WALAILAK JOURNAL
http://wjst.wu.ac.th Information Technology

A Hybrid Immune Genetic Algorithm to Solve University Time Table Problems*

Pongsarun BOONYOPAKORN[1,*] and Phayung MEESAD[2]

[1]*Department of Information Technology, Faculty of Information Technology, King Mongkut's University of Technology North Bangkok, Bangkok 10800, Thailand*
[2]*Department of Information Technology Management, Faculty of Information Technology, King Mongkut's University of Technology North Bangkok, Bangkok 10800, Thailand*

([*]Corresponding author's e-mail: pongsarun.b@it.kmutnb.ac.th, phayung.m@it.kmutnb.ac.th)

Received: 26 September 2016, Revised: 5 April 2017, Accepted: 30 May 2017

Abstract

Timetabling problems are complex optimization tasks which can be difficult to solve, especially when trying to achieve the optimal results. Moreover, they are also known to be NP-hard (Non-deterministic Polynomial-time hard) problems. Usually, problems are defined as assigning a set of lectures to a fixed number of timeslots and rooms such that they satisfy a number of hard and soft constraints. Genetic algorithm (GA) is an approach to optimize learning based on the principles of biological evolution. In the past, GA have been applied to solve instances of timetabling which are combined with various techniques and optimization operators to optimal realistic requirements. In this paper, a new genetic hybrid algorithm which includes an artificial immune system called the Immune Genetic Algorithm (IGA) is proposed. This method mainly aims to construct a feasible timetabling which satisfies all constraints as best as possible. The main idea of this paper is to reduce the number of initial chromosomes to a local search space. In addition, the author has also compared the results with Genetic Algorithm (GA), Immune Algorithm (IA), and Practical Swarm Optimization (PSO) to illustrate that the IGA produces good quality solutions and outperforms similar methods.

Keywords: Timetabling problems, scheduling problems, artificial immune, genetic algorithm, hybrid immune

Introduction

Timetabling problems are known to have NP-hardness (Non-deterministic Polynomial-time hard) which defines a class as hard or soft, to optimize problem solutions of the combinatorial nature. The basic problem involves the allocation and distribution of resources to different tasks subjected to different constraints. Hard constraints are physically constrained and cannot be violated while scheduling is being computed. On the other hand, soft constraints are ones which can be considered as desirable to be addressed in the solution as much as possible but not necessarily essential. Various scheduling problems exist which are complex with high computation time. Educational time table scheduling is one of the most difficult to solve because of many constraints which need to be satisfied in order to get a feasible solution. Usually, resources are comprised of a set of teachers, groups of students, available time slots, and number of rooms. Several approaches have been conducted in this domain with various methodologies to solve timetabling problems such as Operations Research, Artificial Intelligence, and Computational

[*]Presented at 1st International Conference on Information Technology: October 27th - 28th, 2016

Intelligence. Meta-heuristics optimization is one of the algorithms from the Artificial Intelligence area which is the most popular technique designed to solve this problem. This can be categorized as a single-solution and population-based approach. Single-solution approaches can be found in tabu search, simulated annealing, graph coloring-based, and variable neighborhood search. Population-based approaches included genetic algorithm, ant colony, particle swarm optimization, harmony search-based, and scatter search.

Based on a recent survey, the genetic algorithm over recent years has become more widespread as it is used in the solution of complex and the non-linear problems, it based on the natural selection principle through the simulation nature biological evolution process. Each candidate solution to a problem known as chromosome is an ordered array of values for attributes called genes. Each gene is regarded as atomic follows; the set of alleles for that gene is the set of values that the gene can possibly take. The performance of a set of candidate solutions are first evaluated and ordered, then new candidate solutions are produced by selecting candidate as parents and applying mutation which has a slight alteration on the features of an existing parent or crossover to recombine the features of a pair of parent solution operators. Javidi *et al*. [1] analyzed parameters such as population size, crossover probability, and mutation probability based on benchmark functions and traveling salesman problem (TSP). Their experiments shown that the uniform models are terrible for the parameters of a genetic algorithm.

