Jurnal Teknologi, 44(D) Jun 2006: 123–130
© Universiti Teknologi Malaysia

MAINTAINING DIVERSITY FOR GENETIC ALGORITHM: A CASE OF TIMETABLING PROBLEM

ABU BAKAR MD. SULTAN[1], RAMLAN MAHMOD[2], MD. NASIR SULAIMAN[3] & MOHD. RIZAM ABU BAKAR[4]

Abstract. Over the last decade, variant of genetic algorithm (GA) approaches have been used to solve various type of optimization problems. Premature convergence is the main problem for GA performance. A common hyphothesis is that high diversity is important to avoid this problem. Failure to maintain GA population diversity will lead to this problem and affected quality of result will be produced. In this paper, we proposed two-problem representation and two strategies to retain population diversity as well as preventing premature convergence. The algorithm was then applied to timetabling problem and showed promising result.

Keyword: Genetic algorithm, premature convergence, diversity, timetabling, optimization

Abstrak. Lebih sedekad, pelbagai pendekatan algoritma genetik (GA) digunakan bagi penyelesaian pelbagai jenis masalah pengoptimuman. Penyelesaian pramatang merupakan masalah utama kepada prestasi GA. Andaian umum mengatakan diversiti tinggi adalah penting untuk mengelakkan masalah ini. Gagal mengekalkan diversiti populasi mengakibatkan masalah ini dan menjejaskan kualiti keputusan yang dihasilkan. Dalam kertas kerja ini, kami menyarankan dua bentuk perwakilan masalah dan dua strategi untuk mengekalkan diversiti populasi, seterusnya menyelesaikan masalah ini. Algoritma ini kemudiannya diaplikasikan kepada masalah penjadualan dan menunjukkan keputusan yang menggalakkan.

Kata kunci: Algoritma genetik, penyelesaian pramatang, diversiti, penjadualan, pengoptimuman

1.0 INTRODUCTION

Genetic algorithms (GA) are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. GA 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 solution. New solutions are obtained by combining different chromosomes to produce new better offspring or by altering existing member of the population (mutation). A series of

[1,2&3] Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
Email: abakar@fsktm.upm.edu.my[1], ramlan@fsktm.upm.edu.my[2], nasir@fsktm.upm.edu.my[3]
[4] Department of Mathematics, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia.
Email: rizam@fsas.upm.edu.my

evolution then takes place by first evaluating the fitness of each chromosome (individual) and selecting the fittest to survive to next generation.

Among many approaches, GA is the most prominent and widely accepted for various types of timetabling problem according to the number of literatures published. It is proven that standard GA itself is difficult to solve optimization problem including timetabling. Due to this factor, most approaches are hybridized. A major problem in GA is that classic GA has the tendency to converge to local optima. This premature convergence is caused by several algorithmic features, particularly selection pressure and too high gene flow between population member [1]. Population diversity is undoubtedly the key issue in the performance of GA [2]. A common hypothesis is that higher diversity is important to avoid premature convergence and escape local optima. Numerous methods have been applied to combat premature convergence. For instance, [3] proposed a measurement to guide diversity search within the population. This technique emphasize on balancing between crossover and mutation under certain threshold values.

This paper discusses the application of GA to timetabling problem with special attention given to the issue of premature convergence. We proposed two strategies, which employed standard GA to prevent the problem.

2.0 TIMETABLING PROBLEM

Timetabling problem involves assigning event to timeslots and venues that satisfy all hard constraints and minimize as much soft constraint. These kinds of problems mostly exist in educational institution such as school and university. Due to difficulties in solving this type of problem, it was classified as NP complete problem. An effective way to solve this type of problem is by employing optimization technique such as metaheuristic.

GA or evolutionary algorithm have been actively explored for timetabling problem. The fact is over past decades, GA and evolutionary approaches have dominated most of the literatures on this issue. Some researchers such as [4 - 6] considered that GA has been utilized as an effective tool for the timetabling solution. More recently, [1] 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 though most of the approaches retained the basic concepts of GA, none of them are similar in term of implementation. The presentations varied from several aspects, starting from the representation of problem encoding, behaviors of genetic operator, and the processing techniques.

