Indonesian Journal of Electrical Engineering and Computer Science
Vol. 34, No. 1, April 2024, pp. 584~591
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v34.i1.pp584-591

A new hybrid parallel genetic algorithm for multi-destination path planning problem

Luthfiansyah Ilhamnanda Yusuf[1], Aina Musdholifah[2]
[1]Master Program in Artificial Intelligence, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia
[2]Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia

Article Info

Article history:

Received Nov 24, 2023
Revised Jan 21, 2024
Accepted Feb 4, 2024

Keywords:

Genetic algorithm
Multi-destination
Multi-objective
Parallel genetic algorithm
Path planning

ABSTRACT

This paper proposes a new parallel approach of multi objective genetic algorithm for path planning problem. The main contribution of this work is to reduce the population size that effect in decreasing processing times of finding the optimum path for multi destination problem. This is achieved by combining the local population of island parallel approach and global population of global parallel approach. Various experiments have been conducted to evaluate the new hybrid parallel genetic algorithm (HPGA) in solving multi-objective path planning problems. Three different test areas with 2 destinations were used to assess the performance of HPGA. Furthermore, this work compares HPGA and sequential genetic algorithm (SeqGA), as well as compared to other existing parallel genetic algorithm (GA) methods. From experimental results show that proposed HPGA outperform others, in term of processing time i.e., up to 3.6 times speedup faster, and lowest GA parameter values. This proposed HPGA can be utilized to design robots with fast and consistent path planning, especially with various obstecles.

Corresponding Author:

Aina Musdholifah
Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences
Universitas Gadjah Mada
Yogyakarta, Indonesia
Email: aina_m@ugm.ac.id

1. INTRODUCTION

Finding an optimal path in an environment or path planning (PP) is the main problem in developing autonomous robots. PP algorithm allows autonomous robots to move independently without human intervention, thereby increasing efficiency in processes that can utilize autonomous robots. The simplest form of this problem is when path searching is carried out in a static environment [1]. In the real world, autonomous robots have been used in various problems, including exploring other planets and looking for victims in the search and rescue process [2]. Genetic algorithm (GA) is a metaheuristic algorithm inspired by living things and based on Darwin's survival of the fittest theory, where the best individuals will survive after undergoing long adaptation [3]. GA can be used to solve optimization problems, one of which is PP problems [4], and many studies have shown GA can be good for solving PP problems.

In GA, population generation is the essential first step and dramatically influences the results. Populations in GA are solution candidates that will be optimized. The solutions are stored in the chromosomes of each individual. These chromosomes can be encoded to simplify the optimization progress. Patle *et al.* [5] proposes matrix-binary encoding in GA to solve PP problem and gets a comparable result to other intelligence navigational controllers. Population generation using heuristics has been tested and performs better than fully random generation [6]. Next in the step are genetic operations. The primary genetic operations are parent selection, crossover, and mutation. Parent selection and crossover are used to obtain new solutions from an existing solution in the population. Intuitively, good alleles from parents can be combined into better offspring if the suitable alleles are selected in crossover. If not, the offspring produced will not necessarily be better than the parent's solution. Therefore, to maintain a good solution in the mating pool, all population members do not carry out a crossover [7]. Mutation operators introduce uncertainty into the obtained solution to explore new solutions. A good mutation operator needs to have 3 criteria: reachability, unbiasedness, and scalability [4]. Xin *et al.* [8], proposed multi domain inversion to make more offspring from chosen parents and pick the most optimum to be retained in the population. This method speeds up convergence, gives better results, and is more robust to conventional GA.

Many studies have proposed many modified genetic operations. Lamini *et al.* [9], the same point crossover is modified so that the offspring will produce the optimal solution from the selected parent. This method improves the performance of the GA by reducing the iterations required for similar results. A new crossover operator has also been proposed with the addition of local search. Apart from considering the solution from the parent, the offspring also looks for solutions from neighboring parents. This modification accelerates the convergence of solutions produced [10]. The closest node pairing crossover (CNCP) proposed by [11] considers the distance between two genes in the crossover operation. CNCP gives a better solution and a faster convergence rate. A mutation operator that mutates all genes inside the chromosome instead of just one gene has been tested and gives a better result than other mutation methods [12]. A new mutation operator has also been proposed where the most optimal parent is considered more during the calculation as it is closer to the global optimum. The result is increased performance in convergence speed and stability of the algorithm compared to ordinary GA [13].

