2009 Fourth International Conference on Computer Sciences and Convergence Information Technology

# On Improvement of Effectiveness in Automatic University Timetabling Arrangement with Applied Genetic Algorithm

Pariwat Khonggamnerd
Department of Computer Engineering
Thai-Nichi Institute of Technology
Bangkok, Thailand
pariwat@tni.ac.th

Supachate Innet
Department of Computer and Multimedia Engineering
University of the Thai Chamber of Commerce
Bangkok, Thailand
supachate_inn@utcc.ac.th

***Abstract*— Arranging university course's timetable is problematic. It differs from other timetabling problems in terms of conditions. A complete university timetable must reach several requirements involving students, subjects, lecturers, classes, laboratory's equipments, etc. This paper proposes a genetic algorithm model applied for improving effectiveness of automatic arranging university timetable. Hard constraints and soft constraints for this specific problem were discussed. In addition, the genetic elements were designed and the fitness function was proposed. Three genetic operators: crossover, mutation, and selection were employed. A simulation was conducted to obtain some results. The results show that the proposed GA model works well in arranging a university timetable. With 0.70 crossover rate, there is no hard constraints appeared in the timetable.**

***Keywords- automatic arrangement, genetic algorithm, university timetabling problems***

## I. INTRODUCTION

At present, universities, especially private universities in Thailand open several courses in order to serve students' requirements. Each year, the number of students in each course is changeable, depending on fashion and nation's requirements. Arranging university course's timetable is, therefore, becoming more complicated. University timetabling problems are differing from other timetabling problems in terms of conditions [1] and are also differing from university to university. Several parameters must be considered in order to obtain a satisfied solution. Such parameters include students' groups, subjects, instructors, and class rooms. In addition, time to accomplish this task depends on personal experiences. One who is familiar with this task or has knowledge in course scheduling may propose more effective solutions.

Automatic university course timetable scheduling has been interested by several researchers as it can save a lot of man-hour works. A large number of different schemes has been proposed in the literature for automating effective timetable. These include Genetic algorithm (GA) [2]-[5], Tabu Search [6], Simulated annealing [7], Constraints-based Reasoning [8], and Multi-agent System [9]. [10] reports the design and implementation of a PC-based computer system to help constructing university course-examination timetable. It used an integer programming (IP) model to assign courses to timeslots and rooms. From all, GA has been widely used and it has been implemented for solving university timetabling problem since 1990 [1]. This is because it is a powerful algorithm to find optimized solution and enables to solve problem in element-level conditions. Hence, it is of interest to be employed to solve problems in this paper.

There are actually two different but related problems: courses, and examination timetable. This paper will focus only on course's scheduling. In addition, it aims on improving effectiveness in automatic university timetabling arrangement by means of an applied genetic algorithm, which extended from the work proposed in [5]. The proposed model was tested using real-world data of University of the Thai Chamber of Commerce, a Thailand's private university, involving the weekly scheduling of all courses in 2008 - 2009 curriculum of bachelor degree of School of Engineering of this university.

The paper is organized as follows. It starts with a general discussion about how this research comes from and some previous work related to this problem. Section II provides more discussion about university timetabling problems in details. The proposed model of GA used in this work is explained in Section III. The section provides how the GA elements are designed, what the definition of fitness function used in the simulation is, and which GA operators are employed to solve the problems. In section IV, the results of simulation are included and some discussion of the results are stated. Finally, the conclusion of this work is presented in Section V.

## II. PROBLEM STATEMENTS

To compete with large and famous universities, small universities, where having limits number of classrooms, and instructors, must provide varieties of curriculums and facilities in enabling to meet students' interests. As curriculums are changing every year, arranging a course timetable must be done appropriately semester by semester to cover changed conditions. This is a very tedious and

978-0-7695-3896-9/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCIT.2009.202

laborious task as it is about arrangement of inadequate resources (rooms, humans, and times) to meet never-satisfied human's requirements. A complete timetable must reach as many requirements as possible if not all. The requirements involves in this problem include requirements of student, subject, lecturer, class and laboratory room, period, and time space.

