Hybridizing Genetic Algorithm and Record-to-Record Travel Algorithm for Solving Uncapacitated Examination Timetabling Problem

Munther Hameed Abed[1], Alicia Y. C. Tang[2]
[1]College of Graduate Studies
[2]College of Information Technology
Universiti Tenaga National
munt1979@yahoo.com, aliciat@uniten.edu.my

Abstract – **Examination timetabling is one of the most important administrative tasks in academic institutions. They are used to schedule examinations into timeslots and rooms. Many methods have been developed to solve examination timetabling problems. Metaheuristics have shown good results especially if they are hybridized with other methods. Genetic Algorithms (GAs) are one of the techniques that have been used in optimization problems. Record-to-Record Travel (RRT) is another optimization method that has been introduced for local search. In this paper, we describe the combined use of GA and RRT, called GARRT. In particular, the process of hybridization of the two algorithms to solve the uncapacitated examination timetabling problem is discussed. GARRT aims to balance the global search (by GA) and the local search (by RRT). This work uses Carter's benchmark datasets as the testbed. Simulation results showed that GARRT performed better when compared to the results generated by the GA approach alone. A good result is achieved by minimizing the violation of the soft constraints.**

Keywords – Examination timetabling; Genetic Algorithm; Record To Record Travel; Optimization

I. INTRODUCTION

The timetabling problem is one of the most difficult NP-hard problems that universities and educational institutions have to face every semester. Timetabling is the process of scheduling for a set of classes or examinations which are organized into several timeslots, subject to numerous constraints [1]. The difficulty in solving problems of examination timetabling is due to the numerous parameters that must be taken into consideration, and a variety of violations in the examination timetable of the various institutions that have different constraints [2]. Examination timetabling can also be described as a process of allocating examinations to fixed venues and timeslots to avoid conflicts.

This '*conflict*' can arise when students are simultaneously arranged to be in the same venue that has been scheduled for the usage of other students or when a student has two examinations at the same time [3]. Planning the examination schedules manually has been widely used to solve the timetabling problem. However, manual approaches are not always feasible as it needs plenty of manpower and the process is time consuming. Metaheuristic methods have been used to overcome the manual solution [4]. A metaheuristic approach is widely used because it usually gives a good solution compared to the sequential heuristic approach within an acceptable time frame [5]. The main advantage of metaheuristic techniques is that they have the ability to handle a wide range of constraints efficiently. Metaheuristic is defined as follows: *"A metaheuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality"*. In this paper we discuss a hybrid metaheuristic approach to solve the uncapacitated examination timetabling problem, where the capacity of the room is not taken into account because there are many rooms usable for examinations. Meanwhile in the capacitated problem, the capacity of the room is regarded as a hard constraint. The proposed algorithm combines two metaheuristic, namely the Genetic Algorithm and Record to Record travel algorithm (GARRT). In the first phase, GA is used to generate a population of solutions in order to maintain the solutions diversity. While in the second phase, RRT is used to enhance the solutions generated by the GA. We choose RRT due to its ability to tune a single parameter [6]. There are many parameters to consider when solving uncapacitated examination timetabling problems such as huge numbers of examinations, insufficient numbers of timeslots to schedule of the examinations and a very large number of students, where each student has one or more examinations but must not have more than one examination in the same timeslot [7].

II. RELATED WORK

Many algorithms have been deliberated on the timetabling problem. According to Wren [8]:

"*Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives".*

Timetable scheduling problems can be divided into several types, and these include education, transport, industry and sport [8]. The educational timetabling can be divided into school timetabling and university timetabling which comprises of course timetabling and examination timetabling. The school timetabling is the problems faced by school, and are related to the weekly scheduling for all the lessons of a school. This problem involves a group of teachers, classes, lessons and weekly periods since they are predefined. The university timetabling has two problems particularly course and examination timetabling, where the problem concerning the course is the procedure of allocating timeslots and rooms to enable students and lecturers meeting. Meanwhile the problem concerning the examination timetabling is the venue and timeslots allocations for the student examinations. The examination timetabling is one of the most substantial administrative tasks in universities. This task requires every examination to be allocated to a set of timeslots under certain constraints. The examination timetabling problem can be described as [9]: *"The assigning of examinations to a limited number of available time periods in such a way that there are no conflicts or clashes".*