Along with the development of GA, many modifications have been made to the essential workings of this algorithm. One of them is the addition of the deletion operator. This deletion operator allows the GA to have flexible chromosome lengths, thus allowing for more optimal solutions [14]. Several studies have used the deletion operator in path planning problems and obtained good results [10], [15], [16]. Sarkar *et al.* [16] uses circuit removal technique to remove any loop form when optimizing the robot path. In a smooth path planning problem, Berzier curve can be used to create a smooth path for the robot. In [17] and [18] use GA combined with Bezier curve method to solve smooth path planning problems with good results. Chen and Gao [19] proposes GA with adaptive crossover mutation rate to solve path planning for soccer robots and obtained good results.

Fitness functions are used to evaluate the generated solution. The fitness function will guide the population toward the optimal solution, so a suitable fitness function is essential [4]. To increase the efficiency of the GA, an appropriate fitness function is needed so that convergence can be achieved quickly while providing the most optimal solution. Path length can be used to find the shortest path in the PP problem and tested to give good results [6], [9], [10], [15], [20], [21]. Path smoothness can be used to minimize the number of turns in the optimum solution and tested to give optimal results [10], [21] and reduce convergence time [15]. Using energy usage as a fitness function where the robot minimizes any energy consumption when maneuvering can also give optimal results [15], [22], [23]. Another fitness function that considers the safety of the path the robot takes has also been tested and produced an excellent solution to the path planning problem [6], [21].

Even though it has been proven to solve PP problems, GA has the disadvantage of high computation and increasing problem complexity. The more iterations of the GA, the more fitness function calls that need to be made, thereby increasing resource requirements [3]. In the problem of finding paths with many destinations (multi-destination path planning), the increase in limited computing time causes the use of algorithms in off-the-line problems [24]. To overcome this limitation, many instances of the GA can be created, which can be run simultaneously to obtain one result, or what is usually called a parallel genetic algorithm (PGA). PGA is an extension of regular GA that can be utilized with the help of multithreading capabilities commonly found in modern processors. PGA can cut computing time without sacrificing results because the algorithms are population-based, so there is no reduction in the search algorithm's capabilities. With good implementation, the PGA will be able to beat ordinary GA in terms of computing time [25]. PGA can also take advantage of the development of modern processors, which have many cores, especially on GPUs [26].

Many studies highlight the use of parallel processing to increase the search speed of GA. PGA has been tested to solve the vehicle routing problem based on cloud computing with the synchronizable kernels method. The method has proven very efficient and can obtain a speedup of more than 13 times compared to the conventional GA [27]. Amassmir *et al.* [28], an intelligence system to assist in making recruitment decisions has been developed with sequential GA and PGA, and it can give a quality decision in a reduced CPU time. PGA has also been used in hyperparameter optimization problems, and it can improve model accuracy [29]. In scheduling problems, PGA has been tested, resulting in faster convergence, preventing premature convergence [30], and improved execution time [31]. In PP problems, PGA has been tested, and the resulting performance makes using GA in PP of unmanned aerial vehicles feasible in real-time [20]. In theory, the reduction in computing time produced by the PGA can be applied to solving PP problems with multiple destinations and objectives (multi-destination path planning with multi-objective).

2. METHOD

A static and fully-observable environment is used in this experiment. The size of the environment and the number and shape of obstacles are determined before the experiment is carried out. The starting point and destination points and their coordinates are determined at the beginning, but the order of the destination points will depend on the fitness obtained by the agent.

The chromosome representation is a whole number representation, where each allele contains the coordinates of the point traversed by the agent. Whole numbers are used to simplify computing. In one individual, there are multiple chromosomes, where the number corresponds to the destination points needed to be reached. The number of alleles inside a chromosome is determined by the parameter chromosome length (*m*), and then the chromosome length is added by 2 (*m+2*) to accommodate the start and destination points. This chromosome configuration is illustrated in Figure 1. The destination point is shuffled when the individual is initialized so that the algorithm can accommodate different arrangements of destination points.

