Jurnal Teknologi

Full paper

# A Rectification Strategy In Genetic Algorithms for Academic Timetabling Problem

Chong-Keat, Teoh[a*], Wibowo, Antoni[b], Ngadiman, Salihin[a]

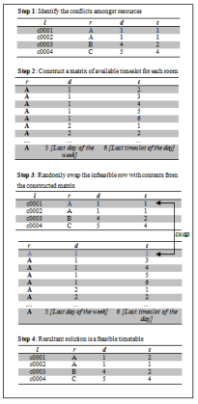
[a]*Dept. of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia*
[b]*Dept. of Decision Sciences, School of Quantitative Sciences, Universiti Utara Malaysia, 06010, Sintok, Kedah, Malaysia*

[*]Corresponding author: ckteoh3@live.utm.my

**Article history**

Received :28 May 2014
Received in revised form :
7 January 2015
Accepted :15 March 2015

**Graphical abstract**



**Abstract**

The university course timetabling problem is both an NP-hard and NP-complete scheduling problem. The nature of the problem concerns with the assignment of lecturers-courses to available teaching space in an academic institution and may take on the form of high school timetabling, examination timetabling or university course timetabling. In this paper, the authors attempt to construct a feasible timetable for a faculty department in a local university in Malaysia which at the present moment; the scheduling task is performed manually by an academic registrar. The feasible timetable is constructed by means of Genetic Algorithm, embedded with a rectification strategy which transforms infeasible timetables into feasible timetables.

*Keywords*: Academic timetabling problem; genetic algorithm; rectification strategy

**Abstrak**

Masalah penjadualan kursus universiti merupakan sebuah masalah penjadualan *NP-hard* dan *NP-complete*. Secara ringkasnya, objektif masalah ini adalah untuk menugaskan pensyarah yang mengajar ke sumber yang sedia ada seperti dewan kuliah. Masalah penjadualan ini juga boleh mengambil bentuk penjadualan sekolah menengah, penjadualan peperiksaan serta penjadualan kursus universiti. Dalam artikel ini, para penyelidik akan membina sebuah jadual waktu yang boleh dilaksanakan untuk sebuah jabatan fakulti universiti tempatan di Malaysia disebabkan tugas penjadualan masih dilakukan secara bertulis oleh pendaftar akademik pada masa kini. Jadual waktu tersebut dibina menggunakan prinsip Algoritma Genetik, yang ditambah dengan strategi pembetulan yang mengubahkan jadual waktu yang silap ke jadual waktu yang boleh dilaksanakan.

*Kata kunci*: Masalah penjadualan akademik; algoritma genetik; strategi pembetulan



## 1.0 INTRODUCTION

The university course timetabling problem (UCTP) is often treated as a NP-hard and NP-complete problem, implying that there is no known polynomial time algorithm which can guarantee the finding of the best solution[1-2]. Basically, the objective of the UCTP is to assign a set of resources which often comprises entities such as lecturers, courses and rooms to available timeslots whilst respecting the stipulated constraints of the institution. Since the problem is typically considered large-sized in nature, possesses a substantial amount of variables and constricted by numerous constraints, it is also referred to as a constraint satisfaction problem[3]. Even till present day, one of the main issues in many institutions is that the scheduling of courses is still performed manually by an academic registrar. Constructing a master timetable can be a tedious, daunting and time consuming process and may be subjected to a high percentage of anomalies.

In a scheduling environment, more often or not there are specific constraints to adhere by. Similarly, in UCTP, there are two types of constraints to adhere by which are hard constraints and soft constraints. The former (hard) constraints are vital and mandatory constraints in which they cannot be violated under any circumstances at all, lest the timetable is rendered infeasible. On the other hand, the latter (soft) constraints, such as the unavailability of lecturers are secondary constraints which can be violated, but preferably not since the satisfaction of these constraints enhances the quality of the timetable. In the literature, the UCTP has been reported to be successfully solved through various metaheuristic algorithms such as Genetic Algorithm[4-7], Ant Colony Optimization[8-9], Particle Swarm Optimization[10-12], Simulated Annealing[13-15] and Tabu Search[16-17] to name a few. The rest of the paper is organized as such: Section 2 presents the model definition, section 3 details the research methodology and section 4 presents the results of the experiments.

