Automatic Timetable Generation System

[1]Deeksha C S, [2]A Kavya Reddy, [3]Nagambika A, [4]Akash Castelino, [5]K Panimozhi
[1,2,3,4]UG Student, [5]Assistant Professor
[1]Dept. of Computer Science and Engineering,
[1]BMSCE, Bangalore, India.

***Abstract* - This paper discusses the various approaches that can be taken to solve the timetable generation problem. Timetable generation is basically a constraint satisfaction problem. The methods discussed in this paper are capable of handling both soft and hard constraints. Once the final timetable has been generated, teachers and students can access it through three different views i.e., class view, lab view and combined view of classes and labs.**

The Timetable Generation System aims at automating the process of timetable generation for the Computer Science and Engineering department of BMS College of Engineering, Bangalore. It takes the number of semesters, sections, subjects, labs and teachers as input and generates a timetable which satisfies all of the hard constraints.

***Index Terms* - Course timetabling, Genetic Algorithm, Tabu Search, Iterative Local Search, GRASP.**

I. INTRODUCTION

Even though most college administration work has been computerized, timetable scheduling is still done manually due to the difficulties involved. The manual timetable scheduling requires considerable time and effort. Timetabling is the allocation, of given resources to objects that are placed in space time, in such a way that they satisfy a desirable set of objectives.

The college lecture-timetabling problem asks us to find some slots and classrooms which satisfy the constraints imposed on offered courses, lecturers, classrooms and so on. The problem is a combinatorial optimization problem belonging to NP-hard class where the computational time grows exponentially as the number of variables increases.

Various approaches have been made in the past decade to solve the problem of constructing timetables for schools and colleges. In our paper this problem is formulated as a constraint satisfaction problem and we discuss the various approaches that are capable of handling both hard and soft constraints.

Hard constraints cannot be violated under any circumstances. For example, two classes cannot be allocated to a single teacher at the same time period, two classes cannot be attended by a student at the same time, more than one class cannot be held at a room at the same time et cetera.

Soft constraints are necessary but not absolutely critical. For example, a timetable must be made in such a way that a group of students don’t have to come to college to attend only one class.

II. LITERATURE SURVEY

There exist various timetable generation problems such as University Timetabling, Employee Timetabling, Sports Timetabling and Examination Timetabling.

Carter and Laporte (1998) considered different categories to solve the timetabling problem. They are – Cluster method, Sequential method, Meta-Heuristics and Constraint Based method. Meta Heuristics is a higher level procedure which is used to provide good enough solutions for optimization problems. On some class of problems, they do not guarantee a globally optimum solution. This method is used when the classical methods are too slow or fail to give a solution. This is achieved at the cost of optimality and precision for speed. In this paper we consider the following Meta-Heuristic methods.

A. Genetic Algorithm (GA)

Genetic Algorithms (GA) was invented by John Holland and has described this idea in his book “Adaptation in natural and artificial systems” in the year 1975 [6].

Genetic Algorithms are inspired by Darwin's evolutionary theory. GA comes under the class of Evolutionary algorithms that use the principle of natural selection to derive a set of solutions towards the optimal solution. It is a search heuristic which generates solutions to optimization problems using techniques inspired by natural evolution like mutation, inheritance, crossover and selection.

Here the algorithm is generally started with a set of candidate solutions called the population. Each solution in the initial population has a set of characteristics (its chromosomes or genotypes) which can be altered and mutated. Solutions from one population are taken and used to make another population, with a hope that the new population will be better than the old one [1]. Solutions are selected for breeding on the basis of their fitness. The fitness function usually identifies the number of constraints violated by a timetable. A timetable is said to be more fit if it violates less number of constraints. In the timetable generation problem, the population is a set of timetables maintained in memory. Each timetable is evaluated by finding the number of times it violates the constraints. Each timetable has an equal chance to participate in breeding.

