Time Table Scheduling using Genetic Algorithms employing Guided Mutation

Vinayak Sapru*, Kaushik Reddy†, B Sivaselvan‡
IIITD&M Kancheepuram,
IIT Madras Campus, Chennai-36.
* vinayaksapru@gmail.com,†kaushikreddy@gmail.com,‡sivaselvanb@iiitdm.ac.in

***Abstract*—Genetic Algorithms, a class of evolutionary optimization techniques offer benefits of being probabilistic, requiring no auxiliary knowledge in comparison to conventional search methods such as calculus based, enumerative and random strategies. This paper discusses a Genetic Algorithm based university time table scheduling algorithm satisfying constraints that avoid clash of faculty, class room slots, etc. The paper exploits the rank based selection scheme to ensure that the time table schedule generated is the feasible global optima as opposed to the stagnant solution setup associated with roulette selection scheme. An application specific encoding structure, rank based selection of time-table schedules and single point crossover to explore new and fitter schedules is used in the proposed algorithm. The proposed guided mutation operator helps in convergence as a result of the increased constraint satisfaction rates and hence better fitness values.**

***Index Terms*—Constraints Satisfaction, Elitism, Evolutionary Algorithms, Genetic Variation, Guided Mutation, Rank Selection**

I. INTRODUCTION

Scheduling is the task of allocation, arrangement or distribution of certain objects, subject to specific constraints, in a pattern of time or space [1]. The aim of such an arrangement may be optimal allocation of available resources in order to achieve minimization of time or cost. Alternately, the aim of a scheduling problem may only be to obtain a feasible solution based on the constraints, since in certain scheduling problems, all feasible solutions have equal cost and are optimal. This is the case with the Timetabling problem dealt with in this paper.

Objects in a scheduling problem may be workers, vehicles, machines, etc. Constraints are the rules that govern the process of scheduling (such as restrictions on time or available resources), and certain constraints may be inviolable for the problem under consideration, while others take the form of principles that should, but need not necessarily be satisfied. A Genetic Algorithm (GA) is a parallel search algorithm based on the Darwinian evolutionary theory [2]. It is used to search a large solution space which may be discontinuous or nonlinear, and/ or where expert knowledge is lacking or difficult to encode. Genetic algorithms form part of a larger category of algorithms called Evolutionary Algorithms (EA), which apply techniques inspired by natural evolution- such as mutation, selection, and crossover, and use them to generate solutions to optimization problems.

The process of Selection in Genetic Algorithms follows the Darwinian principle of Survival of the Fittest, by allotting a large number of copies in the following generations to fitter chromosomes, and allotting none or few copies to unfit chromosomes. This makes the gene pool, on average, much fitter. The process of crossover involves exchange of genetic information between two randomly chosen mates, which goes on to produce one or multiple offspring, which may be distinct from their parent chromosomes. The process of mutation involves random change of allele values in a chromosome, in order to bring about genetic variation.

The Time Table Scheduling Problem is an NP-hard problem, and is hence very difficult to solve by conventional methods [3]. Time Table problems can be of several kinds - for the purpose of transportation, a hospital roster for nurses, or educational institutes. This paper considers an educational time table, specifically for the case of a University. The main task involved in this category of scheduling problem is the assignment of each lecture for a particular student group (i.e. students in the same department and year) to a specific room during a given time slot, while keeping in mind all restrictions on time, the availability of faculty, and other miscellaneous resources.The following constraints would be inviolable for a University Time Table Scheduling problem:

1) No room can have more than one scheduled class in a given time slot.
2) No student group can have more than one scheduled class in a given time slot.
3) No faculty member can have more than one scheduled class at a given time slot.
4) The number of classes per week is a fixed number for each subject, and this number must be met by the prepared schedule.

II. RELATED WORK

The subject of Timetable scheduling has received great interest from the scientific community in recent years. This has led to the creation of a series of conferences called the PATAT (Practice and Theory of Automated Timetabling) [4] in 1995, which has since been held every two years, the most recent edition scheduled in 2010. It also led to the establishment of the WATT (Working Group on Automated Timetabling) [5] by EURO (Association of European Operational Research Societies). In 2002, PATAT helped establish the International Competition of Timetabling (ITC, 2002) and the most recent edition was conducted in 2007 [6]. Since Timetable scheduling

978-1-4244-5967-4/10/$26.00 ©2010 IEEE

is an important problem that has been dealt with in the past, a large number of diverse methods have already been proposed in the literature for solving such problems, as well as several variations of the same.

