Journal of Engineering Science and Technology
Vol. 15, No. 1 (2020) 278 - 292

COMPLEXITY AND PERFORMANCE COMPARISON OF GENETIC ALGORITHM AND ANT COLONY FOR BEST SOLUTION TIMETABLE CLASS

SYAHRUL MAULUDDIN[1,*], ISKANDAR IKBAL[2], AGUS NURSIKUWAGUS[3]

[1,3] Department of Information System, Universitas Komputer Indonesia
[2] Department of Infomatics Engineering, Universitas Komputer Indonesia
[1,2,3] Jl. Dipatiukur No. 102-116, Bandung, 40132, Indonesia
*Corresponding Author: syahrul.mauluddin@email.unikom.ac.id

Abstract

Ant colony and genetic algorithm are two solution methods used to determine the efficiency of the search time for a solution. This study aims to make a comparison of ant colony and genetic algorithm in terms of providing the results of the learning schedule in the department. This comparison uses the same variables to tested on ant colony and genetic algorithm. Variables used for department majors such as the number of lecturers, days, number of classes, time slots per day. This comparison also uses the asymptotic complexity method to see the asymptotic complexity of using ant colony and genetic algorithm. The results of the investigation obtained, the average genetics algorithm performance time obtained by ten tests is 0.9856s and the average number of cycles is 4.9. In ant colony performance, the average time obtained by ten tests is 6.6251s and the average number of cycles is 6.3. This result has obtained for investigation of 10 trials, with each time the experiment reaches 4 to 8 cycles. This investigation uses 96 slots with 109 available slot timetables. Asymptotic Complexity for ant colony is n^4, and genetic algorithm is n^3. A comparison of ant colony and genetic algorithm illustrates that genetic algorithm is the fastest algorithm for scheduling cases. This process gaves a best solution effect speed to get the best scheduling solution.

Keywords: Ant colony, Complexity, Genetic algorithm, Optimization, Performance, Timetable class.

1. Introduction

Preparation of timetables in the department is a combination of the number of lecturers, classes, time slots, and days that arrange based on an acquire the lecture needs. This combination has known as departmental resources, which uses as vary a determinant regulation for timetable allocation. Resources have used to obtain the best solution timetable arrangement with conflicting timetables = 0 with the constraint predefined. The conflicting timetable = 0, it means that timetables have arranged following the available resources have not collided in every slot. The process timetable search has aimed to reduce the number of class attendants without collision with a combination of the number of lecturers, classes, time slots, and days that order by lecture needs. Compilation of these combinations has often made an obstacle when maps a class into a timetable. These combinations to be a prerequisite in the department for arranging the schedule and answering why the optimization of the preparation of class timetables is needed. Repairing and improvement of the process are to be a consideration for producing the schedule and absolutely time and speed. Another consideration is a limited resource in the department so that a fast decision and arranges schedule are needed to display [1].

Determination of time efficiency in the execution of algorithms has determined by the right data structure in a case [2]. The use of the right data structures can affect the running process [3-6]. Determination of the execution time has influenced by the presence of open and closed looping structures in programming [7]. Another thing in the search for solutions for colony algorithms (ACO) and genetics (GA) will be determined by how much data will be selected [8]. So that comparison of intelligence processes for both algorithms is often identified by time parameters, lots of data, number of loops, and a probability of samples [9].

In previous research [1], the use of genetic algorithms (GA) for arranging of lecture timetables, which can conclude that the results have obtained to determine the best solution arrangement with no collision is 0.7s. Assumptions, the give such as mapping, are matching the activities of lecturers with courses, days, space, and availability of time slots. The same constraint sets the process with no collision and an error <= 0.05, and it obtained by baseline from paper [1]. This result is manipulated by genetic engineering with the structure of table and the steady-state genetic algorithm. The success of efficiency using an algorithm does sustainability research to compare between colony algorithm and algorithm genetic [1].

A problem in course timetabling is a collision and efficiency. The manipulation process is carried out by comparison between the course and another course without collision. These occur because many courses should be adjusted and just handled by human touch. Another problem, the existing system did not available to handle the collision and any prerequisite. For further, this paper has proposed the comparison of two algorithms between GA and ACO. The comparison has constrained by volume, some lecturers, availability of days, and slot. In line with the research that has completed on [1-9], we have carried out a count of performance algorithm with the parameters [6].

