

Reinforced Island Model Genetic Algorithm to Solve University Course Timetabling

Alfian Akbar Gozali, Shigeru Fujimura*

Graduate School of IPS, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka 808-0135, Japan
*Corresponding author, e-mail: alfian@tass.telkomuniversity.ac.id

Abstract

The University Course Timetabling Problem (UCTP) is a scheduling problem of assigning teaching event in certain time and room by considering the constraints of university stakeholders such as students, lecturers, departments, etc. This problem becomes complicated for universities which have immense number of students and lecturers. Therefore, a scalable and reliable timetabling solver is needed. However, current solvers and generic solution failed to meet several specific UCTP. Moreover, some universities implement student sectioning problem with individual student specific constraints. This research introduces the Reinforced Asynchronous Island Model Genetic Algorithm (RIMGA) to optimize the resource usage of the computer. RIMGA will configure the slave that has completed its process to helping other machines that have yet to complete theirs. This research shows that RIMGA not only improves time performance in the computational execution process, it also offers greater opportunity to escape the local optimum trap than previous model.

Keywords: university course timetabling problem, island model, genetic algorithm

Copyright © 2018 Universitas Ahmad Dahlan. All rights reserved.

1. Introduction

The University Course Timetabling Problem (UCTP) is a scheduling problem of assigning teaching event in certain time and room by considering the constraints of university stakeholders such as students, lecturers, departments, etc. The constraints could be hard (encouraged to be fulfilled) or soft (better to be fulfilled) constraints. Timetabling itself is considered an NP-Hard problem [4]. Some universities such as Telkom University [5] have a large population of students and lecturers, and so their constraints are also great as a result. This condition could make the problem even more complicated.

In addition, the student body in Telkom University has increased from 6,570 students in 2010 to 21,698 in 2016. The number is a result of the merging of four universities: Telkom Institute of Technology, Telkom Polytechnic, Telkom Institute of Management and Telkom School of Arts. The university timetabling solver must, as a result, meet a new specification: scalability.

One of the most recent researches is the application of genetic algorithms (GAs), which is inspired by the theory of evolution. This method has been used to solve many actual UCTP cases. There are several GA models such as informed GA [10], parallel GA [2], NSGA II [11, 9], Adaptive Real Coded GA [13], Hybrid Fuzzy and GA [6], Quantum Evolutionary Computing [1], and distributed model GA [14] that have been proposed. This research used the distributed model GA, or what is known usually as Island Model GA [14], out of all these models. We have chosen this model for its high scalability.

For the university course timetabling itself, Gozali et al. introduced Asynchronous Island Model GA (AIMGA) [5]. This model succeeded in solving actual UCTP cases in the Telkom University School of Engineering with a satisfying result. However, when it was run under various computer specifications, faster computers were idle after having completed their tasks while the slower ones were still running. This idling problem left an opportunity to be exploited for more efficient performance.

Therefore, we are going to introduce the Reinforced AIMGA (RIMGA) to improve time performance in the computational execution process. At the same time, we also offer greater opportunity to escape the local optimum trap than the conventional AIMGA. Taken together, the contributions of this work are (1) introducing RIMGA as a brand new mechanism to complement common AIMGA, (2) designing the Telkom University UCTP, and (3) analyzing RIMGA performance in handling Telkom UCTP.

This paper consists of seven sections. The remainder of this paper is organized as follows. Section 2 talks about the mechanism of proposed method, RIMGA and its parameters, handles AIMGA's idling problem. Section 3 introduces the research method which is split into two subsections: designing Telkom UCTP and its RIMGA implementation. Section 4 shows how we conducted the experiment, results, and its discussion. The last but not the least, section 5 takes place as the conclusion of this work.

2. The Proposed Method

2.1. Reinforced State

The AIMGA could solve the synchronous model waiting problem, but in reality, it is found that there is still an opportunity to increase the AIMGA efficiency. There is almost no problem if the specification of the computers is not too different. If, however, they are actually under a very different specification, there will be slaves that complete their tasks faster than other slaves. Such a condition will make the faster slaves idle while the slower ones are still running their tasks. The RIMGA was introduced in this paper to increase the AIMGA efficiency.

