Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A COMPARATIVE STUDY OF EVOLUTIONARY ALGORITHMS FOR SCHOOL SCHEDULING PROBLEM

[1]DANIEL NUGRAHA, [2]RAYMOND KOSALA
[1]School of Computer Science, Bina Nusantara University, Jakarta, Indonesia
[2]School of Computer Science, Bina Nusantara University, Jakarta, Indonesia
E-mail: [1]niel_syd@yahoo.com, [2]rkosala@binus.edu

ABSTRACT

The scheduling problem is one of the combinatorial optimization problems that are common in the real world. Evolutionary algorithms, such as Genetic Algorithm and the Bees Algorithm have been used in the literature to solve this problem. In this paper, we apply the Genetic Algorithm, Bees Algorithm, and the combination of these two algorithms, which is Hybrid GA-Bees, to solve school scheduling problem. The algorithms to solve the school scheduling problems were implemented using the object-oriented paradigm and tested to create actual class schedules at a middle school in Jakarta. Our study concludes that the Hybrid GA-Bees can produce better solutions compared to the solutions generated by the Genetic Algorithm and Bees algorithm.

Keywords: *Genetic Algorithm, Bees algorithm, Hybrid GA-Bees algorithm, school scheduling*

1. INTRODUCTION

The scheduling problem is one of the combinatorial optimization problems that are common in the real world, and the scheduling problem in education domain is not new. In general, the scheduling problem can be classified into several types, such as college-level academic scheduling, scheduling middle and elementary schools, scheduling exams, transportation scheduling, scheduling flow of sales or delivery of goods, and others.

The scheduling problem can be categorized in the class of NP-hard problem in which the problems are difficult to solve using exact algorithms [1]. In this class of problems, the computation time required searching for the optimal solution increases exponentially depending on the size of the problem [2, 3]. In the research literature there has been many solutions proposed to solve similar NP-hard problem with various metaheuristic and evolutionary approaches. From those studies we can conclude that the evolutionary methods have been succeeded in solving these problems [4, 5, 6].

In a previous study [7], a new algorithm which is called Hybrid GA-Bees was proposed. The authors in [7] have tested the Hybrid GA-Bees algorithm to solve a university's course timetabling problem, and concluded that it produced in a better schedule optimization when compared with other methods, such as Tabu Search, Bees Algorithm, and Variable Neighborhood search (VNS). In this paper, we focus on comparing some evolutionary algorithms, including the Hybrid GA-Bees algorithm, to solve the real world *class scheduling* problem at a middle school institution.

In this paper, a Hybrid GA-Bees algorithm will be compared with the Genetic algorithms and Bees algorithm alone with a customized fitness computation for scheduling problems at one middle school in Jakarta. The rest of this paper is divided into the following sections: Section 2 describes the scheduling problem in detail, Section 3 describes the algorithms that will be compared, Section 4 shows the experimental results and compares the performance of evolutionary algorithms, and Section 5 concludes the paper.

2. DESCRIPTION OF THE SCHEDULING PROBLEM

There are two main classes of course scheduling or timetabling problems: the *Post Enrolment*-based course scheduling problem and the *Curriculum*-based course scheduling problem [8]. In the literature, the post enrolment-based course scheduling problem is also often called as *class scheduling* problem or *event scheduling* problem.

Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This paper attempts to solve the class scheduling or the post enrolment problem at a middle school in Jakarta. In the manual scheduling, the schools often have to deals with the overlapping schedules that may cause the obstruction of teaching and learning process, make the students come late to the next lesson, and in the end waste the time allocated for learning. The manual course scheduling also has the has to deal with following complications: constraints on the teachers, constraints on classroom space, constraints on the time for study, and constraints due to the regulation about the number of work hours and rest periods of the teachers. In the following, we propose to solve this problem by using evolutionary algorithms.

2.1 Constraints

A class scheduling problem can be described as a set of soft constraints and hard constraints. Hard constraints are the constraints that should be fulfilled, while soft constraints are the constraints that will not disrupt the teaching and learning process if not met, but the schedule would be better and preferable if it could be met. The set of hard and soft constraints are described as follows.