Contribution to this paper has proposed some technic in ACO and GA that can compare to the performance and complexity [6]. The objective of this study is to measure performance and the asymptote complexity of ACO and GA in the case. The distance slot measures technic for comparison between a single timetable slot to many slots. We have concluded that GA has a simple process than ACO; even some

research stated ACO is faster than GA. However, we proved for another conclusion, indeed, is an anomaly with their stated, of course, in our constraint and data.

The rest of the paper has organized as follows: proposed problem comparison technique about complexity and performance in section one. The research method and sequencing flow of process have given in Section 2. Related works and some previous research about GA and ACO have summarized in Section 3. Some experiments and results have given in Section 4.

2. Methods

Investigating on this study, comparison with GA and ACO uses the prepared work steps. This work step discusses how the ACO and GA within a comparison result, which investigates in the case of the best solution to the formation of lecture schedules. In Knut, every searching algorithm has began from brute force model, the the solution has taken from how many combination we have. That was a problem if every solution has to break down and then make bulk a time [10].

Performance has used for time adjustment, and complexity is using Big-O [10]. Big-O notation (with a capital letter O, not a zero), also called Landau's symbol, is a symbol used in complexity theory, computer science, and mathematics to describe the asymptotic behavior of functions. It tells us how fast a function growth or declines [10].Each algorithm in the colony case is unique in the process [11-13]. Combinatorial process which taken by [11] has to solve for traveling salesman. On that research, the genetic modelling has engineered by dynamic mutation [11]. The other research for immunology area, GA was succesfull implemented to help immunologist and mathematicians to solve the problem in immunology [12]. GA has used for university scheduling that succesfull implement with various schedule [13]. The background research from [14] was introduced some paramters to solve the scheduling problem. On that reason, we followed the step to accomplish the process of GA and ACO. The work steps of the research include [1]: a) Defined of variables for GA and ACO; b) Defined constant value for maximum and minimum values; c) Processing instance of ACO and GA; d) Comparison between ACO and GA in time execution and Big-O asymptote complexity (Laundau's symbolic) [10]. Figure 1 expresses the flow of research.

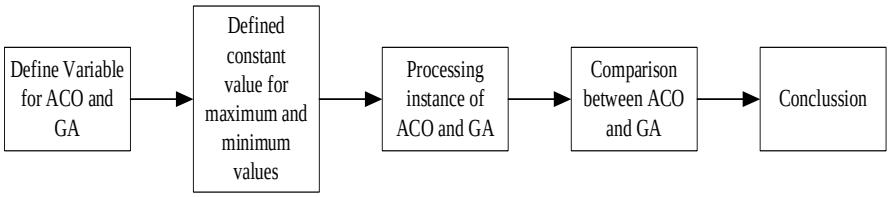

Fig. 1. The flow of research method.

Figure 1 describes the flow of research to get the comparison results. Performance comparison of ACO and GA has to use the same variable. The reason, using the same variable and dimension, is to get significant performance [14]. So, the performance of ACO and GA have more visible for the cases studied. Obtaining the minimum and maximum values have set to be a constant value as a guidance value within execution the ACO and GA [15]. After setting constant value for maximum and minimum, the process continues to execute an instance of ACO and GA.

2.1. Defined constant for maximum and minimum values

Expressing the constant value is stated to set the maximum and minimum value as the limit of the range for each variable [13-15]. The variables used in this study are classroom, days, timeslot, and timetable fit. The variables have presented by the data structure in GA and ACO processes. In this research, testing data uses the number of timetabling as much as 109 arrangements. Limitation for each variable, we set amount for five spaces, six days, 4-time slots. A combination of a variable is presented in Table 1.

Table 1. Constant value Maximum Value for each variable.

Variable Name	Maximum
Slot Timetable	96
Avalaible Classroom	5
Avalaible Days	6
Avalaible Timeslot	4
Fit Timetable	109

In the ACO and GA process, the variable has applied as a count variable. The name of the variable in ACO and GA process is a lecturer (l), day (d), and time (t). We did not take other names in order to make distinct and easy to retrace. We created the algorithm ACO and GA with add some characterized that can be different from the original algorithm. In the process, setting a variable and the other is still the same, but for every cycle in the process. We always make the new random value for every variable and save every single timetable, which has not a collision.

2.2. Processing Iinstance of ACO and GA

This step is to get the performance results of GA and ACO. The implementation of the program has adjusted to the dataset owned. In GA, the time obtained is following many pairwise combinations obtained at the time of execution. Likewise, for ACO, the same thing will be done. We use a time unit in seconds to measure the performance of GA and ACO.

