International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 11 – No. 3, August 2016 – www.ijais.org

An Application of Genetic Algorithm for University Course Timetabling Problem

Sanjay R. Sutar
Asso.Professor, Dr. B. A. T. University, Lonere &
Research Scholar, SGGSIET, Nanded, India

Rajan S. Bichkar
Professor, E&TC and Dean R&D,
G.H.Raisoni College of Engg. & Mgt.,
Pune, India

ABSTRACT
Timetabling problem is a process of assigning given set of events and resources to the limited space and time under hard constraints which are rigidly enforced and soft constraints which are satisfied as nearly as possible. As a kind of timetabling problems, University course timetabling is a very important administrative activity for a wide variety of institutes. Genetic algorithm is an advanced heuristics method which is very effective in many areas. It is frequently deployed meta-heuristics algorithm to solve difficult combinatorial optimization problems. In this paper, genetic algorithm is used to solve university course timetabling problem. At first, a model of problem to be solved is defined. Then, the genetic representation is determined and a fitness function is established according to the constraints. Finally, a case of university course timetabling from real-world is discussed and solved. It is demonstrated that the method proposed in this paper is feasible and efficient.

Keywords
Timetabling problems, Genetic algorithm, Optimization, Heuristic method

1. INTRODUCTION
The course timetabling is one of the major administrative activities for a wide variety of institutions. A scheduling problem can be defined as the problem of assigning a number of events into a number of time periods or time slots. The standard definition is "Timetabling is the allocation, subject to constraints, of given objects in time space, in such a way to satisfy a set of desirable objectives".

Real timetabling problems have many forms like educational timetabling (course and exam), employee timetabling, sports timetabling, timetabling of transportation means, etc. Timetabling problems are scheduling problems, which are computationally complex, constrained optimization problems. They are classified as constraint satisfaction problems, where the main objective is to assure all problem constraints, rather than optimizing a number of objectives. Automated timetabling is of great importance to institutions and organizations, as it can save a lot of man-hours and provide optimal solutions with constraint satisfaction within minutes that can boost productivity, quality of education, quality of services and finally quality of life. High quality large-scale timetables, such as University course timetabling may need great efforts and many hours of work spent by a qualified person or a team. Number of methods had been already proposed in the literature for solving timetabling problems.

In this paper, a genetic algorithm (GA) is proposed to solve a real world university course timetabling problem. It has been used to solve such problems; however, there is no universal timetabling model which can be applied everywhere. The constraints in different universities and academic institutions may vary depending on the respective schemes. In this research work, the model of university course timetabling problem is defined. Then a process of using genetic algorithm to solve this problem is described. A genetic representation is determined to reflect the relationship of teachers, courses, classes, classrooms and time periods and a fitness function is established according to the constraints. Finally, a university course timetabling problem from real-world is solved and analyzed.

2. RELATED WORK
There have been many attempts to solve the course timetabling problem, some examples can be found in [1] [2]. Timetabling is NP-hard problem in a number of ways, as shown in [3]. The potential of the genetic algorithm (GA) in solving highly constrained problems is found in [4]. Hitoshi Kanoh proposed both knowledge and constraints based method to solve the university course timetabling problem efficiently [5]. Alexander Brownlee investigated the problem of class timetabling and attempted to reproduce three different approaches to solve it [6]. An extension of genetic algorithm known as memetic algorithm is also applied to the problem. Maciej Norberciak describes a universal method for solving large, highly constrained timetabling problems from different domains [7]. The solution is based on evolutionary algorithm's framework and employs tabu search to speed up the solution finding process. Hyperheuristics are used to establish operating parameters of the algorithm. The method has been used to solve three different timetabling problems with promising results from preliminary experiments. Results look appealing but it needs improvement in the algorithm such as employing some form of local search.

Mihaela Oprea presented the current state of a research work that involves the development of a multiagent system for University course timetable scheduling [8]. The purpose of the work was to analyze the benefits of using an agent-based approach for the University course timetable scheduling, which involves many communication, cooperation and negotiation processes. He described an architecture of a multi-agent system for University course timetable scheduling, MAS_UP-UCT and briefly discussed an evaluation of the multi-agent system.