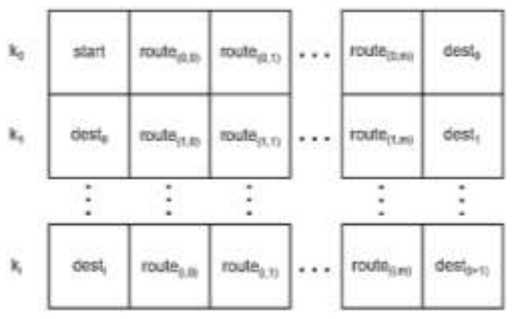
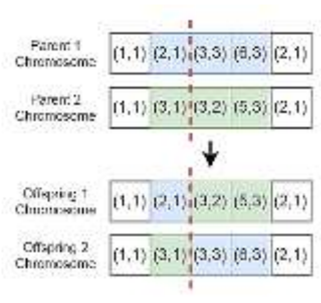
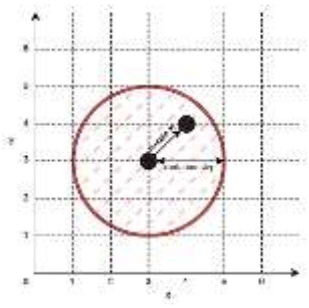


Figure 1. Chromosome configuration inside an individual

To select parents for crossover, tournament selection is used, and the mating pool is chosen to be less than the total population. Then, elitism is used to keep the best individual in every generation to fill the rest of the population. Figure 2 illustrates the single-point crossover and creep mutation process. For single-point crossover, random allele as the crossover point resulting in two chromosomes combined from the two parents, as illustrated in Figure 2(a). A modified creep mutation operator moves the point inside an allele to a random point within a defined range. For example as illustrated in Figure 2(b), when the third allele of Offspring 2's second chromosome in Figure 2(a) undergoes a mutation, it will be mutated from (3,3) to (4,4) and still within mutation step. In both operations, the first and the last allele on each chromosome are static and will not undergo crossover and/or mutation because those two points are the start and destination points.





(a) (b)

Figure 2. Examples of genetic operators used are (a) single-point crossover and (b) modified creep mutation

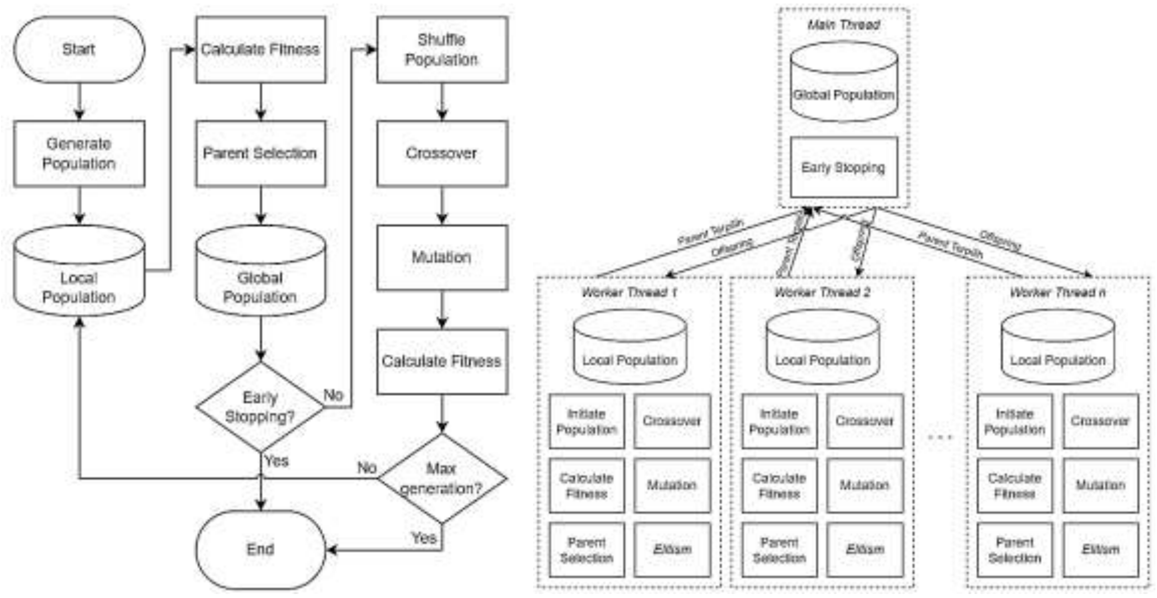
Two fitness functions are used to calculate the quality of every individual. Line segments can be created from every two adjacent alleles in a chromosome, and fitness can be calculated from all the line segments created. The path length is the length of every line segment added, as in (2), and path collision is the number of collisions of the line segment to all obstacles in the environment, as in (3). This path collision is then used as a penalty for the line segment. If there's a collision, the fitness obtained from the path length function is multiplied by the number of collisions obtained from the path collision function as in (1).

