Comparison of Memetic Algorithm and Genetic Algorithm on Nurse Picket Scheduling at Public Health Center

Nico[1]*, Novrido Charibaldi[2], Yuli Fauziah[3]
[1,2,3]Informatics Department, Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia
[1]nicokinazoko@gmail.com*; [2]novrido@upnyk.ac.id; [3]yuli.fauziah@upnyk.ac.id
*Corresponding author

ABSTRACT

One of the most significant aspects of the working world is the concept of a picket schedule. It is difficult for the scheduler to make an archive since there are frequently many issues with the picket schedule. These issues include schedule clashes, requests for leave, and trading schedules. Evolutionary algorithms have been successful in solving a wide variety of scheduling issues. Evolutionary algorithms are very susceptible to data convergence. But no one has discussed where to start from, where the data converges from making schedules using evolutionary algorithms. The best algorithms among evolutionary algorithms for scheduling are genetic algorithms and memetics algorithms. When it comes to the two algorithms, using genetic algorithms or memetics algorithms may not always offer the optimum outcomes in every situation. Therefore, it is necessary to compare the genetic algorithm and the algorithm's memetic algorithm to determine which one is suitable for the nurse picket schedule. From the results of this study, the memetic algorithm is better than the genetic algorithm in making picket schedules. The memetic algorithm with a population of 10000 and a generation of 5000 does not produce convergent data. While for the genetic algorithm, when the population is 5000 and the generation is 50, the data convergence starts. For accuracy, the memetic algorithm violates only 24 of the 124 existing constraints (80,645%). The genetic algorithm violates 27 of the 124 constraints (78,225%). The average runtime used to generate optimal data using the memetic algorithm takes 20.935592 seconds. For the genetic algorithm, it takes longer, as much as 53.951508 seconds.

Keywords : Memetic Algorithm; Genetic Algorithm; Runtime; Data Convergence; Picket Scheduling

Article History		
Received : May, 10th 2022	Revised : May, 25th 2022	Accepted : May, 27th 2022

I. INTRODUCTION

Hospital staff scheduling is a common problem in hospitals and health organizations. Scheduling is one of the essential aspects of workforce management and the most vulnerable to errors or problems because things must be considered, including the availability of nurse time, days off, etc. The nurse Scheduling Problem (NSP) is a variation of the staff scheduling problem that takes the case of nurse scheduling. The purpose of NSP is to increase the work productivity of nurses but still comply with the government [1].

In Kebumen III Public Health Center, a picket schedule is arranged once a month. However, in its implementation, there are several obstacles. The obstacles include making a schedule that is considered quite time-consuming. When there are employees who cannot or are unable to picket on that day, they will change the picket schedule. Another obstacle is when two employees agree to exchange schedules, but they have to change the schedule again for the schedule archive.

From these problems, many cases have been solved using the existing algorithms. However, the algorithms used by the previous researcher do not always produce the same results. Some cases can be solved by algorithm A with a small number of violations. However, if used in other cases, it can even produce convergent genes. Therefore, no algorithm can suit only the same case but must look at other factors such as the amount of data, computation time, number of constraints, and level of accuracy [2]. Of the several algorithms that have been tested, the evolutionary algorithm that is suitable for use for complex constraints and complicated sequence algorithms with fitness values are the Memetic algorithm, Genetic algorithm, Particle Swarm Optimization (PSO) [3], and Tabu Search Algorithm [2].

Several algorithms are combined with other algorithms, such as the Neuro-Fuzzy Genetic Algorithm. This algorithm combines a genetic algorithm with ANFIS (Adaptive Neuro-Fuzzy Inference System). This Neuro-Fuzzy Genetic Algorithm can optimally predict and generate nurse scheduling at the University of Benin Teaching Hospital (UTBH) nursing service department [4]. However, this study is limited to theory, and there is no calculation.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

There are two types of constraints called Hard Constraint and Soft Constraint. Hard Constraints are obstacles that must be avoided. Meanwhile, Soft Constraints are obstacles that can be ignored but can also be avoided. From several existing studies, almost all of the hard constraints used are the same, such as the schedule should not collide with other employees, the employee's working hours should not exceed a certain time, etc. In the previous research, there was a discussion about holiday constraints. But get a day off if you have reached a certain shift.

Convergent data is a phenomenon where individual data from the process that has a high fitness value dominates the population. Convergent data cannot be completely lost because the solution obtained from the evolutionary algorithm is random. Scheduling research using the memetic algorithm can overcome convergent data, but no one has yet explained when data converge. When here is meant, namely in what iteration or generation the data begin to converge. While in scheduling research, using genetic algorithms already exists but is not explained in detail.

In previous studies, when using genetic algorithms for picket scheduling when the 55th generation had started to show data convergence and continued to the next generation and did not change the final results [5]. For the number of violations, the larger the population and generation, the smaller the number of errors [6]. In testing the number of violations, the number of generations used is 1000 with a population of 200, resulting in the longest computation time. Still, the number of violations obtained is getting smaller. Studies that use the memetic algorithm do not explain how many populations are used but only change the number of generations used [7]. The average initial population used by the genetic algorithm and memetics is 100[2]. The average number of iterations used is different. The average genetic algorithm iteration is around 50-100 iterations. The average iteration of the memetic algorithm is around 500-1000. The larger the dataset, the more time it takes and the higher risk of schedule collisions [8].

The genetic-tabu search algorithm was compared with the ant colony algorithm in a previous study. The result is for a faster time achieved by the ant colony algorithm with 12.22 seconds. In comparison, the genetic-tabu search algorithm takes 122.86 seconds. However, for the fitness value, the genetic algorithm - tabu search is better, which is 0.0171 compared to the ant colony algorithm with a fitness value of 0.0135 [9]. In another study, the genetic algorithm took 880.430 seconds (14.7) minutes to meet the soft constraint and 109.270 seconds for the hard constraint with a population of 25 and a generation of 600 [10].

For lecture schedules, a genetic algorithm with a population of 50 and generation 30, the optimal fitness value of 0.667 does not conflict but violates the soft constraint of evening lectures [11]. The average error is 2.3% for the same problem with 16 seconds for scheduling lectures [12]. Still, in the same case, when using the genetic algorithm, the average execution time is 25.86 minutes with three lecture rooms, one hall, 51-course teachers, 18 lecturers, five working days, and 14 effective hours [13].

When the hybrid ant colony algorithm is compared with the memetic algorithm, the hybrid ant colony algorithm gets a penalty value of 10 with a time of 29.55 seconds. The memetic algorithm gets a penalty value of 2397 with 621.10 seconds [14]. When the number of populations is increased, and the number of generations is also the same, it tends to be the result of the genetic algorithm in processing the schedule to produce an optimal solution. The fastest time is achieved in producing an optimal schedule, which is when the population size is increased, and the number of generations is reduced [15]. When the mutation rate is 0.3, and the generation limit is 1000, it produces the highest fitness value of 665 with a suitability rate of 89.38% for generations of 10000 and above. Meanwhile, testing with an initial mutation rate value of 0.5 and a generation limit of 5000 resulted in the highest fitness value of 660 with a suitability rate of 87.37% for generations of 500000 and above [16].

Another study conducted tests with the genetic algorithm eight times with 10,000 evolutions. The system produces the smallest number of violations, as many as 8, and the average violation as much as 22 with a time of 74 seconds [17]. When the firefly improved algorithm is compared with the improved genetic algorithm, the more iterations, the fitness value will decrease to achieve the best fitness value [18].

