ISSN 2348 - 2249
Volume-5, Issue-1, January-March, 2017
International Journal for Multi Disciplinary Engineering and Business Management
(IJMDEBM)

AUTOMATED TIMETABLE SCHEDULING USING GENETIC ALGORITHM

Ritu Walia
Department Of Computer Science,
Mata Gujri College, Fatehgarh Sahib, Punjab, India
de.ritu@yahoo.co.in

Bachandeep Singh Bhathal
University Institute of Computing,
Chandigarh University, Mohali, Punjab, India
bachan0235@gmail.com

Abstract – University time table scheduling is a NP-Hard problem which is a monumental task to do manually. This is a highly constrained combinational and optimisation problem which needs to arrange the events like classes, courses using resources like teaching faculty and venue for class into a number of Time-slots such that any conflict is avoided. There is usually a very large solution set, so conventional search algorithm like calculus based, enumerative and random strategies are not successful. So, we use Genetic Algorithm which revolves around the process of natural selection using operation like crossover and mutation to optimise this problem and create a feasible time table. The genetic algorithm is divided into 3 parts, handling one constraint at a time. Starting with selecting the subject for a time slot for a class. Then in second part appropriate faculty member is assigned for each time slot containing a course considering faculty's subject preferences. Lastly, required room or lab is assigned for each time slot based on subject requirement and teacher preferences. This division of work into 3 parts help us in achieving the final timetable faster as the constraint in each section of genetic algorithm are fairly less when compared to handling all the constraints together. In this way we are aiming to create a time table in an optimised way to implement it in a university and save all those man-hours spent to do the task manually.

I. INTRODUCTION

Time table scheduling has been in human requirements since they thought of managing time effectively. It is widely used in schools, colleges and other fields of teaching and working like crash courses, couching centers, training programs etc. In the early days, time table scheduling was done manually with a single person or some group involved in the task of scheduling it with their hands, which takes a lot of effort and time. Scheduling even the smallest constraints can take a lot of time and the case is even worse when the number of constraints or the amount of data to deal with increases. In such cases, a perfectly designed time table is reused for the whole generation without any changes, proving to be dull in such situations. Other cases that can cause problems is when the number of employers or workers is weak, resulting in rescheduling of the time table or they will need to fill on empty seats urgently.

Academic Institutions (Schools, Colleges, Universities, etc.) are the regular users of such time tables. They need to schedule their courses to meet the need of current duration and facilities that are available to them. However, their schedule should meet the requirement of new course addition and newly enrolled students to fresh batches. This may result in rescheduling the entire time table once again for its entire batches and to be scheduled in the shortest possible time before the batch courses start.

The timetable problem can be categorized into various types (e.g., rail timetable, examination timetable, school timetable, course timetable etc.) by considering different specific constraints, processes and resources. The particular instance chosen for implementation in this project is the lecture timetable. The lecture timetable problem is to organize Subject, Teacher, Room, and Period in order to satisfy the set of constraints. The problem formulation is as follows:

- Set of subjects, S = {s1, s2,…, sn}
- Set of teachers, T = {t1, t2, …, tm}
- Set of rooms, R = {r1, r2, …, ro}
- Set of periods, P = {p1, p2,… pk}

A combination (a, b, c, d) in which a ϵ S, b ϵ T, c ϵ R, d ϵ P, can be stated as "teacher b teaches subject a in room c during period d". The relation of subjects and teachers are fixed in all tuples. The lecture timetable problem can be interpreted as the scheduled rooms and periods in tuples and the solution of this problem is a set of tuples which satisfy all constraints. Assigning lectures to time periods is equivalent to the graph coloring problem. This is one of the approaches or algorithms that shall be investigated. Another approach, a heuristic algorithm, will also be looked at in great detail.

B. Problem Statement

Till now, the time table is being generated manually and is cumbersome task for the time table creator. It takes weeks for a time table to be created considering all the constraints, which in this case are very large in number. The aim of our system is to reduce the workload of timetable creator, and provide him with a simple and effective system that is easy to use.

II. LITERATURE SURVEY

A. Introduction

Timetable creation is a very arduous and time-consuming task. To create timetable, it takes lots of patience and man hours. Time table is created for various purposes like to organize lectures in school and colleges, to create timing charts for train and bus schedule and many more. To create timetable, it requires lots of time and man power. We have tried to reduce these difficulties of generating timetable by Genetics Algorithm. By using Genetic Algorithm, we are able to reduce the time require to generate time table and generate a timetable which is more accurate, precise and free of human errors. The first phase contains all the common compulsory

ISSN 2348 - 2249
Volume-5, Issue-1, January-March, 2017
International Journal for Multi Disciplinary Engineering and Business Management
(IJMDEBM)

classes of the institute, which are scheduled by a central team. The second phase contains the individual departmental classes. Presently this timetable is prepared manually, by manipulating those of earlier years, with the only aim of producing a feasible timetable.

