Journal of ICT, 3 (2), pp: 1-14

A GENETIC ALGORITHM APPROACH FOR TIMETABLING PROBLEM: THE TIME GROUP STRATEGY

*A. B. M. Sultan, R. Mahmod, M. N. Sulaiman, and **M. R. A. Bakar

**Faculty of Computer Science and Information Technology*
University Putra Malaysia, 43400 UPM Serdang Selangor, Malaysia
***Department of Mathematic*
Faculty of Science and Environmental Studies
University Putra Malaysia

E-mail: {abakar, ramlan, nasir@fsktm.upm.edu.my}
E-mail: rizam@fsas.upm.edu.my

ABSTRACT

The university timetabling problems (TTP) deal with the scheduling of the teaching program. Over the last decade variant of Genetic Algorithm (GA) approaches have been used to solve various types of TTP with great success. Most of the approaches are problem dependent, applied only to the institutions where they were designed. In this paper we proposed time group strategy and Simple GA (TGGA) to solve highly constrained TTP. The proposed model promises to solve highly constrained timetabling with less effort. The model is tested and results are discussed.

Keywords : timetabling, heuristic, genetic algorithm.

1.0 INTRODUCTION

Timetabling problems have attracted the continuous interest of researchers mainly because they provide the opportunity of testing combinatorial solution methods in formulations that represent difficult practical problems (Dimopoulou and Miliotois, 2001). The problem of constructing course timetables for higher learning institutions includes the problem of allocating the set of courses offered by university to time periods and classrooms in such a way that no teacher, student or room is used more than once. Specifying timetabling constraints can be difficult for anyone, and often this task is handled by administration staff with little knowledge of mathematics and computing (Ross *et al.*, 1998). The problems typically incorporate many non trivial constraints of various kinds and attracted much interest for solving them using several methods such as graph colorings, heuristics, integer programming, neural networks, constraints satisfaction, genetic algorithm (GA), tabu search and recently case base reasoning. This paper does not attempt to compare such methods. Instead it illustrates how SGA with the assistance of simple heuristic can solve highly constrained timetabling problem.

2.0 RELATED BACKGROUND

Many current successful university timetabling systems are problem dependent, applied only to the institution where they were designed. The fact that the majority of the literatures reported in the area of timetabling focuses on considering more practical aspects of the problem, rather than in exploring in a more methodical way how suitable a particular method is for solving it (Marin, 1998). Another factor is due to the variations in each school's policies and standards, it is not an easy task to generate a satisfactory solution to meet the needs of all related factors (Wang, 2003).

Among many approaches, Genetic Algorithm (GA) was the prominent and widely accepted approach to solve various type of timetabling problems. The fact is over past decades GA and evolutionary approaches have dominated most of the literatures on this issue. For example, the genetic algorithm has been used to solve timetabling problems (Carter and Laporte, 1998; Paechter *et al.*, 1998; Marin, 1998; Carrasco and Pato, 2001; Ueda *et al.*, 2001; and Wang, 2003). Even thought most of the approaches retain the basic concepts of GA, but none of them are similar in terms of implementation such as problem encoding, behaviours of genetic operator and processing techniques.

Some researchers such as Burke and Petrovic (2002), Yu and Sung (2002) and Daskalaki *et al.*, (2003) consider that GA has been utilized as an effective tool for the timetabling solution. More recently Qu (2002) made a conclusion that the stochastic algorithms perform generally well with respect to the solution quality but it depends on the representations of problem and the employed operators. Even most of the discussions above reported solutions that are problem dependent, the author believed that some of the problems share common instances, therefore GA can be applied to various types of problem instances. A number of surveys done by (Carter *et al.*,1998) and (Schaerf, 1999) gave a complete description of these approaches.