Adilah Binti Abdullah provided a review of the current manual timetable systems and developed a web based

timetable system using genetic algorithm [9]. Pariwat Khonggamnerd proposed a genetic algorithm model for improving effectiveness of automated University timetable [10].Hard constraints and soft constraints for the specific problem were discussed, the genetic elements were designed and the fitness function was proposed. Three genetic operators: crossover, mutation, and selection were employed to obtain optimized results. The results show that the proposed GA model works well in making the University timetable; no hard constraints appeared in the timetable with crossover probability 0.70. However, modifications are required to satisfy the soft constraints.

Nabeel R. established a new hybrid algorithm to solve course timetabling problem based on genetic algorithm and Great Deluge algorithm [11].He applied the method on standard benchmark problems and were able to produce promising results. The experiments carried out in the work demonstrated that the method obtained better results. The proposed method produced a feasible and good quality timetable. Moreover, it gave consistently good results across various benchmark problems.

3. PROBLEM STATEMENT

In order to compete with various large scale institutions, small academic infrastructure faces bigger challenges due to their limited number of classrooms, instructors, resources etc. for meeting student's interest. As curriculum keeps changing, a course timetable must be updated appropriately to reflect changed conditions. It is a very tedious and time consuming task due to inadequate resources (rooms, faculty and time). A complete and efficient timetable must meet as many requirements of students, courses and instructors as possible, if not all. University course timetabling problem can be defined as a combinatorial optimization problem of assigning a given set of teachers, classes, courses, and classrooms into a limited number of time slots to satisfy almost all constraints. Generally, the constraints represent the limitations on resources, time slots and equipments. The constraints are divided into two categories: hard constraints and soft constraints. Hard constraints are rigidly enforced and soft constraints are a series of optimized conditions which is not absolutely essential. Examples of soft constraints include: senior teachers' such as professors and associate professors, demands are preferred, the classroom of all courses for a class should be same, the course should not be scheduled in the evening for effective teaching, the teachers' demands should be satisfied as many as possible, the courses shouldn't be scheduled in the weekends, etc.

The University course timetabling is one of the common educational timetabling problems. Since it is NP-hard, a variety of approaches have been adopted, achieving varying levels of success. It is a search problem, in which courses must be arranged around a set of time slots, in order to satisfy given constraints and optimize a set of objectives. The NP-hard problems are very difficult to solve using conventional techniques. Therefore a better solution could be achieved using evolutionary algorithm.

4. PROPOSED METHODOLOGY

The proposed model is designed using genetic algorithm employing a constructive heuristic approach. Although Genetic Algorithm is a search technique used to find exact or approximate solutions, the results are often not the best but generally "acceptably good" solutions.

Genetic algorithm (GA) is a heuristic search technique to solve the optimization and search problems. It is the simulation for the process of natural selection and biological evolution. Specially, the process of solving problem using genetic algorithm is described as follows.

Before starting the genetic evolution, a genetic representation and a fitness function should be determined. Genetic representation, called chromosome, expresses mode of the solution. The fitness function which is based on the problem is defined to measure the quality of the represented solution. It is always genetic representation dependent. Once the genetic representation and the fitness function are defined, the evolution which is an iterative process would start.

Initialization- This is a process to initialize the population formed by many individual solutions.

Selection- In this step, a proportion of the existing population in current generation is selected to breed a new generation.

Reproduction- Once those individual solutions to breed a new generation are selected, the next generation population of solutions would be generated through genetic operators: crossover and mutation.

Termination- The above evolutionary process is repeated until a termination condition is reached.

Replace_By_Generation parents are selected randomly for crossover. Crossover randomly selects crossover points in two chromosomes. The information is swapped between the parents, rendering an offspring.

The parameters Replace_By_Generation and Best_ Chromosomes in population are provided along with number of chromosomes, number of crossover points, mutation size and crossover, mutation probabilities. Replace_By_Generation parents chosen randomly are replaced by the offspring produced by crossover operation. Best_ Chromosomes will be retained in the population. The timeslots under each timetable forms the alleles of the chromosome. The different combination of alleles gives the chromosome its distinct identity, shown in Figure-1.

Organization of information into a hash structure helps the Genetic Algorithm to perform operations very quickly. Since the entire timetable is stored in a hash structure, the retrieval of an individual class timetable or individual timeslot may be performed by manipulation of array index. This enables a direct addressing and exchange of information during crossover operation. The use of pointers also helps in a quick exchange.