The main idea of the RIMGA is how the idle island (computer) can be utilized further to help another running island. The idle island as an island that has reached its stop condition has to find another island which is still running. The idle island will reinforce that island to complete its process more quickly. Figure 1 illustrates the difference between the asynchronous and the RIMGA.

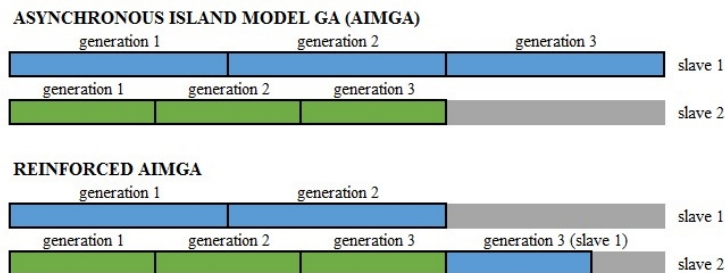


Figure 1. The difference of two island GA Models

The AIMGA isolates each computer to run separately so that each computer can complete its task without waiting for other computers to complete theirs. The reinforced model tries to utilize the idle time (gray cell) of slave 2 from the asynchronous one. After slave 2 has completed its task, it would help slave 1 to finish its task (generation 3).

As shown in the Master Island state diagram in Figure 2-master state, the RIMGA implements the reinforced function. It tries to find an island that has not completed its task to be helped by the island that has. This attempt aims to maximize the productivity of the model by optimizing the utility of the idle island that has completed its task.

Figure 2-slave state shows that the RIMGA starts when there is an island that has completed all its process. The master will evaluate and choose which of the islands that has not completed

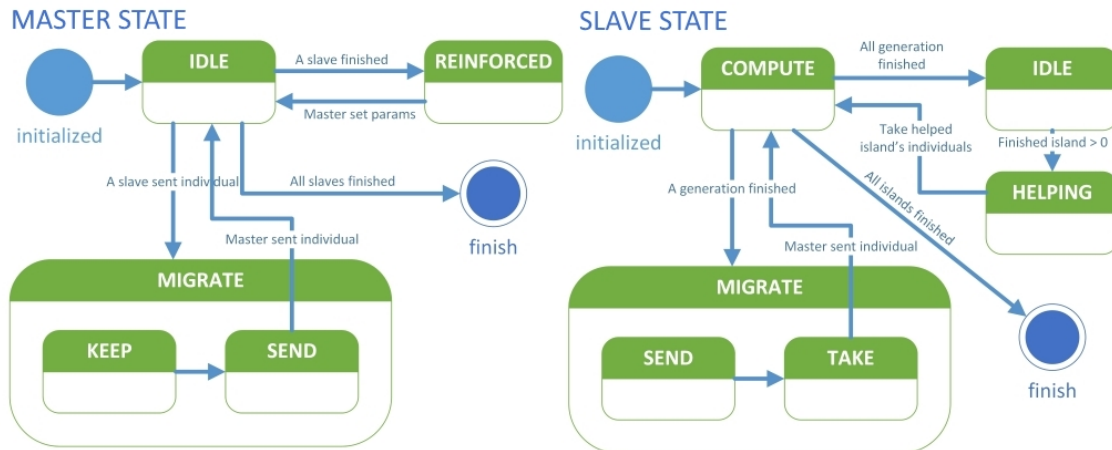


Figure 2. Master-Slave Island State diagram

its task to be helped by the island that has. There are several considerations in picking the island to be helped and how to generate its population. Such considerations are presented as *reinforced parameters*.

2.2. Reinforced Parameters

Reinforced parameters are parameters that control how the reinforced state behaves. The master island will use the reinforced parameters to control how the islands that have completed their tasks to help those that have not. There are three reinforced parameters: the island state, island priority, and individual picking method. The reinforced parameters are defined as follows.