$$F = \sum_{i=0}^{j} \left(\sum_{m=0}^{n-1} L\left(P_{i,m}, P_{i,m+1}\right) \times \left(C\left(P_{i,m}, P_{i,m+1}\right) + 1 \right) \right) \tag{1}$$

$$L\left(P_{i,m}, P_{i,m+1}\right) = \sqrt{\left(x_{P_{i,m}} - x_{P_{i,m+1}}\right)^2 + \left(y_{P_{i,m}} - y_{P_{i,m+1}}\right)^2} \tag{2}$$

$$C\left(P_{i,m}, P_{i,m+1}\right) = \sum_{obs=obstacle[0]}^{obstacle[max]} \begin{cases} 10, & L\left(P_{i,m}, P_{i,m+1}\right) \cap obs \in obs \\ 0, & L\left(P_{i,m}, P_{i,m+1}\right) \cap obs = \emptyset \end{cases} \tag{3}$$

The parallelization model used is a hybrid between the island and global parallelization model with some modifications as illustrated in Figure 3. The process begins by creating several local populations inside the slave nodes. After individuals are initialized in the local populations, each individual's fitness will be calculated. Then, a selection is carried out for each local population and combined with a global population. After all the parents are combined, an equal number of parents will be distributed back to the local population to produce offspring by crossover and mutation inside the local population, as illustrated in Figure 3(a). Figure 3(b) shows the thread execution of each of the processes. The experiment is using early stopping technique and will trigger when there is no change in fitness of the best individual after some generation (referred as patience value). This early stopping technique can fasten the execution time and measure the convergence rate.



(a) (b)

Figure 3. Inner working of the proposed hybrid parallel genetic algorithm (HPGA) with (a) the general flow of the algorithm and (b) processes on each thread

The experiment of this research was execute using procedures shown in Figure 4. The proposed method HPGA is compared to sequential genetic algorithm (SeqGA), island parallel genetic algorithm (IPGA), and global parallel genetic algorithm (GPGA). Three different environments with an area of 15×12 and 2 destination points are used, shown in Figure 5. Parameter testing is performed on all the methods tested to find the best parameter, then best parameter is used to create a new model and performance comparison is made between all the method tested.

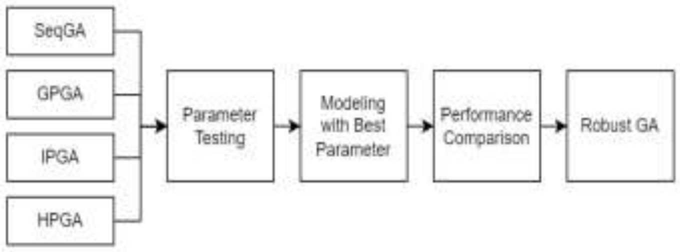


Figure 4. Test design to evaluate performance of the proposed method

3. RESULTS AND DISCUSSION

To evaluate the performance of all the methods, grid search was used. All parameter tested is shown in Table 1 and the best parameters shown in bold. The best parameter is chosen by evaluating the minimum fitness achieved, consistency of delivering an optimal and collision-free path, and the execution time. The best parameters are then used to compare all the methods tested and the results are shown in Table 2. Other methods use higher maximum population, mating pool, tournament length, and patience value to achieve a comparable result to HPGA.