## 2.0 MODEL DEFINITION

The model addressed in this paper is adopted from a faculty department in a local university in Malaysia. This section outlines the background and properties of the dataset, the various entities involved in the problem formulation, the constraints which govern the problem and the mathematical formulation of the problem.

### 2.1 Properties of Dataset

In the university's academic system, the intake of fresh undergraduates takes place only once a year, which means that at any time, students are always in their odd (first, third, fifth) or even (second, fourth, sixth) semesters (Note that the industrial training semester is omitted in this model). In this work, we attempt to schedule the courses for students who are in their first, second and third year for five various departments in a faculty in a local university in Malaysia. To model the data in a distinct manner, the dataset is separated into three respective problem instances and consists of 4 entities namely course, day, timeslot and room – The first instance consists of courses and the amount of lectures for year 1 students, the second instance consists of year 1 and year 2 students and the third dataset consists of year 1, year 2 and year 3 students.

**Course (*c*):** The data used in this experiment are the total courses which cater to all students of the faculty, ranging from the 1st – 3rd year. The data structure tailored in such a way that the lecturers are bound together with the course entity, thus ensuring that they do not conflict with one another.

**Day (*d*):** There are 5 working days for the staffs which is from Monday till Friday.

**Timeslot (*t*):** The duration for a timeslot is 50 minutes and there are 9 usable timeslots every day except for Friday, which has only 8 timeslots due to reasons of religious prayers. In total, there are 44 usable timeslots for the entire week from 8.00am until 5.00pm.

**Room (*r*):** There is a total of 17 usable rooms, all fit to be used for lecturing activities, which include computing activities.

The aforementioned entities are then assigned to specific sets of events such that they do not conflict with one another according to the constraints as follow:

- $H_1$ - All lectures belonging to a course must be scheduled to distinct periods. A violation occurs if a lecture is not scheduled or two lectures are scheduled simultaneously (Hard Constraint)
- $H_2$ - Lectures belonging to a course cannot be scheduled at the same room simultaneously. A violation occurs if two lectures are scheduled simultaneously and additional violation constitutes additional violation score (Hard Constraint)
- $H_3$ - Lectures cannot be scheduled in between 1.00pm-1.50pm (period 6) from Monday till Thursday (Hard Constraint)
- $H_4$ - Lectures cannot be scheduled in between 1.00pm-2.50pm (period 6-7) on Friday (Hard Constraint)
- $S_1$ - Lectures should not be scheduled at the last period of the day (Soft Constraint)

The aforementioned constraints delimitate the search space and govern the search direction of the algorithms. The objective is therefore to locate the feasible solution location with the least objective function value.

### 2.2 Model Formulation

The value of the objective function is synonymous to the fitness value of a solution which connotes the quality of the solution. The model formulation in this work is adapted from the previous works reported in the literature.[18] In this paper, the objective function is calculated over all candidate solutions as the summation of violated constraints for all courses multiplied with the weights attached to both the hard and soft constraints respectively. In this paper, the hard constraint weight is denoted as $\alpha$ and given the value 10; soft constraint weight denoted as $\beta$ and given the value 1. The model formulation is described in Equation (1).

$$Min\ G(x) = \alpha * \sum_{c}\sum_{i} f\left(A_i^h\right) + \beta * \sum_{c}\sum_{i} f(A_i^s) \quad (1)$$

s.t.
$\forall$ ($c_i$, $r_i$, $d_i$, $t_i$) and ($c_j$, $d_j$, $t_j$, $r_j$)
$A^{H_2}$ : ($d_i$=$d_j$) and ($t_i$=$t_j$) and ($r_i$=$r_j$)
$A^{H_3}$ : [($d_i$=1) or ($d_i$=2) or ($d_i$=3) or ($d_i$=4)] and ($t_i$=6)
$A^{H_4}$ : ($d_i$=5) and [($t_i$ =6) or ($t_i$ =7)]
$A^{S_1}$ : [($d_i$=1) or ($d_i$=2) or ($d_i$=3) or ($d_i$=4) or ($d_i$=5)] and ($t_i$=9)