The set of (slave) islands is expressed by S . The set of islands that have completed their tasks is represented by F , those that have not by U , and those that are helped by H . Let $F \subset S$, $U \subset S$, and $H \subset S$ where $F = \bar{U}$. The reinforced parameters are expressed as $\{p|p = \{true, false\}\}$ such that p_1 is the island state, p_2 the island priority, and p_3 the individual picking method.

The first parameter is the **island state**. The island state is a condition to determine whether an island that is being helped by another island or not. This parameter will determine whether the *reinforcement direction* is divergent (balance for all islands that have not completed their tasks) or convergent (islands that have, one by one). An island that is helped by another island expressed as H_i . $R(a, b)$ is a function that determines whether island a must help island b or not. $R(a, b)$ will return true if p_1 is true and island b has not been helped by another island yet. If $R(a, b) = true$, island a helps island b and the state of island b will be changed to being helped. $help(a, b)$ is a procedure in which island a helps island b .

$$R(f:\forall F, u:\forall U) = \begin{cases} true, & \text{if } (p_1 = true) \cap (u \notin H) \\ false, & \text{otherwise} \end{cases} \quad (1)$$

$$\text{if } (R(f, u) = true) \Rightarrow help(f, u), u \leftarrow h$$

The second parameter is **island priority**. It has two choices: those could be based on the least number of iterations or on the poorest (largest) fitness. These choices refer to *the factors are the most important*, the number of iterations or fitness value. If the least number of iterations were activated ($p_2 = true$), the island that has completed its task will find and help another island which has the least number of iterations. Otherwise, if the poorest fitness were activated ($p_2 = false$), the island that has completed its task will find and help another island that has the lowest fitness value. Let iteration(s) return the current iteration of island s and fitness return the best current

fitness of island s .

$$s = \begin{cases} \text{argmin}(\text{iteration}(u : \forall U)), & \text{if } (p_1 = \text{true}) \\ \text{argmin}(\text{fitness}(u : \forall U)), & \text{otherwise} \end{cases} \quad (2)$$

The third parameter is **individual picking method**. This parameter determines the way of picking individuals from an island that going to be helped (reinforced). There are two ways of picking individuals. The first ($p_3 = \text{true}$) is by picking the best individual from helped island and duplicating it into as many as the individual numbers of a population. The second ($p_3 = \text{false}$) is by picking the best population from helped island or usually the last population from it. The second way has the consequence of the best individual bucket in the master having to be changed into the best population bucket. In other words, the master island must keep the best population from every island rather than the best individual.

3. Research Method

3.1. UCTP in Telkom University

The UCTP in Telkom University is a student-level timetabling (student sectioning) problem. As referenced from Tomas Muller et al., Student sectioning is the problem of assigning students to classes of a subject while respecting individual student requests along with additional constraints. For example, a student cannot attend two classes which overlap in time[7]. Therefore, the fulfillment of each of the students' preference is encouraged as well. This approach has been implemented in previous researches [10, 5] in Telkom University and other universities such as Purdue [8] and Waterloo University [3].

Like any other UCTP, Telkom University is a minimum optimization problem. The objective is to minimize all the predefined constraint violation for each of the teaching events. Such that a teaching event is an event of a lecturer l in a room r at time t class c for a set of students S . Which is defined by following notation:

$$e = (l, r, t, c, S) \quad (3)$$

With reference from previous researchers [10, 5, 2], this research used two types of constraints, the hard and the soft constraints. Hard constraint (HC) is a constraint that must be satisfied. Soft constraint (SC) ought more to be satisfied to improve the quality of the timetabling. As the constraints are working in the same UCTP cases, the types of HCs and SCs used for this research are exactly the same as the previous research conducted by Suyanto [10] and Gozali [5].