Hard Constraints

H1. Teacher: One educator cannot teach in two places or classes on the same time slot on the same day.

H2. Overlap: On the same day, there should be no courses taught more than once.

H3. Maximum working day: The educators cannot teach more than four time slots in a single day.

Soft Constraint

S1. Break: The educators should be given time off before the next classes.

2.2 Fitness Function

The formulation of the fitness function is adapted from [9] and can be written as follows:

$$f(X) = \sum_{i=1}^{n} w_i d_i \qquad (1)$$

The function $f(X)$ is a fitness or objective function that specifies the quality of a scheduling, where:

- X is a scheduling (timetable) which is checked against a set of hard and soft constraints mentioned above,
- w_i are weights, where the hard constraint has a weight of 100 and soft constraint has a weight of 1,
- d_i is violation number of the i^{th} constraint (how many times a constraint is violated), and
- n is the total number of constraints.

The lower the value of the fitness function in Formula 1 will yield a better solution. Thus this fitness function will be minimized during the search of the timetabling solution, and the best solution will have a fitness value of 0 (zero).

3. THE ALGORITHMS

3.1 Genetic Algorithms

Genetic algorithm, which was proposed by John Holland in 1975, is a branch of evolutionary computation (EC) and is based on the principle of natural selection [9].

The genetic algorithm [10] that is shown in Figure 1 begins by generating an initial population in the form of a collection of individuals who were randomized. Each individual or commonly called chromosome has a fitness value that measures how good the individual is as a solution. Each individual is represented as the arrangement of genes. Finally, each gene contains a value or a particular trait called allele.

1) *Choose the initialize population of individuals*
2) *Evaluate the fitness of each individual in that population*
3) *Repeat*
4) *Selection: Select pair of parent individuals form a population*
5) *Crossover: With a crossover probability crossover the parents to generate a new individual (Children). If no crossover was performed, the individual is an exact copy of parents*
6) *Mutation: With a mutation probability mutate new individual*
7) *Evaluate the individual fitness of new individuals*
8) *Replace least-fit population with new individuals*
9) *Until terminations condition met*

Figure 1. The Genetic algorithm

In every loop or one generation, each individual undergo selection, crossover, and mutation are then

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the fitness values are re-evaluated. The selection operator chooses the parents randomly using roulette wheel selection method, in which individuals with better fitness values will have higher probabilities of being selected. The selected parents that are chosen probabilistically then crossed over. In the end, with a small probability a mutation is carried out to the individual chromosome.

3.2 Bees Algorithm (Bees)

Bees Algorithm is a metaheuristic search technique based on population. The algorithm is inspired by the foraging behavior of the honey bee in combination with a randomized local search around in the neighborhood of the solution to get the best fitness value [1]. The parameters in this algorithm are: the number of scout or search bees (*n*), the number of sites that will be used for the neighborhood search (*m*), the number of elite bees (*e*), the number of bees that will be employed for patches visited by the elite bees (*nep*), number of bees that are employed for other (*m-e*) selected patches (*nsp*), and the size of patches (*ngh*) [11].

The Bees algorithm that we use [12] is shown in Figure 2. The algorithm starts with setting the number *n* bees that are conducting random searches of honey into the population, and then back for the fitness calculation. The bees that produce the best fitness values later become elite bees (*e*), then the elite bees choose their members (*nep*) to carry out the neighborhood search in *m* sites.

1) *Initialize population with random*
2) *Evaluate fitness of the population*
3) *While (stopping criterion not met)*
   // Forming new population
4) *Select sites for Neighborhood Search*
5) *Recruit bees for selected sites (more bees for the best e sites) and evaluate fitness.*
6) *Select the finest bees from each patch*
7) *Assign remaining bees to search randomly and evaluate their fitness*
8) *End While.*

Figure 2. The Bees Algorithm (Bees)

The neighbor solution is the best result from the new fitness calculation, and this will replace the old result. Next, the rest of the bees other than the elite bees will be sent to search randomly into the population and then back for the fitness calculation. This condition is repeated until the fitness value equal to 0.

In the Neighborhood Search algorithm [12] shown in Figure 3, the bees choose the neighbor randomly, put the result in set X', then swap or move at random. The best results will be used for further processing.

