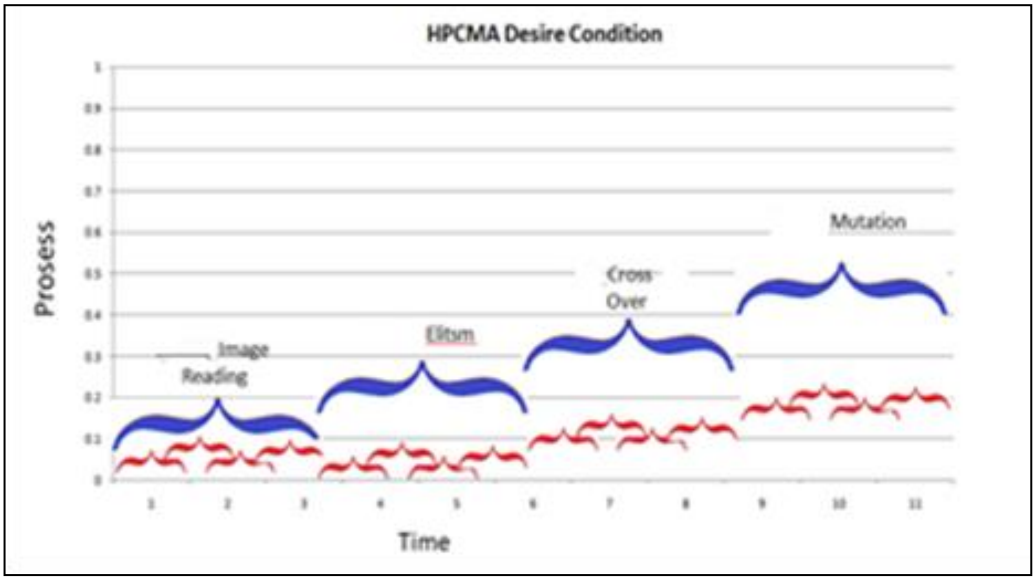

Figure 3. HPCMA2 model Process

Based on figure 3, HPCMA2 model processes are divided into several macro and micro processes, and those are described below:

The first macro process is to convert image data fingerprint to binary code. It consists of several steps as follow:

1. Read the folder and sub-folder that contain fingerprint image data then saved it to the data class.
2. Measure the number of created data classes, it refers to the amount of data.
3. The conversion process from image to array string in HPC mode.
4. The conversion process from array string to binary code in HPC mode.
5. Save the binary data to the database server.
6. Make sure that all conversion processes are well done. If there are any unprocessed data, the step will be looped to number 3.

7. Retrieve all converted fingerprint data.
8. Measuring the amount of processed data then compared to the amount of all image fingerprint data. If it equals all data we can go to the next step, the second macro process, and if not then we will wait for all processes to finish.

The second macro process is the selection or elitism process. We use 2% of the population to obtain the fittest parents candidate.

9. Decide elitism data with a 2% data sample of total binary data. We can change the number of samples we need.
10. We will know the amount of elitism data after we finish process number 9.
11. Then with those data, we will select parents candidates randomly in HPC mode.
12. Save the selected data from the elitism process to the database server.
13. If it all well done we can go to the next step and if not the process will be repeated to process number 11.
14. Retrieve the selected data.
15. Then if the elitism process is finished, measure the number of processed data then compared to the number of elitism defined before, if it does not equal, wait until all the process finishes.

The third macro process is crossover. Crossbreeding process to obtain the new offspring.

16. Select the parent candidates from the previous process. We divide to parent x and parent y with the same number of data.
17. Multiplying parent x and parent y then we know the number of possible crossbreeds that can be performed.
18. Crossover. Move 100 bytes binary data from parent x to parent y with a specified transfer location that will overwrite the original 100 bytes in parent y and those are the new offspring in HPC mode.
19. Save the new offspring to the database.
20. If the crossover process is finished then we can go to the next process and if not the process

will be repeated to process number 18.

21. Retrieve all the processed data.
22. Measure the amount of the processed data then compared to the amount of the multiplication result of parent “X” and parent “Y”. If it equals, go to the next process, and if it does not wait until the process is finished.

The fourth macro process is mutation. This is a mutate process of several bytes of binary data from the crossover to obtain new offspring.