During the last decade, there have been several algorithms to solve examination timetabling problems. In 1975, an initial survey proposed by Miles [10] identified a beneficial bibliography of computer aided timetabling early development. In 1979, Schmidt and Strohlein proposed another survey comprising more than 200 references of timetabling works before that particular year [11]. Numerous mathematical models have been developed to tackle course timetabling and examination timetabling using graph coloring based method [12].

The survey mentioned above was updated to concise algorithmic methods from 1986 to 1996. Those methods can be classified into four categories: sequential methods, metaheuristics methods, constraint-based methods, and cluster methods [9]. The recent research was outlined on examination timetabling and course timetabling that had been carried out consisting of hybrid evolutionary algorithms, metaheuristics, multi-criteria techniques, case-based reasoning methods and adaptive techniques [13]. Most of the approaches have been tested on the 13 instances that were introduced by Carter [9] in 1996. Table 1 shows the 13 datasets.

Table 1. Carter benchmark examination timetabling dataset

Data	No. of Timeslots	No. of Exams	No. of Students
car-f-92	32	543	18419
car-s-91	35	682	16925
ear-f-83	24	190	1125
hec-s-92	18	81	2823
kfu-s-93	20	461	5349
lse-f-91	18	381	2726
pur-s-93	42	2419	30032
rye-s-93	23	486	11483
sta-f-83	13	139	611
tre-s-92	23	261	4360
uta-s-92	35	622	21266
ute-s-92	10	184	2750
yor-f-83	21	181	941

III. Problem Definition and Formulation

A set of examinations = (e_1 … e_E) are required to be scheduled within a certain number of timeslots = (t_1 … t_T), which may or may not be fixed beforehand subject to a variety of hard and soft constraints. Hard constraints must be satisfied in order to produce a feasible timetable, whilst violation of soft constraints should be minimized and provides a measure of how good the solution is via the objective function. The uncapacitated examination timetabling problem of Carter benchmark datasets are used which contain 13 real world examinations timetabling problems from various instances, where room capacity requirement is unnecessary because it has a large number of rooms usable for examinations. To calculate the objective function for each instance, we use the following inputs [4]:

- E: Number of examinations
- T: Number of timeslots
- S: Number of students
- C: Conflict matrix, each element in conflict matrix are representative by Cij, and i,j∈ {1….. E}. A group of students takes the examinations i and j.

The hard constraint in uncapacitated examination timetabling is usually represented by the following:

"*A student must not have two or more examinations scheduled in same timeslot*".

A clash-free requirement is illustrated in (1), in which the students will not be asked to take two examinations at the same time.

$$\boldsymbol{Clash-free} = \sum_{i=1}^{E-1} \sum_{j=i+1}^{E} c_{ij} \times x(t_i, t_j) = 0 \qquad (1)$$

$$\boldsymbol{x(t_i, t_j)} = \begin{cases} 1 & if\, t_i == t_j \\ 0 & otherwise \end{cases} \qquad (2)$$

Soft constraints refer to those that are desirable to be fulfilled but cannot be entirely fulfilled in general. Soft constraint are diversified and dependent on the particularities of each university since it will assess the most desirable and feasible timetable. Soft constraints generally deal with the *Objective* function that measures a given feasible solution, in addition to the different solutions which can be compared and improved. The main soft constraint in uncapacitated examination timetabling is as follows:

"*Spreading examinations over timeslots which attempts to give students enough time for revision."*