Create a Neighborhood Solution from *X:*
1. *Randomly select neighbor*
2. *Set X' = X*
3. *Randomly choose a kind of move in two kind of moves (Single move, Swap move)*
4. *Take a best solution X' in the neighborhood of X created by the chosen kind of move with a fraction of available set of block element of X*
5. *Return X'*

Figure 3. The Neighborhood Search Algorithm

3.3 GA-Bees Algorithm

The GA-Bees hybrid algorithm shown in Figure 4 is an algorithm that combined the genetic algorithm and Bees algorithm [7].

The first step the combined GA-Bees algorithm shown in Figure 4 is to initialize the population randomly. Then the fitness of each individual in the population is calculated. In one generation (one loop), the Hybrid GA-Bees algorithm does two processes in sequence. The loop starts with the Bees algorithm from steps 4 to 7 that perform local search as well as to improve the fitness value as many as *e* (the number of elite bees) individuals. Then the improved individuals are returned to the population. Finally the genetic algorithm processes (Selection, Crossover, and Mutation) are started from steps 8 to 12.

4. EXPERIMENTS

The algorithms above were implemented using the Java SDK 1.7 and their performances were tested using the data from Tunas Muda School in Jakarta during the academic year 2011/2012.

The data consists of a total of 12 classes from grade 7 to grade 12 of middle (junior high) school. Each grade level has parallel classes. The number of classrooms used for teaching and learning operations was 25 rooms and special facilities including laboratories. There were 32 teachers that teach no more than 16 sessions per week. The duration of one class session was 70 minutes.

Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The experiments were executed using a computer with the following specification: Intel Core i7 - 2630QM processor running at 2.00GHz with a 32-bit Windows 7 Operating System.

1) *Initialize population with random*
2) *Evaluate fitness of the population*
3) *While (stopping criterion not met)*
   // Bees algorithm
4) *Select sites for neighborhood search*
5) *Recruit bees for selected sites (more bees for the best e sites) and evaluate fitness.*
6) *Select the finest bees from each patch*
7) *Assign remaining bees to search randomly and evaluate their fitness.*
   // Genetic algorithm
8) *Randomly select pair of parent individuals from a population*
9) *Crossover: With a crossover probability crossover the parents to generate a new individual (children). If no crossover was perform, the individual is an exact copy of parents*
10) *Mutation: With a mutation probability mutate new individual at each position in chromosome*
11) *Evaluate the individual fitness of new individuals*
12) *Selection: Replace least-fit population with new individuals*
13) *End while.*

Figure 4. The GA-Bees (hybrid) Algorithm

Table 1 shows the representation of each class as a matrix timetable with size x times y, where x is the number of sessions per day and y is the number of working days in a week. So there were 25 genes used for one allele and they were initialized randomly at the start. A chromosome or individual was composed of 12 alleles that represent the amount of class. So the string that represents an individual consists of 300 genes. Figure 5 shows the string representation of the individual in the population.

Table 1 . Scheduling Class 7C

	1 07:50-09:00	2 09:20-10:30	3 10:40-11:50	4 12:30-13:40	5 13:50-15:00
Monday	Ms. Astri **SCIENCE**	Mr.Timothy **MUSIC** Music Room	Mr.Caroline **ENGLISH** 24	Mr.David **MATHEMATICS**	Mr.Chistopher **HUMANITIES** 44
Tuesday	Mr.David **MATHEMATICS**	Ms. Astri **SCIENCE**	Ms.Amalia **BAHASA** 31	Mr.Philip **TECHNOLOGY** DT	Ms. Astri **SCIENCE**
Wednesday	E/A/H/I **RELIGION** 21/22/23A/23B	Mr.David **MATHEMATICS**	Mr.Caroline **ENGLISH** 24	Mr.Chistopher **HUMANITIES** 44	Ms.Amalia **BAHASA** 31
Thursday	Mr.Chistopher **HUMANITIES** 44	Mr.Caroline **ENGLISH** 24	Mr.Timothy **MUSIC** Music Room	Mr.David **MATHEMATICS**	Mr.Nigel **ASEMBLY** Hall
Friday	Ms.Amalia **BAHASA** 31	Mr.Maria **CARE**	Ms. Astri **SCIENCE**	Mr.Philip **TECHNOLOGY** DT	Mr.Ibnu **PE** Field