Newall (1999) highlighted what is probably most lacking in the timetabling problem is a truly general method of implementing constraints in a timetabling system. This is due to the fact that the number of violations and constraints involved in real problems is rather large and sometimes very difficult to manage, posses obstacles both in how to formulate the problem and how to approach the problem for its solution (Marin, 1998). From the above discussion we believed that constraints management plays a vital role to produce approaches that can operate at higher level of generality and be problem independent. One clear solution suggested by (Deris *et al.*, 1999) is to allow standard GA to play its role as a tool to generate promising solution and let constraints propagation techniques handle the constraints. This technique follows the concept of GA as general optimization tools because it does not involve modification of standard GA procedure, thus it is generic and problem independent. Ross *et al.* (1998) have done extensive studies to discover limitation on GA based approach and realized that GA is worthwhile direction for timetabling research. Marin (1998) added that hybridization of GA with other techniques and strategies from Constraints Satisfaction Problems provide an important direction for investigation. A review made recently by (Qu, 2002) shows that hybridized methods often perform better than individual approaches as they are benefited from the advantages of both techniques with careful design. More recently Burke *et al.* (2002) reported that the literature on timetabling and evolutionary algorithms suggest that a combination of heuristics is a promising research direction. With wide acceptance, the research on GA approaches should continue to produce methods that can operate at higher levels of generality. To achieve these goals, the methods must be widely accepted, easy to understand, adopt and implement in whatever situation.

3.0 GENETIC ALGORITHM (GA)

Genetic Algorithm is a stochastic search algorithm and a general-purpose optimization method based on Darwin Theory of evolution. GA operates on a population of solutions represented by some encoding. Each individual in the population is known as chromosomes that represent a set of solutions. New solutions were obtained by combining different chromosomes to produce new better offspring or by altering an existing member of the population (mutation). A series of evolution then takes place by first evaluating the fitness of each chromosome (individual) and then selecting the fittest to survive to the next generation.

Simple GA itself is difficult to solve the timetabling problem. Due to these factors most of the GA approaches are hybridized with other methods such as Constraint Logic Programming, Modified GA, Non-dominated Sorted GA, etc. A major problem in GA is that simple GA have tendency to converge to local optima. This premature convergence is caused by several algorithmic features, particularly selection pressure and too high gene flow within the population (Ursem, 2002). Population Diversity is undoubtedly a key issue in the performance of GA (Popovic,1997) A common hypothesis is that higher diversity is important to avoid premature convergence and to escape local optima. Numerous methods have been applied to combat premature convergence. Ursem (2002) proposed a measurement to guide diversity search within the population. When the diversity reaches below certain threshold, crossover function stops temporarily and let the mutation operator increase the diversity back to certain level before crossover resumes its function.

This paper will contribute to the discussion of using time group strategy combined with simple GA to solve highly constrained timetabling problem.

4.0 PROPOSED MODEL

In this study we proposed a simple GA (TGGA) to solve highly constrained timetabling problem by assigning events to a group of timeslots. This method used simple GA to find feasible and near optimal solution for university course timetabling. Weekly slots were divided into 16 groups of timeslots. Each of it consists of three-hour timeslots to occupy three-hour slots for each course. The competition within the subject to occupy the slot no longer exists because the combination of timeslots automatically reduce the number of constraints up to two third of the overall constraints. The other benefit is that the name of the lecturer for the subject will not appear more than once for a day. Thus, by default we have satisfied hard constraint h1. In common form of representation, each subject was separated by number of teaching hours per week. For instance, subject A consists of three units slot, thus the representation in GA chromosomes took three genes instead of one by grouping. Time group technique is not new, in (Dimopoulou and Miliotois, 2001) the time group was applied with integer programming to solve timetabling problem with a great success. This method reduces the size of array and therefore increases the processing speed and reduces the usage of computer resources. For example, if we have fifty subjects, the size of array is one hundred and fifty (50*3=150) without grouping the timeslot. In addition the competition within subjects to occupy reasonable slot without violating the hard constraints need also to be taken into account.

The proposed Simple Genetic Algorithm (TGGA) is as follows.

```
Procedure Simple_GA
          Begin
             Generate random population at g1
              Evaluate population(g1)
              Repeat
              Begin
                  g1 = g1 + 1;
                   crossover population at (G1);
                   mutation  at G1;
                   new population.
              end
          end
```

Fig. 1: Simple genetic algorithm

4.1 Chromosomes Structure

The direct representation for the structure of the chromosomes was used. Each gene holds compact information to the solution. Each numeric integer represents the information about venue, timeslot, lecturer, subject, session and room as illustrated in Figure 2.