2.3. Complexity and Performance Comparison between GA and ACO

The comparison of both algorithms has obtained through performance and complexity. Execution time is depending on the number of collisions obtained. Process looping will stop when ACO and GA have reached collisions = 0. Because the number of collisions = 0, then the result validation will reach 100% of the arranged schedule without any collisions.

3. Related Works

3.1. GA algorithm

The algorithm has found at the University of Michigan, the United States, by John Holland [11]. In a manner, the genetics process follows the step like a) Generate population, b) Construct new generation with use selection operator, c) Build crossover and mutation operation, d) Evaluate every population with fitness value to fulfilled criteria stop . The process will stop with fitness. Default value of fitness is = 0. The number population for GA is 109 chromosomes. Number generation,

we only created with a random number for every gen such as lecturer, day, and time. When the process has a fitness = 0, then the solution matches. Figure 2 shows the GA process.

The proposed technique has pursued research about timetabling to arrange the schedule [13,14]. The research has used GA process to gain the results with many constraints like limitation between lecture and exercise in the same object are not the same time, priority learning in the laboratory if did not occupy, and for the processor, waiting time, should not exceed more than two hours [13,14]. This prerequisite names as a soft constraint. The experiment has shown that the coarse-grained algorithm fulfilled both hard and soft constraints in 60% of cases, while fulfilment of hard constraints and not all soft constraints has achieved within 95% of cases [13].

3.2. ACO algorithm

Introducing research in ACO (ant colony) has applied on parallel job scheduling for industry. The research assigned to solve the scheduling for type production, multiprocessor, flow of shop, and flexible job shop [16]. It reviews improved ant colony algorithm and its application on several scheduling problems,and the characteristics of the improved strategy. Finally, the development trend and existence problems of production scheduling are discussed ACO algorithm and previous research have taken from [16-18]. In the next sub section, we show the sequential process of ACO that following the proposed model by [16-18]. To abbreviation the parameter, we used many symbols and acronym to abbreviate the process means.

3.2.1. Setting parameter on first step

Parameters setting is constant value which has an important initial before run ACO process, we occupied the parameters such as: (τ_{ij}) = 0.01, timetable order (n=96), d_{ij}, (Q) = 1, (α) = 0.01, $\alpha \geq 0$, (β) = 0.01 , $\beta \geq 0$, Ants (m), (ρ) = 0.03, $0 < \rho < 1$, ($NCMax$ = collision = 0) NC =1 to NC = $NCMax$ [17].

Between visibility timetable and timetable= $1/[d_{ij} (\eta_{ij})]$ (1)

3.2.2. Setting dataset on the second step

On this step, we should to set structure data and called dataset. We constructed a dataset in the table structure. We designed the matrix with columns and rows. Yields, for the first timetable from the first cycle, have to input as the first element in the tabu list. The objective in this step is making the indexing table in order to easy to find the track. The variable tabu1 would be consisted indexing from 1 until n and employ in the first cycle [16].

3.2.3. Tracking on the third step

In the third step, ACO process will include a compilation of a route for visiting timetable to each ant. The ant, which in the table, will set to adequately selected track on 109 travels from origin timetable into the target. In the final travel, tabuk has contained ant colony. If k is indexed visiting and timetable is stated N-tabuk, then visiting probability would be counted as following equation [17].

$$P_{ij}^{k} = \left[[\tau_{ij}]^{\alpha} . [\eta_{ij}]^{\beta}\right] / \left[\sum_{k^{i\epsilon\{N-tabu_k\}}} [\tau_{ik'}]^{\alpha} . [\eta_{ik'}]^{\beta}\right] \quad (2)$$

$P_{ij}^{k} = 0, for\ next\ j$, where *i* has indexed for origin timetable, and *j* has targeted timetable.