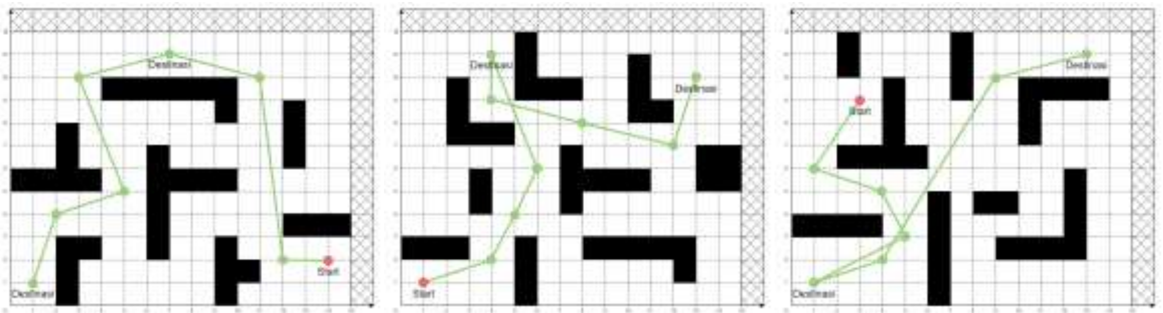
Table 1. Parameters tested for the best performance on each method

No	Parameter	Tested parameters			
		HPGA	SeqGA	GPGA	IPGA
1	Chromosome length	3, 4	3, 4	3, 4	3, 4
2	Maximum population	2000, 3000, 4000, 5000	2000, 3000, 4000, 5000, 6000	2000, 3000, 4000, 5000, 6000	2000, 3000, 4000, 5000, 6000
3	Mating pool	400, 800	400, 800, 1200, 1600	400, 800, 1200, 1600	400, 800, 1200, 1600
4	Tournament length	2, 3	2, 3, 4, 5	2, 3, 4, 5	2, 3, 4, 5
5	Mutation probability	0.1, 0.2, 0.3	0.1, 0.2, 0.3	0.1, 0.2, 0.3	0.1, 0.2, 0.3
6	Mutation step	2	2	2	2
7	Migration count	-	-	-	2, 3, 4, 5, 6
8	Patience	10	10, 25	10, 25	10, 25

Table 2. Performance of all the methods in all test area with different scenarios

Area	Method	Best fitness from 50 trials			Average	Average execution
		Minimum	Maximum	Average	Generation	Time (sec)
Area 1	SeqGA	30.02	58.72	31.57	45	5.00
	GPGA	30.02	64.29	32.59	45	2.25
	IPGA	30.02	58.19	31.07	50	2.48
	HPGA	**30.02**	**31.96**	**30.23**	74	**1.55**
Area 2	SeqGA	26.43	32.01	26.67	43	5.29
	GPGA	26.43	28.90	26.72	43	2.24
	IPGA	26.43	35.41	26.97	49	2.45
	HPGA	**26.43**	**27.69**	**26.50**	70	**1.49**
Area 3	SeqGA	30.20	56.13	30.99	45	4.72
	GPGA	30.20	32.43	30.37	46	2.20
	IPGA	30.20	32.01	30.42	51	2.45
	HPGA	**30.20**	**31.69**	**30.33**	76	**1.53**

Figure 5 illustrates output paths with the lowest minimum fitness value generated on the three different areas. The generated path on first area in Figure 5(a) shows the generated path in area 1 start from (14,2) and goes through (12,2), (11,10), and arrive at the first destination (7,11) then continues to (13,10), (5,5), (2,4) and arrive at the second destination (1,1). Figure 5(b) shows the generated path in area 2 start from (1,1) and goes through (4,2), (5,5), (6,6) and arrive at the first destination (4,11) then continues to (4,9), (8,8), (12,7) and arrive at the second destination (13,10). Figure 5(c) shows the generated path in Area 3 start from (3,9) and goes through (1,6), (4,5), (5,3) and arrive at the first destination (1,1) then continues to (4,2), (4,2), (9,10) and arrive at the second destination (13,11). All the generate path shows a good overall path with no collision with the obstacle.