In the past, Genetic Algorithms have been applied to solve instances of timetabling. Salwani and Hamza [2], they proposed the combination of a Genetic Algorithm and sequential local search to solve course timetabling problems. Wutthipong *et al*. [3] compared the performance of GA's crossover operators including partially matched crossovers, order crossovers, and cycle crossovers. Their experimental results demonstrated that order crossovers work more effectively than the others when producing feasible timetables, cycle crossovers are a great operator to find the optimal feasible timetable. Thi *et al*. [4], the formulation of the problem based on realistic requirements with multiple objectives of Genetic Algorithm to solve timetabling in Faculty of Science and Technology, Hanoi University were shown. A novel approach using Genetic Algorithms for timetabling was proposed in [5,6], they presented techniques to solve hard and soft constraints with operators to obtain an optimal, simple scalable, and parameterized results for multiple courses.

Various hybrid techniques to improve the performance of a solution have been proposed. Sheung *et al*. [7], the Genetic Algorithm and simulated annealing were presented to solve a prototypical timetabling problem. Chu and Fang [8] and Vinayak *et al*. [9], investigated and compared the Genetic Algorithm and Tabu Search approaches to solve these kinds of tabling problems. A new hybrid GA-Bees Algorithm was presented by [10], they obtained results which confirmed this approach was able to produce better high quality solutions. Meysam *et al.* [11], a fuzzy genetic algorithm with a local search was presented to solve timetabling. This was achieved by using fuzzy logic to measure the violation of soft constraints in the fitness function to deal with inherent uncertainly and vagueness involved in real life data. Om *et al.* [12] they proposed a bacterium and Genetic Algorithm combination to represent a point in the n-dimensional search space where each point was a potential solution to the time table.

As mentioned above, timetabling problems have arisen in many fields which could be related to several techniques. Currently, the most popular technique employed is the Genetic Algorithm which is considered as a pure technique but could be modified into a hybrid to overcome the above issues and become more effective and efficient. In the next paragraph, a review of the Genetic Algorithm to solve the problem and a closer look at the Artificial Immune System (AIS) which will be used to improve the population while initializing the chromosome data set is presented.

The AIS is based on the natural immune system and is used as a metaphor to solve computational problems [13]. Several application domains of AIS are detection, recognition, network security, data mining, optimization, and time scheduling. The AIS has been modified to work in a variety of different techniques such as Artificial Immune Network, Clonal Selection Algorithm, and Negative Selection Algorithm. Juan *et al*. [14] they presented an Artificial Immune Network which computed university course timetabling. Moreover, they also showed the benchmark of an AIS implementation technique.

The objective of this paper is to describe the modified GA which is a combination of an AIS to form an Immune Genetic Algorithm (IGA) to reduce the search space and achieve efficient searches.

Performances of the IGA and 2 other techniques will be compared. This paper is divided as follows: Section 2 describes materials and methods of the evolutionary algorithm that can solve timetabling problems such as encoding, objective function, operators, and initial population. Section 3 presents the experimental results and discussion. Finally, conclusion and pinpoints of future directions for further work are in section 4.

Materials and methods

In this section, a description of the proposed evolutionary algorithm for the IGA to solve timetabling problems will be presented. Firstly, a statement of the problem will be described.

Problem description

The timetables to be treated in this paper are constructed from courses of the Faculty of Information Technology, King Mongkut's University of Technology North Bangkok. Students are grouped according to their major and arrival year. Assuming that there are 3 timeslots a day, each time slot is 3 h and there is 7 work days per week. Each subject is lectured in one successive time slot. The problem is defined using 6 main entities: Room, Class, Group, Professor, Module, and Timeslot. The author assigned one professor to one timeslot and module in such a way that there is no conflict between the room, class, group, professor, module, and timeslot. The lecture timetable formula is as follows: Problem = $\{T, R, P\}$, where $T = \{t_1, t_2, t_3, \ldots, t_n\}$ is a set of timeslots per week, $R = \{r_1, r_2, r_3, \ldots, r_n\}$ is a set of rooms and $P = \{p_1, p_2, p_3, \ldots, p_n\}$ is a set of Professors. The considerations consist of hard and soft constraints. In this paper, hard constraints deal with the following:

Each professor can take only one class at a time.

1. A classroom can have only one subject assigned to it at a time.
2. Clashes should not occur between the modules for students of one group.
3. Each module must be assigned to a professor that can teach that module.
4. Each module must be assigned to a timeslot that the professor can attended.

There are also soft constraints which are penalized equally by their occurrences:

1. Each professor should be assigned to their preferred courses.
2. There should be sufficient numbers of seats in each room for all students present.

The goal of this technique is to minimize the violation of a feasible solution.

