Penerbit Akademia Baru

Journal of Advanced Research in Computing and Applications

Journal homepage: www.akademiabaru.com/arca.html

ISSN: 2462-1927

Akademia Baru
Advanced Research in Computing and Applications

Scheduling Difficult to Schedule Examination using Memetic Algorithm

Open Access

Naimah Mohd Hussin[1,*]

[1] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 02600 Arau, Perlis, Malaysia

ARTICLE INFO

Article history:
Received 5 March 2018
Received in revised form 4 April 2018
Accepted 12 July 2018
Available online 1 August 2018

ABSTRACT

The main objective in solving an examination timetable problem is to develop a conflict-free timetable where no students are sitting for more than one examination at the same time. UiTM Examination Timetabling problem is a complex problem due to its size and constraints. Part of the solution is to schedule a few selected courses that are difficult to schedule and schedule them using travelling salesman model. A travelling salesman model is implemented where vertices represent examinations while the edge between two vertices represents the number of students sitting for both examination (vertices). Memetic Algorithm was implemented to simulate the shortest path between two given vertices and try to find a reasonable solution. Experiments are performed to determine the performance of the algorithm with respect to its solution quality. The results show that it is able to produce good optimal solution.

Keywords:
Examination timetabling problem, memetic algorithm, travelling salesman

1. Introduction

1.1. Examination Timetabling Problem

Examination timetabling problem is a difficult combinatorial problem that must normally be faced by educational institution up to twice a year. The problem of constructing a conflict-free examination timetable is a difficult task and the problem also varies with respect to the size, structure and constraints and with the importance of each constraint. Therefore, a known solution approach for a problem dataset may not be suitable for another problem instance. At Universiti Teknologi MARA (UiTM), the main examination timetable (containing common subjects) is prepared centrally, and subjects that are non-common are scheduled by the respective faculties[1][2]. Common subjects are subjects taken by more than one program and non-common subjects are subjects that are taken by only one program.

[*] *Corresponding author.*
E-mail address: Naimah Mohd Hussin (naimahmh@perlis.uitm.edu.my)

1.2. Solution Approach

A solution approach to the examination timetabling problem at UiTM has been designed and implemented and can produce a good feasible examination timetable. This paper illustrates the first phase of the solution where a set of difficult to schedule examinations (with large enrollment and high number of examinations in conflict) are scheduled as only one examination per slot. The problem is modelled as a shortest path problem where the vertices represent the examinations (v_1, v_2, v_3, …. v_n) and the weight of its edges, e_{ij} represent the number of students sitting for both examinations *i* and *j* (number of students in conflict). The problem of finding the minimum number of students siting for adjacent slot is the same as the shortest path problem. Shortest path problem is the problem of finding a path between two vertices such that the sum of the weights of its edges is minimized [1]. Shortest path is similar to Travelling Salesman problem (TSP) except that the last vertex visited does not return to the start vertex.

There are several algorithms used to solve this common problem such as Dijkstra's algorithm, Bellman-Ford algorithm, Genetic algorithm (GA) and Heuristic algorithm[3]. Other interesting algorithms that has been applied in optimization problems which is biologically inspired is Artificial Neural Network [4], Genetic Algorithm [5] and Memetic algorithm (MA) [6]. MA represents a 'metaheuristic optimization paradigm' based on the systematic exploitation of knowledge about the problem being solved, and the synergetic combination of ideas taken from other population-based and trajectory-based metaheuristics. In other words, MA is a hybridized technique of genetic algorithm and local search algorithm (LS) to produce required solutions for an optimization problem [6].

In this paper, the solution of the shortest path problem using MA is implemented. The implementation is written in Java programming language. A background and the implementation of the MA will be described and presented in the following sections.

2. Literature Review

Carter and Laporte [7] defined the basic problem in examination timetabling as "the assigning of examinations to a limited number of available time periods in such a way that there are no conflicts or clashes" The conflicts are associated with constraints that should not be violated (hard constraints) or constraints that can be violated if necessary (soft constraints). The timetabling problem is an optimization problem since is objective is to minimize the number of violated constraints or penalties. The objective function is an aggregated weighted penalty where a weighted penalty is associated with each constraint (a higher weight if the constraint is very important).

In most of the literature related to the educational timetabling problem, many researchers propose solutions that have been developed for schools or universities. These proposals were implemented by applying various techniques that were developed for some instance of a real problem. Several surveys on automated timetabling problems have been published that classify timetabling problems and their solution methodologies.