23. Retrieve all offspring from the crossover process to be mutated.
24. Measure the number of candidates that will be mutated.
25. Mutate all the candidates by exchanging the byte from binary data to obtain the new offspring in HPC mode.
26. Save all new offspring data to the database server.
27. If the mutation process is finished then we can go to the next process and if it does not, the process will be repeated to process number 25.
28. Retrieve all the mutated data.
29. Ensure that the mutation process is finished then measure the number of processed data and compared it to the number of candidates. If it equals, then the process is accomplished and if it does not, wait until the process is finished.
30. Finish the threading process in the background.

Figure 4 below is the flowchart of the HPMCA2 model.

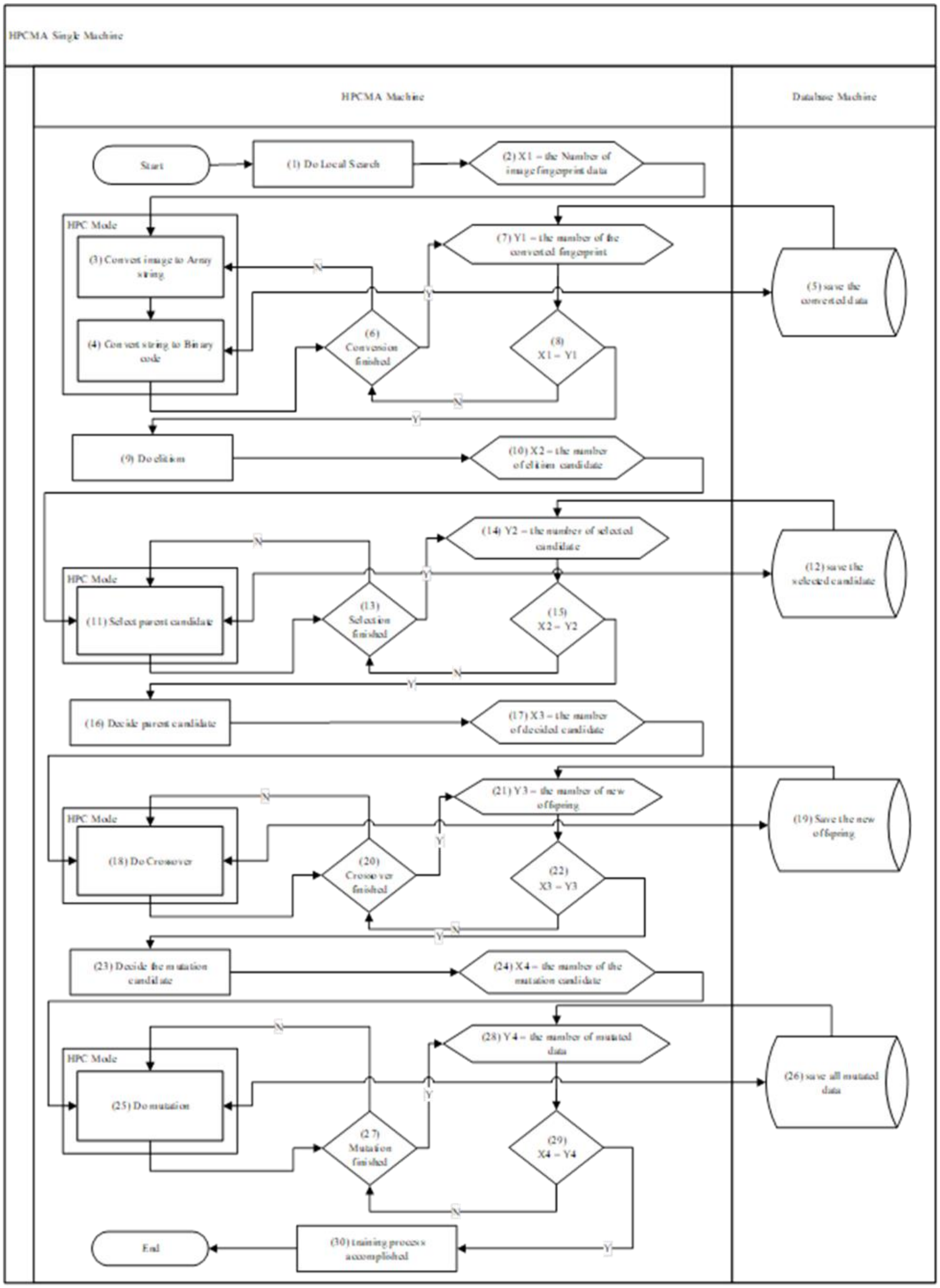
HPCMA Single Machine
HPCMA Machine
Database Machine
Start
(1) Do Local Search
(2) X1 – the Number of image fingerprint data
HPC Mode
(3) Convert image to Array string
(4) Convert string to Binary code
(5) save the converted data
(6) Conversion finished
(7) Y1 – the number of the converted fingerprint
(8) X1 = Y1
N
Y
(9) Do elitism
(10) X2 – the number of elitism candidate
(11) Select parent candidate
(12) save the selected candidate
(13) Selection finished
(14) Y2 – the number of selected candidate
(15) X2 = Y2
(16) Decide parent candidate
(17) X3 – the number of decided candidate
(18) Do Crossover
(19) Save the new offspring
(20) Crossover finished
(21) Y3 – the number of new offspring
(22) X3 = Y3
(23) Decide the mutation candidate
(24) X4 – the number of the mutation candidate
(25) Do mutation
(26) save all mutated data
(27) Mutation finished
(28) Y4 – the number of mutated data
(29) X4 = Y4
(30) training process accomplished
End