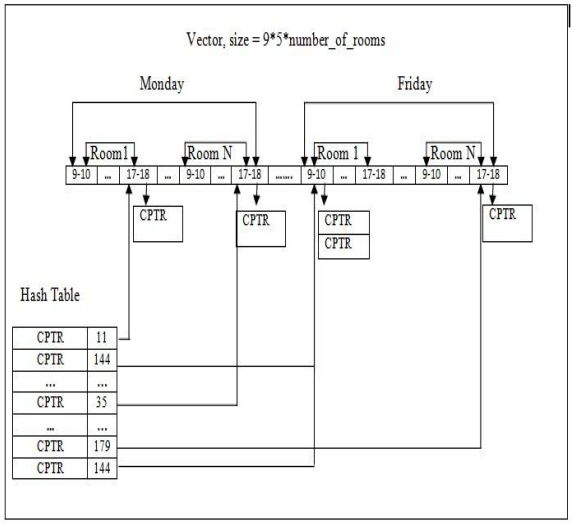

Figure 1: Schematic chromosome representation of a class timetable added to a hash map

Representation of chromosome for a class schedule will need a slot (time-space slot) for each hour (the class time is one hour), for every room, every day. Also, classes begin at 9am, and should finish before or at 6pm (9 hours total) and working days are from Monday to Friday (5 days total). A standard vector with a size 9x5x (Number_of_rooms) is used. The slot is a standard list because during the execution of algorithm, multiple classes during the same time-space slot are allowed. There is an additional hash map which is used to obtain the first time-space slot at which a class begins (its position in vector) from the address of the class object.

Each hour of a class has a separate entry in the vector, but there is only one entry per class in the hash map. For instance, if a class starts at 1pm and lasts for two hours, it has entries in 1pm and 2pm slots.

5. IMPLEMENTATION SCENARIO

The proposed algorithm is implemented using VC++ on Intel Core2 Duo and 32-bits Windows OS. The code given in code project has been customized to solve the problem. A configuration file consists of various input parameters e.g. names of professors, course names, and number of rooms, lab resources and available seats. The population operated by genetic algorithm is maintained in memory. Hard constraints are used for evaluating each generated time table by estimating number of times it breaches them. Therefore, each generation will provide a new time table with minimum number of constraint violations. A class is a structure with three fields' information, where each class will have a certain size and faculty number and the group to which it belongs.

The initial population will comprise of a number of chromosomes equal to the population size. Each chromosome will be designed using the constructive heuristic approach and is represented as a three-dimensional matrix. Then, the value of each slot of the matrix represents allotment scheduled on a day in the corresponding room and period. The initialization procedure uses the input data for chromosome representation. The result of initialization process is to obtain possible chromosomes that may meet the requirements of several hard constraints. A course can be scheduled only once in a day and student's conflict is defined as a student should not be assigned for more than a course at the same slot. In professor's conflict, he/she should never be allotted with more than one class including labs at the same slot. In class-room conflict, classes and laboratories of various courses scheduled at physical location must not overlap. Also, laboratory periods should come in the continuous timeslots.

The program is designed in such a way that the goodness of the chromosomes increases with the increase in the fitness score i.e. the chromosome having the highest score is said to be more optimal than others. The mathematical notation of the fitness score calculation based on soft constraints can be represented as:

$$Minimize \qquad f(t) = \sum_{j=1}^{n \in SC} p(j) * V(j)$$

Where

p (j) - Penalty cost of soft constraint *j* on *T*.

V (j) - Number of violations of Soft constraint *j*.

If $j \in SC$ on *T* is satisfied, then *V (j) = 0*.

Only hard constraints are used to calculate the fitness of a class schedule as follows:

Each class can have 0 to 5 points.

- If a class uses a spare classroom, its score is incremented.
- If a class requires computers and it is located in the classroom with them, or it doesn't require them, increment the score of the class.
- If a class is located in a classroom with enough available seats, increment its score.
- If a professor has no other classes at the time, increment the class's score once again.
- If the student group has no other class at the same time, increment the score of the class.
- If a class violates constraints at any time slot that it occupies, its score is not incremented.
- The total score of a class schedule is the sum of scores of all classes.
- The fitness value is calculated as schedule score/maximum score, and maximum score is number_of_classes*5.