Timetable problems have been modeled as graphs, where the events are ordered, and then assigned sequentially into valid time slots [7]. Modelling a timetabling problem as a set of variables (events) to which values (resources such as teachers and rooms) have to be assigned in order to satisfy a number of hard and soft constraints [8]. Meta-heuristic methods, such as genetic algorithms (GA) [3], simulated annealing [9], tabu search [10], and other heuristic approaches, that tend to be inspired from nature or by natural processes and phenomena, and apply nature-inspired and nature-like processes to solutions or populations of solutions, in order to slowly make them evolve towards and attain optimality.

This paper discusses a Genetic Algorithms based solution to the Time Table Scheduling problem. The main reasons for this are that Genetic Algorithms require no gradient or other information; they also do not require a continuous or differentiable data set or objective function. Additionally, a GA would evolve the solution from one population in one generation to another and hence be capable of producing multiple optima rather than a single local one [2], [11]. Additionally, Genetic Algorithms can, with small variations, be made efficient for dealing with a variety of problems, while maintaining a degree of robustness [12]. These characteristics make GA a well-suited tool for the class of timetable scheduling problem discussed in this paper.

III. PROPOSED GA BASED TIME TABLE SCHEDULER

For the timetabling problem, lectures must be assigned to faculty members and arranged in time slots, such that the resulting time table is feasible. Hence, we are searching for any given feasible solution, since all feasible solutions are treated as equivalent by the algorithm. Further refinement in the feasible solutions obtained using Genetic Algorithms (GA) based on user preferences is possible. A detailed discussion of the encoding scheme and the GA operators used is given in the following subsections:

A. Encoding Scheme

Each individual chromosome is, in itself, a time table. This means that every chromosome would carry a large amount of data- regarding subjects, faculty, and rooms allotted to each slot for each student group in a week. The structure of the chromosome would be a 5 x 8 matrix, since it must represent a time table for a 5- day week, with 8 allotted time slots per day. Additionally, each time slot must carry separate information about each student group. We assume that there are four student groups in the university representing four different batches of undergraduate students.

Under each student group, would be information about whether the class slot is active or vacant. If the slot is vacant, then the remaining information is to be ignored. If it is active, the information regarding the faculty code, subject code, and room code where that class would be held is necessary. All information (whether a class is active or vacant, room codes, etc.) is in binary. The fitness of each chromosome is scaled between 0 (infinite constraints violated by the chromosome) and 1 (no constraints violated, i.e., feasible timetable). The fitness of a chromosome is calculated using the formula in Equation 1, where F is the fitness of the chromosome and c is the number of violated constraints.

$$F = \frac{1}{1+c} \tag{1}$$

. The value of c is arrived at as follows:

- c is incremented every time multiple classes are scheduled in the same room in a given time slot; and
- c is incremented every time multiple classes are scheduled for the same faculty in a given time slot; and
- Every time the number of classes per week of a given subject is not equal to the predefined number, c is incremented by the absolute value of the difference between the two.

Hence, any chromosome having a fitness of 1.0 would be a feasible solution.

Each time slot is modelled as a structure in C containing the following members (i) a 1 bit variable which describes whether or not the slot is occupied (has an active class),(ii)a 3 bit variable describing the subject code of the active class, (iii) 3 bit variable describing the faculty code of the active class, and a 2 bit variable describing the room code of the active class. Each chromosome would be a 5 x 8 array of a single time slot. The total population would be an array of chromosomes. The chromosomes are initialized randomly. The simulation uses a coin toss function (equally probable event with two outcomes) in order to generate a 0 or 1. Accordingly, each bit in every chromosome is populated randomly with a 0 or a 1.

B. Selection Operator

The selection operator in a GA chooses the next generation of the GA based on the fitness of the present generations. It stochastically allocates higher number of copies in the following generation to highly fit strings in the present generation. For the purpose of this timetabling problem, a rank selection operator was implemented. Rank selection chooses the best few (five or six) chromosomes in a generation, and allots maximum copies to them, based on their fitness.

In most cases, rank selection operator is used in order to reduce elitism, i.e. selection of only a few elite (highly fit) strings, which leads to a lack of genetic variation. In case of the timetabling problem, rank selection operator is used by us for the opposite problem: due to the large amount of data in each chromosome, and the random nature of the initialization of the chromosomes, all of them have approximately the same values of fitness. As a result, if we use the roulette wheel selection operator, we end up with exactly one copy of each string in the following generation. Hence, the rank selection increases the amount of elitism, and does not let the population in succeeding generations become stagnant, unlike the roulette wheel selection.

C. Crossover Operator

A single- point crossover operator is utilised in this implementation. It works quite well, and provides the much needed genetic variance. The point of crossover is a randomly generated number, and is distinct for each occurrence of crossover. Crossover pairs are generated randomly for the entire population.

The crossover occurs separately for each row of the chromosome, rather than for the chromosome as a whole. A random point is selected among the eight slots for each day in a given time table. All of the information in the slots following the crossover site is exchanged between the pair, i.e the entire time slot details for every single student group. Hence, crossover would occur 8 times for a single chromosome. This simple yet effective modification of the conventional crossover provides the necessary genetic variation.