In previous studies, no one has discussed data convergence in picket scheduling and variations in the use of the initial population used in the initial population. Then in the genetic algorithm, the number of populations used is only 200, which is quite a bit. From the description above, it will produce an application that produces two picket schedules for public health center employees, each using the Genetic Algorithm and the Memetics Algorithm. From this application, a comparison Table will be made between the Genetic Algorithm and the Memetics Algorithm to determine which method is better in Nurse Picket Scheduling. The reasons for taking the Memetics Algorithm can overcome hard constraints and soft constraints. It is a genetic algorithm but added with a local search so that the results can be more optimal [19], get a small error[18], and check when the data starts to converge. While the reasons for taking the Genetic Algorithm can overcome hard constraints and soft constraints, and it is an algorithm that is suitable for scheduling

There will be additional constraints, namely holidays and constraints for each employee and different picket codes. From the two algorithms, the results will be compared based on the parameters that have been determined. Comparing the two algorithms is to compare which algorithm is better for scheduling pickets for public health center employees. Another reason is to find out which algorithm has the least number of violations and how many iterations the data begin to converge. There will be variations in the number of the initial population and the number of iterations used to be used as a reference in comparison.

II. METHOD

The flow of the research method in this study can be seen in Fig. 1, and the research method contains the steps taken in the study, from data collection to testing, so that finally, the research conclusions can be drawn.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

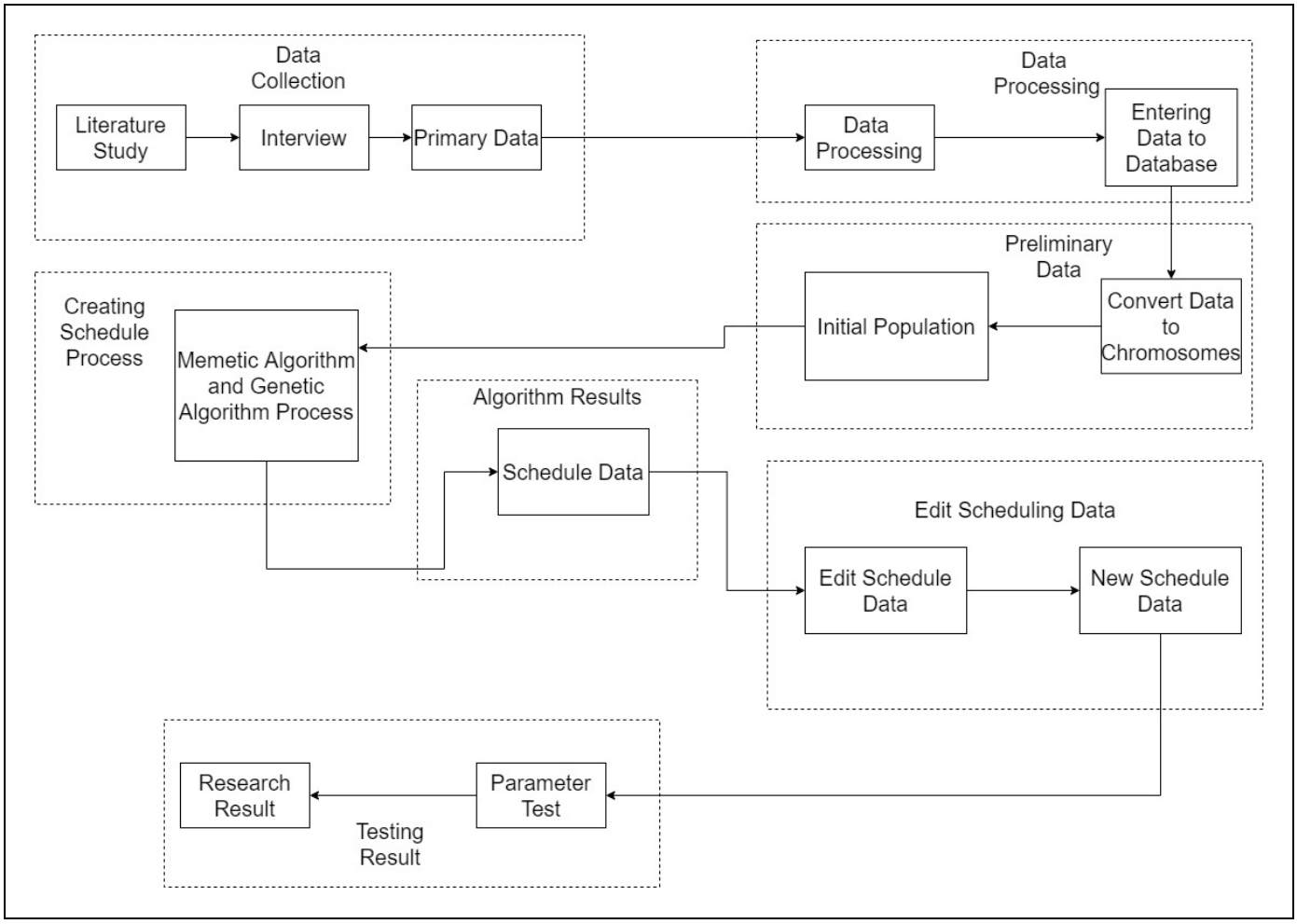

Fig. 1. Research Method

The data obtained are primary. Data were obtained from the Kebumen III Public Health Center in employee data, picket data, and picket provisions obtained from interviews with scheduler employees. The data obtained are 13 employee data, 15 picket data, and 21 picket provisions. After getting the data, the picket and employee data will be saved into the database. The provision of picket data will be made into data constraint, which will later be used as calculations in the genetic algorithm and memetic algorithm.

After all, data is entered into the database except for the data constraint, then when the user wants to create data, the process will start. The data constraint will be used in the process without entering the constraint data into the database. The process starts by taking three data randomly, starting from employee data, picket data, and date data, as many as the number of generations that have been inputted to make the initial population chromosomes. After getting the chromosomes, the fitness value of each chromosome will be calculated.

The function used to calculate the fitness value in the Genetic Algorithm, and the Memetics Algorithm in this study uses the same function. In some cases, the fitness value function is not precisely the same. The determination of the fitness value function is based on how well the solution fits the existing state of all available solutions [20]. The fitness value calculation is in Equation (1). Where *x* variable is employee chromosome fitness value, *y* variable is picket chromosome fitness value, and z variable is chromosome fitness value day

$$Fitness\ Value = \begin{cases} x + y + z, & x, y, z \neq \{\} \\ -1, & z = Sunday \\ -1, & x\ or\ y\ or\ z = \{\} \end{cases} \quad (1)$$

The maximum value for the fitness value is three (3), and the minimum value for the fitness value is a negative one (-1). A value of three (3) must be taken for the system, while a negative value of one (-1) must be avoided. The calculation of the fitness value

can be illustrated as follows:

- On October 16, 2021, employee A received picket A.
  a. October 16, 2021, is Saturday, not Sunday, fitness value + 1
  b. Employee data is in the database; fitness value + 1
  c. Date and constraint don't meet requirements, fitness value – 1 and Total Fitness Score = 1 + 1 - 1 = 1
- On October 24, 2021, Employee B received a picket C.
  October 24, 2021, is a Sunday, fitness value – 1
  Total Fitness Score = -1
- On October 18, 2021, Employee C got picket B
  a. October 18, 2021, is Monday; fitness value + 1
  b. Employee Data is in the database; fitness value + 1
  c. Date and constraint meet requirements, fitness value + 1, and Total fitness value = 1 + 1 + 1 = 3

In conducting system testing, the system is tested three times. The system is tested separately between the genetic algorithm and the memetic algorithm. The data entered into the system are in Table I:

TABLE I
LIST OF DATA ENTERED

Parameter	Genetic Algorithm	Memetic Algorithm
Total Population	500	500
	1000	1000
	5000	5000
	10000	10000
Total Iteration	50	50
	100	100
	500	500
	1000	1000
	5000	5000
Crossover Rate	0.6	0.6
Mutation Rate	0.01	0.01

For calculation of the accuracy for each algorithm is in Equation (2). Where total fitness 3 is the number of chromosomes that have a fitness value of 3 in the schedule, and total maximum fitness three available is total maximum fitness three available (124)

$$Accuracy = \frac{total\ fitness\ 3}{total\ maximum\ fitness\ 3\ available}\ x\ 100\% \qquad (2)$$

The maximum number of fitness value three data points that can be retrieved is 124. How to get at the value 124, which can be done by computing all of the possible outcomes given the limitations that are already in place. It is computed using the number of fitness 3 in the schedule and the number of fitness three that is possible given all of the constraints. This ensures that the result is accurate. In order to determine which algorithm is more accurate, a comparison will be made between the genetic algorithm and the memetic algorithm. First, the genetic algorithm will be used to determine which schedule data has the highest fitness value of 3, and the memetic and genetic algorithms will be compared against one another to determine which algorithm is more accurate.