The sizes of chromosomes depend on the number of subjects taught for the semester. For example, if we have 63 subjects, we need 64 slots (16*4 = 64). Sixteen (16) is the total number of combination timeslots for our case. The reason to choose sixteen combinations is because in our, case almost ninety percent of the subjects offered are three-hour lectures. Therefore the entire slot available is divided into three and hence sixteen combination slots were

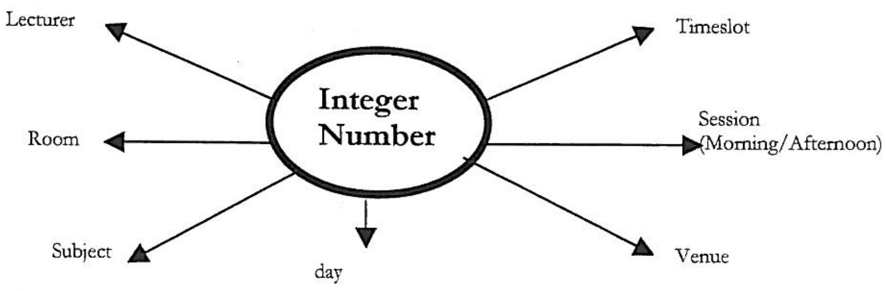

Fig. 2: Genes structure

obtained. The timeslot grouping is the concept of combining several slots into one to reduce the size of the problem. This implies that at least 4 rooms are required to occupy all 64 subjects. If the number of rooms is more than enough (for example in this case more than 4), the size of chromosomes (array) is (16*number of available rooms).

The chromosome represents all available rooms and timeslots. At the initial stage, random numbers were generated to represent the solution. Each number consists of compact information about the schedule as shown in Figure 2. Simple heuristic was employed to ensure that there is not more than one equal number to appear in each chromosome. Each number is only allowed to appear once in the chromosomes. Figure 3 shows chromosomes structure and description about information represented by each integer. For instance, number 4 in slot 2 of room1 represents that lecture-subject code 4 was scheduled in room 1 at the time slot Monday (9-10), Wednesday (9-10) and Friday (9-10).

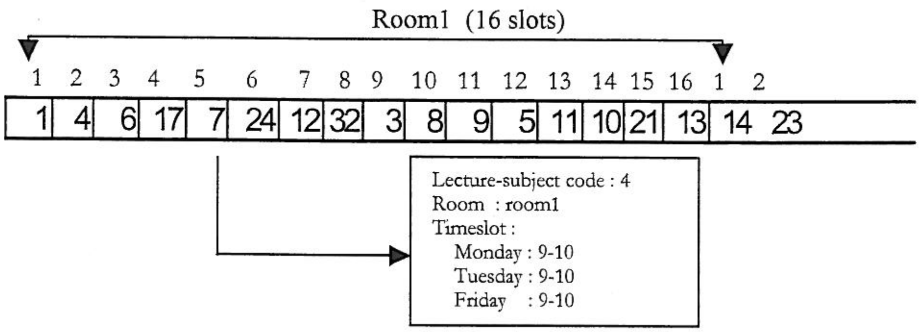

Fig. 3: Chromosomes Structure

4.2 Objective Function

The purpose of the objective function is to guide searching the solution space by reducing constraints violation during the evolution. The number of each constraint violated was determined by the value of penalty function. For this purpose, simple objective function was used to measure constraint violated for each generation. The objective is to reduce constraint violated by maximizing the cost function. Weights are assigned for each constraints violated. There is one type of hard constraint which remains to be solved here, which is h2. Time group strategy has solved the hard constraint h1.

Cost function (f) = maximum_cost - penalty_function(O)

where

Penalty_function (O) = $\sum sgv + \sum rne$ *= number of constrainets violated*

maximum_cost = Estimation of maximum total cost if all constraints are violated

sgv = total cost for subject group being violated

rne = total cost for room capacity being violated.