3.2.4. Measurement adequatly track on the fourth step

Every selected moving, the ant always measures the length of the route. L_k for every ant has done after one cycles have completed. The equation for the count in one cycle completed as following [16]:

$$L_k = d_{tabu_{k(n),},tabu_{k(1)}} + \sum_{s=1}^{n-1} tabu_{k(s),}, tabu_{k(s+1)} \quad (3)$$

where d_{ij} is the distance for timetable *i* and *j* that has formulated as following:

$$d_{ij} = \left[(x_i\text{-}x_j)^2 + (y_i\text{-}y_j)^2 \right]^{\frac{1}{2}} \quad (4)$$

Equation (4) for searching the minimum route. After L_k has been counting, we have gained $L_{min}NC$ and L_{min} and update the pheromone. Ant footprint that has imprinted will sign as a track. A total of epavoration and difference ants are known as the process for updating the intensity. The equation for evaporation can write as following [18]:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (5)$$

$$\Delta\tau_{ij}^{k} = [\mathrm{Q}]/[L_k] \quad (6)$$

For indexed *i* and *j* as origin timetable and target timetable. $\Delta\tau_{ij}^{k} = 0$, for other *i* and *j*.

3.2.5. The intensity of ant footprints on the fifth step

In the previous step, the ant always updates the evaporation and differences. This event can be a path to update the intensity of ant footprints. The process has called as global update for pheromone and footprint. Following the previous research, the formula can be written at below [18]:

$$\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij} \quad (7)$$

Reordering intensity for ant footprint always orders to the default value of 0.

3.2.6. Looping to the second step in the sixth step

Empty of tabu list is the process for removing value into an empty list, and it is carry on looping at the second step. The tabulist of a short list has occupied a tabulist for the new cycle. Every cycle, ant colony optimization, must empty the tabu list. It can assure if the sum of a cycle does not reach or does not convergence. The algorithm goes loop to the second step of the new footprint intensity. We have defined the equation for collision and bound of iteration as following [16]:

$$collision = \sum timeslot\ (l,d,t) + \sum timeslot\ (c,d,t) \quad (8)$$

3.3. Timetabling

In the timetabling process carried out by [19], constraint parameters included instructors, unique rooms, empty slots, rest periods, and maximum instructors to fill in class. The results obtained in the study [19] are that the approach with the

heuristics search algorithm provides solutions in mapping timetabling in the department. The use of the 2007 ITC dataset, the ITC 2002, and Ben Paechter dataset illustrate timetabling solutions by avoiding collections [19].

In research regarding the arrangement of timetabling, it provides a minimal penalty solution [20]. The use of GA to complete the timetabling focuses on minimal collision conditions. Some studies that provide minimal penalty results are carried out by [20]. The GA approach produces several notes in the study, namely periodic constraint changes, and is useful in class determination [20].

4. Results and Discussion

Based on the reference from GA and ACO [18-20], the results obtained will certainly be different. The first step of ACO in Fig. 2 and GA in Fig. 3 is to make a variable setting. Variables are determined as mentioned, put in Table 1. Each variable will convert into a numeric value.

In Table 2 shows the day that it will convert with a numerical value between 1 to 6. The same is true for space, time slots, and lecturers. The space mentioned is only a maximum of five spaceses, and the numerical value is between 1 to 5. Likewise, for time slots with four time slots means a numerical value between values of 1 to 4.

In Table 3 represents the schedule for lecturers who occupy space, teach in class, subject to subjects, days, and time slots. In Fig.2 and Fig.3, the dataset produced in the initial stage of execution has used in the process of getting a timetable solution [14-16]. The conversion dataset is still the original data for the temporary schedule. So that the results of the interim arrangement still allow collisions to occur.

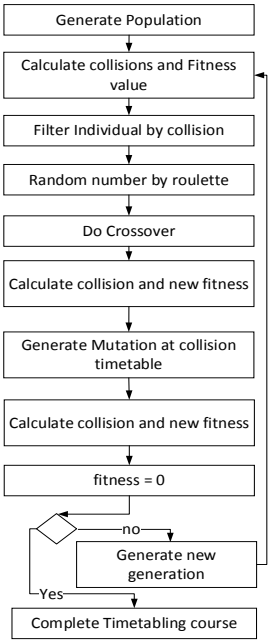

Fig. 2. GA process [1].

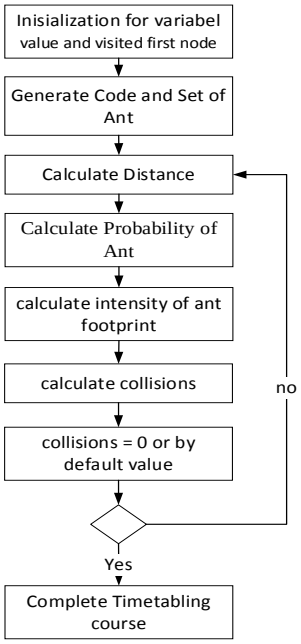

Fig. 3. ACO process [14].