The IGA is proposed based on the theory of immunity in biology which mainly constructs an immune operator accomplished by 2 steps: a vaccination and an immune selection. The aim of a leading immune concept and methods taken from the GA is to theoretically utilize the local characteristic information to seek ways and means of finding the optimal solution when dealing with difficult problems [15]. In this paper, the algorithm will be developed by completing all of the steps and processes in sub sections. It first generates a random detector, and then the initial population. Next performs selection, crossover, and mutation upon the population for a number of generations, until termination criterion is met.

Immune genetic algorithm

In the IGA, each possible solution $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ is considered as a cell. The steps in the IGA are as follows:

1. Initialize detector D, each of which fails to be a random value.
2. Calculate fitness of each cell in the population and rank them.
3. Chose the best candidate to be detector D.
4. Initialize a population P of gene, each set will have a random value.
5. Perform negative selection in any P which matches D.
6. Calculate fitness of each chromosome in P and rank them.
7. Perform crossover and mutation.
8. Return to step **6** if termination condition is not met, else **stop**.

Chromosome representation

Each individual chromosome is, in itself, a time table which means every chromosome would carry an amount of data regarding class, timeslot, room, and professor. The structure of the chromosome would be a 7×3 matrix, since it must represent a time table for 7 days, with 3 allotted time slots per day. Under each class group, there will be information about whether the class slot is active or vacant. If the class is active, the information regarding the timeslot id, room id, and professor id will be clearly shown. **Figure 1** shows encoding for the problem of which timeslot is scheduled, professor teaching and the classroom. By splitting the chromosome into chucks of 3, the information can be retrieved for each class.

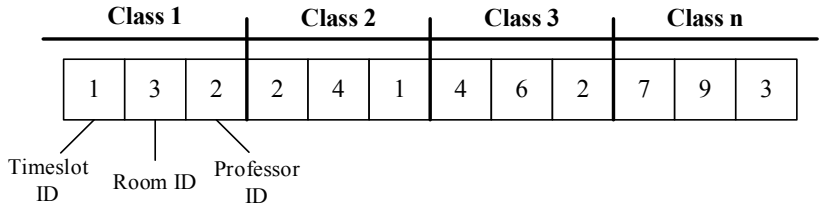

Figure 1 Chromosome representation.

Negative selection algorithm

Artificial negative selection is a computational imitation of self/nonself discrimination, using a form of detector set as the detection mechanism [16]. Matching rules used to generate detector in this paper were r-contiguous bits called r contiguous which match symbols in corresponding positions, this is the main characteristic of a learning algorithm. The value of r can be used to balance between more generalization and specification. **Figures 2** and **3** shows an outline of a typical negative selection algorithm.

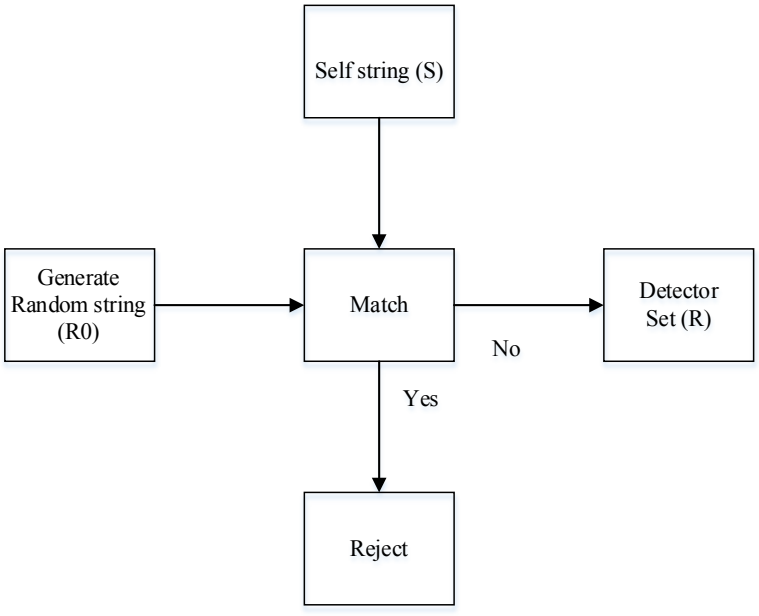

Figure 2 Generation of detector set.