Let $i = 1..5$ be hard constraints and $i = 6..12$ be soft constraints. As hard constraints must be far bigger rather than soft constraints, the objective function becomes:

$$\text{Minimize } V = \sum_{i=1}^5 MV_i + \sum_{i=6}^{12} V_i \quad (4)$$

where V is a violation value for each i constraint. The symbol M means a very big number so that MV_i is much larger than V_i .

With regard to the Telkom University UCTP in this research, the HC values are set much higher than SC so that the GA will prioritize poor fitness more because of HC. By treating the SCs this, they became the focus after all HCs have been satisfied. The penalty value of SCs is designed to be proportional to its influence. For example, as a lecturing event has a lecturer and around 50 students, the ratio of lecturer SC to student SC should be 1:50.

3.2. RIMGA for UCTP

Directed chromosome will be used for the chromosome representation. Directed chromosome mimics the real-world representation which, in this case, is the university timetabling representation.

Thus, the chromosome, as shown in Figure 3, is the representation (mapping) from the real-world timetable.

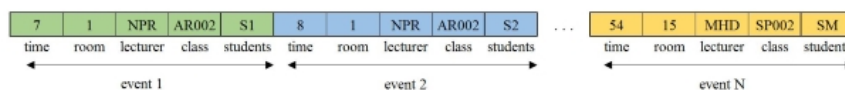


Figure 3. Directed chromosome

Furthermore, since the AIMGA model was derived from the single model, the AIMGA core (individual reproduction and evaluation of fitness) was the same as the single model. The AIMGA implemented Informed GA core, which applied local search and only used directed mutation over crossover [10].

Derived from previous researchers [5, 10], this research also uses two stages of the informed GA, class-level timetabling (stage 1) and student-level timetabling (stage 2). The significant difference between these two stages lies in stage 1 which excludes student constraints. That is because the student evaluation time process took much longer time due to a large number of students. The next stage will include student constraints. Directed mutation which is an additional way to improve the process efficiency of the GA was also used.

4. Result and Discussion

This research experiment has three goals: to implement the AIMGA concept into the Telkom University UCTP, analyze RIMGA parameters, and compare the RIMGA with the ordinary AIMGA. We conduct the first three experiments with the regard of implementing and analyzing RIMGA parameters, while the last experiment is for comparing with conventional AIMGA. Dataset used in this research was Telkom University Engineering School, a faculty in Telkom University, at odd semester for enrollment years 2011/2012. This faculty had characteristics as explained in Table 1.

Table 1. Informed GA Scheme

No	Attributes	Value
1	Classes	813
2	Rooms	80
3	Students	6570
4	Average number of classes per students	6.481
5	Average number of meetings per class	2.752
6	Lecturers	316
7	Average classes per lecturers	2.582

4.1. Result and Discussion

This section explains the test scenario which aims to test three reinforced state parameters: the island state, island priority and individual picking method. Each test will be run thrice with 5 islands [5] in sequence. This number is considered sufficient because there are already statistically significant differences by using the 3 test results. The specifications of the computer used for this test are two computers with Core i5 and 4GB memory, one computer with Core2Duo (2.8 GHz)

and 2 GB memory, and two computers with Core2Duo (2.6 GHz) and 1 GB memory but from different manufacturers. We use fitness and execution time as the evaluation parameters.

4.1.1. Island State Test

Table 2 shows the results for the island state test. From this table, ignoring the island state is better than considering the island state. By ignoring the island state, more than one faster slave can help the slowest slave so that the slowest slave can reach its stopping criteria sooner. The result is in line with the computer specifications of the islands that are very different. Therefore, balancing the process by considering the island state is not as effective as ignoring it. Thus, the next test scenario used the ignoring of the island state configuration.