5.0 TIMETABLING PROBLEM AT FSKTM

Course timetabling problems involve assigning events to timeslots and venues that satisfy all hard constraints and minimize as much soft constraint. These kinds of problems mostly exist in educational institutions such as school and university. Timetabling is known to be a difficult real world problem and it was classified as NP Complete due to the associated constraint. An effective way to solve this type of problem is by employing an optimization technique such as metaheuristic.

The Faculty of Computer Science and Information Technology (FSKTM) at Universiti Putra Malaysia (UPM) consists of four departments in which each of the departments has its specialization area. The weekly teaching hours are from 8.00 am to 7.00 pm each day from Monday to Friday. The duration slot for each class is 50 minutes before the next class begins. Ninety percent of the subjects offered are three hour lectures per week. Each lecturer will be assigned to a maximum of two different subjects for every semester except for those who are holding management duties such as Dean and Heads of Departments. Each undergraduate program is of a four-year duration. The first year students will follow almost the same subjects even if they are undertaking different programs. Classes with a huge number of students will be divided into several groups. Each group will be taught by a different lecturer and can be run simultaneously. The faculty has 2 large lecture halls and 3 medium sized rooms for the classes. Previously, the administrative officer did the timetabling planning manually. The scheduling of the next semester subjects begins during the previous semester once the faculty curriculum committee agrees on the subjects to be offered for the coming semester. The process is time consuming and tedious especially when changes occur frequently. Figure 4 shows the weekly timeslot for FSKTM and the timeslot combination based on the proposed model. Each number represents one combination of timeslot. For example, number 4 is a combination of timeslots consisting of Tuesday (8-9), Thursday (10-11) and Friday (11-12). The designed time group then will be converted to chromosomes structures.

Timeslot	8-9	9-10	10-11	11-12	12-1	1-2	2-3	3-4	4-5	5-6	6-7
Monday	1	2	3	5	8	▨	9	10	11	13	16
Tuesday	4	5	6	8	7	▨	12	13	14	16	15
Wednesday	1	2	3	6	7	▨	9	10	11	14	15
Thursday	5	6	4	7	8	▨	13	14	12	15	16
Friday	1	2	3	4	▨	▨	▨	9	10	11	12

▨ : Free slots

Fig. 4: Weekly timeslot and 16 combinations

The hard constraint at any cost must be satisfied. Typical hard constraints are:

h1 : No lecturers or students or rooms appear more than once at the same timeslot.

h2 : Subjects followed by the same group of students should not be scheduled at the same time

The soft constraint on the other hand should be minimized; failure to achieve this objective will make the timetable not feasible. Typical soft constraints are:

s1: Room capacity is enough to accommodate the number of students for a particular subject.

s2: The same subject, taught by different lecturer for different group of students can be assigned the same timeslot at the different room.

The aims of optimization algorithm are to obtain a feasible and near optimal solution that satisfies all these constraints.

6.0 DEVELOPMENT

A simple system has been developed to test the proposed model. GNU C programming was used as the development tool. The data was taken from the previous session (May-2002) of the Faculty of Information System and Computer Science UPM (FSKTM). Figure 5 shows the system architecture for the proposed algorithm.

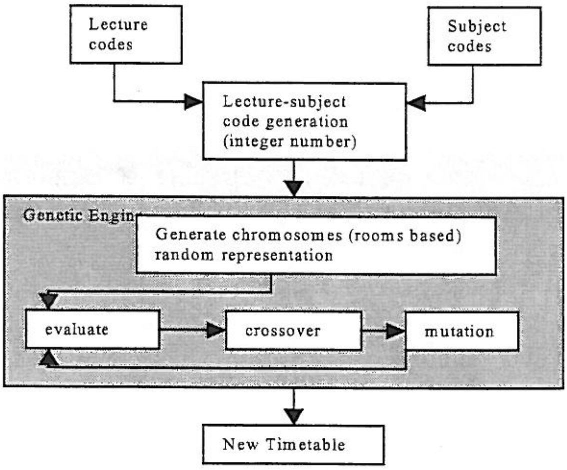
Lecture codes
Subject codes
Lecture-subject code generation (integer number)
Genetic Engin
Generate chromosomes (rooms based) random representation
evaluate
crossover
mutation
New Timetable

Fig. 5: System architecture

7.0 RESULT AND DISCUSSION

A series of experiments and comparative studies were conducted between TGGA with the GA without time grouping (CGA). Standard GA parameters were given for each series of experiments. Detail description of GA parameters is given in Table 1.