III. RESULT AND DISCUSSION

The test results data will be found which algorithm has the optimal solution. Optimal here means that the requirements for the amount of schedule data with a minimum fitness value must be met first, the amount of schedule data with a maximum fitness value is quite a lot, does not take up a lot of time, and there is no data convergence. The following is an explanation of the parameters used as an optimal comparison:

- Number of generations
- Population
- The amount of data with the maximum fitness value. The best-selected chromosome is a chromosome with a fitness value of three (3). The more there are, the better.
- Number of chromosomes with the minimum fitness value
  a. The chromosome with the minimum fitness value combines schedule data with a negative fitness value of one (-1) and an empty fitness value. A negative fitness value of one is obtained if the schedule data is Sunday, and an empty fitness value is obtained if the schedule data does not exist.
  b. The recommended number is the number of employees times the number of Sundays in that month. For example, data for January 2022 will be used for test data, which is five (5) Sundays. The number of the best minimum fitness values is $Total\ Fitness\ Minimum\ Value = 13 * 5 = 65$

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

c. If more than 65, then the schedule data is not good and is better avoided

- Runtime (process time). The time used for the process must be the shortest but has the most optimal results.
- Data convergence. There will be a check done on the data for the schedule to see whether or not there is convergence in the data. Even if the optimal criteria are satisfied, the outcomes will not be optimal since there will be no diversity in the schedule data. This will be the case even if the optimal conditions are met.

For details of the comparison of the algorithms used are as follows:

- Runtime with total generations for a comparison of the number of generations with the runtime of the memetic algorithm, it can be seen in Fig. 2 where the more the number of generations, the longer the runtime, with the longest time being 64.19791 seconds for the shortest time that is for 1.05668 seconds. If the number of generations is 5000, the processing time will be more than 30 seconds, and the maximum is 64 seconds.

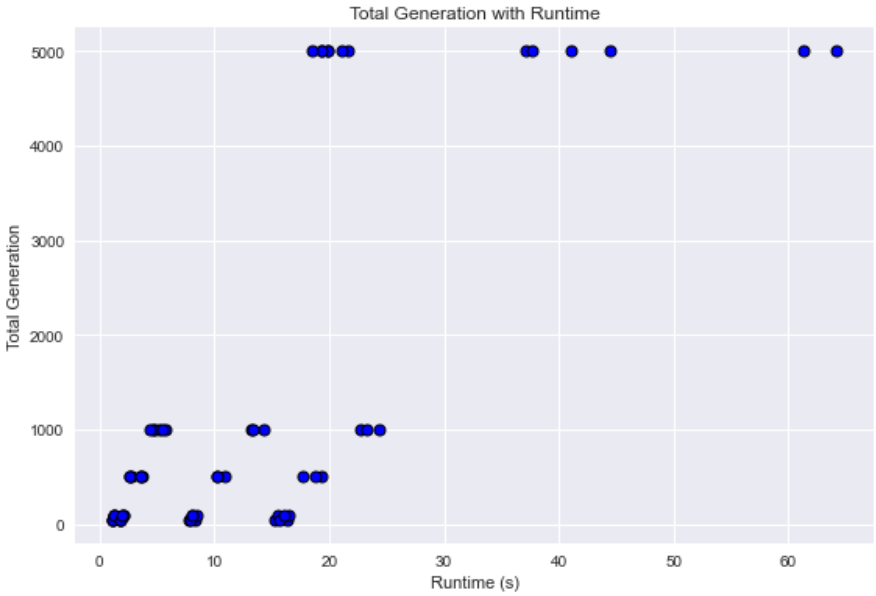

Fig. 2. Runtime with Total Generations with Memetic Algorithm

A comparison of the number of generations with the runtime of the genetic algorithm can be seen in Fig. 3. The longest runtime is obtained from the number of generations of 5000, 361.70901 seconds. The shortest runtime is in the 50th generation for 1.0492 seconds. If the number of generations is 5000, the runtime can last for more than 360 seconds. However, there is also a population of 5000 with a short runtime of 24.77688.

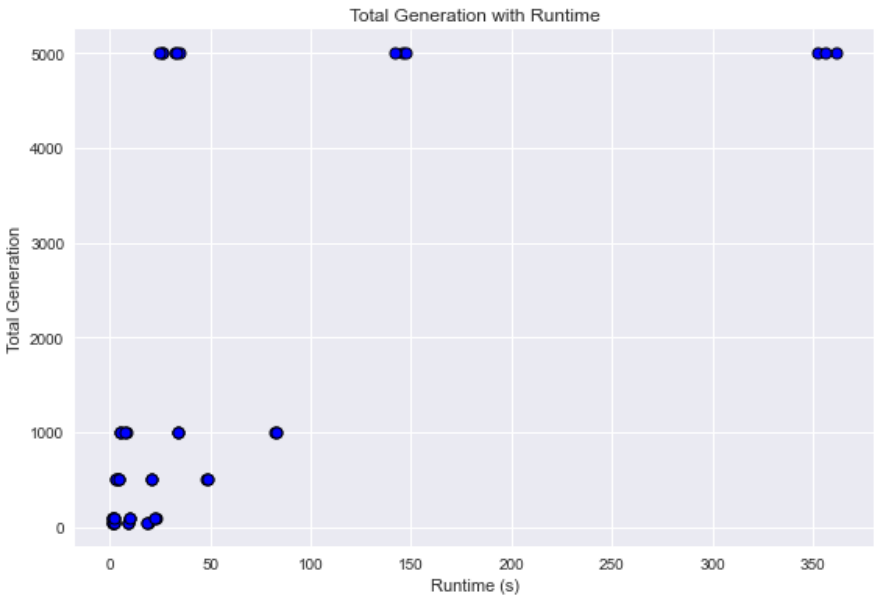

Fig. 3. Runtime with Total Generations with Genetic Algorithm

- Runtime with total population for a comparison of the population with the runtime of the memetic algorithm in Fig. 4. If the population is less than 2000, the runtime is less than 30 seconds. The longest time was in a population of 10000 with a runtime of 64.19791 seconds, while the shortest time was in the total population of 500 with a runtime of 1.05668 seconds.