3.0 PROPOSED MODEL

In this study, we proposed GA namely SGA to solve highly constrained timetabling problem with two-problem representation and two strategies. The two-problem representations are grouping techniques, and odd-even number representation. The grouping techniques involves assigning event to a group of timeslot. Detail description of this grouping techniques is discussed in Section 4.0 and can also be found in [7]. We also proposed the use of odd-even number representation. The concept behind the odd and even number is to differentiate between morning and afternoon session as well as to increase population diversity. The representation seems like each lecturer is given a pair of numeric numbers that is odd and even. Detail description of odd-even number representation can be found in [8].

The chromosome represents all the rooms and available timeslot. Each gene inside the chromosomes hold a numeric number, either odd or even. For instance, number one may represent event, lecture, room, time, day, and subject. We have employed simple heuristic to each generation to make the chromosomes represent dual form. For each room, the chromosomes will have two types of representation that is morning session (even numeric) followed by afternoon session (odd numeric). Each room is divided into two sections; the first section contains a sequence of even integers; the second section contains a sequence of odd integers. In this case, if we have 5 rooms, this means that we have five sequences of even number and five sequences of odd number for the entire chromosomes. Each room consists of 8 genes representing even (e) number and 8 genes representing odd (o) number. Any free time such as lunch break is not included in the chromosome thus making the size of the problem smaller. Figure 1 illustrates the chromosome structure.

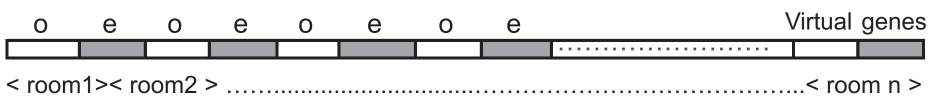

Figure 1 Chromosome structure

3.1 Injection Strategy

To maintain population diversity, we proposed simple injection strategy to the population. Here we used fix point injection, for certain number of generations, we inject new random number to the population to maintain the population diversity. The injection strategy should be carefully designed to avoid overlapping to the genes that have occupied the feasible slot. Simple heuristic will change the random numbers into odd or even number accordingly.

3.2 Sorting Strategy

For every generation, after recombination and mutation, sorting strategy is employed to reschedule the integer into ascending order. Sorting strategy will ensure that reasonable gap duration between two different subjects taught by the same lecturer in the same day are being scheduled in the morning and evening session. Sorting happen within each type of sequence. Figure 2 shows the process of sorting for integer inside the chromosome.

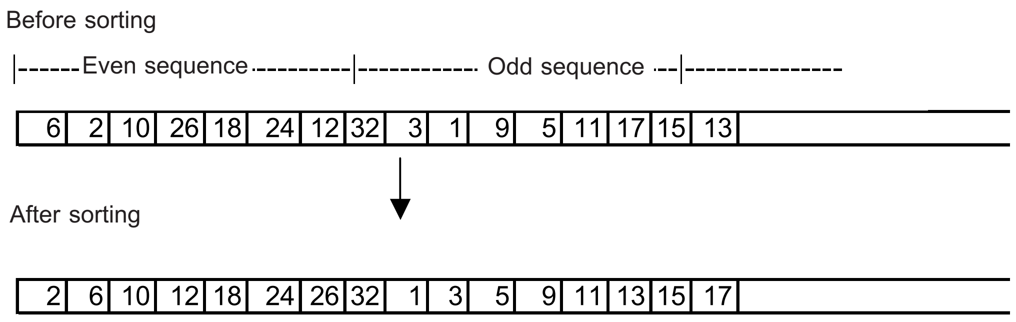

Figure 2 Illustration of sorting process

3.3 Proposed Algorithm

The algorithm below describes general overview about SGA. The algorithm began with random initial population. Initial population then was evaluated to measure the degree of solution quality and selection was made for the next iteration until certain criteria condition was met. For every iteration, each chromosome will undergo four different processes that is crossover, mutation, injection, and sorting. Injection and sorting are two new operators introduced for enhancement of GA performance. Unlike other operators, injection process occurs on certain fix number of generation. This being specified in advanced along with the percentage of injection rate.

```
Procedure SGA
    Begin
      Generate random p(t)
      Evaluate p(t)
      Repeat
      Begin
        t = t+1; n=n+1;
        if n=? inject (%) p(t);
        crossover p(t);
        mutation p(t);
```

```
            sort_accordingly p(t);
            evaluate p(t);
        end
    end
```