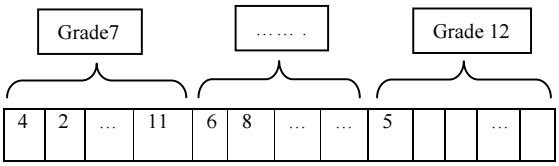

Figure 5. String representation of a chromosome

We have conducted the experiments to test the performance of the three algorithms. They were tested with the same chromosome model to generate good timetables, while complying with the hard and soft constraints that were mentioned earlier in this paper.

The Genetic Algorithm (GA) was tested for up to 100 generations. Early initialization was done using random data entered in each class to form the individual or chromosome. The GA operators that were used are the following:

a. Selection: the selection process was done using roulette dish to choose parents randomly. Chromosomes that have better fitness values had greater opportunities to be selected.

b. Crossover: the crossover method that was used is the probabilistic *n*-point crossover, in which crossover point derived from the number of classes that are available, and the selection probabilities were randomly selected from among the parents. Figure 6, 7 and 8 show two examples of how the crossover operator works.

c. Mutation: the mutation process that was used is a swap mutation of 2 random points. Figure 9 shows an example of how the mutation operator works.

In Figure 6, the resulting Child chromosomes do not always follow the pattern of their parents or follow a certain pattern because they were randomly selected from the chromosomes of the parents. Figure 7 and 8 show two examples of crossover selection models of the children.

Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Parent1:

1	2	3	4	5	6	7	8

Parent2:

11	12	13	14	15	16	17	18

Child1:

1	12	3	14	5	16	7	18

Child2:

11	12	13	14	15	16	7	8

Figure 6. An example of the crossover operation

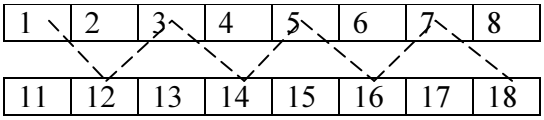

Figure 7. Crossover selection model of Child1

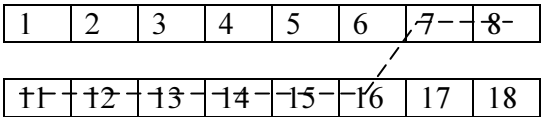

Figure 8. Crossover selection model of Child2

Before:

1	2	3	4	5	6	7	8

After:

1	**8**	3	4	5	6	7	**2**

Figure 9. Swap Mutation model

If an individual meets the criteria mutation probability, then every allele of its chromosome will be swap randomly. So the represented class in the individual may be swapped around.

In order to do the experiments, Figure 10 shows the implementation of a user interface (UI) for the timetabling solution. Figure 11 shows the UI to generate the initial population, and Figure 12 shows the UI of the resulting timetable.