where:
$G(x) = G(\alpha,\beta, A_i^h, A_i^s)$ = fitness function value,
$\alpha$ = 1= hard constraint weight,
$\beta$ = 0.5 = soft constraint weight,
$c_i$ = course corresponding to the i$^{th}$ course,
$d_i$ = day corresponding to the i$^{th}$ course,
$r_i$ = room corresponding to the i$^{th}$ course,
$A_i^h$ = hard constraint corresponding to the the i$^{th}$ course,
$A_i^s$ = soft constraint corresponding to the the i$^{th}$ course,
$i$ = 1, 2, ..., $n$.
$n_c$ = number of courses.

Through the representation of the chromosome as the candidate solution, the first constraint ($H_1$) is naturally satisfied and therefore excluded from the mathematical formulation.

## 3.0 SOLUTION METHODOLOGY

This section describes the proposed algorithm in detail to solve the university course timetabling problem. It illustrates the chromosome representation, the effect of the genetic operators toward the search procedure and also the rectification strategy employed to transmute the infeasible solutions into feasible solutions.

### 3.1 Chromosome Representation

In Genetic Algorithm, the chromosomes are the fundamental exploring agents which form the population of the algorithm. A well represented chromosome is essential in aiding the exploration of the search space and might enhance the efficiency of the search space exploration. The chromosome, $x$ in this work consists of a set of resources such as lecture, room, day and timeslot as given in Equation (2).

$$x_i = l_i, r_i, d_i, t_i \ / \ i=1:n_c \quad (2)$$

where $n_c$ is the number of courses

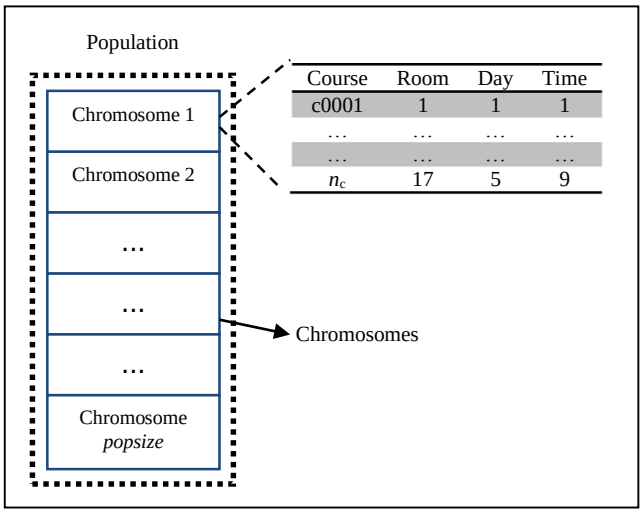



**Figure 1** Population illustration and the encoding of chromosome

A matrix representation of size $var*n_c$ represents the chromosome where *var* is the number of decision variables used and takes the value of 4 in this case. Figure 1 illustrates the representation and encoding of the chromosome.

### 3.2 Crossover and Mutation

The process of taking two best chromosomes (known as parents) and producing an offspring from them is known as crossover. The intention of crossover is to retain the best traits of the parents and to ensure that they are being passed down to the offspring.[19] A single-point crossover which swaps a set of rows amongst the two chromosomes is often used and is employed in this implementation as well. The mutation operator is often regarded as vital as it prevents the algorithm to stagnate in local optima. It functions by randomly selecting a set of rows, usually performed with a few rows to avoid jeopardizing the entire chromosome, and transmutes the entity values with respect to its permissible value and domain. In this case, the entity values represented are the number of courses, rooms, day and timeslots. The crossover rate and mutation rate in this experiment take the value of 0.5 and 0.1 respectively.