4.0 SYSTEM DEVELOPMENT

A prototype system has been developed to experiment our model. GNU C programming was used as the development tool and run under personal computer Pentium III. The data was taken from previous session of the Faculty of Information System and Computer Science, Universiti Putra Malaysia (FSKTM, UPM).

The FSKTM, UPM consists of four departments in which each of the department has its field of 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 hours lecture 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 head of department. Each undergraduate program is four years duration. The first year students will follow almost the same subjects even though they are majoring in different areas. Classes with huge number of students will be divided into several groups. Each group will be taught by different lecturer and can be run simultaneously. The faculty has 2 large lecture halls and 3 medium sizes of rooms for the classes. Previously, the administrative officer did the timetabling planning manually. The scheduling of the next semester subject begins at the previous semester once the faculty curriculum committee agrees the subject to be offered for the next coming semester. The process is tedious and time consuming especially when changes occur frequently. Figure 3 shows the weekly timeslot for FSKTM and the timeslot combination based on our 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).

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

Figure 1 Weekly timeslot and 16 combinations

5.0 RESULT AND DISCUSSION

We have conducted series of experiments and comparative studies between SGA with three other GA method. Each of the methods comprising one additional technique described above to see whether each additional technique contributes significantly to the performance of GA. As the interest of this paper to GA, we are not considering another metaheuristics method for benchmarking. The three GA methods are:

(i) GA + time group (TGGA)
(ii) GA + time group + injection strategy (TGA)
(iii) GA + time group + odd-even representation (RSTGA)
(iv) GA + time group + odd-even representation + injection and sorting (SGA)

Standard GA parameters were given for each series of experiment. Detail description of GA parameters is given in Table 1.

Table 1 Genetic algorithm parameters

Items	**Parameters**
Size of population	40
Number of generation	1000
Crossover rate	0.7
Mutation rate	0.3
Injection rate	10% of pop. size for each 10 generation

Table 2 shows the result from 20 trials run made for each method. Each value in the table is a number of generations converge during the process. The generation values that do not reach maximum generation, meaning that optimal solution is found.

The graph in Figure 4 shows that SGA almost reached 100% success rate for escaping from local optima problem thus making it guaranteed to produce conflict free solutions all the time. The odd-even representation also gave significant effect to population diversity when RSTGA come second in preventing similar problem even without injection strategy. This indicates, odd-even number with injection strategy gives significant impact on preventing premature convergence. Standard genetic algorithm, even with minimum constrained proves to be worsening without any hybridizations or modification. The performance of TGGA and TGA further proves this claim. TGA seems to perform a little bit better when embedded with the injection strategy. This also gives us evidence that injection contributed to maintain population diversity to GA. The time taken to evolve from one generation to the next is about one second and almost similar to all cases.

Table 2 Data from test run

Trials	TGGA	TGA	RSTGA	SGA
1	1000	1000	1000	29
2	1000	1000	1000	12
3	63	1000	493	8
4	1000	1000	292	252
5	1000	1000	188	632
6	1000	1000	401	1000
7	1000	1000	1000	113
8	97	334	1000	35
9	1000	64	378	3
10	1000	1000	1000	1000
11	1000	222	106	203
12	710	187	1000	3
13	1000	1000	1000	7
14	1000	1000	8	1000
15	1000	166	94	12
16	1000	1000	1000	65
17	1000	1000	372	419
18	1000	630	20	707
19	1000	533	1000	265
20	1000	1000	212	372

The odd-even number combined with sorting strategy has shown significant effect to the quality of timetable produced. The result produced by TGGA and TGA just solves a conflict free (hard constraint), further effort is needed to compete SGA and RSTGA in terms of satisfying soft constraints. The experiment proved that with injection and sorting strategies, we could achieve optimal solution frequently.