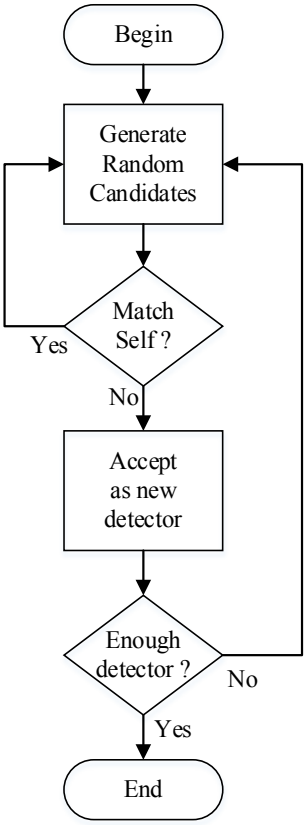

Figure 3 Flow of generation of detector set.

For example, *r*-contiguous matching symbols corresponds to self-strings of length l = 30 and matches length r = 8. This can be summarized by the following;

Self-string 010101001001110010001111110100
Random string 111010101101110010100010011110

Initial population

In general, the initial population is constructed as random and each individual considered as having no hard constraint violation. In this paper, the author selected a population from the immune method which had a negative selection algorithm as discussed in the previous section to reduce search space. This approach considerably saves time when searching for the best solution.

Selection

In this paper tournament selection of size 2 is used, where 2 parents are selected from the population and fitter one is used as a parent. At each generation, the higher an individual's fitness is, the greater the chance that the individual will be chosen for crossover. Tournament selection selects its parents by running a series of "tournaments". First, individuals are selected from the highest population fitness values and entered into a tournament. Next, these individuals can be thought to compete with each other by comparing their fitness values, then the individual with the highest fitness will be chosen as for the parent.

Two-point crossover

Two-point crossover is an alternative crossover method to the uniform which is a very simple crossover method in which a single position in the genome is chosen at random to define which genes come from which parent.

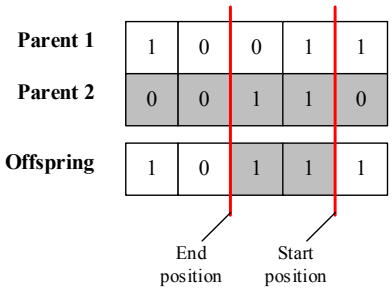

Figure 4 Two point crossover.

Mutation

In this paper, a chromosome is built of a specific room, professor, and timeslot IDs; which means random numbers cannot be simply select. Additionally, because the rooms, professors, and timeslots all have a different range of IDs, they also cannot be a random number between 1 and "X". In this method a uniform crossover will be used to solve the mutation problem. In the uniform crossover, genes are selected at random from an existing and valid parent. The parent might not be the fittest individual in the population, but at least it's valid.

Fitness function

The fitness is calculated by the following function;

$$F_{cost}(x) = \max \sum_{t=1}^{n} P_c + P_t + P_o + P_p \quad (1)$$

where, $F_{cost}(x)$ is a penalty cost of x^{th} generation
x is the generation of exam timetable under evaluation
$P_{(c,t,o,p)}$ is the sub-function of constraints

In addition, the constraints are formulated as follows;

$$P_c = \sum_{c=1}^{n_c} \sum_{g=1}^{n_g} isNotInTimeSlot[(C_c, G_g)] \quad (2)$$

n_c is the number of courses and n_g is the number of groups of students enrolled in the course. The function equals **1** if $isNotIntimeSlot[(C_c, G_g)]$ return true and **0** otherwise.

$$P_t = \sum_{c=1}^{n_c} \sum_{g=1}^{n_g} \sum_{t=1}^{n_t} isCourseDifferenceTime[(C_c, G_g), T_t] \quad (3)$$

n_c is the number of courses, n_g is the number of groups of students enrolled in the course, n_t is the number of time slots, and T_t is the t^{th} time slot. The function equals **1** if *isCourseDifferenceTime*[$(C_c,G_g),T_t$] != (C_c,G_g), *x* and **0** otherwise.

$$P_o = \sum_{t=1}^{n_t}\sum_{c=1}^{n_c}\sum_{g=1}^{n_g} isCapacityOverLoad[(C_c, G_g),|\, G_g \,|>|\, R_g \,|] \tag{4}$$

n_c is the number of courses, n_g is the number of groups of students enrolled in the course, n_t is the number of time slots, and t^{th} is a time slot. The function equals **1** if *isCapacityOverLoad*[$(C_c,G_g),|G_g|>|R_g|$] return true and **0** otherwise.

$$P_p = \sum_{t=1}^{n_t}\sum_{c=1}^{n_c}\sum_{g=1}^{n_g} isLecturer\,\mathrm{Pr}\,efer[(C_c, G_g), T_t] \tag{5}$$

n_c is the number of courses, n_g is the number of groups of students enrolled in the course, n_t is the number of time slots, and T_t is the t^{th} time slot. The function equals **1** if *isLecturerPrefer*[$(C_c,G_g),T_t$] != ($C_{c'}$,$G_{g'}$), $T_{t'}$ and **0** otherwise.