Figure 4. HPCMA2 model flowchart

4. RESULT AND DISCUSSION

Both models, HPCMA1 and HPCMA2 are used to process synthetic fingerprint image data of FVC2006. This algorithm runs on a computer system with (R) Core(TM) i7-6600U CPU @ 2.6GHz and 16BG of memory. The algorithm utilizes the multi-thread feature of the processor. The first experiment is using the HPCMA1 model and the result is shown in Figure 1.

According to Figure 1, the result shows that micro-processes can be parallelized using a multi-thread feature of the processor, as well as macro-process. Ideally, macro-processes run in the sequence where each process runs after the previous process is finished, but it does not run in the HPCMA1 model so there are overlaps in each process, as shown in Figure 1, the second macro-process, elitism, runs before the first macro-process is complete. It causes data integrity to be unmaintained well and cause inconsistencies in the number of data.

To address the limitation of the HPCMA1 model, we apply the HPCMA2 model for the second experiment. The illustration of the result can be seen in Figure 3. All the macro-process in the HPCMA2 model run in sequence, the second macro-process, elitism, runs after the previous process is complete, as well as the third and the fourth macro-process.

To maintain the data integrity, as seen in Figure 3, at the end of the completion process, we check the number of the processed data (X) and compare to the number of the entire data (Y), if the number is equal, the process will be continued. HPCMA2 model outperforms the HPCMA1 model, more efficient in runtime and data because there are no overlaps in every macro-process.

5. CONCLUSION

In this paper, we proposed the two new models of the memetic algorithm that run in a high-performance computing system (HPCMA) environment by utilizing the multi-threads feature of processor, so-called HPCMA1 and HPCMA2, to process fingerprints image dataset. The result shows that HPCMA1 runs with high inconsistency data caused by its overlaps in the macro-process and to address the problem from HPCMA1 we use the HPCMA2 model. Then we check the number of the processed data (X) and compared it to the number of the entire dataset (Y) and ensure that the amount is equal before we continue to the next processes.

ACKNOWLEDGEMENT

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.

CONFLICT OF INTERESTS

The author(s) declare that there is no conflict of interests.

REFERENCES

[1] R. Dawkins, The selfish gene, Oxford University Press, Oxford, 1976.

[2] A. Mendes, C. Cotta, V. Garcia, P. Franca, P. Moscato, Gene ordering in microarray data using parallel memetic algorithms, in: 2005 International Conference on Parallel Processing Workshops (ICPPW'05), IEEE, Oslo, Norway, 2005: pp. 604–611.

[3] X. Chen, Y.-S. Ong, M.-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (2011), 591–607.

[4] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report C3P 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, USA, (1989).

[5] N. Krasnogor, J. Smith, A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues, IEEE Trans. Evol. Comput. 9 (2005) 474–488.

[6] S.-Y. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem, IEEE Trans. Syst. Man Cybern, Syst. 46 (2016) 139–149.

[7] J. Wang, W. Ren, Z. Zhang, H. Huang, Y. Zhou, A hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows, IEEE Trans. Syst. Man Cybern, Syst. 50 (2020), 4732–

4745.