The *Objective* function was used to calculate the cost of the resulting timetables, and to minimize the average cost per student. The cost was assigned by using proximity values between two examinations [14]. To find the cost for an examination i, (3) is applied:

$$\boldsymbol{CostFunction(i)} = \sum_{j=i}^{E} c_{ij} \times proximity\,(t_i, t_j) \qquad (3)$$

Where the (c_{ij}) is the number of students in conflict that equal to 0 when i = j, and the proximity value is as shown in (4).

$$\boldsymbol{proximity}(t_i, t_j) = \begin{cases} \frac{2^5}{2^{|t_i - t_j|}} & if \quad 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases} \qquad (4)$$

The *Objective* function is also used to spread the examinations for students as much as possible within a limited number of timeslots. It is used by Carter to measure the quality of a solution depending on the sum of proximity costs. For example, for a proximity value of 16, a student has two successive examinations, and to realize a proximity value of 8, a student will have free timeslot between each two examinations. Two timeslots give the proximity value of 4, three timeslots give proximity value of 2 and four timeslots give proximity value of 1.The results calculated by (3) are obtained by taking the summation of the values divided by the number of the students (S), to give an average penalty per student [15], where the penalty cost can be formulated as shown in equation (5).

$$\boldsymbol{Penaltycost} = \frac{\sum_{i=1}^{E} CostFunction(i)}{S} \qquad (5)$$

Figure 1 shows the pseudocode for the constructing the proximity coefficient matrix and calculating the penalty cost for each solution in uncapacitated examination timetabling problems. This pseudocode is designed based on equations (3) – (5).

```
Begin
    Assign 0 to Cost
For all examination $e_i$
For all examination $e_j$
If (Conflict Matrix [$e_i$][ $e_j$] not equal to 0)
 Perform weight=Absolute value of (Timeslot[$e_i$]-Timeslot[$e_j$])
If (weight 1 to 5)
 perform Proximity Matrix [$e_i$][$e_j$]= 2 ^ (5 – weight)
Else
          Assign 0 to Proximity Matrix [$e_i$][$e_j$]
          Perform Cost + = Conflict Matrix [$e_i$][$e_j$] * Proximity
                         Matrix [$e_i$][$e_j$]
    End for
    End for
Perform Penalty = Cost / (Number of Student *2)
End
```

Figure 1. Pseudo code of calculate Penalty cost

Conflict density is used to determine the degree of complexity of each dataset as shown in (6). Table 2 illustrates the conflict density for Carter datasets.

$$Conflict\ Density = \frac{number\ of\ conflicts}{(number\ of\ exams)^2} \qquad (6)$$

Table 2. Conflict matrix density

Dataset	No. of Examination	Conflict Density
car-f-92	543	0.14
car-s-91	682	0.13
ear-f-83	190	0.27
hec-s-92	81	0.42
kfu-s-93	461	0. 06
lse-f-91	381	0.06
pur-s-93	2419	0.03
rye-s-93	486	0.07
sta-f-83	139	0.14
tre-s-92	261	0.18
uta-s-92	622	0.13
ute-s-92	184	0.08
yor-f-83	181	0.29

IV. The Proposed Hybrid Algorithm (GARRT)

The successes of GA in solving a number of optimization problems are the main goal for hybridizing it with RRT local search algorithm to obtain the optimal solution. First, GA is used to solve the uncapacitated examination timetabling problem of Carter benchmark dataset. Subsequently, RRT is used to improve the solutions obtained by GA. The RRT is used after the mutation process is applied, as shown in the dotted line box in Figure 2.