The fitness values are represented by single precision floating point numbers (float) in the range 0 to 1.

Replace_By_Generation parents are selected randomly from the generated population in order to undergo genetic operations like mutation or cross-over. This phase will maintain the diversity of the population and avoids premature convergence.

A crossover 'splits' hash maps of both parents in random sized parts. The number of parts is number of crossover

points plus one. It copies parts from parents to the offspring, and forms a new vector of slots, Figure-2.

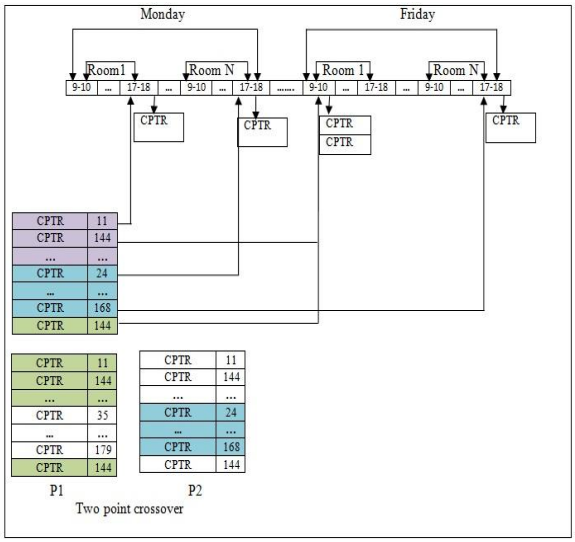

Figure 2: Schematic representation of crossover method

The mutation operation makes a random change in the chromosomes selected to undergo the alterations. Mutation is applied to prevent premature convergence and to find the optimal results.

The algorithm is as follows-

Initialize new population with chromosomes randomly built.

Generation=0.

Repeat

Select parents for crossover Replace_By_Generation times.

Produce the same number of offsprings.

Apply mutation to all.

Select Replace_By_Generation chromosomes and replace them with offsprings by protecting best chromosomes.

Try to add new chromosomes in best chromosomes group.

Generation=Generation+1.

***Until** an algorithm has reached the criteria.*

The generations will continue until one of the possible termination criteria is met, i.e. basically a known optimal or acceptable solution is achieved or maximum number of generations has been performed.

6. RESULTS

The genetic algorithm described above has been tested on the real world dataset of Dr.B.A.Technological University, Lonere, INDIA. It has eight departments, thirty two classes, around two hundred theory and practical courses, hundred teaching and nonteaching staff, twenty four classrooms and various laboratories. Each class is divided into four groups A, B, C, D. All groups in a class attend the respective theory courses together in a classroom, while each group performs the practical course independently in a laboratory. The maximum group and class sizes are twenty five and hundred respectively.

Configuration file specifies the details of professors' workload, group's enrolment, each classroom's capacity, their type etc.

The application tested with different values of the parameters i.e. population size, crossover and mutation probabilities, number of crossover points, mutation size, number of generations, number of chromosomes to be replaced by generation, best chromosomes to be retained in a generation. Finally, a solution which has an optimal fitness is obtained. It has been found that on University departments' dataset the algorithm gives an optimal timetable with 13000 generations, population size 100, 2 crossover points, 40 replacements by generations and mutation size equal to 2. The crossover and mutation probabilities kept constant, 0.8 and 0.03 respectively. Algorithm with more number of replacements converges faster (Figure-3). The population size 100 yields better result than 200 and more (Figure-4). Genetic algorithms effectively demonstrated an ability to solve complex optimization problem.