Table 2. An Example conversion variable value into numeric value for days.

Variable of Days	**Value**
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6

In Table 3 shows the result after conversion on every value. We used a number to simplify the name's value. For instance, name of a lecture on the lecture column, we used a number range from one until six. This numbering will be easiness to process on the algorithm.

Table 3. Dataset content of every variable after conversion.

	Lecture	**Course**	**Class**	**Classroom**	**Day**	**Time**
J1	1	1	2	3	2	2
J2	2	1	2	4	1	4
J3	1	4	3	2	2	2
..	...	...	...	...	...	...
..	...	...	...	...	...	...
Jn	6	3	2	1	4	3

4.1. Performance and complexity GA

The next process is to find out how many collisions occur by calculating the fitness value. Tables 4 and 5 are presented the data after conversion and used for processing in GA.

Table 4. Dateset shows of timetable collision in classroom, day, and time.

	Lecture	**Course**	**Class**	**Classroom**	**Day**	**Time**
J1	1	1	1	4	3	1
J2	3	2	3	**2**	**3**	**1**
J3	2	3	2	3	1	2
J4	1	4	4	**2**	**3**	**1**
J5	4	5	5	3	1	2
Jn	…	…	…	…	…	....

Calculated distance for a single timetable to another, we used Eq.4 for Table 5. If we want to calculate distance J_1 to J_2, then we only assign Lecture in J_1 and Lecture in J_2, the course in J_1, and course in J_2, respectively. For example, d_{12}, it means the distance between slot number 1 and slot number 2. d_{12}= $[(1\text{-}3)^2+(3\text{-}3)^2+(1-1)^2]^{\frac{1}{2}} = [(\text{-}2)^2+(0)^2+(0)^2]^{\frac{1}{2}} = 2$. We generate all distances and find where the short distance to take as a sanctioned track. We handle lecture, day, and time column for testing the path.

In Table 6 describes the results of the generation of GA. On the J_4 schedule, there is a fitness value = 2, and it can conclude that the J_4 schedule has collisions related to lecturers and space. If this happens, the GA process will iterate again, until the overall fitness value is = 0.

Table 5. Dataset shows of timetable collision in lecture, day, and time

	Lecture	Course	Class	Classroom	Day	Time
J1	**1**	1	1	4	**3**	**1**
J2	3	2	3	2	3	1
J3	2	3	2	3	1	2
J4	**1**	4	4	2	**3**	**1**
J5	4	5	5	3	1	2
Jn	…	…	…	…	…	…

Table 6. Dataset Collision in GA from Eq. (8).

	Collision (l,d,t)	Collision(c,d,t)	Fitness
J1	1	0	1
J2	0	1	1
J3	0	1	1
J4	1	1	2
J5	0	1	1
Jn	…	…	…

Constituted by the GA that has done, then the performance results of the GA can be written in Table 7. In Table 7 records the results of the execution of GA by using datasets in Table 3. In the column “cycle” is a lot going on to get a fitness <= 2. The average time spent using GA is 0.9856s, and the average loop is 4.9. Table 7 has obtained from the use of a dataset of 109 schedules.

Table 7. GA performance.

Test	Cycle	Time
1	5	1.159
2	7	1.358
3	4	0.797
4	5	0.991
5	4	0.823
6	4	0.821
7	5	0.996
8	6	1.131
9	5	1.002
10	4	0.778
Avg.	**4.9**	**0.9856**

The complexity calculation for GA is to use Big-O [10]. Based on the GA algorithm, it consists of setting variables (i.e., population), determining parents, crossover, mutation, and fitness calculations. The determination of variable settings, namely the GA algorithm, will make the conversion first in the form of a dataset in Table 3. The results of the conversion algorithm engineered for 96 schedules. Table 8 shows a conversion algorithm for the dataset. The total complexity for the conversion algorithm is [10]:

$$O(n) = T(1) + T(1) + \{T(n)(6T(1))\} = 2T(1) + \{T(n)T(6\times1)\} =$$
$$= T(2) + T(6n) = T(2) + T(6n) = T(2) + T(6n) = T(6n+2) \approx T(n)$$

Thus, the complexity of conversion time is $T(n)$, meaning that if $n = 109$, then the time to be generated is $6n + 2$. Conclusion of complexity for all algorithms in GA, can be seen on the Table 9.