[8] X. Gong, Y. Liu, N. Lohse, T. De Pessemier, L. Martens, W. Joseph, Energy- and labor-aware production scheduling for industrial demand response using adaptive multiobjective memetic algorithm, IEEE Trans. Ind. Inf. 15 (2019) 942–953.

[9] C.-C. Liao, C.-K. Ting, A Novel Integer-Coded Memetic Algorithm for the Set k -Cover Problem in Wireless Sensor Networks, IEEE Trans. Cybern. 48 (2018), 2245–2258.

[10] E. Armstrong, G. Grewal, S. Areibi, G. Darlington, An investigation of parallel memetic algorithms for VLSI circuit partitioning on multi-core computers, in: CCECE 2010, IEEE, Calgary, AB, Canada, 2010: pp. 1–6.

[11] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Trans. Syst., Man, Cybern. B. 36 (2006), 141–152.

[12] Y.-H. Cheng, C.-M. Lai, J. Teh, Memetic algorithm for fuel economy and low emissions parallel hybrid electric vehicles, in: 2017 IEEE 8th International Conference on Awareness Science and Technology (ICAST), IEEE, Taichung, 2017: pp. 219–222.

[13] J. Tang, M. Lim, Y.S. Ong, M.J. Er, Parallel memetic algorithm with selective local search for large scale quadratic assignment problems, Int. J. Innov. Comput. Inform. Control, 2 (2006), 1399–1416.

[14] Y.-H. Cheng, C.-M. Lai, Control strategy optimization for parallel hybrid electric vehicles using a memetic algorithm, Energies. 10 (2017), 305.

[15] V.J. Garcia, P.M. Franca, Ad.S. Mendes, P. Moscato, A parallel memetic algorithm applied to the total tardiness machine scheduling problem, in: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, IEEE, Rhodes Island, Greece, 2006: p. 8.

[16] J. Tang, M.H. Lim, Y.S. Ong, Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems, Soft Comput. 11 (2007), 873–888.

[17] S.A. Mirsoleimani, A. Karami, F. Khunjush, A parallel memetic algorithm on GPU to solve the task scheduling problem in heterogeneous environments, in: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference - GECCO ’13, ACM Press, Amsterdam, The Netherlands, 2013: p. 1181.

[18] J. Ding, L. Shen, Z. Lu, L. Xu, U. Benlic, A Hybrid Memetic Algorithm for the Parallel Machine Scheduling Problem With Job Deteriorating Effects, IEEE Trans. Emerg. Top. Comput. Intell. 4 (2020), 385–397.

[19] P. Assiroj, A.L. Hananto, A. Fauzi, H.L. Hendric Spits Warnars, High Performance Computing (HPC) Implementation: A Survey, in: 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), IEEE, Jakarta, Indonesia, 2018: pp. 213–217.

[20] P. Assiroj, H.L.H.S. Warnars, R. Kosala, B. Ranti, S. Supangat, A.I. Kistijantoro, E. Abdurrachman, The Form of High-Performance Computing: A Survey, IOP Conf. Ser.: Mater. Sci. Eng. 662 (2019), 052002.

[21] J. Tang, M.H. Lim, Y.S. Ong, Adaptation for parallel memetic algorithm based on population entropy, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation - GECCO '06, ACM Press, Seattle, Washington, USA, 2006: p. 575.

[22] M. Blocho, Z.J. Czech, A Parallel Memetic Algorithm for the Vehicle Routing Problem with Time Windows, in: 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE, COMPIEGNE, France, 2013: pp. 144–151.

[23] C.-C. Chyu, W.-S. Chang, A competitive evolution strategy memetic algorithm for unrelated parallel machine scheduling to minimize total weighted tardiness and flow time, in: The 40th International Conference on Computers & Indutrial Engineering, IEEE, Awaji City, Japan, 2010: pp. 1–6.

[24] J. Nalepa, M. Blocho, A Parallel Memetic Algorithm for the Pickup and Delivery Problem with Time Windows, in: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), IEEE, St. Petersburg, Russia, 2017: pp. 1–8.

[25] R. Cheng, M. Gen, Parallel machine scheduling problems using memetic algorithms, in: 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929), IEEE, Beijing, China, 1996: pp. 2665–2670.