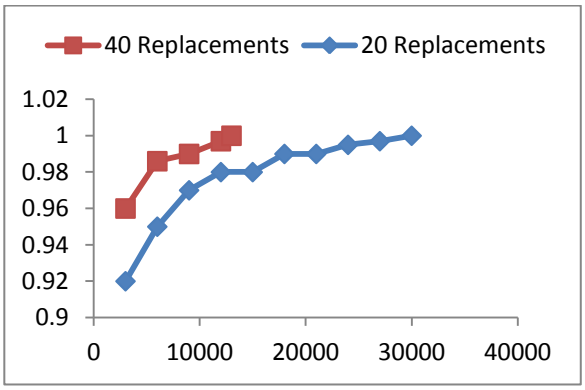

Figure 3: Generations vs. Fitness (Different Replacements)

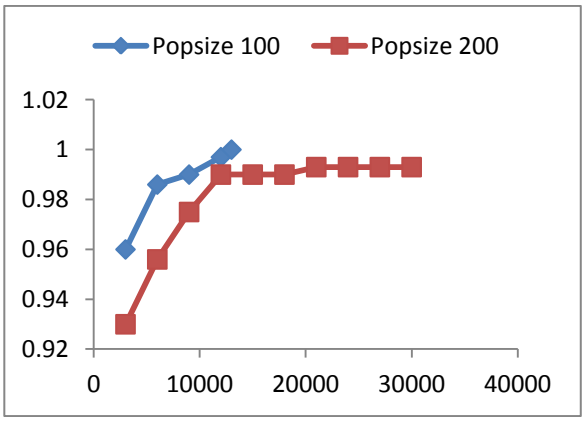

Figure 4: Generations vs. Fitness (Different Population Sizes)

7. CONCLUSIONS AND FUTURE WORK

Timetabling problem is a NP-hard problem. In this paper, university course timetabling problem is discussed in detail from several aspects such as the definition of problem, constraint conditions and the goal of optimization, etc. As an effective heuristic method, genetic algorithm is widely used to solve the optimization problems. Genetic algorithm is applied to solve university course timetabling problem. During the process, a genetic representation and a fitness

function are defined according to the problem. It is demonstrated to be feasible for solving a real-world university course timetabling problem. There are still some points need to be considered in future research although a solution of university course timetabling problem is obtained by using genetic algorithm. First, as genetic algorithm is an iteration based search technique with high computational complexity, it is necessary to improve the performance as much as possible. Second, since there exist other approaches which also can be used to solve this problem, a hybrid method will be studied to solve timetabling problems in the future research.

8. REFERENCES

[1] Burke E.K. and Newall J., "Enhancing Timetable Solutions with Local Search Methods," Burke E.K. and De Causmaecker P. (eds.), Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, 2740, pp. 195-206, 2002.

[2] E.K. Burke, S. Petrovic and R. Qu, "Case Based Heuristic Selection for Examination Timetabling," Proceedings of the Seal'02, 277-281, 18-22, Orchid Country Club, Singapore, 2002.

[3] T. B. Cooper and J. H. Kingston, "The Complexity of Timetable Construction Problems," Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995), LNCS-1153, pages 283–295. Springer Verlag, 1996.

[4] Colorni A., Dorigo and M. Maniezzo, "Genetic Algorithms and Highly Constrained Problems: The Time-Table Case," Parallel Problem Solving from Nature, Goos and Hartmanis (eds.), Springer-Verlag, pp. 55-59, 1990.

[5] Hitoshi Kanoh and Yusuke Sakamoto, "Interactive Timetabling System Using Knowledge Based Genetic Algorithms," IEEE, International Conference on Systems, Man and Cybernetics, 2004.

[6] Alexander Brownlee, "An application of Genetic Algorithms to University Timetabling," Honors Project, 2005.

[7] Maciej Norberciak, "Universal Method for Solving Timetabling Problems Based on Evolutionary Approach," Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 149 – 157, 2006.

[8] Mihaela Oprea, "MAS_UP-UCT: A Multi-Agent System for University Course Timetable Scheduling," International Journal of Computers, Communications and Control, Vol. II , No. 1, pp. 94-102, 2007.

[9] Adilah Binti Abdullah, "Timetable Management System Using Genetic Algorithm," Technical Report submitted at University of Malaya, May, 2008.

[10] Pariwat Khonggamnerd and Supachate Innet, "Improvement of Effectiveness in Automatic University Timetabling Arrangement with Applied Genetic Algorithm," 4th International Conference on Computer Sciences and Convergence Information Technology, 2009.

[11] Nabeel R., "Hybrid Genetic Algorithms with Great Deluge for Course Timetabling," IJCSNS, International Journal of Computer Science and Network Security, Vol.10, No.4, April, 2010.