### 3.3 Rectification Strategy

Infeasible solutions are commonly referred to as solutions whose resources are in conflict and can be dealt with in three distinct manner: (a) Discard them; (b) Heavily penalize them or (c) Rectify them.[20-21] In this paper, the rectification strategy is employed to transform infeasible candidate solutions into valid solutions. The reason for employing the rectification strategy is to immediately rectify the conflicting resources directly, rather than having the algorithm to explore the entire search space all over again, resulting in a significantly improved computational time. The rectification strategy is outlined in Figure 2.

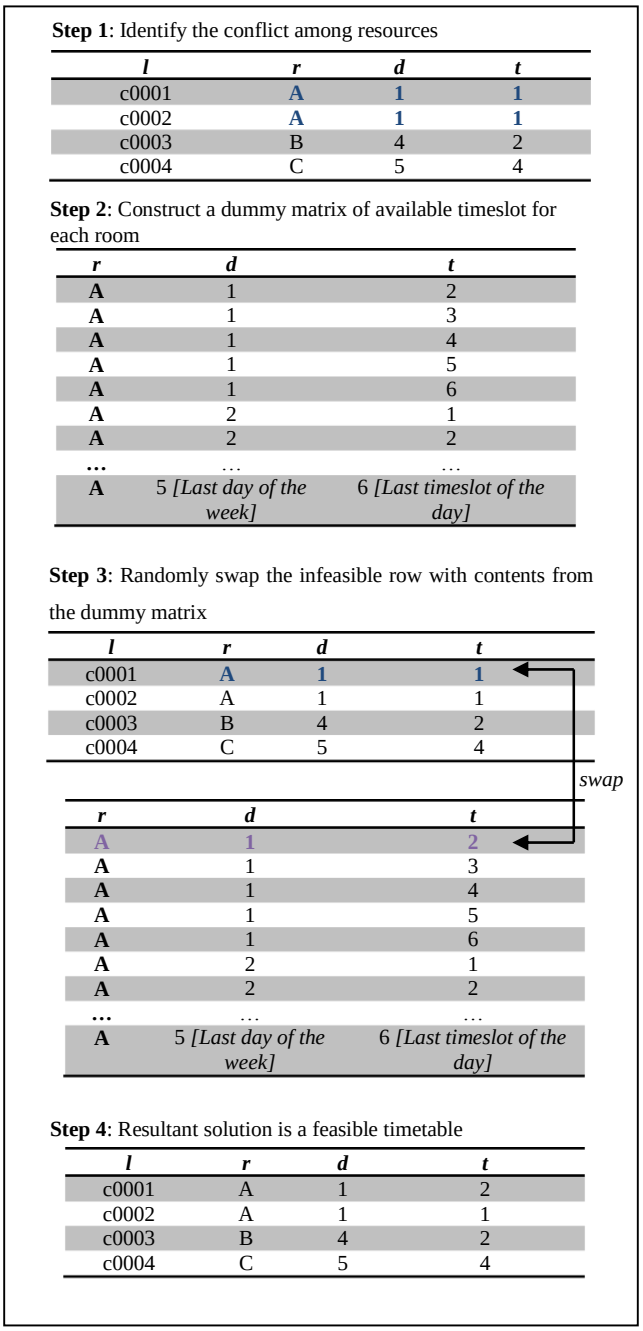



**Figure 2** Rectification strategy for infeasible candidate solutions

In the example illustrated in Figure 2, both courses c0001 and c0002 are assigned to day 1 and period 1 which violates constraint $H_1$. The rectification process is then invoked with the initialization of the construction of a dummy matrix which lists every available period for every available room. Next, the algorithm performs a random swap by simply swapping one of the conflicting rows with any random content of the matrix, resulting in a feasible timetable instantly. In the example, the first row (course c0001) is selected to swap content with the dummy matrix, resulting in a conflict free timetable. At each iteration, the dummy matrix is reconstructed to eliminate the risk of error. The construction of the dummy matrix is crucial as the information it contains ensure that the resulting candidate solution is always conflict-free.