Table 2. Island State Test Result

Test Number	Ignoring island state		Considering island state	
	Best Fitness	Duration (hh:mm:ss)	Best Fitness	Duration (hh:mm:ss)
1	11520	16:55:59	11900	18:48:36
2	11950	16:52:33	11480	18:53:38
3	11840	16:30:32	11940	18:48:41
Average	11770	16:46:21	11773	18:50:18

4.1.2. Island Priority Test

Table 3 shows the test result for this scenario. It indicates that for the duration of execution and best fitness, and the least number of iterations gave better result rather than the poorest fitness. The duration interval between them is just around 9 minutes. The slower slave with the least number of iterations is requires more help so far. This condition shows that the GA performance depends more on the current generation than fitness. Thus, keeping the slower island with fewer generations would be better than poor fitness.

The result, which places the current generation above fitness, means that population in each island is able to keep their diversity. It is easier for the island to avoid the local optimum trap due to its diversity. For this case, therefore, adding generations is better than increasing fitness. Finally, according to the result, the next test will use the least number of iterations.

Table 3. Island Priority Test Result

Test Number	The least iteration numbers		The worst fitness	
	Best Fitness	Duration (hh:mm:ss)	Best Fitness	Duration (hh:mm:ss)
1	11520	16:55:59	13310	17:20:29
2	11950	16:52:33	12410	16:51:01
3	11840	16:30:32	12130	16:34:14
Average	11770	16:46:21	12616	16:55:15

4.1.3. Individual Picking Method Test

The test result for individual picking method test is shown in Table 4. It is shown in this table that the best population copy is superior to the best individual duplication in fitness value. By consuming about 21 more minutes in execution time, the best population copy produced a fitness value 11,650. This was better than the 12,430 from the best individual duplication.

Table 4. Individual Picking Method Test Result

Test Number	The best population copy		The best individual duplication	
	Best Fitness	Duration (hh:mm:ss)	Best Fitness	Duration (hh:mm:ss)
1	11520	16:55:59	12750	16:32:50
2	11950	16:52:33	12440	16:15:05
3	11840	16:30:32	12100	16:28:54
Average	11770	16:46:21	12430	16:25:36

Picking the best population is better rather than copying the best individual into a population mean for this case, the generation trend still tends to trap in a local optimum. Thus, picking a whole best population and continuing the process with it will help the island that did not complete its task to increase its diversity. This parameter, which is similar to the *Island Priority Test* explanation, is also experimental. The Temporal-Salient Scheme [12] could be implemented to direct the solutions this problem. However, like the previous example, the research uses island order instead of the Temporal-Salient Scheme for the sake of time efficiency.

4.1.4. Comparison Test of RIMGA

The last experiment is to compare the performance of the AIMGA and the RIMGA. The goal of this process is to analyze how far the RIMGA could beat the AIMGA in performance. The test was run in two phase: maximum fitness and time constraint. Maximum fitness constraint test was done to compare their time performance to reach same fitness and vice versa. Done in the same way as the previous test, this test was done in three consecutive times to obtain the average.

Table 5-Duration shows the test result with same maximum fitness. The terminate condition of this trial is fitness=10000. Table 5 indicates that AIMGA execution time was almost twice of the RIMGA.

Table 5. Maximum Fitness Limitation Test Result

Model	Duration (hh:mm:ss)	Fitness
RIMGA	16:55:59	9980
AIMGA	24:00:00	9980

Table 5-Fitness shows the result of the 24-hour test. When both ran for 24 hours, there was no significant difference between the RIMGA and the AIMGA in general. We can therefore conclude that, overall, the implementation of the Reinforced function improves the AIMGA performance in obtaining a good result. In addition, the result will be the same if they still have the same GA core.

5. Conclusions

This research has shown that the AIMGA can solve the Telkom University UCTP with acceptable accuracy represented by the GA fitness value. Furthermore, by implementing the RIMGA, the timetabling result accuracy increases in performance to next level. This new approach can obtain the same result as the AIMGA in a faster (about twice) execution time with a somewhat similar effect when they run under the same time constraint. Thus, the reinforced state is an excellent choice if we want to obtain good results more quickly.