Table 8. Complexity conversion algorithm [1] in GA.

Algorithm	Big-O [10]
Index = 1	$T(1)$
NumberRec = 96	$T(1)$
While Index <= NumberRec	$T(n)$
ConversionLecturer(l)	$T(1)$
ConversionClassroom(c)	$T(1)$
ConversionDay(d)	$T(1)$
ConversionTime(t)	$T(1)$
GeneticStream(l,c,d,t)	$T(1)$
Index = Index + 1	$T(1)$
{endWhile}	

In Table 9, the complexity of the GA process is obtained =

$$O(n) = T(6n + 2) + \{T(n) * [T(n) + T(4n^2 - 4n) + T(2n) + T(n) + T(4n^2) + T(4n^2 - 4n) + T(n) + T(4n^2 - 4n)\ ]\}$$
$$= T(6n + 2) + \{T(n) * T(16n^2 - 7n)\}$$
$$= T(16n^3 - 7n^2 + 6n + 2) \approx T(n^3)$$

GA complexity total is = $T(n) + T(n^3) = T(n^3 + n) \approx T(n^3)$

Table 9. GA Complexity in timetable class [1].

Algorithm	Big-O [10]
Setting Variabel (population)	$T(6n + 2)$
While (fitness or collision <>0)	$T(n)$
Counting Collision and Fitness (n)	$T(4n^2 - 4n)$
Filtering Individu	$T(2n)$
Roulette Selection	$T(n)$
Crossover	$T(4n^2)$
Counting Collision and Fitness	$T(4n^2 - 4n)$
Mutation	$T(n)$
Counting Collision and Fitness	$T(4n^2 - 4n)$
{endWhile}	

4.2. Performance and complexity ACO

There is the same algorithm that is for converting values so that the complexity of Big-O will be the same, namely $T(6n + 2)$. The performance results obtained with ACO for each process can see in Table 10. Meanwhile, Table 11 shows about ACO process in our research. We follow state of the art [17] for designing our algorithm. The ACO performance results for timetable cases that have studied can see in Table 10. In Table 10 calculates to achieve the number of clashes = 0, and the required looping average is 6.3 loops. While the average execution time for each test is 6.6251 s.

The results obtained from the calculation of the Big-O notation are as follows:

$$O(n) = T(6n+2) + T(n) + (\ T(n)\ [\ T(n^2+n) + T(n^3) + T(n)]) = T(7n + 2) + (\ T(n^3+n^2) + T(n^4) + T(n^2)) = T(7n + 2) + T(n^4 + n^3 + 2n^2) = T(n^4 + n^3 + 9n^2) \approx T(n^4)$$

Table 10. ACO performance in timetable class.

Test	Cycle	Time
1	6	6.749
2	7	6.055
3	4	5.066
4	6	6.398
5	6	5.556
6	6	10.982
7	8	8.083
8	6	5.216
9	7	6.201
10	7	5.945
Avg.	6.3	6.6251

Table 11. ACO Complexity In timetable class.

Algorithm	Big-O [10]
Setting Variabel	$T(6n + 2)$
Setting Dataset, Eq. (1)	$T(n)$
While (collision <>0)	$T(n)$
Tracking, Eq. (2)	$T(n^2 + n)$
Properly Track, Eq. (3), Eq. (4), Eq. (5), Eq. (6)	$T(n^3)$
Intensity, Eq. (7)	$T(n)$
{endWhile}	

4.3. Comparison GA and ACO

The experiment for comparing both algorithms, we used software generator like Java, computer PC with I3 processor, and 4 Gigabyte memory [15]. In Figs. 4 and 5, it can be seen that the results of GA performance calculations have a faster time than ACO. The overall comparison can be seen in Figs. 4 and 5.

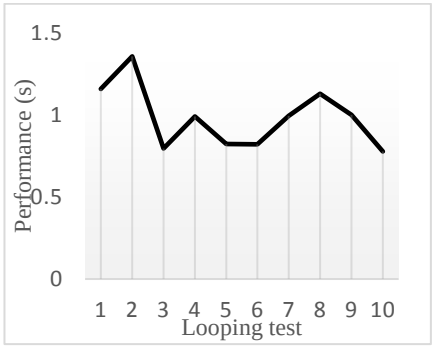

Fig. 4. Running GA every looping test in Java until having performance.