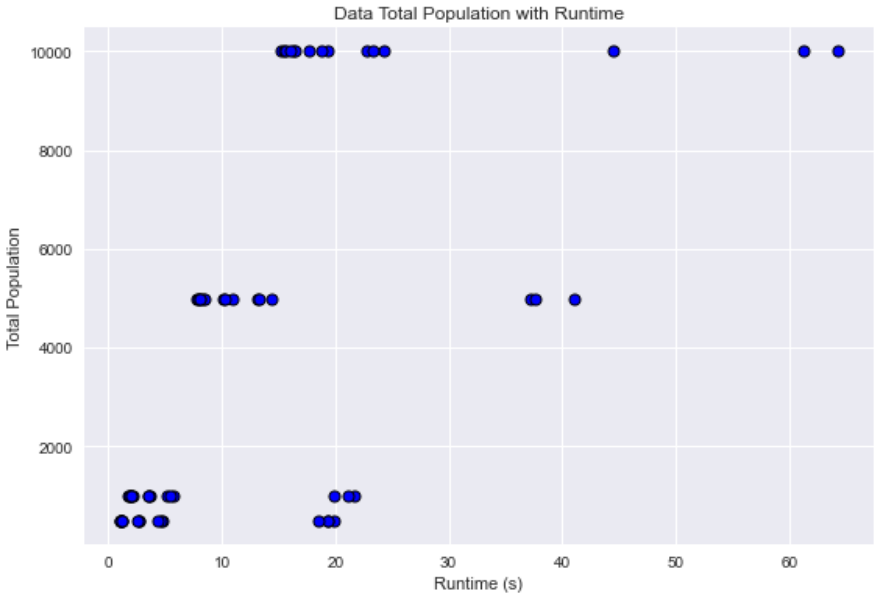

Fig. 4. Runtime with Total Population with Memetic Algorithm

For a comparison of runtime with the total population of the genetic algorithm, see Fig. 5 below. The longest runtime was obtained from a total population of 10000, which was 361.70901 seconds. If the population is less than 2000, then the runtime is not more than 50 seconds, with the longest runtime being 34.45983 seconds.

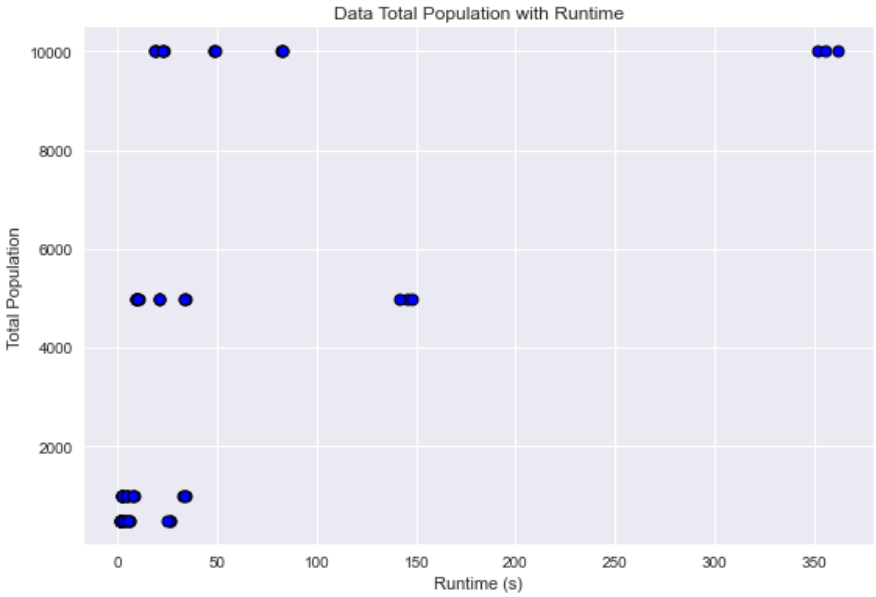

Fig. 5. Runtime with Total Population with Genetic Algorithm

- Runtime with minimum total fitness value. The results of the comparison of schedule data with the minimum fitness value with the runtime algorithm can be seen in Fig. 6 below. The longer the processing time, the fewer scheduled data with the minimum fitness value obtained and met the criteria, which is 65. The fastest runtime to get the minimum number of fitness values that meets the criteria is 7.81527 seconds. When the runtime is more than 30 seconds, the amount of

schedule data with the minimum fitness value obtained is optimal. The amount of schedule data with the optimal minimum fitness value is also in the time range between 7.81527 seconds to 19.34257.

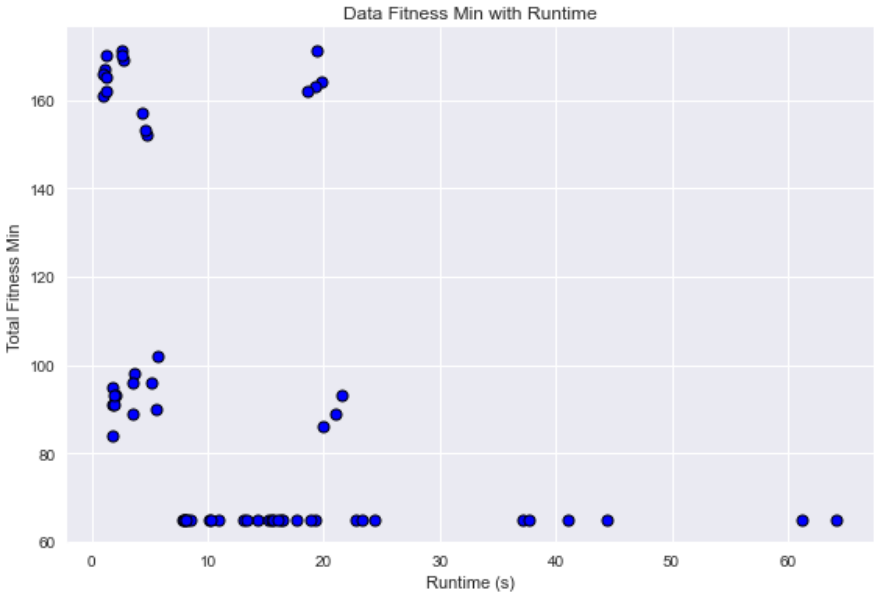

Fig. 6. Runtime with Minimum Total Fitness Value with Memetic Algorithm

For a runtime comparison with the amount of data, the minimum fitness value of the genetic algorithm can be seen in Fig. 7 below. If the runtime exceeds 50 seconds, then the amount of schedule data with the minimum fitness value will always be optimal.

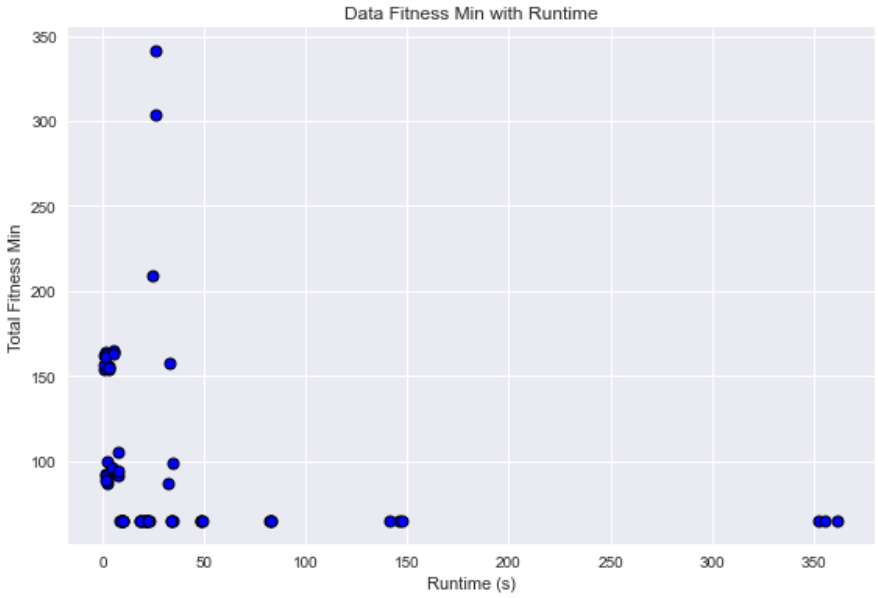

Fig. 7. Runtime with Minimum Total Fitness Value with Genetic Algorithm

- Runtime with maximum total fitness value. The results of comparing the amount of schedule data with the maximum total fitness value with the memetic algorithm runtime can be seen in Fig. 8. The time to get the maximum total fitness value schedules is 15.5523 seconds and 44.42965 seconds. When the runtime runs for more than 60 seconds, the amount of data for the maximum total fitness value is more than the runtime for less than 10 seconds.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