(a)

k0	(14,2)	(12,2)	(11,10)	(7,11)	(7,11)
k1	(7,11)	(3,10)	(5,5)	(2,4)	(1,1)

(b)

k0	(1,1)	(4,2)	(5,5)	(6,6)	(4,11)
k1	(4,11)	(4,9)	(8,8)	(12,7)	(13,10)

(c)

k0	(3,9)	(1,6)	(4,5)	(5,3)	(1,1)
k1	(1,1)	(4,2)	(4,2)	(9,10)	(13,11)

Figure 5. Example of the path generated for (a) area 1, (b) area 2, and (c) area 3 with the chromosome configuration

Table 2 shows that all the methods are able to achieve an optimum and collision-free path shown by the same low minimum fitness. GPGA and SeqGA have comparably higher maximum and average fitness. This means GPGA and SeqGA can't prevent premature convergence because of slow rate of convergence throughout the entire population when using early stopping technique. Meanwhile, HPGA can achieve a low maximum and average fitness compared to other methods. This means that HPGA is very stable in solving a two-destination PP problem and can always deliver a near-optimum path without collision.

When looking at the average generation, other methods have a lower average generation. This is because other methods use a higher population than HPGA to achieve a similar result. The use of a higher population leads to an increase in execution time. While a low generation can mean a faster convergence, it can also lead to premature convergence when combined with an early stopping technique. HPGA can prevent premature convergence even when combined with early stopping with low patience value. When everything is considered HPGA can more consistently produce collision-free path results with faster execution times.

All tested parallel GA methods provide faster execution time than the SeqGA. GPGA could achieve up to 2.4 times speed up, IPGA up to 2.2 times, and HPGA up to 3.5 times speed up. HPGA could prevent premature convergence when combined with early stopping technique. HPGA method has a higher speedup compared to the other three methods with the maximum speedup value achieved in Area 2 with a value of 3.6 times. All the speedup comparisons are shown in Figure 6.

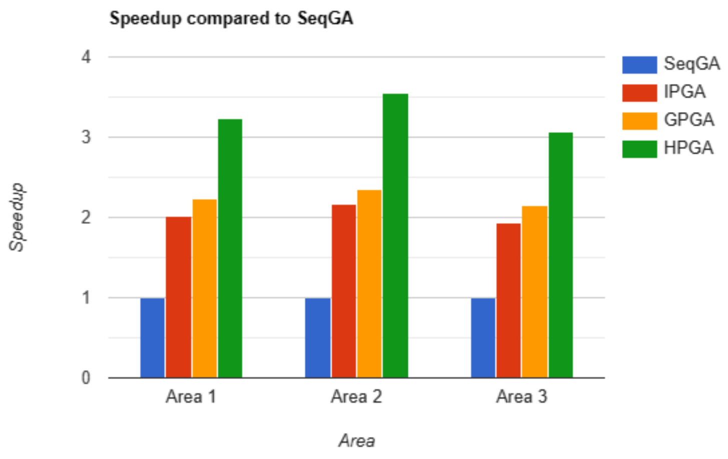


Figure 6. Speedup visualization of all the methods compared to SeqGA

4. CONCLUSION

In this experiment, GA is used to solve PP problems with multi-destination and multi-objective. Then a hybrid parallel method combining island and global parallel was proposed. HPGA resulted in an overall speedup of up to 3.6 times compared to the SeqGA. HPGA also shows a good result when combined with early stopping, preventing premature convergence while improving the overall execution time. This is likely contributed by implementing a global population filled by selecting parents from local populations, resulting in semi-random individuals inside the global population in every generation. The semi-randomness of the global population means that the best individual changes almost every generation and is only unchanged when all the local populations are close to converging. For the next experiment, a more improved and efficient code can be used to further improve the performance of GA for solving PP problems with multi-destination and multi-objective.

ACKNOWLEDGEMENTS

This work was partially supported by Department of Computer Science and Electronics, Universitas Gadjah Mada under the Publication Funding Year 2024.