An early survey by Carter [8] presented an overview of practical applications for the examination timetabling problem from as early as 1964 until 1984. The applications were based on graph colouring heuristics designed to solve specific problems in particular schools. There was no integration and comparison between the approaches and as such a novice examination scheduler had little basis for selecting the best approach to work with. Carter and Laporte [7] extended the survey by classifying the algorithms into: cluster methods, sequential methods, generalised search strategies and constraint based approaches. At that point in time, most algorithms solved only the basic timetabling problem with simple constraints. To encourage more advanced research into timetabling problems, the authors made public a set of test problems that was first tested and reported by Carter et al. [9].

Burke et al. [10] stressed the need and importance of an automated timetabling system and presented a very brief overview of some timetabling methods. Schaerf [11] conducted a survey on automated timetabling, categorizing it into: school, course and examination. The paper gives a mathematical description of the basic search and optimization problems, variants of the problems, and solution approaches published in the literature. Burke and Petrovic [12],[13] presented an overview of university timetabling problems and recent approaches in automated timetabling. These methods include hybridizing heuristic methods, memetic algorithms (incorporating hill climbing, decomposition and the diversity of initial populations), multi-criteria approaches, fuzzy methods, case-base reasoning and hyper-heuristic methods.

3. Methodology

A Memetic Algorithm (MA) was developed that applied Genetic Algorithm (GA) and Local Search (LS) together to find the shortest path[2-4]. Figure 1 shows a generic Memetic Algorithm.

```
1. encode solution space
2. (a) set pop_size, max_gen, gen = 0
   (b) set cross_rate, mutate_rate
3. initialize population
4. while (gen < gensize)
       Apply generic GA
       Apply local search
   end while
5. Apply final local search to best chromosome
```

Fig. 1. A generic Memetic Algorithm [5]

3.1 Initialization

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. In this project, the operators of GA are entered by the user. However, there are default values in the program in cases where the user does not want to enter the parameters or the user enters incorrect values.
Parameters that may optionally be change are:

- Population Size (50 by default),
- Number of generation (1000 by default),

- Mutation rate (0.1 by default),
- Crossover rate (0.5 by default).

3.2 Selection

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

3.3 Reproduction

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally, the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

3.4 Termination

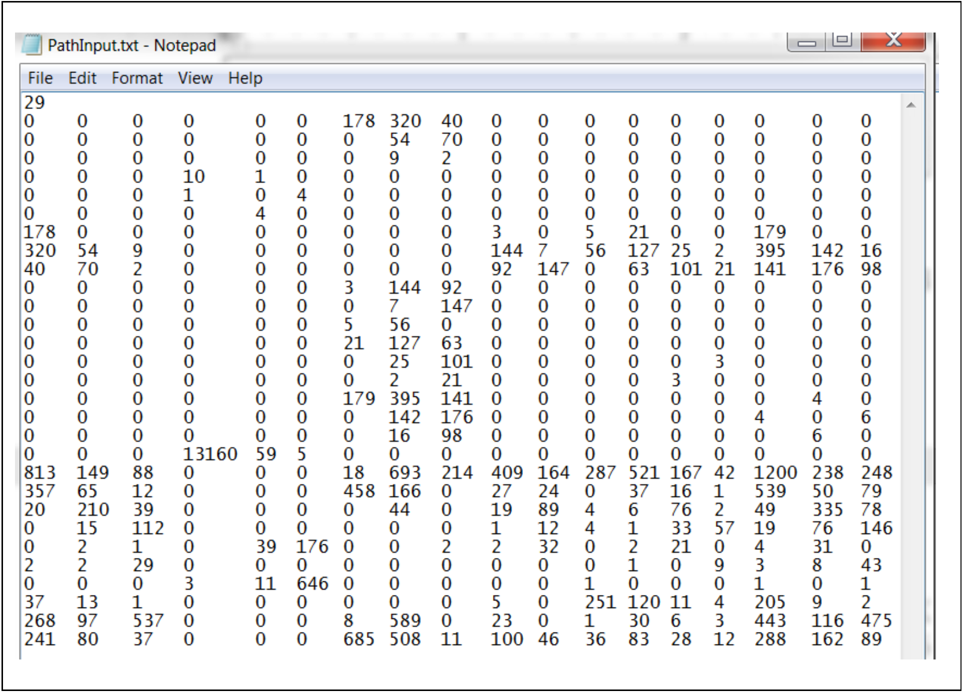
This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria,
- Fixed number of generations reached,
- Allocated budget (computation time) reached,
- The highest ranking solution's fitness is reached or has reached a plateau such that successive iterations no longer produce better results.