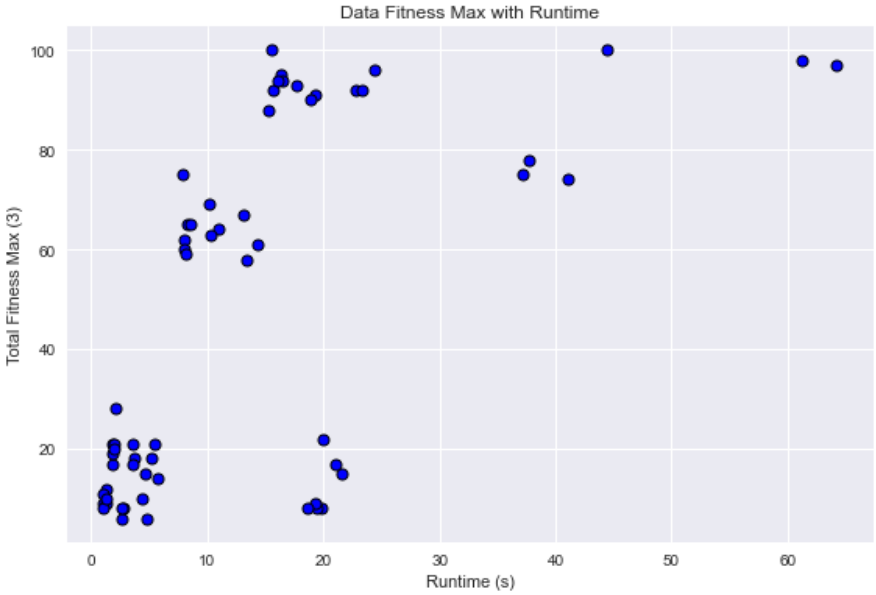

Fig. 8. Runtime with Maximum Total Fitness Value with Memetic Algorithm

The genetic algorithm's fitness value can be seen in Fig. 9 below to compare runtime with the maximum total fitness value. The time to get the highest amount of data maximum total fitness value is 22.05384 with 98. If the time is more than 350 seconds, then the amount of data for the highest fitness value will be more than the small runtime.

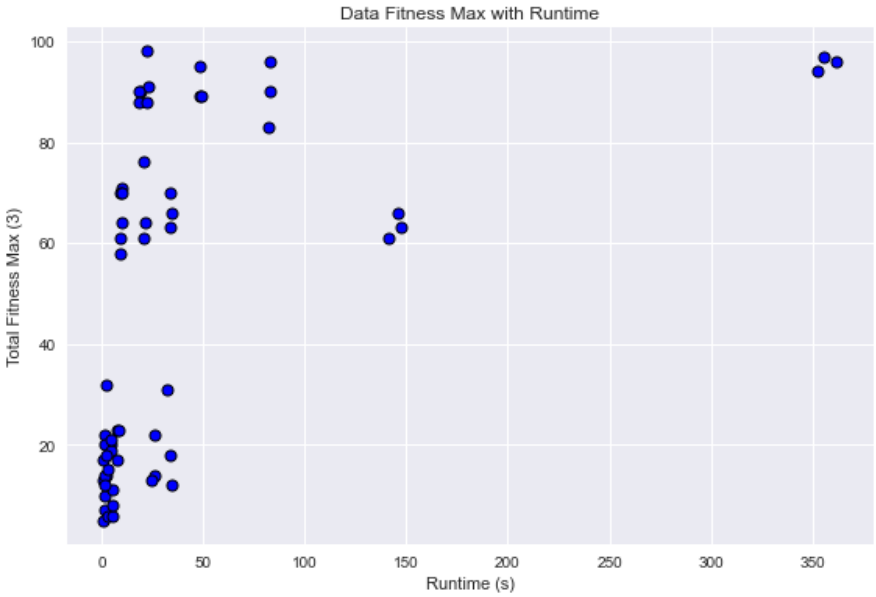

Fig. 9. Runtime with Maximum Total Fitness Value Data with Genetic Algorithm

- The total minimum fitness value and the total population compare the number of generations with the schedule data to the minimum fitness value of the memetic algorithm shown in Fig. 10. The number of populations employed ranges from 5000 to 10000 to obtain the required quantity of scheduled data while maintaining the best minimum fitness value. In the event that the population is lower than that number, the quantity of schedule data with the least fitness value will not be optimal. The amount of schedule data with the lowest fitness value is in the 500th population, and it has 171 in comparison to the other amounts of schedule data.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

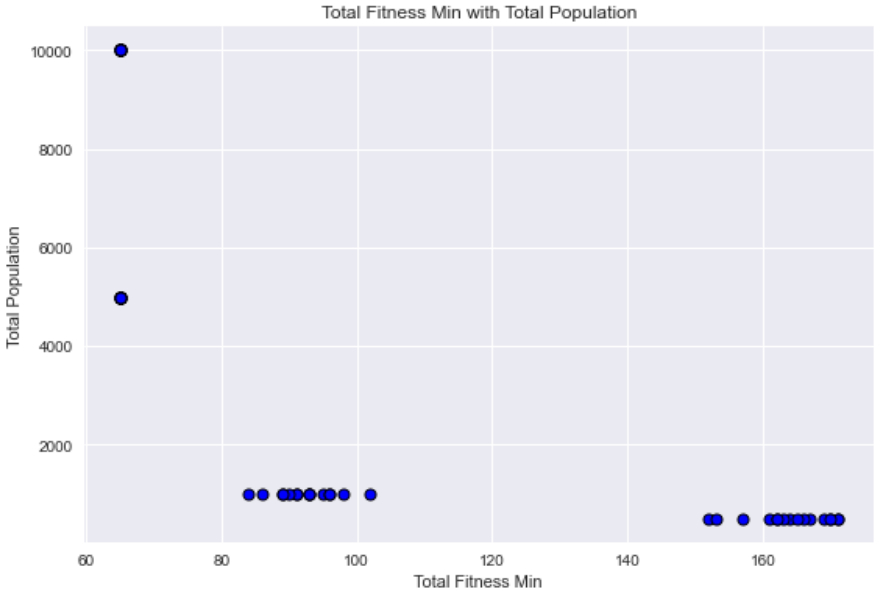

Fig. 10. Total Minimum Fitness Value with Total Population with Memetic Algorithm

To compare of the total minimum fitness value with the population of the genetic algorithm can be seen in Fig. 11 below. The total minimum fitness value is in the total population of 5000 and 10000. If the total population is 1000 and 500, the highest fitness value exceeds the optimal value.

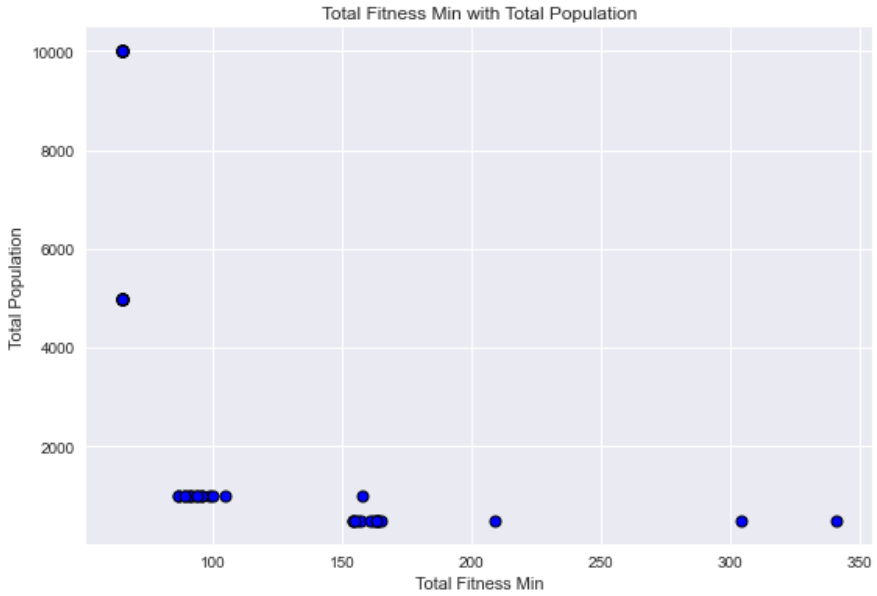

Fig. 11. Total Minimum Fitness Value with Total Population with Genetic Algorithm