Requirements are usually stated in terms of constraints, which are divided into two types: hard constraints and soft constraints. Hard constraints are unacceptable problems, which need obligatory treatment. They are not allowed to occur at any percentages. Soft constraints, on the other hands, are ones that can be accepted within minimization of frequency, more as representing preferences. They fulfill some users' requirements to maximize the perfectiveness of timetable. Examples of soft constraints include student and lecturer's free times between classes, appropriate student and lecturer's lunch time, etc.

Hard constraints and soft constraints specified in this work are stated as follows:

### A. *Hard Constraints*

- An instructor is not allowed to teach different subjects in the same period of day
- A student must not study more than one subject in the same period of day
- A room is not allowed to be occupied by different subjects in the same period of day
- Number of students in a room must be lower than capacity of that room

### B. *Soft Constraints*

- There should not be more than two periods free between the nearby classes during the day of study
- In each day, period 3 or period 4 should be vacant for students' lunch space
- There should not be more than 4 continuing teaching's periods occupied for instructors in a day
- For a subject that separates period of study into two, period 2 of that subject should not locate in the same day of period 1.

These constraints are employed to define fitness function as will be discussed in Section III.

## III. PROPOSED GA MODEL

This section proposes a genetic algorithm model for improving effectiveness of automatic arranging university timetable. The model is developed from the previous work presented in [5]. More hard constraints on limitation of room are taking into consideration. These include room occupation and room capacity which are stated in section II.

### A. *The Elementary Design and Algorithm Description*

University timetable chromosomes are constructed into a group of elements (E). According to this paper, elements comprise three types of data: subject (S), lecturer (L), and group (G). A set of E will represent as constituent elements of chromosomes. So, E = {S, L, G} will represent the elements of chromosome. Each of S, L, and G has subset of each own. It can be shown as $S = \{S_0, S_1, S_2, S_3, \ldots, S_n\}$, which will represent subjects in that semester. In the same way for a set of lecturer, and group, it will do as $L = \{L_0, L_1, L_2, L_3, \ldots, L_n\}$ and $R = \{R_0, R_1, R_2, R_3, \ldots, R_n\}$ respectively. Note that $\{S_0, L_0, R_0\}$ means unoccupied period.

The timetable's chromosomes as discussed above are then operated by means of the proposed algorithm that is applied from genetic algorithm to obtain a complete timetable. Figure 1 shows proposed algorithm

```
begin
   Define fitness function
   Form prototype of chromosome
   gen ← 0
   // create parents chromosome - P1(gen) and P2(gen)
   Initial chromosome P1(gen), P2(gen)
   while (not terminate condition) do
      begin
         // random a probability for operation
         if (random < crossover probability)
            // do crossover
            C1(gen) = Crossover P1(gen)
            C2(gen) = Crossover P2(gen )
         else
            // do mutation
            C1(gen) = Mutate P1(gen)
            C2(gen) = Mutate P2(gen)
         // examine fitness value of each chromosome
         Examine P1(gen), P2(gen), C1(gen), C2(gen)
         gen ← gen + 1  // next generation of chromosome
         // select two best chromosome to be new parents
         Rank P1(gen), P2(gen), C1(gen), C2(gen)
         P1(gen) = SelectBestRank1
         P2(gen) = SelectBestRank2
      end
end
```

Figure 1. Applied genetic algorithm for solving automatic university timetabling arrangement problems.