4. Results and Discussion

The program executes by first entering the dataset filename. Figure 2 shows the dataset used in this project. The first row indicates the number of vertices or the number of examinations. The following data is the matrix A_{mm} where m is equal to the number of vertices (or number of examinations). An element in the matrix, A_{ij} indicates the distance between vertex i and j or the number of students in conflict between examination i and examination j. Number of students in conflict between examination i and examination j is equal to the number of students sitting for both examination i and examination j.

Next, four parameters need to be entered: (1) Population size (50 by default), (2) Number of generation (1000 by default), (3) Mutation rate (0.1 by default), (4) Crossover rate (0.5 by default). The objective of the algorithm is to produce the shortest path for all the vertices. The shortest path is the list of examinations that are scheduled in slot 1 until the last slot.



```
29
0     0     0     0      0     0     178   320   40    0     0     0     0     0     0     0     0     0
0     0     0     0      0     0     0     54    70    0     0     0     0     0     0     0     0     0
0     0     0     0      0     0     0     9     2     0     0     0     0     0     0     0     0     0
0     0     0     10     1     0     0     0     0     0     0     0     0     0     0     0     0     0
0     0     0     1      0     4     0     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0      4     0     0     0     0     0     0     0     0     0     0     0     0     0
178   0     0     0      0     0     0     0     0     3     0     5     21    0     0     179   0     0
320   54    9     0      0     0     0     0     0     144   7     56    127   25    2     395   142   16
40    70    2     0      0     0     0     0     0     92    147   0     63    101   21    141   176   98
0     0     0     0      0     0     3     144   92    0     0     0     0     0     0     0     0     0
0     0     0     0      0     0     0     7     147   0     0     0     0     0     0     0     0     0
0     0     0     0      0     0     5     56    0     0     0     0     0     0     0     0     0     0
0     0     0     0      0     0     21    127   63    0     0     0     0     0     0     0     0     0
0     0     0     0      0     0     0     25    101   0     0     0     0     0     3     0     0     0
0     0     0     0      0     0     0     2     21    0     0     0     0     3     0     0     0     0
0     0     0     0      0     0     179   395   141   0     0     0     0     0     0     0     4     0
0     0     0     0      0     0     0     142   176   0     0     0     0     0     0     4     0     6
0     0     0     0      0     0     0     16    98    0     0     0     0     0     0     0     6     0
0     0     0     13160  59    5     0     0     0     0     0     0     0     0     0     0     0     0
813   149   88    0      0     0     18    693   214   409   164   287   521   167   42    1200  238   248
357   65    12    0      0     0     458   166   0     27    24    0     37    16    1     539   50    79
20    210   39    0      0     0     0     44    0     19    89    4     6     76    2     49    335   78
0     15    112   0      0     0     0     0     0     1     12    4     1     33    57    19    76    146
0     2     1     0      39    176   0     0     2     2     32    0     2     21    0     4     31    0
2     2     29    0      0     0     0     0     0     0     0     0     1     0     9     3     8     43
0     0     0     3      11    646   0     0     0     0     0     1     0     0     0     1     0     1
37    13    1     0      0     0     0     0     0     5     0     251   120   11    4     205   9     2
268   97    537   0      0     0     8     589   0     23    0     1     30    6     3     443   116   475
241   80    37    0      0     0     685   508   11    100   46    36    83    28    12    288   162   89
```