Table 1: Genetic algorithm parameters

Items	Parameters
Size of population	40
Number of Generation	100
Crossover Rate	0.7
Mutation Rate	0.3

Table 2 shows the result from the 20 trial runs made for each method. Each value in the table is a number of generations converged during the process. Here it is assumed that if the generations fail to converge before 1000 generation, this means that the generation was trapped in premature convergence problems and cannot escape from local optima problem.

Table 2: Data from test run

Trials	CGA		TGGA	
	Generation	Constraint	Generation	Constraint
1	1000	12	1000	2
2	1000	18	1000	2
3	1000	6	63	0
4	1000	12	1000	1
5	1000	8	1000	2
6	1000	12	1000	3
7	1000	14	1000	1
8	1000	10	97	0
9	1000	13	1000	2
10	1000	14	1000	2
11	1000	16	1000	2
12	1000	8	710	0
13	1000	10	1000	2
14	1000	8	1000	2
15	1000	8	1000	1
16	1000	10	1000	2
17	1000	14	1000	1
18	1000	15	1000	2
19	1000	16	1000	2
20	1000	8	1000	1

The graph shown in Figure 6 indicates that TGGA can sometimes reach an optimal solution compared with CGA. Even though most of the time TGGA failed to get the optimal solution, the result after 1000 generation shows that the number of violated constrained is not more than two for all cases. This implies that those time groups perform better in terms of handling the constraints. From the processing speed, CGA took three times longer compared with others due to the array size that is three times greater than TGGA.

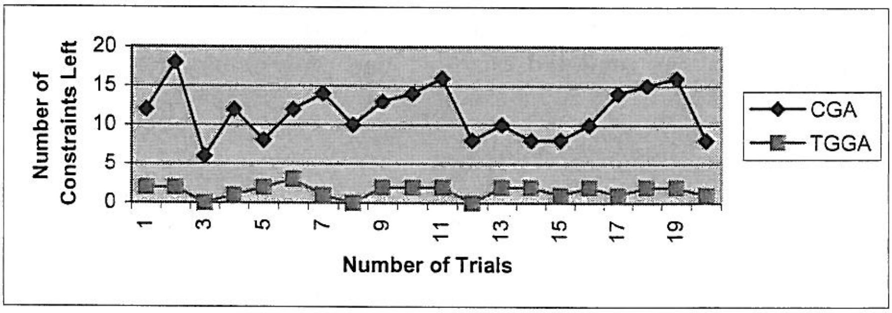

Fig. 6: Graph for constrained violated

From the quality perspective, especially the dispersion of subjects throughout a week, TGGA outperforms CGA. The reason behind this is because the structure of weekly schedule has been properly designed in advance before the GA seeks for the feasible solution. The sequence of schedule produced by TGGA is more structured compared to TGGA. Table 3 shows samples of timetable produced for the year 1 class.

The experiments conducted with several other GA parameters provide almost the same result with small differences. Hence, it can be concluded that the proposed model is quite precise.

Table 3: Timetable for year 1

Time slot	8-9	9-10	10-11	11-12	12-1		2-3	3-4	4-5	5-6	6-7
Mon	1001(r1)	1102(r1) 1100(r4)		1102(r4)		B	1003(r1)	1101(r1)			1101(r4)
Tues		1102(r4)	1304(r3)		1100(r2)	R	1107(r5)		1107(r3)	1101(r4)	1103(r2)
Wed	1001(r1)	1102(r1) 1100(r4)		1304(r3)	1100(r2)	E	1003(r1)	1101(r1)		1107(r3)	1103(r2)
Thur	1102(r4)	1304(r3)		1100(r2)		A	1107(r3)		1107(r5)	1103(r2)	1101(r4)
Fri	1001(r1)	1102(r1) 1100(r4)				K		1003(r1)	1101(r1)		1107(r5)

8.0 CONCLUSION

The simple genetic algorithm model for a highly constrained university course timetabling was proposed. Most of the current and previous methods using GA to solve timetabling employed excessive logic programming (repair) to assist GA find the solution. Other approaches used by researchers are by hybridizing GA with other approaches or by modifying the function of GA operator thus making the method problem dependent.