### B. *Fitness Function*

The chromosome will be examined by fitness function. This function constructs from hard constraints and soft constraints discussed earlier. It determines how well the chromosome meets requirements. The chromosome that provides the best fitness value will be selected to create appropriate university timetable. Equation (1) is the fitness function defined to solve problems specified in this study.

$$f(gen) = \sum_{i=1}^{4} w_i^{hard} \times cost_i^{hard} + \sum_{i=1}^{4} w_i^{soft} \times cost_i^{soft} \quad (1)$$

*where,*

$$cost_1^{hard} = \sum_{g=1}^{n_g}\sum_{p=1}^{n_p}\sum_{s=0}^{n_s}\sum_{l=0}^{n_l} Lecturer\ Clash\ \left[\left((L_l,S_s),P_p\right).G_g\right]$$

$$cost_2^{hard} = \sum_{p=1}^{n_p}\sum_{r=0}^{n_r}\sum_{s=0}^{n_s}\sum_{g=1}^{n_g} Student\ Clash\ \left[\left((G_g,S_s),R_r\right),P_p\right]$$

$$cost_3^{hard} = \sum_{g=1}^{n_g}\sum_{p=1}^{n_p}\sum_{s=0}^{n_s}\sum_{r=0}^{n_r} Room\ Clash\ \left[\left((R_r,S_s),P_p\right),G_g\right]$$

$$cost_4^{hard} = \sum_{r=0}^{n_r}\sum_{s=0}^{n_s}\sum_{g=1}^{n_g} Room\ Overload\left[(G_g,S_s),|R_r| < n_{std}\right]$$

$$cost_1^{soft} = \sum_{g=1}^{n_g}\sum_{p=1}^{n_p}\sum_{s=0}^{n_s} To\ Much\ Class\ Waiting\ Time\left[(S_s,P_p),G_g\right]$$

$$cost_2^{soft} = \sum_{g=1}^{n_g}\sum_{p=3}^{4}\sum_{s=0}^{n_s} No\ Lunch\ Break\ \left[(S_s,P_p),G_g\right]$$

$$cost_3^{soft} = \sum_{g=1}^{n_g}\sum_{p=1}^{4}\sum_{l=0}^{n_l} To\ Much\ Continuing\ Lecture\left[(L_l,P_p),G_g\right]$$

$$cost_4^{soft} = \sum_{p=1}^{n_p}\sum_{s=0}^{n_s}\sum_{g=1}^{n_g} Same\ Day\ Separated\ Class\left[(G_g,S_s),P_p\right]$$

*where,*

$f(gen)$: $fitness\ function\ of\ gen^{th}\ generation$
$w_i^{hard}$ : $weight\ of\ i^{th}\ hard\ constraint$
$w_i^{soft}$ : $weight\ of\ i^{th}\ soft\ constraint$
$n_g$ : $number\ of\ student\ group$
$n_p$ : $number\ of\ period = 30$
$n_s$ : $number\ of\ subject$
$n_l$ : $number\ of\ lecturer$
$n_{std}$ : $number\ of\ student\ in\ class$
$i, l, p, g, s$ : $any\ integer$

In addition, the value of each function can be explained as seen below:

$Lecturer\ Clash\ \left[\left((L_l,S_s),P_p\right).G_g\right]$
Equal 1 : $\left((L_l,S_s),P_p\right).x\ !=\ \left((L_l,S_s),P_p\right)$.y
Equal 0 : others

$Student\ Clash\ \left[\left((G_g,S_s),R_r\right),P_p\right]$
Equal 1 : $\left((G_g,S_s),R_r\right),x\ !=\ \left((G_g,S_s),R_r\right)$,y
Equal 0 : others

$Room\ Clash\ \left[\left((R_r,S_s),P_p\right),G_g\right]$
Equal 1 : $\left((R_r,S_s),P_p\right),x\ !=\ \left((R_r,S_s),P_p\right),y$
Equal 0 : others

$Room\ Overload\left[(G_g,S_s),|R_r| < n_{std}\right]$
Equal 1 : $(G_g,S_s),|R_r| < n_{std} == true$
Equal 0 : others

$To\ Much\ Class\ Waiting\ Time\left[(S_s,P_p),G_g\right]$
Equal 1 : $(S_s,P_p\%6),G_g\ !=\ (S_0,P_p\%6),G_g$
&& $\left(S_p,P_{p+1}\%6\right),G_g == \left(S_0,P_{p+1}\%6\right),G_g$
&& $\left(S_p,P_{p+2}\%6\right),G_g == \left(S_0,P_{p+2}\%6\right),G_g$
&& $\left(S_p,P_{p+3}\%6\right),G_g == \left(S_0,P_{p+3}\%6\right),G_g$
&& $\left(S_p,P_{p+4}\%6\right),G_g\ !=\ \left(S_0,P_{p+4}\%6\right),G_g$
Equal 0 : others