Fig. 2. Dataset

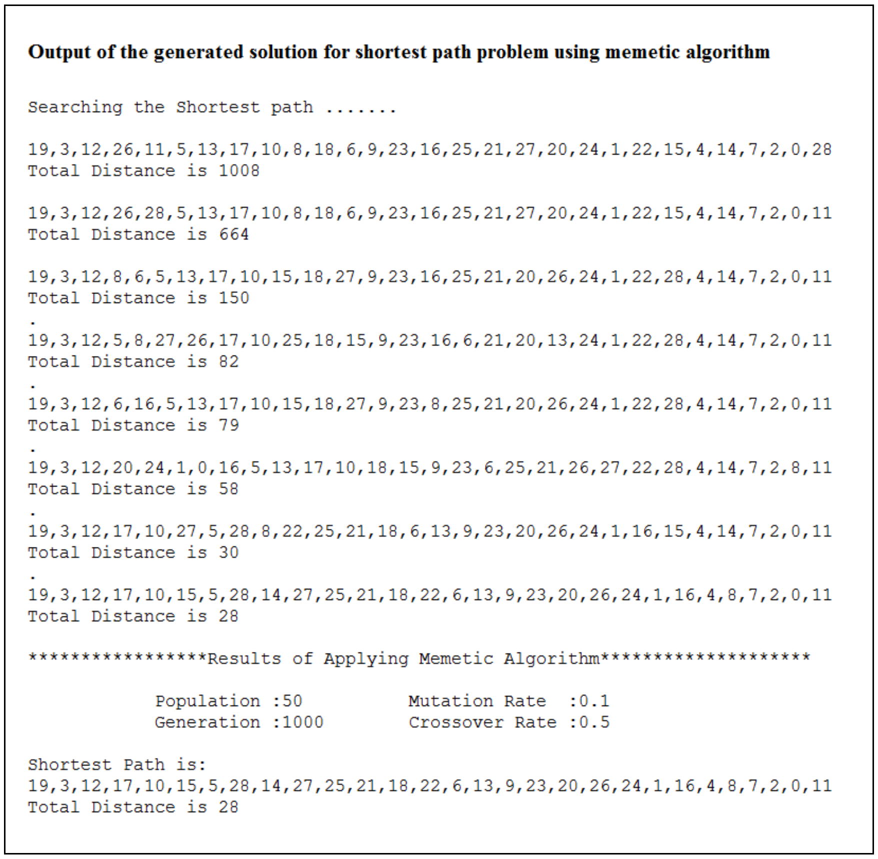
The program applies the memetic algorithm, displays the generated populations and the total distance for each solution. Then, it displays the final results of the shortest path and shows the total distance which is essentially less or equal to the previous solutions.

The sample output appears as shown in Figure 3. Based on the entered parameters, the first solution generated is 1008 while the final solution generated is 28, which indicates the shortest path or the most reasonable found. Figure 3 also shows the different objective values generated for each generation. It illustrates that the objective value for solution decreases as a new generation is produced. In the last few generations, the objective value becomes stable and the final solution is found (value 28). The shortest path value of 28 indicates the number of students who sits for an examination two slots back to back. The shortest path shows the sequence of exams that will be scheduled in slot 1 to slot 29.

Figure 4 shows a graph of the generation runs from the initial solution until the final solution. It starts with a distance 1008 and a rapid improvement of the solution in the first 30 generations and smaller improvement in the next 500 generations and reach a plateau in the next 470 generations. This imply that we can run the algorithm with a lower number of generations and it will not affect the final solution.

4. Conclusion

Memetic algorithm has been implemented in Java language to simulate the shortest path problem and try to find the reasonable solution or the shortest path between two given vertices. The graphical user interface implemented is interactive and flexible where users can change the MA parameters thus producing multiple solutions. The algorithm manages to produce a reasonably good solution where the examinations are sequenced from slot number one to the last slot equivalent to the number of examinations. The objective value of 28 implies that only 28 cases of students having a back to back examination.



Output of the generated solution for shortest path problem using memetic algorithm

```
Searching the Shortest path .......

19,3,12,26,11,5,13,17,10,8,18,6,9,23,16,25,21,27,20,24,1,22,15,4,14,7,2,0,28
Total Distance is 1008

19,3,12,26,28,5,13,17,10,8,18,6,9,23,16,25,21,27,20,24,1,22,15,4,14,7,2,0,11
Total Distance is 664

19,3,12,8,6,5,13,17,10,15,18,27,9,23,16,25,21,20,26,24,1,22,28,4,14,7,2,0,11
Total Distance is 150
.
19,3,12,5,8,27,26,17,10,25,18,15,9,23,16,6,21,20,13,24,1,22,28,4,14,7,2,0,11
Total Distance is 82
.
19,3,12,6,16,5,13,17,10,15,18,27,9,23,8,25,21,20,26,24,1,22,28,4,14,7,2,0,11
Total Distance is 79
.
19,3,12,20,24,1,0,16,5,13,17,10,18,15,9,23,6,25,21,26,27,22,28,4,14,7,2,8,11
Total Distance is 58
.
19,3,12,17,10,27,5,28,8,22,25,21,18,6,13,9,23,20,26,24,1,16,15,4,14,7,2,0,11
Total Distance is 30
.
19,3,12,17,10,15,5,28,14,27,25,21,18,22,6,13,9,23,20,26,24,1,16,4,8,7,2,0,11
Total Distance is 28

****************Results of Applying Memetic Algorithm*******************

            Population :50          Mutation Rate  :0.1
            Generation :1000        Crossover Rate :0.5

Shortest Path is:
19,3,12,17,10,15,5,28,14,27,25,21,18,22,6,13,9,23,20,26,24,1,16,4,8,7,2,0,11
Total Distance is 28
```