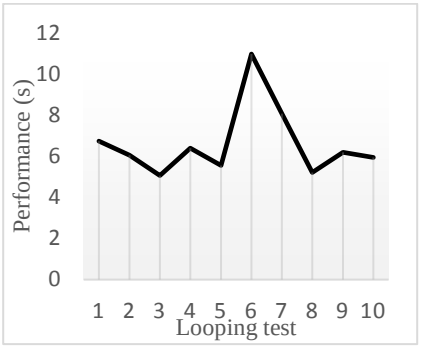

Fig. 5. Running ACO every looping test in Java until having performance.

In Figs. 4 and 5, the graph shows the difference in time. Looping tests that occur also show a big difference. In Fig. 4, the performance of the time ranges from 0.5 s to 1.5 s. While in Fig. 5, ACO performance ranges from 4 s to 12 s. It can claim that the ACO process has a higher complexity compared to GA. So that the scheduling case made will be more efficient when using GA. The overall conclusion of the comparison can be seen in Table 12.

In the fact that ACO is faster than GA for every experiment. Our research is a different result when executing the dataset [16-18]. We have found the anomaly. We have another argument to discuss this happens. Our argument that every process in ACO and GA depends on the dataset and solution algorithm. In the process of finding a solution, we always make new random when a selection of parents in GA.

Table 12. Summarization comparison ACO and GA.

Simulation Test	Asymtotic Complexity	Performance	Avg.Loop
GA	T(n^3)	0.9856s	4.9
ACO	T(n^4)	6.6251s	6.3

Meanwhile, in ACO process is the same; when the condition of collision does not achieve, the process will iterate until collision = 0 [14]. Another reason for this happens ACO should found on the short path in many locations. We defined every single timeslot to be one ant that should find on many selections path. It needs a longer time than GA that only selects the parent [18]. In 109 timeslots, we have to find a minimum of 3 paths to compare with each other. A value of three is a limitation for an ant to continue the track. So, if one ant has three selections, another location, we need 328 cycles for looping for each ant.

5. Conclusions

Investigations conducted to test the performance and complexity of GA and ACO, obtained significant differences. At GA performance, the average time obtained by ten tests is 0.9856s and the average number of cycles is 4.9 cycles. In ACO performance, the average time obtained by ten tests is 6.6251s and the average number of cycles is 6.3. While asymptotic complexity for GA is n^3, and ACO is n^4. The difference in results obtained is due to several things as mentioned below.

- In GA, looping occurs at all stages, starting from population determination, crossover, mutation, and fitness. The thing that uses the most time is when there are three processes for determining fitness and crossover - the asymptotic complexity evidence this event for fitness at $T(12n^2 - 12n)$ or n^2.
- At ACO, all ACO stages have looped. So that the time gained is getting bigger. The event is evident in the investigation using asymtotic complexity, which gets the time of each stage is n^2 and n^3.

In future research, various kinds for selectable parameter which can occupy. We can change the parameters, to see effect of the process into result. Dynamic time and day can apply to execute the model. We can approximation the time occupancy for the process if the lecture pursue for the time and day. To simplify process and reduction the time, we can encompass the ACO only one track in selection. The impact, process in ACO can reduce the time almost n^2.

Acknowledgment

The authors would like to thank the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia for providing a research grant in 2018 to fund this research.

Nomenclatures

Collision	Number timetable which has the same plotting
c	Variable of classroom code conversion
d	Variable of day code conversion
dij	Distance between timeslot and another timeslot
i, j	Index track
Jn	Plotting timetable at n position
k	Index Visiting
Lk	The measure of length closed tour
LminNC	Minimum closed length route
Lmin	Whole closed length route
l	Variable of lecture code conversion
m	Ants (timetable fix slot)
NCMax	Constraint Maximum Cycle or collision = 0
NC	Indexed number of cycles
N-tabuk	Timetable at index visiting
P_{ij}^{k}	Visiting Probability at track i,j
n	Number of timetable order
Q	The cycle of ant constant
Timeslot(l,d,t)	Sequencing value at timetable
t	Variable of time code conversion
tabuk	Temporary data structure for timetable processing (tabu list)
(*x, y*)	Seqeuence of setting variable GA and ACO

Greek Symbols

α	Track intensity constant
β	Visibility constant
τ_{ij}	Path intensity and change at track *i, j*
η_{ij}	Probability Visible intensity at track *i, j*
ρ	Evaporation constant
$\tau_{ij}(t)$	Pheromone concentration of the timetable *i* and *j* at time t
$\Delta\tau_{ij}$	Update of pheromone in the timetable

Abbreviations

ACO	Ant Colony Optimization
Big-O	Laudau's symbolic or Big-O search
GA	Genetic Algorithm

References

1. Mauluddin, S.; Ikbal, I.; and Nursikuwagus, A. (2018). Optimasi aplikasi penjadwalan kuliah menggunakan algoritma genetik. *Jurnal Resti*, 2(3), 792-799.
2. Pandey, K.K.; and Pradhan, N. (2014). A comparison and selection on basic type of searching algorithm in data structure. *International Journal Computation Science Mobile Computing*, 3(7), 751-758.