B. Theory Associated with Problem Area

Time Table Scheduling is an NP-hard problem and hence polynomial time verifiable using genetic algorithms. It a typical scheduling problem that appears to be a tedious job in every academic institute once or twice a year. In earlier days, time table scheduling was done manually with a single person or some group involved in task of scheduling it manually, which takes a lot of effort and time. Planning timetables is one of the most complex and error-prone applications. Timetabling is the task of creating a timetable while satisfying some constraints. There are basically two types of constraints, soft constraints and hard constraints. Soft constraints are those if we violate them in scheduling, the output is still valid, but hard constraints are those which if we violate them; the timetable is no longer valid. The search space of a timetabling problem is too vast, many solutions exist in the search space and few of them are not feasible. Feasible solutions here mean those which do not violate hard constraints and as well try to satisfy soft constraints. We need to choose the most appropriate one from feasible solutions. Most appropriate ones here mean those which do not violate soft constraints to a greater extent. In this project hard-constraints have been taken care of strictly and it has been ensured that soft-constraints are as well followed as much as possible.

C. Existing Systems

Wise Time Table: Combine manual and unbelievably capable automatic generating to achieve complete schedules fast and without any conflicts. System read your old data and together with your additional parameters creates a perfect result. Even really complex situations are resolved easily. Create final exam schedule/seating automatically. We have also a special web utility (use demo, demo for login) for planning exam dates.

Time Tabling Turbo: Best for HIGH, MIDDLE and ELEMENTARY schools. 100% AUTOMATIC timetable construction with an unparalleled Artificial Intelligence algorithm. Supports many kinds of constraints like gaps, activities per day, unavailability, range of periods and more. Allows interactive timetabling and daily changes' tracking. Suitable for any kind of SCHOOL with a recurring weekly timetable with up to 16 periods a day.

D. Problem Identified

Timetabling is known to be a non-polynomial complete problem i.e. there is no known efficient way to locate a solution. Also, the most striking characteristic of NP-complete problems are that, no best solution to them is known. Hence, in order to find a solution to a timetabling problem, a heuristic approach is chosen. This heuristic approach, therein, leads to a set of good solutions (but not necessarily the best solution).

These constraints are of two types Hard and Soft constraints. Hard constraints include those constraints that cannot be violated while a timetable is being computed. For example, for a teacher to be scheduled for a timeslot, the teacher must be available for that time slot. A solution is acceptable only when no hard constraint is violated. On the other hand, soft constraints are those that are desired to be addressed in the solution as much as possible. For example, though importance is given to a teacher's scheduling, focus is on setting a valid timetable and this can lead to a teacher going free for a time slot. Thus, while addressing the timetabling problem, hard constraints have to be adhered, at the same time effort is made to satisfy as many soft constraints as possible. Due to complexity of the problem, most of the work done concentrates on heuristic algorithms which try to find good approximate solutions.

There exist various algorithms for the same:

1. Naive methods
2. Graph coloring
3. Genetic algorithm

III. RESEARCH GAPS

- Current research tackles the course, room, subject and time slot concurrently which in return increases the time complexity of the algorithm as complexities are multiplied for each sub domain.
- Some systems allot subjects or rooms first which in turn consume more resources.

IV. IMPLEMENTATION

Genetic algorithms are adaptive systems inspired by natural evolution. They can be used as techniques for solving complex problems and for searching of large problem spaces. Genetic algorithms are belonging to guided random search techniques, which try to find the global optimum. J.H. Holland presented this concept in early seventies. The power of genetic algorithms and other similar techniques (simulated annealing, evolutionary strategies) lies in the fact that they are capable to find global optimum in multi-modal spaces (spaces with many local optimums). Classical gradient methods will always gravitate from starting position to some local optimum, which could also be global, but it cannot be determined for certain. Genetic algorithms are working with the set of potential solutions, which is called population. Each solution item (individual) is measured by fitness function. The fitness value represents the quality measure of an individual, so the algorithm can select individuals with better genetic material for producing new individuals and further generations.

The simulation of evolution allows survival of better individuals and extinction of inferior ones. Evolution's goal is to find better individuals in each generation. The process of evolution is maintained by selection, crossover and mutation. In terms of genetic algorithms those processes are called genetic operators.