## 4.0 RESULTS AND DISCUSSION

The results of the experiment are presented in this section. The algorithm is set to terminate when the fitness value of the candidate solution reaches zero or when the evaluation time exceeds fifteen minutes (whichever comes first). Table 1 tabulates the results which comprise the fitness scores which are measured in terms of constraint violation, running time which is recorded in seconds and the number of iterations lapsed to reach feasibility. Figure 3, Figure 4 and Figure 5 collectively illustrate the progression of the fitness score to highlight the stark contrast between the two algorithms.

**Table 1** Fitness score and computational time results for the experiment

| Problem Instance | w/o Rectification Strategy | | | With Rectification Strategy | | |
|---|---|---|---|---|---|---|
| | Score | Time(s) | Iteration | Score | Time(s) | Iteration |
| 1 | 0 | 3.43 | 5 | **0** | **2.92** | **2** |
| 2 | 0 | 95.39 | 35 | **0** | **32.08** | **7** |
| 3 | 9 | 900.42 | 172 | **0** | **143.11** | **19** |

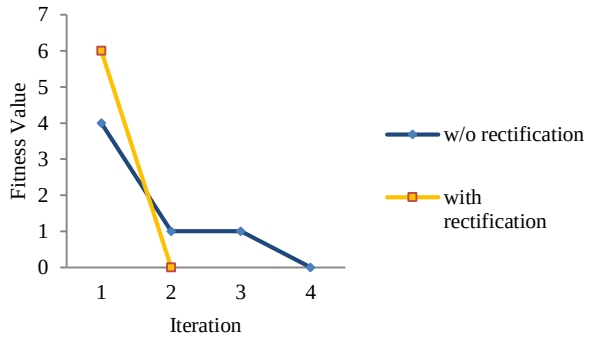



**Figure 3** Fitness progression for the first problem instance

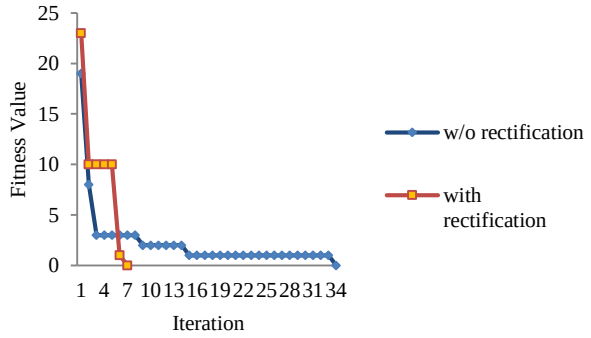



**Figure 4** Fitness progression for the second problem instance


**Figure 5** Fitness progression for the third problem instance

It is evident that the algorithm which features the rectification strategy is superior compared to the one without. In all the tested instances, the rectification strategy outperforms in terms of fitness score, evaluation time and the number of iteration as well. It can be observed that the computational time for the algorithm without the rectification strategy spans two times longer and the number of iterations required to achieve feasibility marks a great difference as compared to the algorithm with the embedded rectification strategy. This demonstrates the superiority of the rectification strategy in solving a highly-constrained problem. It should also be noted that the normal algorithm could not find a feasible timetable for the third problem instance, which attempts to schedule courses for students in their first, second and third year.

## 5.0 CONCLUSION

This paper presents a Genetic Algorithm which features a rectification strategy to solve a university course timetabling problem for a faculty department in a local university in Malaysia. The algorithm is implemented on three distinct problem instances and based on the experimental results; the algorithm which features the rectification strategy outperforms the normal Genetic Algorithm in terms of fitness score and records a significantly reduced computational time and number of iterations. For future works, the authors would consider to implement the algorithm on a more complex and constrained dataset and compare it with other best-known metaheuristic algorithms.

## Acknowledgement

The authors would like to express their gratitude to Universiti Teknologi Malaysia and Ministry of Higher Education (MOHE) Malaysia for the myBrain scholarship and Research University Grant (RUG), number Q.J130000.2528.07H84. In addition, the authors would also like to thank the Research Management Center (RMC) – UTM for supporting this research project.