REFERENCES

[1] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: a general overview," in *Mechanisms and Machine Science*, 2015, pp. 3–27. doi: 10.1007/978-3-319-14705-5_1.

[2] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path planning for autonomous mobile robots: a review," *Sensors*, vol. 21, no. 23, p. 7898, Nov. 2021, doi: 10.3390/s21237898.

[3] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[4] O. Kramer, *Genetic algorithm essentials*, vol. 679. in Studies in Computational Intelligence, vol. 679. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-52156-5.

[5] B. K. Patle, D. R. K. Parhi, A. Jagadeesh, and S. K. Kashyap, "Matrix-binary codes based genetic algorithm for path planning of mobile robot," *Computers & Electrical Engineering*, vol. 67, pp. 708–728, Apr. 2018, doi: 10.1016/j.compeleceng.2017.12.011.

[6] K. Li, Q. Hu, and J. Liu, "Path planning of mobile robot based on improved multiobjective genetic algorithm," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–12, Apr. 2021, doi: 10.1155/2021/8836615.

[7] T. V. Mathew, "Genetic algorithm," 2012. [Online]. Available: http://datajobstest.com/data-science-repo/Genetic-Algorithm-Guide-[Tom-Mathew].pdf

[8] J. Xin, J. Zhong, F. Yang, Y. Cui, and J. Sheng, "An improved genetic algorithm for path-planning of unmanned surface vehicle," *Sensors*, vol. 19, no. 11, p. 2640, Jun. 2019, doi: 10.3390/s19112640.

[9] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018, doi: 10.1016/j.procs.2018.01.113.

[10] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Systems with Applications*, vol. 115, pp. 106–120, Jan. 2019, doi: 10.1016/j.eswa.2018.08.008.

[11] M. Chattoraj and U. R. Vinayakamurthy, "A self adaptive new crossover operator to improve the efficiency of the genetic algorithm to find the shortest path," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 23, no. 2, p. 1011, Aug. 2021, doi: 10.11591/ijeecs.v23.i2.pp1011-1017.

[12] S. Ullah, A. Salam, and M. Masood, "Analysis and comparison of a proposed mutation operator and its effects on the performance of genetic algorithm," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 25, no. 2, p. 1208, Feb. 2022, doi: 10.11591/ijeecs.v25.i2.pp1208-1216.

[13] H. Guo, Z. Mao, W. Ding, and P. Liu, "Optimal search path planning for unmanned surface vehicle based on an improved genetic algorithm," *Computers & Electrical Engineering*, vol. 79, p. 106467, Oct. 2019, doi: 10.1016/j.compeleceng.2019.106467.

[14] Q. Li, W. Zhang, Y. Yin, Z. Wang, and G. Liu, "An improved genetic algorithm of optimum path planning for mobile robots," in *Sixth International Conference on Intelligent Systems Design and Applications*, IEEE, Oct. 2006, pp. 637–642. doi: 10.1109/ISDA.2006.253911.

[15] Y. Li, Z. Huang, and Y. Xie, "Path planning of mobile robot based on improved genetic algorithm," in *2020 3rd International Conference on Electron Device and Mechanical Engineering (ICEDME)*, IEEE, May 2020, pp. 691–695. doi: 10.1109/ICEDME50972.2020.00163.

[16] R. Sarkar, D. Barman, and N. Chowdhury, "Domain knowledge based genetic algorithms for mobile robot path planning having single and multiple targets," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 4269–4283, Jul. 2022, doi: 10.1016/j.jksuci.2020.10.010.

[17] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *Journal of Computational Science*, vol. 25, pp. 339–350, Mar. 2018, doi: 10.1016/j.jocs.2017.08.004.

[18] J. Ma, Y. Liu, S. Zang, and L. Wang, "Robot path planning based on genetic algorithm fused with continuous bezier optimization," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–10, Feb. 2020, doi: 10.1155/2020/9813040.

[19] X. Chen and P. Gao, "Path planning and control of soccer robot based on genetic algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 12, pp. 6177–6186, Dec. 2020, doi: 10.1007/s12652-019-01635-1.