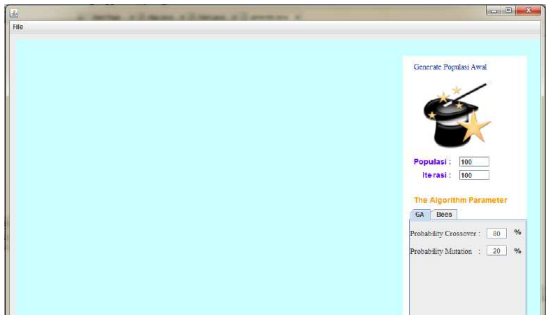

Figure 10. The User Interface of the Solution

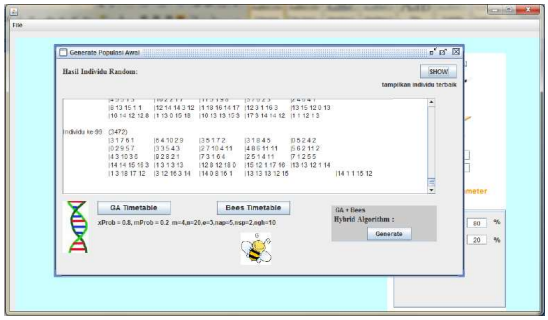

Figure 11. The User Interface to Generate the Initial Population

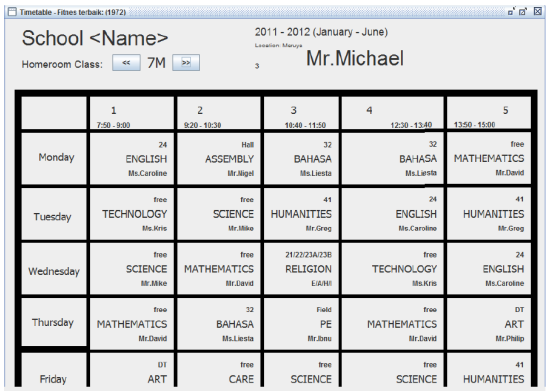

Figure 12. The User Interface of the Resulting Timetable

In this experiment, we used a hold out method to find the best combination of the parameters. The parameters for the Genetic algorithms are selected from the following values based on the previous research done in [7]: crossover threshold probability {50%, 80%, 90%} and mutation probability of {1%, 2%, 3%}. The tests above were performed on all possible combinations of the specified values, and each experiment was done ten times, and then

Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

we use the average values. The best combination of parameters for the Genetic algorithm was crossover threshold probability of 80% and mutation probability of 2%. Total number of generations was limited to 100 iterations, and the value 0 is the best fitness value.

The test result of the Genetic algorithm (GA) in Figure 13 shows the fitness values over generations that form a learning curve. We can see that the learning curve of the maximum fitness values has a high variance. The possible reason for this is because all the operations in the GA, which are the crossover and mutation, are done randomly. The green curve, which is the average fitness value of each generation, tends to improve with lower average fitness value at each generation. This means in general, we can see that the GA produces better population from generation to generation. However, the GA could not produce a good enough timetabling solution after 100 generations because no individual with a fitness value of zero was generated.

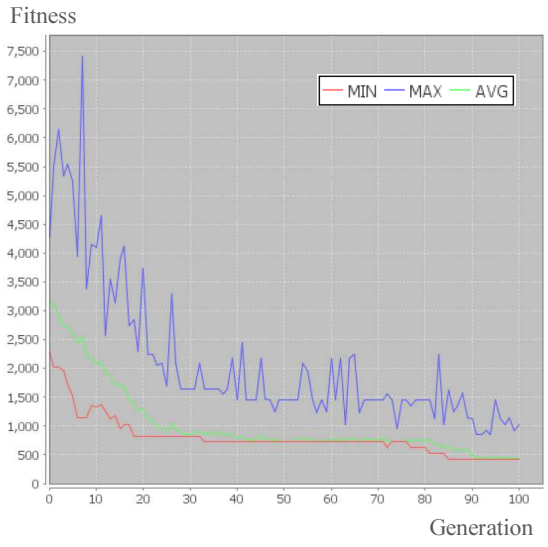

Figure 13. The Genetic Algorithm Test Results

For the Bees algorithm, the best combination of parameters are selected from the following values: the number of scout bees (n) = {20, 30} , the number of elite bees (e) = {3, 4} , the number of selected sites (m) = {4, 5} , the number elite patches (nep) = {4, 5}, the number of other bee patches (nsp) = {2, 3}, the size of the patch (ngh) = {5, 10}. The tests above were performed on all possible combinations of the specified values, and each experiment was done ten times, and then we use the average values. The best combination of parameters for the Bees algorithm was: n = 30, e = 4, m = 5, nap = 4, nsp = 2, ngh = 10. Total number of generations was limited to 100 iterations, and the value 0 is the best fitness value.

As can be seen from the test results of the Bees algorithm in Figure 14, the maximum fitness curve fell rapidly. That is because of the neighborhood search (NS) done by a number of bees that were not employed by the elite bees (m-e), and this affected the shape of the maximum fitness curve. All elite bees ensure that the overall fitness value improves, and if the (m-e) bees obtained good results, then the fitness curve will drop significantly. The fitness curve may go up if the results of the (m-e) bees were not good. However, the fitness curve would not go up significantly because we use only a small number of (m-e) bees. The red curve is a minimal fitness curve that was always constant or goes down. This is because the elite bees maintain the same or better fitness value. To conclude, the Bees algorithm has better results on our scheduling problem than the genetic algorithm over 100 generations in our experiments.