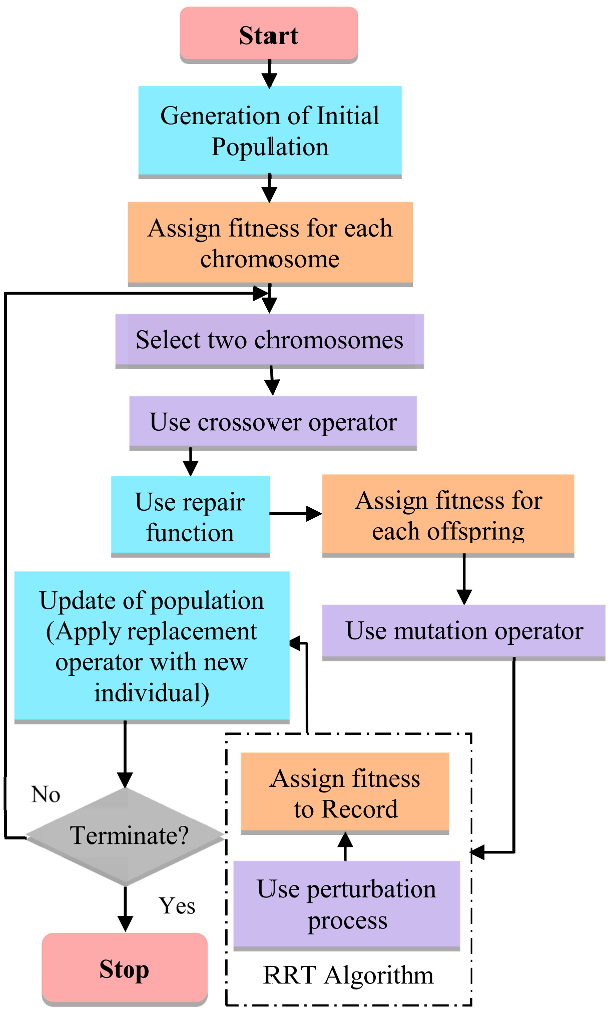

Figure 2. Flow chart of GA and RRT

A. *Genetic Algorithm (GA)*

GA is one of the best optimization algorithms. It is a general algorithm that works well in any search space and is able to produce a high quality solution [16]. GA employs the principles of selection and evolution to create several solutions to a particular problem.

a) *Chromosome Encoding*

Chromosome encoding is a method that is required to encode the possible solutions to any problem in a form so that a computer can process it. A chromosome consists of a set of genes that contain information about the problem needed to be solved. Each gene is a variable in the search space to this problem. Chromosome encoding is the first process in implementing genetic algorithms. In this paper the chromosome representation depend on the number of examinations for each dataset. Each gene contains information on what examinations are scheduled for a certain timeslot, where every gene is represented by the two variables (i.e. the number of examination and timeslot). The permutation encoding is used since it deals with only numbers. Figure 3 shows the chromosome encoding used in this research, where a vertical line "|"represents the chosen crossover point.

Exams	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Timeslots	5	3	4	3	1	2	4	1	3	2	6	1	6	5

Figure 3. Chromosome encoding

b) *Initial Population*

The initial population is known as accepted solutions that started the GA. Each solution is denoted by a chromosome. Generally, this initial population is produced randomly to generate all possible solutions. Initial population has a large number of potential solutions. Optimal solutions may already exist within those potential solutions.

c) *Fitness Function*

The fitness function is one of the most important parts in GA implementation. Fitness function allows the algorithm to characterize the fittest chromosome among others and leads the GA into a better solution [17]. Every chromosome needs to be evaluated according to the assumed constraints in order to identify the fittest chromosome among others. The fitness function can be used by GA to select the chromosome for crossover and mutation processes. In this work, the fitness function is used to calculate the cost of the resulting timetables, and to minimize the average cost per student as shown in (3).

d) *Roulette Wheel Selection Operator*

Roulette wheel selection is considered the most popular type of selection (Figure 4). It is used for selecting possibly useful solutions for crossover and it is also the best standard approach for parent selection [18]. The purpose of a selection is to provide parents (individuals) for crossover and mutation operations to build a new population. The individuals that have high fitness may be chosen many times. The probability for each individual is calculated based on (7).

$$\boldsymbol{p_i} = \frac{f_i}{\sum_{i=1}^{n} f_i} \tag{7}$$

In the equation, fi is the value of fitness for each individual, n is the number of individuals in population also called population size and $\sum_{i=1}^{n} f_i$ is the total of fitness to individuals.