ISSN 2348 - 2249
Volume-5, Issue-1, January-March, 2017
International Journal for Multi Disciplinary Engineering and Business Management
(IJMDEBM)

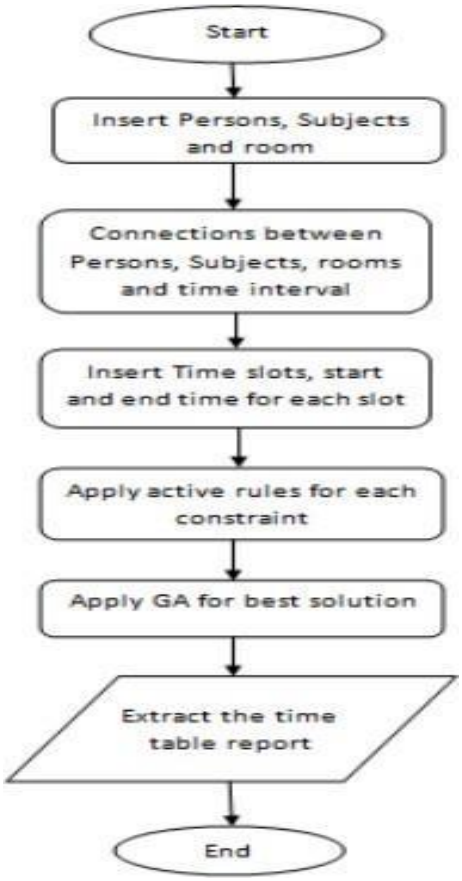
The selection chooses superior individuals in every generation and assures that inferior individuals extinct. The crossover operator chooses two individuals from current population (parents) and creates a new individual (child) based on parents' genetic material. Selection and crossover operators will expand good features of superior individuals through the whole population. They will also direct the search process towards a local optimum. The mutation operator changes the value of some genes in an individual and helps to search other parts of problem space. In the algorithm presented here, each individual in the population represents one timetable. The algorithm starts from an infeasible timetable, and tries to get the feasible one.

The fitness value of an individual is calculated as fitness(individual) = Total Classes - Number of conflicts.

The number of conflicts shows how many constraints have been violated within the current individual. When an individual reaches zero conflicts, that means that it represents a feasible timetable and that there are no collisions of classes [1]. The quality of the timetable is determined by earliness of scheduled classes. Students have better ability to learn in morning hours and after that, the interest for learning is continually decreasing. That is why the best quality value is set to the early hours and worse values are set for late hours. The genetic algorithm will try to schedule classes as early in the morning as it can, indirectly minimizing the number of holes in a student's schedule. The main goal of the genetic algorithm presented here is to achieve a feasible timetable. That is why the feasibility function will mainly try to minimize the number of conflicts in an individual. This is achieved by multiplying the number of conflict by a large constant K. The quality of the timetable is of the lesser importance. Individuals in a population are sorted by ascending value, so the best individual has the smallest fitness value. The program uses eliminating selection, which chooses and eliminates bad individuals from the current population, making room for new children that will be born.

There is some probability (though very small) that eliminating selection deletes the best individual. That would ruin the algorithm efforts and put its work back for some number of generations. Therefore, protection mechanism for best individuals has to be made, so the good genetic material is sustained in population. It is called the elitism. The authors' choice was to keep top eight individuals.

The reproduction operators constitute a very important part of genetic algorithms. Those operators make use of good individuals (which remained in population after selection) and construct new, better individuals and overall population. The crossover operator operates on individuals (called parents) and make new, child individual from their genetic material. This operator fills up empty places in population that remained after elimination. If parents are good, it is likely that their child will also be good. Uniform crossover operator was chosen as the best option for this kind of problem [8].



for each gene in (parent1, parent2) {
if(parent1[gene]==parent2[gene]){
child[gene]=parent1[gene]; }else{ child[gene] = random(parent1 ,parent2)[gene]; } }

Uniform crossover operator checks all genes of both parents. If parents have equal values of a gene, this value is written to the child. If values from parent genes differ, then the algorithm randomly chooses one parent as a dominant one and takes its gene. The program uses simple roulette wheel parent selection algorithm. The probability of selection of one individual is proportional to its fitness value [4].

The mutation is also a typical operator for the genetic algorithm. It takes one or more genes from an individual and changes its value. The probability of the mutation is an input parameter for genetic algorithm. The presented algorithm iterates through every gene of every individual in the population. For each gene a random number form the interval (0, 1) is generated. If the generated value is smaller than the given probability of the mutation (pm), the gene changes value to a random value which denotes a different day-time value or room combination.

ISSN 2348 - 2249
Volume-5, Issue-1, January-March, 2017
International Journal for Multi Disciplinary Engineering and Business Management
(IJMDEBM)

V. EXPERIMENTAL RESULTS

For experimental purposes data from Faculty of Thapar University was used. Algorithm was tested on the small and large instances of problem (Table 1). The large problem is a full size Thapar University schedule for the odd semester. The small size problem was obtained from the large size with exclusion of about 70% of classes from the scheduling process. Both the small and large problem were solved without any conflicts.