3. Petruseva, S. (2006). Comparison of the efficiency of two algorithms which solve the shortest path problem with an emotional agent. *Yougoslav Journal of Operations Research*, 16(2), 211-226.
4. Jia, F.; and Lichti, D. (2017). A comparison of simulated annealing, genetic algorithm and particle swarm optimization in optimal first-order design of indoor TLS networks. *Proceedings of ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences Conference of Geospatial*, Wuhan, China, 18-22.
5. Truong, V. (2012). A comparison of particle swarm optimization and differential evolution. *International Journal on Soft Computing (IJSC)*, 3(3), 13-30.
6. Cazacu, R. (2016). Comparison between the performance of GA and PSO in structural optimization problems. *America Journal of Engineering Research*, 5(11), 268-272.
7. Kundu, S.; and Atha, R. (2018). Pid tuning by particle swarm optimization technique and comparison with classical methods. *International Research Journal of Engineering Technology*, 5(7), 687-692.
8. Iwan, M.; Akmeliawati, R.; Faisal, T.; and Al-Assadi, H.M. (2012). Performance comparison of differential evolution and particle swarm optimization in constrained optimization. *Proceedings of International Symposium On Robotics And Intelligent Sensors*, Kuala Lumpur, Malaysia, 1323-1328.
9. Sharma, I.; Kuldeep, B.; Kumar, A.; and Singh, V.K. (2016). Performance of swarm based optimization techniques for designing digital fir filter: A comparative study. *Engineering Science and Technology, an International Journal*, 19(3), 1564-1572.
10. Knuth. (1997). *The art of computer programming.* Canada: Addison Wesley Longman.
11. Xu, J.; Pei, L.; and Zhu, R.-z. (2018). Application of a genetic algorithm with random crossover and dynamic mutation on the travelling salesman problem. *Procedia Computer Science*, 131, 937-945.
12. Mccall, J. (2005). Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, 184 (1), 205-222.
13. Berisha, A., Bytyci, E.; and Tershnjaku, A. (2017). Parallel genetic algorithms for university scheduling problem. *International Journal of Electrical and Computer Engineering* (*IJECE*), 7(2), 1096-1102.
14. Sidik, R; Fitriawati, M.; Mauluddin, S.; and Nursikuwagus, A. (2018). A schedule optimization of ant colony optimization to arrange scheduling process at certainty variables. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 9(12), 318-323.
15. Chernigovskiy, A.S.; Kapulin, D.V.; Noskova, E.E.; Yamskikh, T.N.; and Tsarev, R.Y. (2017). Production scheduling with ant colony optimization. *Proceedings Of Innovations And Prospects Of Development Of Mining Machinery And Electrical Engineering. International Conference*, Saint Petersburg, Russia, 1-6.
16. You-Xin, M.; Jie, Z.; Zhuo, C.; and Idea, A.B. (2009). An overview of ant colony optimization algorithm and its application on production

schedulingproduction timetable. *Proceedings of International Conference on Innovation Management*, Wuhan, China, 135-139.

17. Subekti, R.; Sari, E.R.; and Kusumawati, R. (2018). *Ant colony algorithm for clustering in portfolio optimization. Journal of Physics : Conference Series*, 983(1), 1-6.
18. Zwaan, S.V.D.; Pais, A.V.; and Marques, C. (2018). Ant colony optimisation for job shop scheduling. Retrieved September 20, 2018, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.3586.
19. Babaei, H.; Karimpour, J.; and Hadidi, A. (2015). Survey of approaches for university course timetabling problem. *Computers & Industrial Engineering* 86, 43-59.
20. Feng, X.; Lee, Y.; and Moon, I. (2016). An integer program and a hybrid genetic algorithm for the university timetabling problem. *Optimization Methods and Software*, 32(3), 625-649.