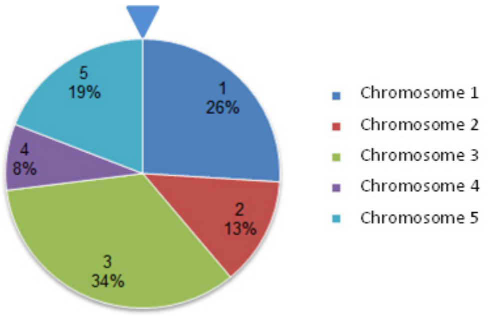

Figure 4. Roulette wheel selection example

e) *Uniform Crossover Operator*

Crossover is a major operator in the genetic algorithm that combines two individuals (parents) to create a new individual (offspring) [19], where genes are selected from both parents to produce a new offspring. The main purpose of crossover is to produce a new individual which is possible to be superior to their parents if it takes the good characteristics of each parent. Uniform crossover determines what are the values given to gene in the offspring individuals with some probability- known as the crossover rate. It also permits the parent individuals to be combined at the gene level unlike some crossover types which is combined at the segment level (Figure 5). An examination is chosen from one of the chromosomes (parent) and set in the one of the chromosomes (offspring). Another examination of the other chromosome (parent) is to be set to the other chromosome (offspring). The choice of the examination is random depending on the crossover rate.

Figure 5. Uniform crossover

Parent 1	E	5	8	12	6	10	2	4	9	3	7	1	14	11	13
	T	5	3	4	3	1	2	4	1	3	2	6	1	6	5
Parent 2	E	7	11	3	6	14	9	2	4	1	12	13	10	5	8
	T	1	1	1	2	2	3	3	3	4	4	5	5	6	6
Offspring 1	E	2	7	4	10	1	3	11	6	14	9	13	5	8	12
	T	1	1	2	2	3	3	3	4	4	5	5	6	6	6
Offspring 2	E	1	11	13	5	8	3	7	4	10	2	9	6	12	14
	T	1	1	1	2	2	3	3	4	4	5	5	6	6	6

f) Mutation Operator

Mutation is employed to keep genetic diversity from one population of chromosomes generation to the other. Mutation changes one or more chromosome gene values from its initial state (Figure 6). The new gene values may enable genetic algorithm to reach a solution better than the previous solution. In general the mutation avoids genetic algorithm from falling in local optima. This type of mutation is called insertion mutation where the examination is randomly selected from one of timeslot and its insertion in another timeslot, which does not cause a conflict. Mutation process is very important to give the individual good fitness which may be better than the previous individual.

Original Offspring1	E	2	7	4	10	1	3	11	6	14	9	13	5	8	12
	T	1	1	2	2	3	3	3	4	4	5	5	6	6	6
Original Offspring2	E	1	11	13	5	8	3	7	4	10	2	9	6	1 2	14
	T	1	1	1	2	2	3	3	4	4	5	5	6	6	6
Offspring1	E	2	7	4	10	1	3	11	6	8	14	9	13	5	12
	T	1	1	2	2	3	3	3	4	4	5	5	6	6	6
Offspring2	E	1	11	13	5	8	3	7	4	10	2	9	14	6	12
	T	1	1	1	2	2	3	3	4	4	5	5	6	6	6

Figure 6. Mutation operator

g) Repair Function

After the end of crossover operation, is the possible appearance of the errors in the timetable. Some of the errors are the repetition or loss of one examination, and some students have two or more examinations at the same time. Therefore, the repair function is used to prevent infeasible timetables to survive in subsequent generations. The function guarantees that all the constraints that need to be satisfied will be fulfilled.