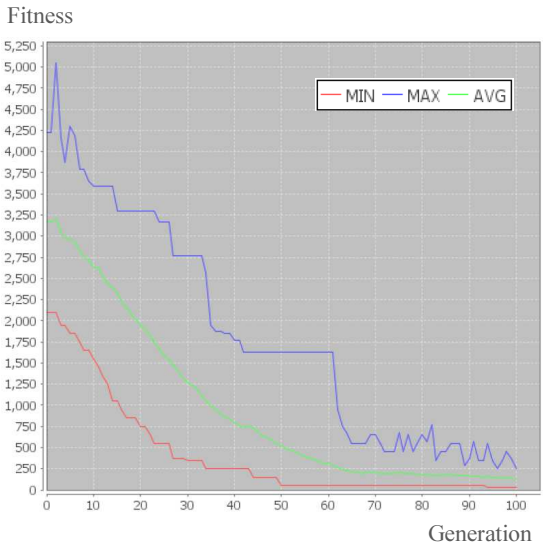

Figure 14. The Bees Algorithm Test Results

For the GA-Bees algorithm, the best combination of parameters are: n (the number of scout bees) = 30, e (the number of elite bees) = 4, m (the number of selected sites) = 5, nep (the number of bees recruited for patches visited by bees elite e) = 5, nsp (the number of bees recruited for the other (m-e) selected patches) = 2, and ngh (the size of the

Journal of Theoretical and Applied Information Technology
30th September 2014. Vol. 67 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

patches) = 10. From the resulting best parameter of the combination of Genetic algorithms and Bees algorithms, averaging over ten runs, we obtained some fitness curves as shown in Figure 15.

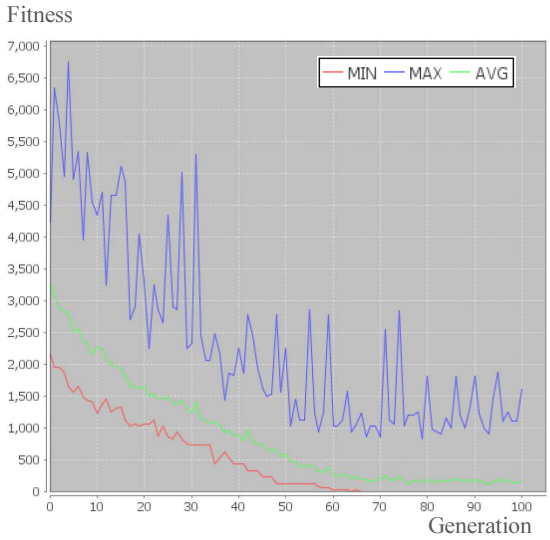

Figure 15. The GA – Bees Test Results

We can see from the experiment results of the GA-Bees algorithm in Figure 15 that GA-Bees algorithm gave better results compared to the results of the previous two algorithms. The average minimal fitness curve touches the point 0, which means that it was able to produce a timetabling that meets all the given constraints before 100 generations, that is at the 64th generation. We can also see that the maximum fitness curve has a higher variance than the maximum fitness curve of the genetic algorithm. The possible explanation for this higher variance is because the randomness of the neighborhood search in the Bees algorithm is intensified by the randomness of the crossover and mutation operators in the Genetic algorithm.

We also calculated and compared the gradients of the learning curves to compare the three algorithms. The gradients of the learning curves were calculated using the following formula:

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \qquad (2)$$

Table 2 shows the results of the calculation of the average gradient of the learning curve using formula (2) by averaging the results starting with i = 1 (the 1st generation) up to i = 100 (the 100th generation). In this experiment we also tested each algorithm as much as ten times with the same variable and the same initial random population. Table 2 shows negative numbers because on average the gradients were decreasing over generations. The larger the negative value, the greater the slope of the learning curve and the algorithm also gives better results faster. On average, the GA-Bees algorithm had the largest gradient.