Results and discussion

In order to test the effectiveness of IGA, IA, and GA when solving the timetabling problem, a comparison with the PSO algorithm was performed to investigate trends of performance. All coding was written in Java and the test case focused on the 3 above algorithms. All tests were executed on a 3.30 GHz Intel core i5 processor with 16 GB of ram. The convergence graph for IGA, IA, GA, and PSO shows progress until a valid solution for each of the algorithms were discovered. The specifications of the problem are shown in **Table 1**.

Table 1 General data used by the Genetic Algorithm.

No.	Operator	Quantity/Type
1	Number of individuals	1000
2	Crossover probability	0.9
3	Mutation probability	0.02
4	Number of generations	1000
5	Selection mechanism	Tournament selection
6	Crossover type	Two point crossover

The sample data test of the problem was acquired from the Faculty of Information Technology, King Mongkut's University of Technology North Bangkok which included 23 classes, 33 lessons, 20 professors, 10 rooms, and 18 timeslots for testing.

Table 2 Comparison of run time.

Generation	GA	AIS	IGA	PSO
100	11 (sec)	10 (sec)	**8 (sec)**	9 (sec)
200	23 (sec)	22 (sec)	**20 (sec)**	23 (sec)
400	46 (sec)	39 (sec)	**37 (sec)**	39 (sec)
800	95 (sec)	80 (sec)	**78 (sec)**	82 (sec)
1000	121 (sec)	92 (sec)	**85 (sec)**	93 (sec)

From **Table 2**, it is clear to see that IGA greatly reduced the running time. The specifications of the problem were also similar in **Table 1**. This is mainly because IGA detected and reduced the number of individuals during the initialize process. Off course, the search space could be less and also the generation of run time more effective.

When performing the simulations, it was found that the convergence of the algorithm depends on a number of timetabling initials including the number of subjects, professors, and hours per week of each subject. The graph in **Figure 5** provides a comparison of the proposed algorithm with the conventional population operator based algorithm.

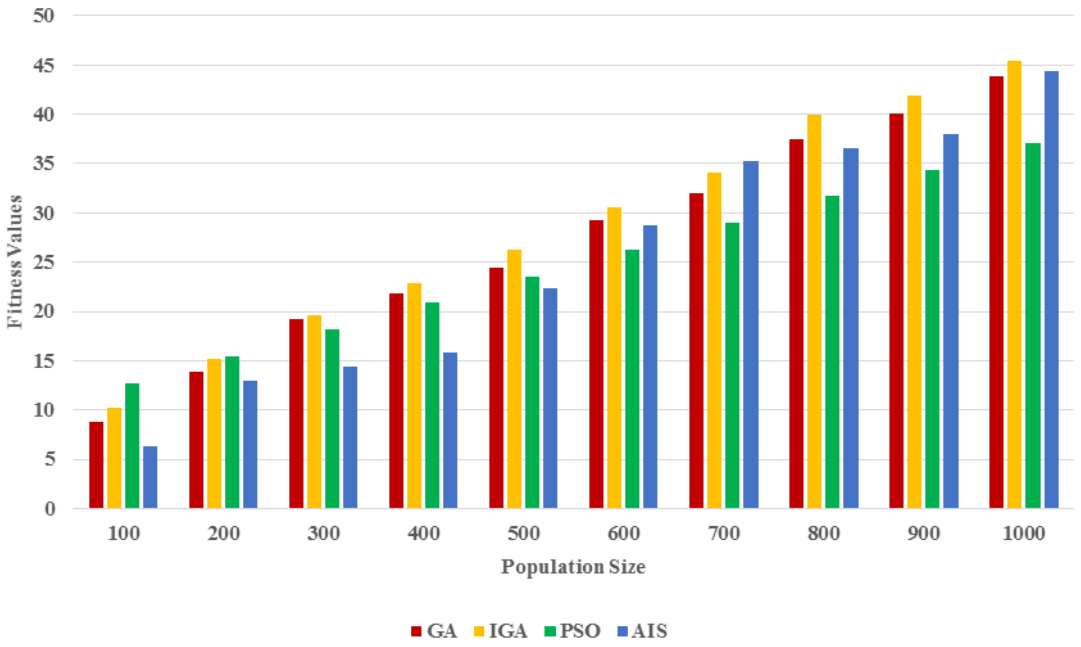

Figure 5 Algorithm comparison between fitness values and the number of populations.