## References

[1] Bardadym, V. A. 1996. Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science. *Computer-Aided School and University Timetabling: The New Wave*. 1153: 22–45.
[2] Ismayilova, N. a., Sağir, M., & Gasimov, R. N. 2007. A Multiobjective Faculty–Course–Time Slot Assignment Problem with Preferences. *Mathematical and Computer Modelling*. 46(7–8): 1017–1029.
[3] Teoh, C. K., Wibowo, A., & Ngadiman, M. S. 2013. Review of State of the Art for Metaheuristic Techniques in Academic Scheduling Problems. *Artificial Intelligence Review*. doi:10.1007/s10462-013-9399-6.

[4] Agustín-Blas, L. E., Salcedo-Sanz, S., Ortiz-García, E. G., Portilla-Figueras, A., & Pérez-Bellido, Á. M. 2009. A Hybrid Grouping Genetic Algorithm for Assigning Students to Preferred Laboratory Groups. *Expert Systems with Applications*. 36(3, Part 2): 7234–7241.
[5] Jain, A., Jain, S., & Chande, P. K. 2010. Formulation of Genetic Algorithm to Generate Good Quality Course Timetable. 1(3): 248–251.
[6] Kohshori, M., & Abadeh, M. 2012. Hybrid Genetic Algorithms for University Course Timetabling. *International Journal of Computer Science*. 9(2): 446–455.
[7] Suyanto, S. 2010. An Informed Genetic Algorithm for University Course and Student Timetabling Problems. *Proceedings of the 10th international conference on Artificial Intelligence and Soft Computing*. 229–236.
[8] Lutuksin, T., & Pongcharoen, P. 2010. Best-Worst Ant Colony System Parameter Investigation by Using Experimental Design and Analysis for Course Timetabling Problem. *2010 Second International Conference on Computer and Network Technology*. 467–471.
[9] Thepphakorn, T., Pongcharoen, P., & Hicks, C. 2014. An ant colony based timetabling tool. *International Journal of Production Economics*. 149(0): 131–144.
[10] Qarouni-Fard, D., Najafi-Ardabili, A., & Moeinzadeh, M.H. 2007. Finding Feasible Timetables with Particle Swarm Optimization. *Innovations in Information Technology, 2007. IIT '07. 4th International Conference on*. 387–391.
[11] Shiau, D.F. 2011. A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences. *Expert Systems with Applications*. 38(1) : 235–248.
[12] Tassopoulos, I. X., & Beligiannis, G. N. 2012. Solving Effectively the School Timetabling Problem Using Particle Swarm Optimization. *Expert Systems with Applications*. 39(5): 6029–6040.
[13] Aycan, E., & Ayav, T. 2009. Solving the Course Scheduling Problem Using Simulated Annealing. *Advance Computing Conference*. 6–7.
[14] Frausto-solís, J., Alonso-pecina, F., & Mora-vargas, J. 2008. An Efficient Simulated Annealing Algorithm for Feasible Solutions of Course Timetabling. 675–685.
[15] Zhang, D., Liu, Y., M'Hallah, R., & Leung, S. C. H. H. 2010. A Simulated Annealing with a New Neighborhood Structure Based Algorithm for High School Timetabling Problems. *European Journal of Operational Research*. 203(3): 550–558.
[16] Alvarez-Valdes, R., Crespo, E., & Tamarit, J. M. 2002. Design and Implementation of a Course Scheduling System Using Tabu Search. *European Journal of Operational Research*. 137(3): 512–523.
[17] Lü, Z.P., & Hao, J.K. 2010. Adaptive Tabu Search for Course Timetabling. *European Journal of Operational Research*. 200(1) : 235–244.
[18] Othman, M. 2010. Universal Tool for University Course Schedule Using Genetic Algorithm. *International Journal*. 2(6): 1–6.
[19] Sivanandam, S., & Deepa, S. 2007. *Introduction to Genetic Algorithms*. *Springer*. 15–60
[20] Pongcharoen, P., Promtet, W., Yenradee, P., & Hicks, C. 2008. Stochastic Optimisation Timetabling Tool for University Course Scheduling. *International Journal of Production Economics*. 112(2): 903–918.
[21] Blum, C., & Roli, A. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys [CSUR]*. 1–4.