h) Genetic Algorithm Parameters

Crossover rate, mutation rate and population size are among the many parameters of GA. The crossover rate aims to determine amount chromosomes, which will remain for the next generation. The offspring can be a replica of their parents if there is no crossover. If the crossover exist, the offspring can be considered as made of both parent's chromosome. The rate of crossover occurring is usually 60% to 70% [20]. The mutation rate is used to determine the frequency of the mutated chromosome part. If the mutation does not exist, the offspring are created directly after crossover without any alteration (replica). If the rate of the mutation is 100%, then the entire chromosome will be altered. The rate of mutation occurring is usually 0.1% to 1% [20]. The population size is considered one of the important parameters in genetic algorithm that is used to determine the number of chromosomes in the population (in one generation). The population size relies on the nature and the form of the problem. If population size is small, the search process in the genetic algorithm is fast because a small part of search space is explored. Conversely, the search process slows down if the population size is big because the number of exploration is large.

B. Record to Record Travel (RRT) Algorithm

RRT algorithm is a local search method. It is a deterministic optimization algorithm which is inspired from simulated annealing. The algorithm depends on the cost function which is gradually improved by exploring the neighborhoods. Initial solutions of the algorithm are usually randomly selected, and then apply a perturbation mechanism to obtain a better solution compared to the current best solution found so far (called a "record"). This process is done by searching for the neighborhood of the current best solution. If the new solution is better than the current solution, then the new solution will be marked as a "record", otherwise the current solution will remain (without being replaced).

If the new solution is not much worse than the current solution, it will be accepted as a neighborhood[21]. RRT has only one parameter (i.e., deviation value). In this case, if the deviation value is small it will generate poor results within a reduced search time, while if the value is high it will generate good results after an important computational time. The deviation value used in this work is $d = 0.01 \times \text{Record}$, and d represents the maximum allowed deviation that determines the amount of worse value which is acceptable as compared to the current record [20].

V. EXPERIMENTAL RESULTS

The GARRT approach was experimented for 13 Carter benchmark datasets. Table 3 illustrates the parameters used in our approach.

Table 3. Parameters setting for GARRT

No. of iteration	10000
Crossover rate	0.7
Mutation rate	0.01
Population size	20
Deviation value	0.01 × record

In the approach, the current solutions in the population are used to produce new solutions. The best solution is then determined, and the other solutions are ignored based on the minimization criteria. Table 4 shows the results for each of the best penalty, average penalty and standard deviation for GARRT that are applied to the 13 benchmark dataset. The GARRT algorithm was run 5 times for each dataset.

Table 4. The results obtained by GARRT

Datasets	Best Penalty	Average Penalty	Standard deviation
car-f-92	4.54	4.88	0.20
car-s-91	5.38	5.67	0.21
ear-f-83	36.27	39.09	1.41
hec-s-92	10.73	11.42	0.44
kfu-s-93	15.17	15.76	0.36
lse-f-91	11.87	12.81	0.47
pur-s-93	6.87	7.08	0.21
rye-s-93	8.6	9.08	0.27
sta-f-83	158.16	158.85	0.39
tre-s-92	8.86	9.13	0.18
uta-s-92	3.59	3.95	0.19
ute-s-92	25.34	26.48	0.62
yor-f-83	40.26	40.70	0.34

The best results were obtained by running the GARRT algorithm for each of the 13 Carter's datasets, as presented in Table 5. The approaches selected for comparison with GARRT algorithm are: Carter's sequencing heuristics with backtracking [15]; Tabu search by Di Gaspero and Schearf [22]; Local search-based method which includes an optimization step after allocating each examination by Caramia et al. [23]; Hybrid constraint programming, simulated annealing and hill climbing by Merlot et al. [24]; GRASP [25]; Asmuni's fuzzy multiple ordering criteria [26]; Ahuja–Orlin's large neighbourhood search approach [3]; A new neural network based construction heuristic by Corr et al. [27], and a graph-based hyper heuristic by Burke et al. [28].