- Total minimum fitness values are divided by the number of generations to compare the number of generations to schedule data along with the total minimum fitness value shown in Fig. 12. When the number of generations is less than 1000 and greater than 5000, the data for the minimal fitness value can be considered ideal. On the other hand, the quantity of data on the minimal fitness value is still subject to change between the two generations, so it cannot serve as a reference.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

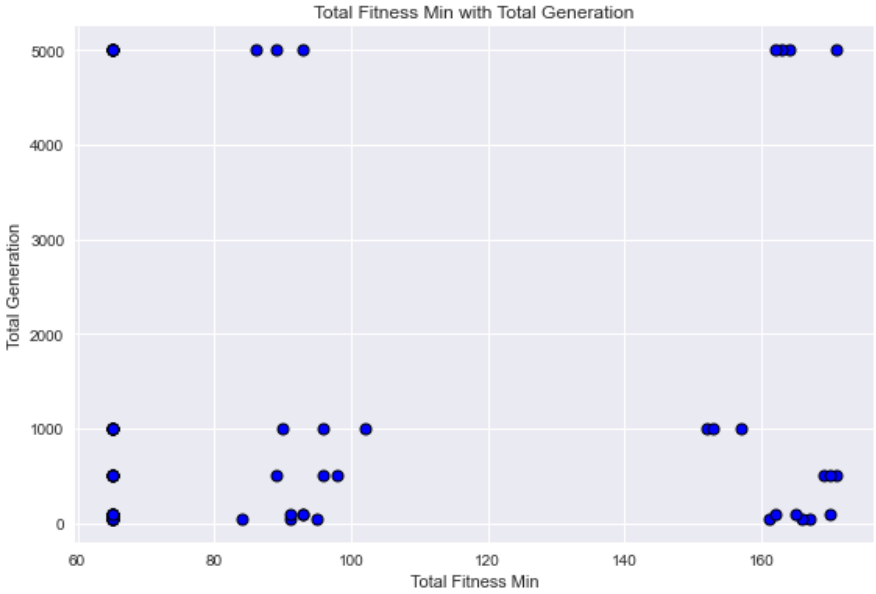

Fig. 12. Total Minimum Fitness Values by Number of Generations with Memetic Algorithm

To compare of the total minimum fitness value with the number of generations of the genetic algorithm can be seen in Fig. 13 below. For the total minimum fitness value, as many as 341 are in the number of generations of 5000. For a population of 5000 total minimum fitness value varies.

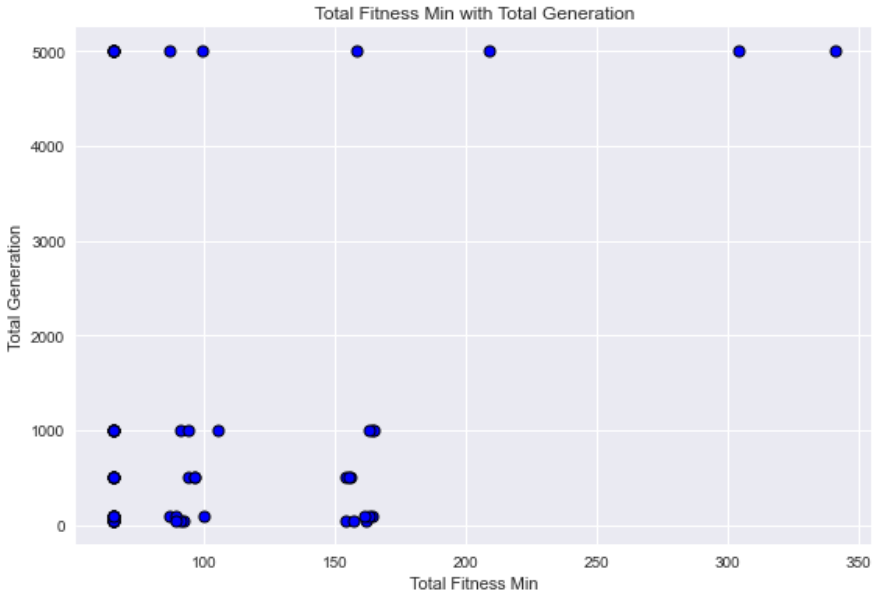

Fig. 13. Total Minimum Fitness Values by Number of Generations with Genetic Algorithm

- Total maximum fitness value with total population to compare the population with schedule data with the highest fitness value, the memetic algorithm can be seen in Fig. 14. The larger the population, the more data with the highest fitness value. The amount of data with the highest fitness value is 100 with a population of 10000.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

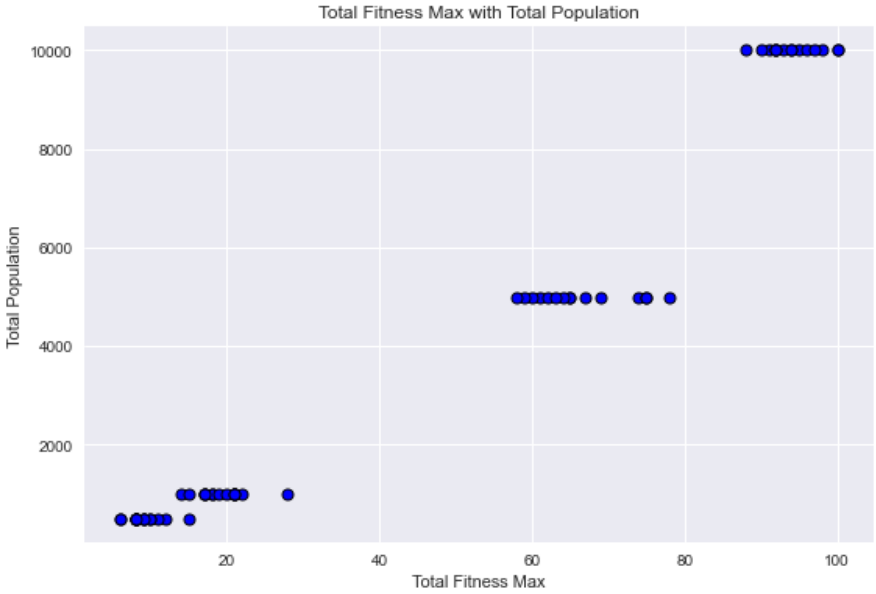

Fig. 14. Total Maximum Fitness Values by Total Population with Memetic Algorithm

For a comparison of the highest number of fitness values with the genetic algorithm population, it can be seen in Fig. 15. The highest number of fitness values, 98, is in a population of 10000. The larger the population, the greater the number of the highest fitness values obtained.