	SMALL	LARGE
Classes	100	770
Rooms	20	41
Group	50	114
Faculty	35	157

VI. CONCLUSION

The application will make the procedure of time table generation easier consistently which may otherwise need to be done using spread sheet manually which might lead to constraints problem that are strenuous to establish when time table is generated physically. The purpose of the algorithm is to generate a timetable schedule mechanically. The algorithm includes many techniques, aimed at improving the efficiency of the search operation. It also addresses the chief hard constraints. Most of the non-rigid soft constraints are also productively handled. Keeping in mind the generality of the algorithm operation, it can further be modified to more particular scenarios, e.g. University, examination scheduling, etc. A number of hours which are spent on creating a fruitful timetable can be reduced ultimately through the mechanization of the timetable issue. The most fascinating future direction in the evolution of the algorithm lies in its addendum to constraint propagation.

VII. FUTURE WORK

- To also include exam scheduling in the system. Scheduling of exams can be as complicated as classes, so our system, in the future, can also schedule examinations of every class group within the given time frame.
- System can further be modified to handle more constraint like teacher time or room preference.
- We can also add a feature to reschedule a lecture or a lab in case faculty has to cancel one due to various unforeseen reasons.
- Application can be developed as a distribution medium to deliver time table to students and faculty.
- Right now, the software works on an independent platform from the college's intranet server. So, in the future we would like to export our software directly to college so that they can manage the software as they want within their provided freedom. This will make the software even more easy to access for students and teachers.

REFERENCES

[1] Jinsoo Han; Chang-Sic Choi; Ilwoo Lee, "Automatic Timetable Generation System", IEEE Transactions on, vol.57, no.1, pp.85,89, February 2011.

[2] Erdem, H.; Uner, A., "Some experiments with simulated annealing for coloring graphs", IEEE Transactions on, vol.55, no.4, pp.2184,2189, November 2009.

[3] Yuksekkaya, B. Kayalar, A.A. Tosun, M.B. Ozcan, M.K. Alkar, A.Z., "Optimization of resource allocation and leveling using genetic algorithms," *IEEE Transactions on Consumer Electronics*, vol.52, no.3, pp.837,843, Aug. 2006.

[4] Chia-Hung Lien; Ying-Wen Bai; Ming-Bo Lin, "Multi-mode resource-constrained discrete time–cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm," *IEEE Transactions on Consumer Electronics*, vol.53, no.4, pp.1634,1641, Nov. 2007.

[5] Vernon, S.; Joshi, S.S., "Local search techniques for large high school timetabling problems," *IEEE Transactions on Information Technology in Biomedicine*, vol.15, no.4, pp.531,538, July 2011.

[6] Faundez-Zanuy,"Timetable construction–an annotated bibliography", *IEEE Aerospace and Electronic Systems Magazine*, Vol. 19, No.4, pp. 17-21, April 2004.

[7] IFaundez-Zanuy, M., "Genetic Algorithm to Generate the Automatic Time-Table–An Over View ", *IEEE Aerospace and Electronic Systems Magazine*. Vol. 20, No.4, pp. 13-17, April 2005

[8] Faundez-Zanuy, M., "A graph coloring algorithm for large scheduling problems", *IEEE Aerospace and Electronic Systems Magazine*, Vol. 20, No.2, pp. 1315, February 2005.

[9] P. B. Saurabh and D.S.Chaudhari, "An automated approach based on bee swarm in tackling university examination timetabling problem", *International Journal of Engineering and Advanced Technology*, vol. 1, pp. 91-94, 2012.

[10] S. M. Prakash and C. J. Kalpna, "The problem of assigning students to course sections in a large engineering school", *International Journal of Artificial Intelligence Knowledge Discovery*, vol. 1, pp. 25-28, 2011.

[11] A. M. Martinez and A. C. Kak, "A memetic algorithm for university exam timetabling" , *LOS ALAMITOS*, pp. 228-233, 2001.

[12] J. Mazanec, "A Novel approach for automatic timetable generation", Journal of Electrical Engineering , vol. 59, pp. 203-209, 2008.

[13] K. H. Pun, "University Timetable Generator Using Tabu Search", Biometric Technology for Human Identification II, vol. 5779, 2005.

[14] R. Ibrahim and Z. M. Zin, "Automated Scheduling System for Thesis and Project Presentation Using Forward Chaining Method with Dynamic Allocation Resources", IEEE 3rd International Conference in Communication Software and Networks (ICCSN), Beijing, China, 2011, pp. 132- 136.

[15] W. Shimin and Y. Jihua, "The problem of assigning students to course sections in a large engineering school", IEEE Conference Proceedings, 2010, pp. 2618-2621.