$No\ Lunch\ Break\ \left[(S_s,P_p),G_g\right]$
Equal 1 : $(S_s,P_3),G_g \neq (S_0,P_3),G_g\ ||\ (S_s,P_4),G_g \neq (S_0,P_4),G_g$
Equal 0 : others

$To\ Much\ Continuing\ Lecture\left[(L_l,P_p),G_g\right]$
Equal 1 : $\left(L_l,P_p\%6\right),G_g == \left(L_l,P_{p+1}\%6\right),G_g$
$== \left(L_l,P_{p+2}\%6\right),G_g == \left(L_l,P_{p+3}\%6\right),G_g$
Equal 0 : others

$Same\ Day\ Separated\ Class\left[(G_g,S_s),P_p\right]$
Equal 1 : $(G_g,S_{s2}),P_p < (G_g,S_{s1}),P_{p+6}$
Equal 0 : others

*where,* $x, y$ : $any\ integer$
$x\ \%\ y$ : $remaining\ of\ x/y$
$S_{s1}$ : $subject\ S\ period\ 1$
$S_{s2}$ : $subject\ S\ period\ 2$

### C. Genetic Operator

Three genetic operators are employed to solve this problem: Crossover, Mutation, and Selection.

- Crossover

Crossover is a method that exchanges element of two parent chromosomes, say Parent 1 and Parent 2. The process of crossover is as follows:

1. Select a number of positions of element in one of the prototype chromosome randomly. These elements will be operated by crossover method. Figure 2 shows that bit 1, 3, 4, and 7 of Parent 1 were randomly chosen, and the value of those bit are E1, empty (E0), E4 and E7 respectively.
2. Duplicate those elements to a child chromosome, say Child 1.
3. Check the neighborhood bit (both sides). If it is the same element, duplicate it to the child chromosome as well. It not, continue step 4.
4. For the other parent chromosome (Parent 2), delete all the elements (not the bit position) of the same value as were selected in Parent 1. From Fig. 2, E1, E2, and E7 were deleted.
5. Put all the residual element of Parent 2 into the next space of the Child 1. For two-bit elements, stick them together. If the space is not enough to put elements, take the next available space. In Fig. 2, E6 occupies two bits. Therefore, they can not fit in bit 6 of Child 1. They have to be copied in bit 8 and 9.
6. Use the same principles to the second chromosome of parent (Parent 2) to yield the chromosome of the other child (Child 2).

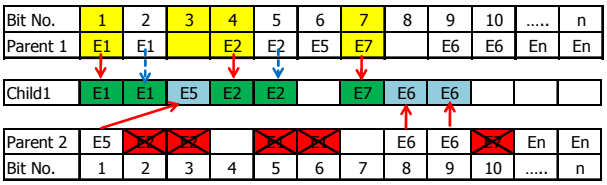


Figure 2. Example of Crossover Process.

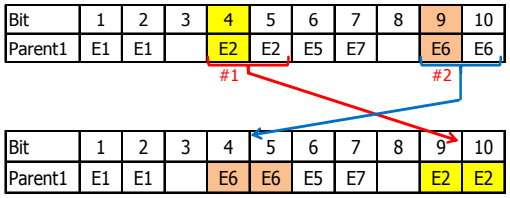


Figure 3. Example of Mutation Process.

- Mutation

The process of chromosome mutation was done by altering the elements in the chromosome to create a new chromosome. The process of mutation can be explained in detail below.