The last row in the Table 5 shows the results of the approach used in this research. As shown in Table 2, GARRT outperformed some of the results generated by other approaches in terms of smaller penalty values obtained. The good result is achieved by minimizing the violation of the soft constraints.

Table 5. Comparison of GARRT with other approaches

	car-f-92	car-s-91	ear-f-83	hec-s-92	kfu-s-93	lse-f-91	rye-s-93	sta-f-83	tre-s-92	uta-s-92	ute-s-92	yor-f-83
Carter et al, 1996	6.2	7.1	36.4	10.8	14	10.5	7.3	161.5	9.6	3.5	25.8	41.7
Di Gaspero&Schearf, 2001	5.2	6.2	45.7	12.4	18	15.5	-	160.8	10	4.2	29	41
Caramia et al, 2001	6	6.6	29.3	9.2	13.8	9.6	6.8	158.2	9.4	3.5	24.4	36.2
Metrol et al, 2003	4.3	5.1	35.1	10.6	13.5	10.5	8.4	157.3	8.4	3.5	25.1	37.4
Casy& Thompson, 2003	4.4	5.4	34.8	10.8	14.1	14.7	-	134.9	8.7	-	25.4	37.5
Asmuni et al, 2004	4.56	5.29	37.02	11.78	15.81	12.09	10.35	160.42	8.67	3.57	27.78	40.66
Abdullah et al, 2006	4.4	5.2	34.9	10.3	13.5	10.2	8.7	159.2	8.4	3.6	26	36.2
Corr et al, 2006	6.24	7.21	49.44	13.57	19.9	14.99	-	159.28	10.77	4.48	31.25	-
Burke et al, 2007	4.53	5.36	37.92	12.25	15.2	11.33	-	158.19	8.92	2.88	28.01	41.37
GARRT	4.54	5.38	36.27	10.73	15.17	11.87	8.6	158.16	8.86	3.59	25.34	40.26

VI. Conclusion and Future Work

The combined use of GA and RRT (GARRT) for solving uncapacitated examination timetabling problem using the Carter's datasets has been described. The proposed approach aimed to enhance GA efficiency by minimizing the penalty cost for each dataset. To accomplish this aim, GA is used in the first stage to explore the search space while RRT algorithm is used in the second stage to enhance the solution generated by the GA operators. The paper also presented comparison results of GARRT with various approaches used by other researchers. Experimental results showed that GARRT performed better than most of the approaches described in this paper in which the number of violation is significantly reduced. The good performance can be attributed to the nature of GA and RRT, which tries to balance the global search (by GA) and the local search (by RRT). In future work, we will apply GARRT to solve the capacitated examination timetabling problem and courses timetabling problem.

References

[1] H. L. Fang, "Genetic algorithms in timetabling and scheduling," in *Department of Artificial Intelligence*. vol. PhD: University of Edinburgh, 1994.

[2] A. Dammak, A. Elloumi, and H. Kamoun, "Lecture timetabling at a Tunisian university," *International Journal of Operational Research,* vol. 4, pp. 323-345, 2009.

[3] S. Abdullah, S. Ahmadi, E. K. Burke, and M. Dror, "Investigating Ahuja- Orlin's large neighbourhood search approach for examination timetabling," *OR Spectrum,* vol. 29, pp. 351-372, 2006.

[4] S. Abdullah, "Heuristic approaches for university timetabling problems," in *School of Computer Science and IT*. vol. *PhDUnited Kingdom*: *University of Nottingham*, 2006.

[5] R. Qu, E. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling,* vol. 12, pp. 55-89, 2009.

[6] S. Seitz and P. Orponen, "An efficient local search method for random 3-satisfiability," *Electronic Notes in Discrete Mathematics,* vol. 16, pp. 71-79, 2003.