In the PSO calculation, the practice set size of population for GA, IA, and IGA with $c_1 = 2$, $c_2 = 2$, was $w = 1/(2\times\log_2)$. In the GA and IGA calculation, mutation rate = 0.02 and crossover = 0.9. By observing the simulation results and the graphs, it is inferred that, for most of the populations, there was no great difference between the execution times and fitness values for the GA, IA, IGA and PSO. Until the population reached 500, the IGA had an increase of time whereas PSO remained linear. However, the GA and IGA fitness values rose more than PSO when the population increased. In IA, it still has less fitness values at the beginning but then rose higher when the population increased. This might stem from the fact that the parameters of GA, IA and IGA were different from PSO such as velocity, global and local search space, and so on. Over all, when comparing between IGA and GA, it is cleared to see that IGA had less fitness than GA but IGA consumed less time to reach the optimal solution than GA.

For the next investigation, the fitness value was studied focusing on the effectiveness of crossover probability varying from 0.0 - 1.0 increasing in 0.1 increments at each step, as shown in **Table 3**. The size of problem was then reduced to a medium size, the number of individuals and generations were 500 but the others were the same as in **Table 1**. In this evaluation, the fitness values were determined when it was stable at 1.0. The results illustrated that the highest fitness value occurred when crossover probability was in the range of 0.9. In the other hand, the IA and PSO could not be evaluated in **Tables 3 - 5** because these 2 algorithms contained different processes and parameters.

Table 3 Effects of crossover probability to fitness value.

Crossover probability	Number of generations		Fitness value	
	GA	IGA	GA	IGA
0.0	195	454	23344	22471
0.1	399	104	23720	23493
0.2	24	208	24028	23860
0.3	77	311	24270	22735
0.4	75	178	24101	22954
0.5	78	42	24568	24182
0.7	59	16	24954	18532
0.9	75	20	**25306**	**24413**
1.0	75	22	25107	24285

Table 4 illustrates the effects of mutation rates with 2 different crossovers with probability at 0.9, which is the best result of fitness value as found in **Table 3**. The result shows that the highest fitness value occurred when mutation rates were in the range of 0.02 with double point crossover. This is the optimal value to solve the problem because of the soft constrain violations from the fitness function described in the previous section.

Table 4 Effects of mutation rate with different types of crossover probability to fitness value.

Mutation rate	Single point crossover		Double point crossover	
	GA	IGA	GA	IGA
0.02	**25282**	**18989**	**25646**	**18184**
0.04	25219	18603	25439	18085
0.06	24593	18520	25212	17343
0.08	24543	17741	24067	17666
0.10	24758	17898	24065	17138
0.12	24532	16988	24377	17286
0.14	24426	17293	23988	17815
0.16	24199	17434	24251	17397
0.18	24066	17459	24109	16820
0.20	23989	17098	24026	16389

Finally, **Table 5** illustrates the effects of mutation rates with 2 types of selection methods. In this evaluation, the double point crossover was used and increased 0.02 at each step of the mutations. The result shows that the highest fitness value occurred when mutation rate was in the range of 0.02.

Table 5 Effects of mutation rate with different selection methods and double point crossover probability to fitness value.

Mutation rate	Roulette-wheel selection		Tournament selection	
	GA	IGA	GA	IGA
0.02	**24019**	**17723**	**25646**	**18185**
0.04	24003	17465	24439	18084
0.06	23941	17077	25212	17343
0.08	23890	16986	24067	17666
0.10	23591	16045	24065	17138
0.12	23421	16249	24377	17286
0.14	23322	16514	23988	17815
0.16	23363	16136	24251	17397
0.18	23450	16122	24109	16820
0.20	23597	16155	24026	16389

Conclusions

In this paper the author has proposed a new hybrid approach, which is a combination of a steady-state in GA, IA with an IGA technique and PSO to solve timetabling problems. This new hybrid could be used to improve the quality of initial solutions which are generated randomly. The algorithm was tested on a set of courses from the Faculty of Information Technology, KMUTNB, Thailand. Experiments were carried out to investigate the effect when combining the PSO search, IA search, and GA search with IGA in terms of performance based on a set of instances. The results revealed that the proposed hybrid approach was able to find a feasible solution for problem instances and provided superior performance for small instances by generating the best solution with no violation of constraints existing in the solution.