D. Mutation Operator

Mutation is performed via the novel Guided Mutation operator for the Time Table scheduling problem. The conventional mutation operator is associated with a mutation probability. Through simulations, it was found that the ideal value of this probability was 0.05. The mutation operator would do the following:

- Examine the first binary digit of a chromosome.
- Toggle that bit with a probability of 0.05, or leave it unchanged, with a probability of 0.95.
- Repeat the process with every bit of the chromosome.
- Similarly, mutate all the chromosomes in the population.

The advantage of mutation is that it explores new spaces, and brings about a large degree of genetic variation. The main disadvantage is that, due to the random nature of this mutation operator, it is equally likely to bring about variation that would increase or decrease the fitness of a given chromosome. To solve this issue, we introduced the Guided mutation operator. It does the following:

- Create a copy of the chromosome under consideration.
- Examine the first binary digit of the copy of the chromosome.
- Toggle that bit with a probability of 0.05, or leave it unchanged, with a probability of 0.95.
- Repeat the process with every bit of the chromosome copy.
- Compare the fitness of the (mutated) copy with that of the (un-mutated) original.
- If the (mutated) copy is fitter, replace the (un-mutated) original with the copy. Otherwise, discard the copy.
- Similarly, perform guided mutation on all the chromosomes in the population.

IV. Proposed Genetic Algorithm

The algorithm puts together all of the steps and processes described previously. It first generates a random initial population, and then performs selection, crossover and guided mutation upon that population for a number of generations, until some termination criterion is met. In this case, we keep a limit on the number of generation as a termination criterion. We can also set the termination criterion as that point when a chromosome having a fitness of 1.0 is obtained. Pseudocode of the proposed guided mutation based genetic algorithm for time table scheduling is as given in Figure 1.

1) Initialize generation count and the maximum number of generations.
2) A random initial population is generated.
3) Fitness of each chromosome is evaluated.
4) Reproduction operation is performed using Rank Selection.
5) Fitness of each chromosome is evaluated.
6) Random pairs of chromosomes for crossover operator are selected and single point crossover is performed.
7) Fitness of each chromosome is evaluated.
8) Guided mutation is performed and followed by fitness evaluation.
9) Generation count is incremented.
10) Iterate Steps 2-9 till either maximum limit of generations or fitness required is reached.

Fig. 1. Proposed Genetic Algorithm based Time Table Scheduler

V. Simulation Results

On performing the simulations, it was found that the convergence of the algorithm depends heavily on a number of parameters, including the number of subjects under consideration for each student group, as well as on the number of hours per week of each subject. Accordingly, the algorithm was tested for a total of four separate cases- each with different number of subjects and/or a different number of contact hours per week for each subject.

The graphs in Figures 2-5 provide a comparison of the proposed algorithm with the conventional mutation operator based algorithm for the four cases discussed The x- axis, in case of all graphs, is a logarithmic (base 10) scale. The algorithm is simulated for 10,000 generations.

By observing the simulation results and the graphs, it is inferred that, for the first few generations, there is no great difference between the fitness values for the algorithm using Guided Mutation as opposed to the algorithm using conventional Mutation. In fact, the algorithm using conventional mutation may even produce chromosomes having slightly higher values of fitness. Eventually though, the algorithm using conventional mutation begins to spawn new generations having the same constant values of fitness as the previous generation. Genetic variation becomes less, and increases in fitness become rare. However, the same does not occur in case of the algorithm using guided mutation. The fitness values for each generation keep varying, and this genetic variation that