[20] V. Jamshidi, V. Nekoukar, and M. H. Refan, "Analysis of parallel genetic algorithm and parallel particle swarm optimization algorithm UAV path planning on controller area network," *Journal of Control, Automation and Electrical Systems*, vol. 31, no. 1, pp. 129–140, Feb. 2020, doi: 10.1007/s40313-019-00549-9.

[21] K. S. Suresh, R. Venkatesan, and S. Venugopal, "Mobile robot path planning using multi-objective genetic algorithm in industrial automation," *Soft Computing*, vol. 26, no. 15, pp. 7387–7400, Aug. 2022, doi: 10.1007/s00500-022-07300-8.

[22] R. Shivgan and Z. Dong, "Energy-efficient drone coverage path planning using genetic algorithm," in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, IEEE, May 2020, pp. 1–6. doi: 10.1109/HPSR48589.2020.9098989.

[23] A. V. Le, N. H. K. Nhan, and R. E. Mohan, "Evolutionary algorithm-based complete coverage path planning for tetriamond tiling robots," *Sensors*, vol. 20, no. 2, p. 445, Jan. 2020, doi: 10.3390/s20020445.

[24] Y. Xue and J.-Q. Sun, "Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm," *Applied Sciences*, vol. 8, no. 9, p. 1425, Aug. 2018, doi: 10.3390/app8091425.

[25] T. Harada and E. Alba, "Parallel genetic algorithms: a useful survey," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–39, Jul. 2020, doi: 10.1145/3400031.

[26] J. R. Cheng and M. Gen, "Accelerating genetic algorithms with GPU computing: a selective overview," *Computers & Industrial Engineering*, vol. 128, pp. 514–525, Feb. 2019, doi: 10.1016/j.cie.2018.12.067.

[27] M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and J. M. Koushyar, "An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems," *Journal of Cloud Computing*, vol. 9, no. 1, p. 6, Dec. 2020, doi: 10.1186/s13677-020-0157-4.

[28] S. Amassmir, S. Tkatek, O. Abdoun, and J. Abouchabaka, "An intelligent irrigation system based on internet of things (IoT) to minimize water loss," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 25, no. 1, p. 504, Jan. 2022, doi: 10.11591/ijeecs.v25.i1.pp504-510.

[29] M. Farsi *et al.*, "Parallel genetic algorithms for optimizing the SARIMA model for better forecasting of the NCDC weather data," *Alexandria Engineering Journal*, vol. 60, no. 1, pp. 1299–1316, Feb. 2021, doi: 10.1016/j.aej.2020.10.052.

[30] J. Luo, S. Fujimura, D. El Baz, and B. Plazolles, "GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem," *Journal of Parallel and Distributed Computing*, vol. 133, pp. 244–257, Nov. 2019, doi: 10.1016/j.jpdc.2018.07.022.

[31] M. R. Rathomi and R. Pulungan, "A coarse-grained parallelization of genetic algorithms," *International Journal of Advances in Intelligent Informatics*, vol. 4, no. 1, p. 1, Apr. 2018, doi: 10.26555/ijain.v4i1.137.

BIOGRAPHIES OF AUTHORS

Luthfiansyah Ilhamnanda Yusuf is a master student in artificial intelligence at Universitas Gadjah Mada (UGM). Received a bachelor degree in computer science from Universitas Amikom Yogyakarta in 2022. His research interests include genetic algorithms, machine learning, and natural language processing. He can be contacted at email: luthfiansyahilhamnandayusuf@mail.ugm.ac.id.

Aina Musdholifah received a bachelor and master degrees in computer science, Universitas Gadjah Mada (UGM), Yogyakarta, Indonesia, in 2003 and 2006, respectively, and Ph.D. in Computer Science from Universiti Teknologi Malaysia (UTM). She is currently an Associate Professor and a member of the Intelligent System Laboratory, Department of Computer Science and Instrumentation, UGM, Yogyakarta. Her research interests include genetics algorithm, machine learning, fuzzy logic, and bioinformatics. She can be contacted at email: aina_m@ugm.ac.id.