Acknowledgements

This research was supported by the Ministry of Science and Technology, whom provided funds throughout this work. Gratitude is also due to the Faculty of Information Technology, King Mongkut's University of Technology North Bangkok for the use of computers to run these experiments.

References

[1] MM Javidi, RH Fard and M Jampour. Research in random parameters of genetic algorithm and its application on TSP and optimization problems. *Walailak J. Sci. & Tech.* 2015; **12**, 27-34.

[2] A Salwani and T Hamza. Generating university course timetable using genetic algorithms and local search. *In*: Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology. Busan, South Korea, 2008, p. 254-60.

[3] C Wutthipong, K Soradech and S Nidapan. Performance comparison of genetic algorithm's crossover operators on university course timetabling problem. *In*: Proceedings of the 8th International Conference on Computing Technology and Information Management. Seoul, South Korea, 2012, p. 781-6.

[4] TBH Thi, DP Quang and DP Duy. Genetic algorithm for solving the master thesis timetabling problem with multiple objectives. *In*: Proceedings of the Conference on Technologies and Applications of Artificial Intelligence. Tainan, Taiwan, 2012, p. 74-9.

[5] S Alves, S Oliveira and A Neto. A novel educational timetabling solution through recursive genetic algorithms. *In*: Proceedings of the Latin America Congress on Computational Intelligence. Curitiba, Brazil, 2015, p. 1-6.

[6] I Supachate. A noval approach of genetic algorithm for solving examination timetabling problems: A case study of Thai Universities. *In*: Proceedings of the 13th International Symposium on Communications and Information Technologies. Surat Thani, Thailand, 2013, p. 233-7.
[7] J Sheung, A Fan and A Tang. Time tabling using genetic algorithm and simulated annealing. *In*: Proceedings of the IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering. Beijing, China, 1993, p. 448-51.
[8] SC Chu and HL Fang. Genetic algorithms vs. Tabu search in timetable scheduling. *In*: Proceedings of the 3rd International Conference Knowledge-Based Intelligent Information Engineering Systems. Adelaide, Australia, 1999, p. 492-5.
[9] S Vinayak, R Kaushik and B Sivaselvan. Time table scheduling using genetic algorithms employing guided mutation. *In*: Proceedings of the Computational Intelligence and Computing Research. Coimbatore, India, 2010, p. 1-4.
[10] B Nguyen, T Nguyen and T Tran. A new hybrid GA-Bees algorithm for a real-world university timetabling problem. *In*: Proceedings of the Intelligent Computation and Bio-Medical Instrumentation. Hubei, China, 2011, p. 321-6.
[11] S Meysam, S Mohammad and S Hedieh. A fuzzy genetic algorithm with local search for university course timetabling. *In*: Proceedings of the Data Mining and Intelligent Information Technology Applications. Macao, China, 2011, p. 250-4.
[12] PV Om, G Rohan and SB Vikram. Optimal time-table generation by hybridized bacterial foraging and genetic algorithm. *In*: Proceedings of the Communication Systems and Network Technologies. Rajkot, India, 2012, p. 919-23.
[13] LN de Castro and J Timmis. Artificial immune systems: A novel paradigm to pattern recognition. *In*: JM Corchado, L Alonso and C Fyfe (eds.). Artificial Neural Networks in Pattern Recognition. University of Paisley, UK, 2002, p. 67-84.
[14] S Juan, DS Yi and X Min. Reach on application of IGA (Immune Genetic Algorithm) to the solution of course-timetabling problem. *In*: Proceedings of the Computer Science & Education. 2009, p. 1105-9.
[15] AR Hafizah, I Zaidah and MH Naimah. Bipartite graph edge coloring approach to course timetabling. *In*: Proceedings of the Information Retrieval & Knowledge Management. Selangor, Malaysia, 2010, p. 229-34.
[16] J Zhou. Revisiting negative selection algorithms. *J. Evol. Comput.* 2007; **15**, 223-51.