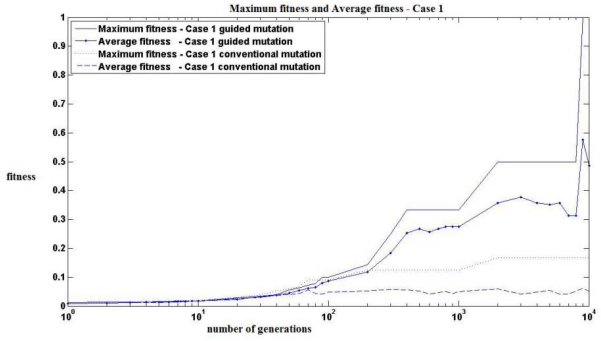

Fig. 2. Effect of Guided Mutation - Case 1

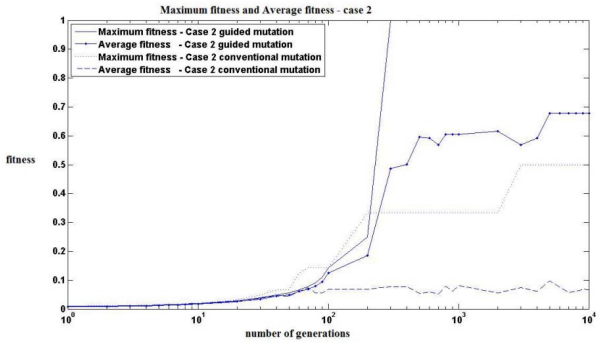

Fig. 3. Effect of Guided Mutation - Case 2

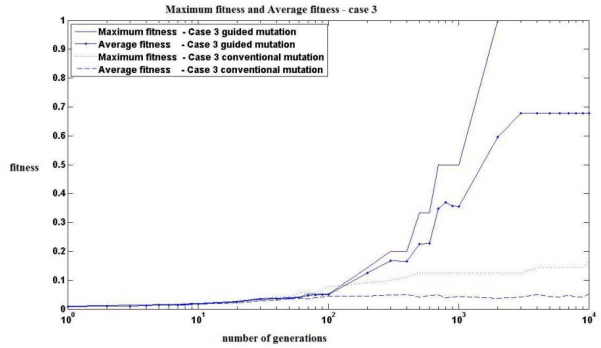

Fig. 4. Effect of Guided Mutation - Case 3

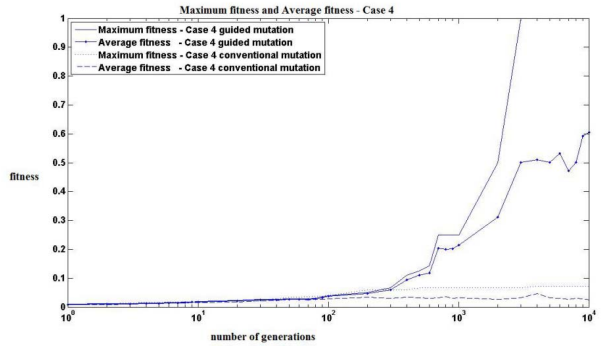

Fig. 5. Effect of Guided Mutation - Case 4

occurs in every new generation allows for the search of new points in the solution space, eventually leading to a feasible solution. Repetition of fitness values in succeeding generations is much rarer. Time Tables scheduled using conventional mutation do not converge to a feasible solution, although in some cases they came close to doing so. Additionally, the Time Table generally takes a much higher number of generations to reach high values of fitness. However, with guided mutation, the algorithm converges to a feasible solution in all cases, and does so much faster. The average fitness of the entire population is much higher in case of the algorithm with guided mutation (in the range of 0.5 to 0.6). Based on the formula for fitness (equation 1), a fitness value of 0.5 denotes only one violated constraint. Hence, an average fitness greater than 0.53 for a population of the order of 20 implies the existence of multiple feasible solutions.

VI. CONCLUSIONS

This paper discussed a suitable encoding scheme and operators, including the novel Guided Mutation operator, for solving the Time Table scheduling problem. The algorithm was found to be effective for providing a solution, and the Guided Mutation operator showed a marked improvement in the algorithms performance under a variety of conditions. Problems that are similar in nature to the Time Table scheduling problem may be solved using representation and operators similar to those discussed above and a performance study of the proposed algorithm will be reported in the future.

REFERENCES

[1] A. Wren, "Scheduling, timetabling and rostering– a special relationship?" in *1st International Conference on the Practice and Theory of Automated Timetabling*, 1995, pp. 474–495.

[2] D. Goldberg, *Genetic Algorithms in Search, Optimization,and Machine Learning*. Addison Wesley, 1989.

[3] S. Abdullah and H. Turabieh, "Generating university course timetable using genetic algorithms and local search," in *3rd International Conference on Convergence and Hybrid Information Technology*, 2008.

[4] "International series of conferences on the practice and theory of automated time tabling (patat)," www.asap.cs.nott.ac.uk/patat/patat-index.shtml.

[5] "International timetabling competition," www.cs.qub.ac.uk/itc2007/indexfiles/overview.htm.

[6] "The working group on automated timetabling," www.asap.cs.nott.ac.uk/watt.

[7] M. W. Carter, "A survey of practical applications of examination timetabling algorithms," *Operations Research*, vol. 34, pp. 193–202, 1986.

[8] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *European Journal of Operational Research*, vol. 119, pp. 557–581, 1999.

[9] D. Zhang, Y. Liu, R. MHallah, and S. Leung, "A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems," *European Journal of Operational Research*, vol. 203, no. 3, pp. 550–558, 1999.

[10] G. White, B. Xie, and S. Zonjic, "Using tabu search with longer-term memory and relaxation to create examination timetables," *European Journal of Operational Research*, vol. 153, no. 1, pp. 80–91, 2004.

[11] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[12] D. A. Coley, *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific, 1999.