In this paper, the problem involved an assignment for weekly periods to a combination of timeslots of university courses. The three-hour slot was combined into some pattern to form a number of timeslot combinations. Without the grouping techniques it is almost impossible to achieve a feasible result due to highly constrained problem. The combination of timeslot reduces the number of complexities and constraints of the problem. Future work may include investigating an efficient method for satisfying more soft constraints.

Here Simple Genetic Algorithm has shown to have some potential to be implemented in wider range of university course timetabling problems and other scheduling problems with less effort.

REFERENCES

Burke, E. K., & Petrovic, S. (2002). Recent research direction in automated timetabling. *European Journal of Operational Research 140*, 266-280.

Carrasco, M. P., & Pato, M.V. (2001). A multiobjective genetic algorithm for class/teacher timetabling problem. in E.Burke and W.Erben (Ed.): *The International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT) 2000, Lecture Notes for Computer Science 2079*, 3-17.

Carter, M. W., & Laporte, G. (1998). Recent development in practical course timetabling. In: E Burke E.: M. Carter (Ed.) : *The International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT)1997, Lecture Notes for Computer Science 1408*, 3-19.

Daskalaki, S., Birbas, T., & Housos E. (2003). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, Article in Press.

Deris, S., Omatu, S., Ohta, H., & Saad P. (1999). Incorporating constraint propagation in genetic algorithm for university timetabling planning. *Engineering Application of Artificial Intelligence 12*, 241-253.

Dimopoulou. M., & Miliotois, P. (2001). Implementation of a university course and examination timetabling system. *European Journal of Operational Research 130*, 202-213.

Marin, H.T. (1998). Combinations of GAs and CSP strategies for solving the examination Timetabling Problem. *Ph.d Thesis, Department of Computer Science*, University of Monterrey, Mexico.

Newall, J P. (1999). Hybrid Methods for Auto Timetabling. *Ph.D Thesis, Department of Computer Science*, University of Nottingham UK.

Paechter, B., Rankin, R. C., & Cumming, A. (1998). Improving a lecture timetabling System for University. *Lecture Notes for Computer Science 1408*, 156-165.

Popovic, D. (1997). Retaining diversity of search point distribution through a breeder genetic algorithm For Neural Network Learning. *IEEE international Conference on Neural Network 1*, June 9 – 12. (pp. 495-498) , Houston, USA: IEEE Press.

Qu, R. (2002). Case-Based Reasoning for course timetabling problems. *PH.D Thesis, Department of Computer Science*, University of Nottingham UK

Ross, P., Hart, E., & Corne, P., (1998). Some observation about GA-based exam timetabling. in E.Burke, M.Carter (Eds) , *The International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT)1997, Lecture Notes for Computer Science 1408*,115-129.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review 13(2)*, 87-127.

Ueda, H., Ouchi. D., Takashi, K., & Miyahara, H. (2001) A co-evolving timeslots/room assignment genetic algorithm technique for university timetabling. In E.Burke and W.Erben (Ed.), *The International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT) 2000, Lecture Notes for Computer Science 2079*,48-63.

Ursem, U. K. (2002). Diversity-guided evolutionary algorithms. *Lecture Notes for Computer Science 2439*, 462-471.

Yu, E., & Sung, K. (2002). A genetic algorithm for a university weekly courses timetabling. *International Transactions in Operational Research(9)*,703-717.

Wang, Y.Z. (2003). Using Genetic Algorithm Methods to solve Course Scheduling, *Journal of Expert system with Application(22)*, 295-302.