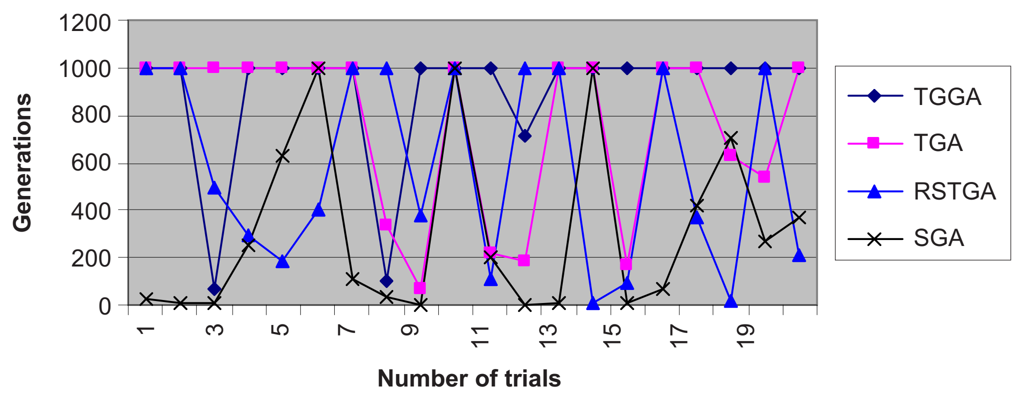

Figure 4 Graph for success rate

The experiments conducted with several other GA parameters almost provide the same result with small differences.

6.0 CONCLUSION

We have proposed standard genetic algorithm with two-problem representation and two strategies to overcome the problem of premature convergence. The experiment shows proper problem representation reduces problem complexities in number.

To ensure the diversity of the population always at higher level before the algorithms converge to global optima, we introduced simple injection strategy and sorting strategy. The introduction of injection techniques gives promises to counter premature convergence. Our future research is to enhance the capability of injection techniques to produce better and flexible injection. The series of experiments made gave us evidences that there are relationships between population size, crossover rate, mutation rate, and the number of generation converges with the rate of random number injected to the population. The intention here is to find more precise mathematical formulation for this technique.

There are many interesting directions for future investigation. For instance, more detailed and thorough analysis for improving our proposed approach should be explored.

ACKNOWLEDGEMENTS

This research is sponsored by grant IRPA-04-02-04-0802-EA001 from Ministry of Science, Technology and Innovation Malaysia (MOSTI).

REFERENCES

[1] Qu, R. 2002. Case-based Reasoning for Course Timetabling Problem. Ph. D. Thesis. Department of Computer Science. University of Nottingham, UK.

[2] Popovic, D. 1997. Retaining Diversity of Search Point Distribution Through a Breeder Genetic Algorithm for Neural Network Learning. IEEE International Conference on Neural Network. Houston, Texas.

[3] Ursem, U. K. 2002. Diversity-guided Evolutionary Algorithms. Lecture Notes in Computer Science. 2439: 462-471.

[4] Burke, E. K., and S. Petrovic. 2002. Recent Research Direction in Automated Timetabling. *European Journal of Operational Research*. 140: 266-280.

[5] Daskalaki, S., T. Birbas, and E. Housos. 2003. An Integer Programming Formulation for a Case Study in University Timetabling. *European Journal of Operational Research*. Article in Press.

[6] Yu, E., and K. Sung. 2002. A Genetic Algorithm for a University Weekly Courses Timetabling. *International Transactions in Operational Research*. 9: 703-717.

[7] Md Sultan, A. B., R. Mahmod, M. N. Sulaiman, and M. R. Abu Bakar. 2004. A Genetic Algorithm Approach for Timetabling Problem: The Time Group Strategy. *Journal of Information Comunication Technology*. 3(2): 1-14.

[8] Md Sultan, A. B., R. Mahmod, M. N. Sulaiman, and M. R. Abu Bakar. 2004. Odd and Even Number Representation to Solve Highly Constrained Course Scheduling Using Genetic Algorithm. *Jurnal Teknologi Maklumat*. 16(2): 21-44.