1. Select two positions of element in one of the prototype chromosome randomly. In Fig. 3, bit 4 and bit 9 of Parent 1 were randomly chosen as an example.
2. Check the neighborhood bit (both sides) of both selected position. If they contain an element that occupies the same number of bits, alter their position. Both of bit 4 and 5 are group and alter with the elements in bit 9 and 10. All the residue elements in Parent 1 are no change.
3. If not, reselect the position and redo step 2.

- Selection

Selection is the method to choose required chromosome from the mating pool. The chromosomes that are selected will be the parents' chromosome of the next generation. In this paper, two chromosomes that have best (or lowest) fitness value are.

## IV. Simulation and Results

The problems considered in this paper is specific to real-world courses from University of the Thai Chamber of Commerce, a Thailand's private university, and involves the weekly scheduling of all courses in 2008 - 2009 curriculum of bachelor degree of school of Engineering of this university. The specifications of this problem are shown in Table I. Table II shows the value of weight for each constraint that is used for obtaining the simulated results. The values of weight do not significant in terms of quantity, but they represent each constraint. They are used for differentiating; and hence, counting the quantity of constraints appeared in the results.

TABLE I. Scheduling Problem Specification

| No | Description | Quantity |
|---|---|---|
| 1 | Number of student's groups | 28 |
| 2 | Number of class rooms | 4 |
| 3 | Number of computer rooms | 2 |
| 4 | Number of teaching days in a week | 5 |
| 5 | Number of time-period within a day | 6 |
| 6 | Number of subjects | 55 |
| 7 | Number of instructor | 23 |

TABLE II. Hard Constraints and Soft Constraints Weighting

| Weight | Value |
|---|---|
| $w_1^{hard}$ | 10000 |
| $w_2^{hard}$ | 10000 |
| $w_3^{hard}$ | 10000 |
| $w_4^{hard}$ | 10000 |
| $w_1^{soft}$ | 1 |
| $w_2^{soft}$ | 1 |
| $w_3^{soft}$ | 1 |
| $w_4^{soft}$ | 1 |

An effect of crossover probability on the fitness value was study by varying it from 0.00, 0.30, 0.40, 0.50, 0.60, 0.65, 0.70, 0.75, and 1.00. Five hundred generations of process were conducted. The fitness values were determined when it was stable. The minimum process generations that provide stable fitness values were recorded. All the results were tabulated in Table III.

From the results shown in Table III, it can be seen that the lower fitness value are occurred when the crossover probability are in range 0.65 to 0.75. Figure 4 shows how the fitness value decreases when the generation increases at three different crossover probabilities of 0.65, 0.70 and 0.75. In addition, the graph shows that crossover probability of 0.70 provides the best result in this specific problem. Therefore, the simulation was repeated at 0.70 crossover probability to average the fitness value. The result is illustrated in Table IV. The result shows that by using 0.70 crossover rate, hard constraints will never occurred, but some soft constraints will be unavoidable. To get better results, some facilities, such as more classrooms, more instructors, or more studying periods may require.

TABLE III. EFFECT OF CROSSOVER PROBABILITY TO FITNESS VALUE

| Crossover Probability | Number of Generations | Fitness Value |
|---|---|---|
| 0.00 | 438 | 110012 |
| 0.30 | 316 | 50016 |
| 0.40 | 368 | 90020 |
| 0.50 | 494 | 60012 |
| 0.60 | 442 | 70014 |
| 0.65 | 477 | 40017 |
| 0.70 | 445 | 34 |
| 0.75 | 478 | 30026 |
| 1.00 | 163 | 220023 |

TABLE IV. AVERAGE NUMBER OF CONSTRAINTS OCCURRED AT 0.70 CROSSOVER RATE.

| No | Generations | Fitness Value | Average Hard Constraints | Average Soft Constraints |
|---|---|---|---|---|
| 1 | 445 | 34 | 0 | 25.1 |
| 2 | 438 | 28 | | |
| 3 | 441 | 26 | | |
| 4 | 458 | 21 | | |
| 5 | 462 | 19 | | |
| 6 | 468 | 17 | | |
| 7 | 444 | 22 | | |
| 8 | 459 | 29 | | |
| 9 | 455 | 28 | | |
| 10 | 442 | 27 | | |

## V. CONCLUSION

This paper presents improvement of effectiveness in automatic university timetabling arrangement by applying genetic algorithm. The algorithm was developed in enabling to obtain more optimized results. The elements of chromosome were redesigned. More hard constraints were included and the fitness function was revised. The results show that the proposed GA model works well in arranging a university timetable. With 0.70 crossover rate, there is no hard constraints appeared in the timetable. However, more facilities may required to be able to remove all the soft constraints.