[7] E. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined local search approach to exam timetabling problems," *IIE Transactions,* vol. 36, pp. 509-528, 2004.

[8] A. Wren, "Scheduling, Timetabling and Rostering - A special relationship?," *Practice and Theory of Automated Timetabling,* pp. 46-75, 1996.

[9] M. Carter and G. Laporte, "Recent developments in practical examination timetabling," *Practice and Theory of Automated Timetabling,* pp. 1-21, 1996.

[10] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing

systems," *Parallel Computing and Transputer Applications,* vol. 1, pp. 177-186, 1992.

[11] K. Socha, M. Sampels, and M. Manfrin, "Ant algorithms for the university course timetabling problem with regard to the state-of-the-art," *Applications of evolutionary computing,* pp. 334-345, 2003.

[12] D. de Werra, "An introduction to timetabling," *European Journal of Operational Research,* vol. 19, pp. 151-162, 1985.

[13] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *European Journal of Operational Research,* vol. 140, pp. 266-280, 2002.

[14] E. Burke, *Practice and theory of automated timetabling: first international conference, Edinburgh, UK, August 29-Septmber 1, 1995: selected papers* vol. 1: Springer - Verlag, 1996.

[15] M. W. Carter, G. Laporte, and S. Y. Lee, "Examination timetabling: Algorithmic strategies and applications," *Journal of the Operational Research Society,* vol. 47, pp. 373-383, 1996.

[16] P. Bajpai and D. R. M. Kumar, "Genetic Algorithm - an Approach to Solve Global Optimization Problems," *Indian Journal of Computer Science and Engineering,* vol. 1, pp. 199-206, 2010.

[17] N. Nuntasen and S. Innet, "Application of genetic algorithm for solving university timetabling problems: a case study of Thai universities," 2007, pp. 128-133.

[18] A. N. Alexandrova and A. I. Boldyrev, "Search for the Li n 0/+ 1/-1 (n= 5-7) Lowest-Energy Structures Using the ab Initio Gradient Embedded Genetic Algorithm (GEGA). Elucidation of the Chemical Bonding in the Lithium Clusters," *Journal of Chemical Theory and Computation,* vol. 1, pp. 566-580, 2005.

[19] J. Zhang, H. S. H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *Evolutionary Computation, IEEE Transactions on,* vol. 11, pp. 326-335, 2007.

[20] T. El-Ghazali, "Metaheuristics: from design to implementation," *Jonh Wiley and Sons Inc., Chichester,* 2009.

[21] D. Wong, H. W. Leong, and C. L. Liu, *Simulated annealing for VLSI design* vol. 42: Springer, 1988.

[22] L. Di Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," *Practice and Theory of Automated Timetabling III,* pp. 104-117, 2001.

[23] M. Caramia, P. Dell'Olmo, and G. Italiano, "New algorithms for examination timetabling," *Algorithm Engineering,* pp. 230-241, 2001.

[24] L. Merlot, N. Boland, B. Hughes, and P. Stuckey, "A hybrid algorithm for the examination timetabling problem," *Practice and Theory of Automated Timetabling IV,* pp. 207-231, 2003.

[25] S. Casey and J. Thompson, "GRASPing the examination scheduling problem," *Practice and Theory of Automated Timetabling IV,* pp. 232-244, 2003.

[26] H. Asmuni, E. Burke, J. Garibaldi, and B. McCollum, "Fuzzy multiple heuristic orderings for examination timetabling," *Practice and Theory of Automated Timetabling V,* pp. 334-353, 2005.

[27] P. Corr, B. McCollum, M. McGreevy, and P. McMullan, "A new neural network based construction heuristic for the examination timetabling problem," *Parallel Problem Solving from Nature-PPSN IX,* vol. 4193, pp. 392-401, 2006.

[28] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research,* vol. 176, pp. 177-192, 2007.