Fig. 15. Total Maximum Fitness Values by Total Population with Genetic Algorithm

- The total maximum value of fitness is divided by the number of generations, with the goal of comparing the number of generations to the schedule data with the highest fitness value in Fig. 16. When there are 100 and 5000 generations, the amount of data obtained has the largest fitness value. This is the case regardless of the number of generations.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

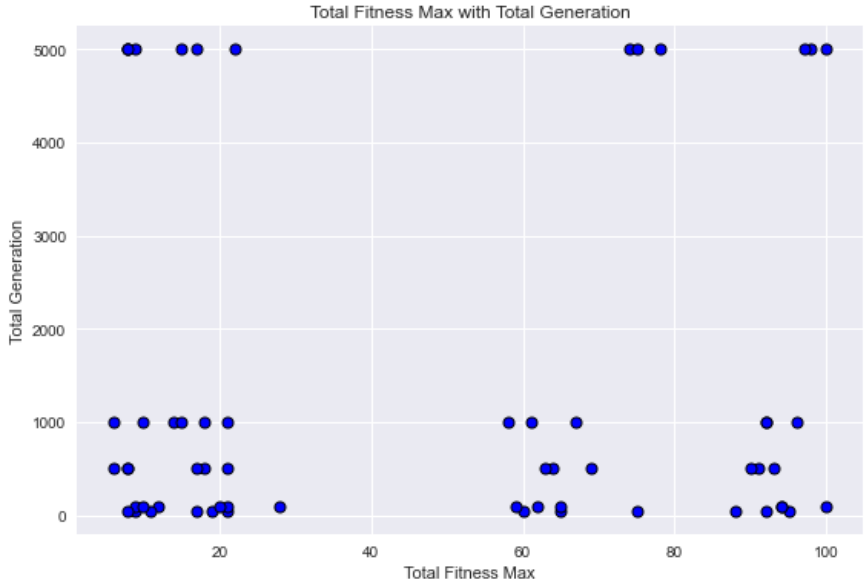

Fig. 16. Total Maximum Fitness Value by Number of Generations with Memetic Algorithm

For a comparison of the highest number of fitness values with the number of generations of genetic algorithms in Fig. The number of generations, which can be as high as 100, has the highest number of fitness values of any category, 98. The data obtained to produce good scheduling results are in Table II from the comparison results above.

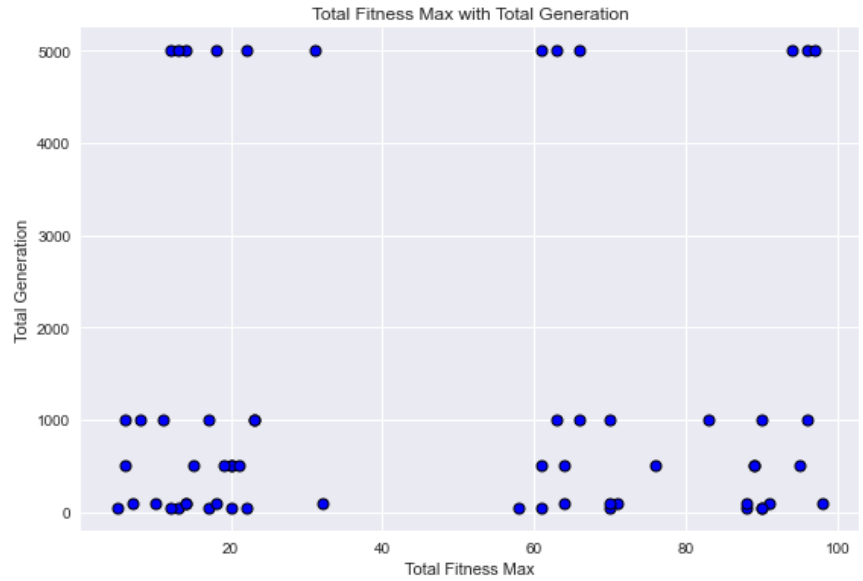

Fig. 17. Total Maximum Fitness Value by Number of Generations with Genetic Algorithm

TABLE II
DATA CONVERGENCE TABLE

Parameter	Memetic Algorithm	Genetic Algorithm
Total Population	More than 5000	More than 5000
Total Generation	Variative	Variative
Maximum Total Fitness	100	98
Minimum Total Fitness	171	341
Best Runtime (seconds)	More than 30 seconds	More than 48 seconds
Fastest Runtime (seconds)	1.05668	1.09492
Longest Runtime (seconds)	64.19791	361.70901
Runtime Average	13.692163	41.739482

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323

From Table VI, the minimum population used is 5000, with a varied number of generations. The minimum population of 5000 is to achieve the optimal amount of data for the minimum fitness value (65), that is, when the total population used is 5000. It is not the same for the best runtime chosen to produce the minimum number of fitness values between the memetic algorithm and the memetic algorithm. For the memetic algorithm will produce an optimal schedule, the best runtime is if the runtime is more than 30 seconds. For genetic algorithms to produce an optimal schedule, the best runtime is if the runtime is more than 48 seconds.

From the results of Table VI, it is still impossible to conclude which algorithm is the best between the memetic algorithm and the genetic algorithm because there are parameters that have not been used as checks, namely data convergence. The best algorithm can be chosen if there is no data convergence. For checking, the memetic and genetic algorithm's scheduling data will be taken with a minimum population of 5000. From three attempts in Table I, one data from each generation is taken with the most fitness value 3. Table III summarizes the population and the number of generations used and whether the data are convergent or not.

TABLE III
DATA CONVERGENCE TABLE

Total Population	Total Generation	Data Convergence / Not		Total Maximum Fitness		*Runtime*	
		Memetic Algorithm	Genetic Algorithm	Memetic Algorithm	Genetic Algorithm	Memetic Algorithm	Genetic Algorithm
5000	50	No	Yes	75	70	7.81527	9.05295
	100	No	Yes	65	71	8.46885	10.1639
	500	No	Yes	69	76	10.17643	20.97574
	1000	No	Yes	67	70	13.13913	33.69411
	5000	No	Yes	78	66	37.67181	145.55149
10000	50	No	Yes	95	90	16.27991	18.92156
	100	No	Yes	100	98	15.5523	22.05384
	500	No	Yes	93	95	17.63324	48.31349
	1000	No	Yes	96	96	24.34925	82.77366
	5000	No	Yes	100	97	44.42965	355.82328

From the results of Table III, it can be concluded that the memetic algorithm with a minimum population of 5000 with various generations can produce scheduling data without data convergence. The best scheduling data is when the total population is 10000, with the number of generations being 100 and 5000 producing a maximum fitness value of 100. For the fastest runtime, the number of generations is 100, which is 15.5523 seconds.

For genetic algorithms, from the number of generations, all generations experience data convergence with the data that appears the most, namely the "PCare RPU" and "RPU" picket data. When the total population is 10000, the maximum fitness value is more than 90. For scheduling with the highest maximum fitness value of 98, the population is 10000, with the number of generations being 100. For the fastest runtime, the population is 10000, with a total population of 10000. generation as many as 100 that is for 22.05384. For a comparison between the runtime and the accuracy of each algorithm, see Fig. 18.

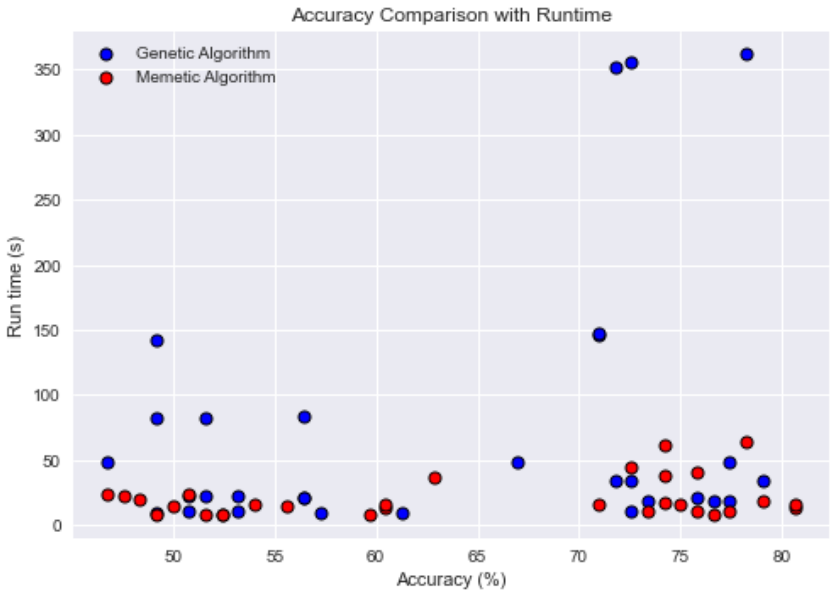

Fig. 18. Comparison Between Accuracy and Runtime of Each Algorithm

From the picture above, it can be seen that the memetic algorithm has a fairly short runtime, which is under 60 seconds. While genetic algorithms "can" compete, there is a runtime that reaches more than 100 seconds, but its accuracy is fewer than a runtime of fewer than 50 seconds. The results of comparing the memetic algorithm and the algorithm are in Table IV.