## REFERENCES


[1] A Colorni, M. Dorigo, and V. Maniezzo, "Genetic algorithms – A new approach to the timetable problem," In Lecture Notes in Computer Science – NATO ASI Series, vol. F no. 82, Combinatorial Optimization, (Akgul et al eds), Springer-Verlag, pp. 235-239, 1990.

[2] E.K. Burke, D.G. Elliman, and R.F. Weare, "A genetic algorithm based university timetabling system," the 2nd East-West International Conference on Computer Technologies in Education, 1994.

[3] E.K. Burke, K.S. Jackson, J.H. Kingston, and R.F. Weare, "Automated timetabling: The state of the art," The Computer Journal, vol. 40 no. 9, pp. 565-571, 1997.

[4] S. Kazarlis, "Solving university timetabling problems using advanced genetic algorithms," 5th International Conference on Technology and Automation, 2005.

[5] S. Innet, N. Nuntasen, "University timetabling using evolutional computation," WSEAS Transaction on Adavnces in Engineering Education, issues 12 vol. 4, pp 243-250, Dec. 2007.

[6] A. Herth, "Tabu search for large scale timetabling problem," European Journal of Operational Research, 1992.

[7] A. Abramson, "Constructing school timetables using simulated annealing sequential and parallel algorithms," Management Science, 1991.

[8] H.S. Fen, I. S. Deris, and S. Z. Hashim, "Investigating constraint-based reasoning for university timetaling problems," International MultiConference of Engineers and Computer Scientists, vol 1, 2009.

[9] M.Dimopoulou, and P. Miliotis, "Theory and methodology: Implementation of a university course and examination timetabling system," Eroupean Journal of Operational Research vole 130, pp. 202-213, 2001.


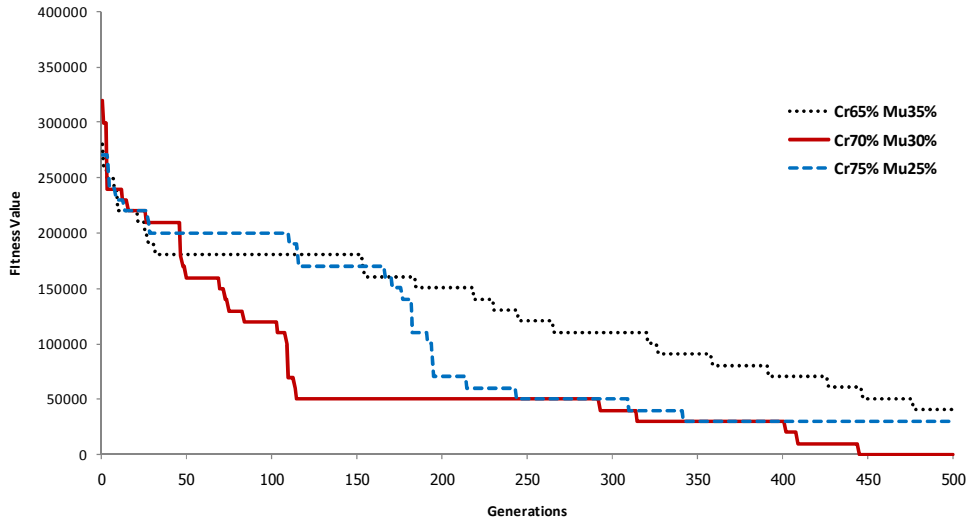


Figure 4. Change of Fitness Value when varying crossover probablility to 0.65, 0.70, and 0.75.