The optimum configuration parameters of the RIMGA for Telkom UCTP are Island State: ignored, Island Priority: the least iteration numbers, and Individual Picking: the best population copy. Although this study focuses on the Telkom University UCTP, the findings may well have a bearing on other UCTPs with similar characteristics as the Telkom University UCTP.

Taken together, this research confirms that the RIMGA can solve the UCTP with scalability issues. This study also contributes additional evidences that encourage the use of the reinforced function in the AIMGA. The results of the experiment could serve as the basis for future researchers in setting its parameters. In addition, further studies still need to be conducted for the RIMGA for better real-world implementation. Further studies on its network cost is also necessary to investigate its real computational time and cost. The RIMGA also still needs to be implemented in a simpler problem to study the correct considerations in the parameter adjustment.

Acknowledgments

Indonesia Endowment Fund for Education (LPDP), a scholarship from Ministry of Finance, Republic of Indonesia, supports this work. We conduct this research while at Graduate School of Information, Production, and Systems, Waseda University, Japan.

References

- [1] N. K. Aryani, A. Soeprijanto, I. M. Y. Negara, and M. Syai'in. Economic dispatch using quantum evolutionary algorithm in electrical power system involving distributed generators. *International Journal of Electrical and Computer Engineering*, 7(5):2365–2373, Oct. 2017.
- [2] K. Banczyk, T. Boinski, and H. Krawczyk. Parallelisation of genetic algorithms for solving university timetabling problems. In *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, pages 325–330, Sept 2006.
- [3] M. W. Carter. A comprehensive course timetabling and student scheduling system at the university of waterloo. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers*, pages 64–82, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [4] M. Garey and D. S. Johnson. *Computer and Intractability*. W.H. Freeman and Company, New York, 1979.
- [5] A. A. Gozali, J. Tirtawangsa, and T. A. Basuki. Asynchronous Island Model Genetic Algorithm for University Course Timetabling. In *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling*, pages 179–187, York, 2014. PATAT.
- [6] Q. Kotimah, W. F. Mahmudy, and V. N. Wijayaningrum. Optimization of fuzzy tsukamoto membership function using genetic algorithm to determine the river water. *International Journal of Electrical and Computer Engineering*, 7(5):2838–2846, Oct. 2017.
- [7] T. Muller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010.
- [8] K. Murray, T. Muller, and H. Rudova. Modeling and Solution of a Complex University Course Timetabling Problem. In E. K. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, number 3867 in Lecture Notes in Computer Science, pages 189–209. Springer Berlin Heidelberg, Aug. 2006. DOI: 10.1007/978-3-540-77345-0_13.
- [9] A. W. Pratama, A. Buono, R. Hidayat, and H. Harsa. Estimating parameter of nonlinear bias correction method using nsga-ii in daily precipitation data. *TELKOMNIKA*, 16(1):241–249,

- Feb. 2018.
- [10] Suyanto. An Informed Genetic Algorithm for University Course and Student Timetabling Problems. *Artificial Intelligence Soft Computing Lecture Notes of Computer Science, Springer Berlin Heidelberg*, 6114:229–236, 2010.
- [11] F. Titel and K. Belarbi. A mixed binary-real nsga ii algorithm ensuring both accuracy and interpretability of a neuro-fuzzy controller. *International Journal of Electrical and Computer Engineering*, 7(5):2614–2626, Oct. 2017.
- [12] S. Tsutsui, Y. Fujimoto, and A. Ghosh. Forking genetic algorithms: Gas with search space division schemes. *Evolutionary Computation*, 5(1):61–80, March 1997.
- [13] Umar, Firdaus, A. Soeprijanto, and O. Penangsang. Optimal expenditure and benefit cost based location, size and type of dgs in microgrids systems using adaptive real coded genetic algorithm. *TELKOMNIKA*, 16(1):10–17, Feb. 2016.
- [14] D. Whitley, S. Rana, and R. B. Heckendorn. Island Model Genetic Algorithms and Linearly Separable Problems. *Lecture Notes in Computer Science, Springer Berlin Heidelberg*, 1305:109–125, 1997.