TABLE IV
COMPARISON TABLE OF MEMETIC ALGORITHM AND GENETIC ALGORITHM

Parameter	Memetic Algorithm	Genetic Algorithm
Average Number of Generations	50-5000	50-5000
Initial Population	5000	5000
Average Running Time (seconds)	20.935592	74.771624
Convergent Data Started	More than 10000 total population and 5000 generations	Total population 5000 and number of generations 50
Best Accuracy (%)	80.645 (100 of124)	78.225 (97 of 124)

Table IV shows that the Memetic Algorithm is better than the genetic algorithm when viewed from the five parameters used. It is practically the same for the average number of generations and initial population initialization between the memetic algorithm and the genetic algorithm. The average runtime of the memetic algorithm is faster than the genetic algorithm, with an average runtime of the memetic algorithm of 20.935592 seconds. Compared with the memetic algorithm, the average runtime of the genetic algorithm is 74.771624seconds.

Data convergence has not been obtained for scheduling using the memetic algorithm if the population is 10000 and the number of generations is 5000. For the genetic algorithm, data convergence has started at the 5000th population, and the number of generations is 50. Of the total number of constraints, as many as 124, the memetic algorithm is better than the genetic algorithm because the number of violations in the memetic algorithm is less than the genetic algorithm. The best accuracy of the memetic algorithm is 80.645%, 2.24% higher than the genetic algorithm, which is only 78.225%.

IV. CONCLUSION

Based on the research that has been done. The better algorithm between the memetic algorithm and the genetic algorithm used for picket scheduling is the memetic algorithm because the memetic algorithm still chooses chromosomes with poor fitness values to be replaced with mutated chromosomes with better fitness values. In contrast, the genetic algorithm chooses chromosomes randomly to be replaced with chromosomes with better fitness values. The accuracy of the memetic algorithm is higher than the genetic algorithm. The memetics algorithm has an accuracy of 80.645% compared to the genetic algorithm, only 78.225%. The number of violations is small, which is 24 of the 124 total constraints (0.1935 %) compared to the number of violations of the genetic algorithm, which is 27 of the total constraints (0.2177 %). For a faster average runtime, the memetic algorithm with an average runtime of 20.935592 seconds compared to the genetic algorithm with an average runtime of 74.771624 seconds, with a note that the average runtime here is calculated based on the runtime needed to get schedule data. Optimal (the minimum number of fitness values is 65). The genetic algorithm has experienced data convergence when the population is 5000 with 50 generations. Compared to the memetic algorithm with a population of 10000 and the number of generations of 5000, there is no data convergence. Initial population initialization of the two algorithms starts from a total population of 5000 with various generations. The larger the population, the better the results because the initial population is very large, and each algorithm has more chromosome selection options as well. In future works, it is possible to add optimization methods to the genetic algorithm to reduce the high data convergence.

REFERENCES

[1] A. A. El Adoly, M. Gheith, and M. Nashat Fors, "A new formulation and solution for the nurse scheduling problem: A case study in Egypt," *Alexandria Eng. J.*, vol. 57, no. 4, pp. 2289–2298, Dec. 2018.
[2] A. S. Koli, T. Rajodwala, and A. Mittal, "Comparison of Evolutionary Algorithms for Timetable Scheduling," pp. 1351–1359, 2020.
[3] R. Hariyanto and M. Z. Sarwani, "Optimizing K-Means Algorithm by Using Particle Swarm Optimization in Clustering for Students Learning Process," *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 6, no. 1, pp. 65–68, 2021.
[4] F. I. Amadin, D. Ph, M. E. Bello, and B. Sc, "A Genetic Neuro Fuzzy Approach for Handling the Nurse Rostering Problem," vol. 19, no. June, pp. 198–205, 2018.
[5] Y. Yudriani, E. C. Djamal, and R. Ilyas, "Optimalisasi Penjadwalan Jaga Dokter dan Tenaga Medik di Rumah Sakit Dustira Menggunakan Algoritma Genetika," no. September, pp. 40–44, 2017.
[6] Andriansyah, N. Alfadilla, P. D. Sentia, and D. Asmadi, "Optimization of Nurse Scheduling Problem using Genetic Algorithm: A Case Study," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 536, no. 1, 2019.
[7] A. Y. E. Dodu, D. W. Nugraha, and S. D. Putra, "Penjadwalan Tenaga Kebidanan Menggunakan Algoritma Memetika," *J.*

Sist. Inf. Bisnis, vol. 8, no. 1, p. 99, 2018.
[8] C. T. TAURAI, "A Timetable Planner for Bindura University of Science Education using Genetic Algorithm," no. June, 2017.
[9] N. I. Kurniati, A. Rahmatulloh, and D. Rahmawati, "Perbandingan Performa Algoritma Koloni Semut Dengan Algoritma Genetika – Tabu Search Dalam Penjadwalan Kuliah," *Comput. Eng. Sci. Syst. J.*, vol. 4, no. 1, p. 17, 2019.
[10] Y. Sari, M. Alkaff, E. S. Wijaya, S. Soraya, and D. P. Kartikasari, "Optimasi Penjadwalan Mata Kuliah Menggunakan Metode Algoritma Genetika dengan Teknik Tournament Selection," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 6, no. 1, p. 85, 2019.
[11] H. Husada, I. Cholissodin, and F. A. Bachtiar, "Optimasi Penjadwalan Kuliah Pengganti Menggunakan Algoritme Genetika," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 9, 2018.
[12] A. Haifan and dkk, "Sistem Penjadwalan Praktikum Menggunakan Algoritma Genetika," *Tek. Inform. – ITN Malang*, vol. 4, no. 1, pp. 1–8, 2018.
[13] F. Mone and J. E. Simarmata, "Application of Genetic Algorithm in Scheduling Subjects," vol. 15, no. 4, pp. 615–628, 2021.
[14] M. Cai and Y. Zuo, "A hybrid ant colony optimization algorithm based on MapReduce," vol. 3, no. 3, pp. 305–326, 2016.
[15] M. R. M. Umar Hasan, Teguh Iman Hermanto, "PENJADWALAN MATA KULIAH MENGGUNAKAN ALGORITMA GENETIKA DI STT WASTUKANCANA PURWAKARTA," 2018.
[16] H. Pranata, "Sistem penjadwalan shift kerja anggota kepolisian menggunakan algoritma genetika," 2019.
[17] A. Yudistira, E. C. Djamal, and R. Yuniarti, "Optimalisasi Penjadwalan Audit di Inspektorat Daerah Kabupaten Cianjur Menggunakan Algoritma Genetika," *Semin. Nas. Apl. Teknol. Inf.*, pp. 1907–5022, 2017.
[18] A. Goli, E. B. Tirkolaee, I. Mahdavi, and M. Zamani, "Solving a University Exam Scheduling Problem Using Genetic and Firefly Algorithms."
[19] V. Sahargahi and M. F. Drakhshi, "Comparing the Methods of Creating Educational Timetable," vol. 16, no. 12, pp. 26–36, 2016.
[20] S. E. Haupt, "Introduction to genetic algorithms," *Artif. Intell. Methods Environ. Sci.*, pp. 103–125, 2009.

DOI: http://dx.doi.org/10.25139/ijair.v4i1.4323