Table 2. The Gradients of the learning curves

Experiment Number	GA	Bees	GA-Bees
1	-17	-23	-22
2	-19	-21	-23
3	-15	-15	-21
4	-17	-20	-25
5	-13	-29	-41
6	-14	-22	-22
7	-15	-20	-21
8	-18	-21	-19
9	-14	-22	-23
10	-13	-19	-27
Average	-15.5	-21.3	-24.4

The average gradient obtained by the Bees algorithm was around 37.42% better than the average gradient obtained by the Genetic algorithms. The average gradient obtained by the combined or hybrid GA-Bees algorithm was around 14.55% and 57.42% better than those of the Bees algorithm and the Genetic algorithms respectively. Thus, we can conclude that the GA-Bees algorithm was the best algorithm for solving our timetabling problem.

There is still an open research issue, whether the hybrid GA-Bees algorithm is better than the other evolutionary algorithms for other scheduling problems.

The limitation of this study is that we assume on each day there is the same number of sessions for different class levels. For schools that require different session numbers, our methods above can still be used by giving additional information to the algorithm about the lengths of the alleles in the genes.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. CONCLUSION

In this paper, we try to find good solutions to the class scheduling problem, which is one of the most common combinatorial optimization problems in the real world. Specifically, we compared an algorithm combined of the Genetic and Bees, which is called GA-Bees, with the Genetic algorithms and the Bees algorithm alone with a customized fitness computation for scheduling problems at a middle school in Jakarta. Based on our experiments, the hybrid GA-Bees algorithm produced better and optimal solutions when applied to the problem of academic scheduling of a middle school in Jakarta.

Some future research in this area includes applying the hybrid GA-Bees algorithm for scheduling exams, bus scheduling, or scheduling the purchase of goods.

REFERENCES:

[1] D.S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co. Boston, MA, USA, 1997.

[2] A. Schaerf. A Survey of Automated Timetabling. *Artificial Intelligence Review*, April 1999, Volume 13, Issue 2, pp. 87-127.

[3] S. Abdullah and H. Turabieh, "Generating University Course Timetable Using Genetic Algorithms and Local Search". *Proceedings of the Third International Conference on Convergence and Hybrid Information Technology, 2008. ICCIT '08,* pp.254-260.

[4] Y. Zhou, J. Zhang, and Y. Wang. Performance Analysis of the (1+1) Evolutionary Algorithm for the Multiprocessor Scheduling Problem. *Algorithmica*, June 2014, pp. 1-21.

[5] M. Ragera, C. Gahmb, F. Denz. Energy-oriented scheduling based on Evolutionary Algorithms. *Computers & Operations Research*, Available online 17 May 2014.

[6] J.A. Soria-Alcaraz, M. Carpio, and H. Puga. "A new approach of Design for the Academic Timetabling through Genetic Algorithm". *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010*, pp.96-101.

[7] N.B. Phuc, T.T.M.K. Nguyen, and T.H.N. Tran, "A New Hybrid GA-Bees Algorithm for a Real-world University Timetabling Problem". *Proceedings of the International Conference on Intelligent Computation and Bio-Medical Instrumentation (ICBMI), 2011*, pp.321-326.

[8] B. McCollum, P. McMullan, B. Paechter, R. Lewis, A. Schaerf, L. Di Gaspero, A. J. Parkes, R. Qu, and E. Burke. "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition". *Technical Report, International Timetabling Competition* 07-08, 2007.

[9] M. Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.

[10] G. Zhang and W. Liu. "A genetic Algorithm Base on a New Real Coding Approach". *Proceedings of the second International Conference on Intelligent System Design and Engineering Application (ISDEA)*, 2012, pp.88-92.

[11] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. "The Bees Algorithm – A Novel Tool for Complex Optimisation Problems", *Proceedings of IPROMS 2006 Conference*, pp.454-461.

[12] N.T.T.M. Khang, N.B. Phuc, and T.T.H. Noung, "The Bees Algorithm for a Practical University Timetabling Problem in Vietnam". *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, 2011, pp. 42-47.