Fig. 3. Sample output

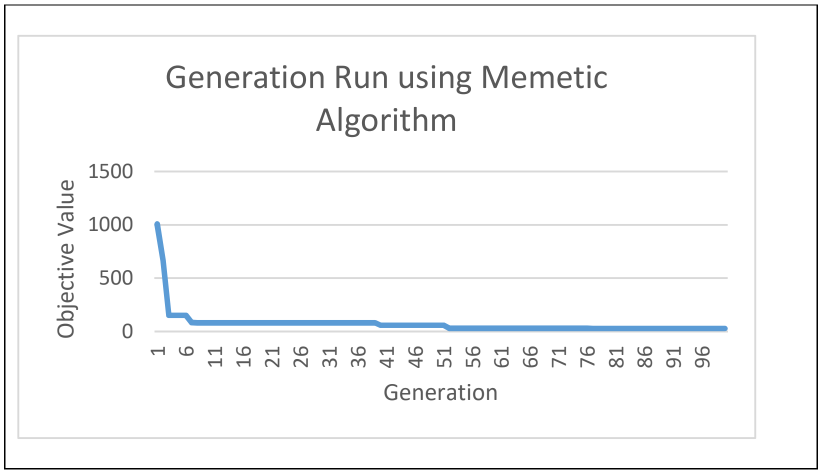


Fig. 4. Generation Run Using Memetic Algorithm

References

[1] Kendall, G. and Mohd Hussin, N., A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA University of Technology. In: Practice and Theory of Automated Timetabling, E. K. Burke and M. Trick (Eds), Lecture Notes in Computer Science, Volume 3616, Springer-Verlag, 2005, 270-293.

[2] Cowling, P., Kendall, G. and Mohd Hussin, N., A survey and case study of practical examination timetabling problems. Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), Gent, 2002, 258-261.

[3] Cherkassky, B.V., Goldberg, A.V. and Radzik, T. (1996), Shortest paths algorithms: Theory and experimental evaluation. Mathematical Programming 73, 129-174.

[4] Parveen, R., Nabi, M., Memon, F.A., Zaman, S. and Ali, M., A review and survey of artificial neural network in medical science, Journal of Advanced Research in Computing and Applications, Vol 3, No 1, 2016, 7-16.

[5] Larrañaga, P., Kuijpers, C. M. H. , Murga, R. H. , Inza, I. & Dizdarevic, S. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review, Kluwer Academic Publishers, Vol. 13(2), 1999, pp. 129-170.

[6] Moscato, P. A., Gentle Introduction to Memetic Algorithms. International Series in Operations Research and Management Science, New York: Springer, Vol. 57, 2007, pp. 105-144.

[7] Carter, M. W. and Laporte, G. (1996) Recent developments in practical examination timetabling. In: The Practice and Theory of Automated Timetabling, E. K. Burke and P. Ross (Eds), Lecture Notes in Computer Science Vol. 1153, Springer-Verlag, pp. 3-21.

[8] Carter, M. W. (1986) A survey of practical applications of examination timetabling algorithms. Operations Research Society of America, Volume 34 No 2, pp. 193-202.

[9] Carter, M. W., Laporte, G. and Lee, S. Y. (1996) Examination timetabling: algorithmic strategies and applications. Journal of the Operational Research Society Volume 47 Issue 3, pp. 373-383.

[10] 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

[11] Schaerf, A. (1999) A survey of automated timetabling. Artificial Intelligence Review. n.13, pp. 87-127.

[12] Petrovic, S. and Burke, E. K. (2004) University timetabling. In: The Handbook of Scheduling: Algorithms, Models, and Performance Analysis. J. Leung (Ed.)

[13] Burke, E. K. and Petrovic, S. (2002) Recent research directions in automated timetabling. European Journal of Operational Research 140: pp. 266–280.

[14] Cotta, C. & Fernandez, J.A. Memetic Algorithms in Planning, Scheduling, and Timetabling. Studies in Computational Intelligence, Berlin: Springer, Vol. 49, (2007):1-30.

[15] Andreatta, A.A., Carvalho, S.E.R. & Ribeiro, C.C. An object-oriented framework for local search heuristics. Proceedings Technology of Object-Oriented Languages, (1998): 33–45.

[16] Poonam Garg, A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard Algorithm, International Journal of Network Security & Its Applications (IJNSA), Vol.1, No 1, April 2009