*Algorithm **1**: Genetic Algorithm GA ( )*

```
[Start]Produce a random population of n chromosomes (suitable solutions for the problem)
    1) [Fitness] Compute the fitness f(x) of each chromosome x in the initial population.
    2) [New Population] Generate a new population by repeating the following steps until the new population is complete.
        (a) [Selection] Select two parent chromosomes from the existing population on the basis of their fitness. (Chromosomes with highest fitness are selected.)
        (b) [Crossover] Considering a crossover probability, crossover the parents to form new solution.
        (c) [Mutation] Considering a mutation probability, mutate new offspring at each locus (position in chromosome).
        (d) [Accepting] Add the new offspring in the new population.
    3) Use the newly generated population for further run of the algorithm.
    4) If the end condition is satisfied, return the existing best solution from the current population.
    5) [Loop] Go to step 2.
```

The efficiency of the genetic algorithm mainly depends on the fitness function [20, 25]. Mutation and crossover are the two main parts of the algorithm. They are called the operators of GA. The task of Crossover is the realization of the deterministic search [5]. From the previously selected pool, two parent solutions are selected for breeding. The new solution is obtained by the method of crossover and mutation and shares many characteristics of the parent solution. New parents are selected for every new child and this process continues until a population of appropriate size is generated. Only the best solutions from the previous pool are selected for breeding, along with a small number of less fit solutions to ensure genetic diversity. The creation of new population can be stopped when a solution which satisfies the minimum criteria can be found. In our case, the process can be stopped when a timetable satisfying all the hard constraints is found.

Advantages: Provides diverse values of solutions and reaches global maxima. Mutation is used to induce diversity. Saving best solution is helpful.

Disadvantages: It is complex to implement.

B. Tabu Search

Fred Glover proposed the Tabu Search in 1986. Tabu search can be directly applied to virtually any kind of optimization problem. Tabu search technique is one of the popular local search method based on neighborhood search algorithm [2].

Tabu search basically avoids getting trapped at local maxima. That is why this search allows non-improving moves when it is trapped in local optima. Another advantage of Tabu Search method is that it prevents cycling back to the previously visited solutions by the use of memories thus creating more chances of improvement.

Tabu search uses a search space which is the space of all possible solutions that can be considered. The search space could also simply be a set of feasible solutions to the problem. In Tabu search one of the basic elements is the "Tabu". Tabus are used to prevent cycling while moving away from local optima and through non-improving moves. Tabus are also used to move away from the previously visited portions of the search space and thus help in exploring other regions. Tabus are stored in a short term memory called the Tabu list. Usually the Tabu list is implemented as a circular list of a predefined length. The moves are considered as Tabu only for a certain number of iterations and this number is called the "tabu tenure". At each iteration the Tabu tenure value is randomly chosen in the range of 0.25Tb to 0.5 Tb (Tb is the square root of the total number of courses). Also sometimes the best solution found so far using the Tabu search procedure may not actually be "the best". This is because Tabus are too powerful that they might prohibit attractive moves even when there is no danger of using those moves. This may lead to overall stagnation of the search process. Thus a criterion called "Aspiration Criteria" allows canceling the Tabus. The simplest and most commonly used aspiration criterion is allowing a move, even if it is Tabu [17].

The overall quality of the timetable is evaluated based on a function of soft constraints' violation called the objective function.

The objective function f (q) is defined as:

$$f(q) = \sum w_i q_i \qquad (1)$$

where w_i and q_i are the weight and violation number of the i^{th} soft constraint, and i ranges from 1 to n (n is the total number of soft constraints). A neighborhood solution satisfies aspiration criterion if and only if f (Qnei) < f (Qbest) (Qbest is the best solution found so far).

Algorithm 2 : Tabu search - F(x)

```
1) Building an initial solution using greedy algorithm.
    (a) Non-preassigned courses are split into blocks and are assigned to appropriate courses.
    (b) Non-preassigned courses are assigned to appropriate rooms based on results of (1).
2) Applying Tabu search algorithm to improve the initial solution.
```

Some of the courses may be preassigned, that is their periods and rooms are predetermined by people, while the other courses are not. These preassigned courses are not re-scheduled again.

The searching process is stopped when it meets one of the following criteria:

The number of iterations passed is beyond a certain predefined number or when the best solution found so far satisfies all the constraints or when the number of consecutive unimproved iterations has reached a certain number [18].

The results of the timetable obtained from the Tabu Search algorithm shows significant improvements when compared to handmade results. Therefore, automated timetables obtained from Tabu Search algorithm are much better than handmade timetables due to the high improvement rate.

Advantages: It is faster than any genetic algorithm and reaches local maxima. Also it avoids local optima.

C. Iterative Local Search

The optimization problems and combinatorial search are tackled by Local Search meta-heuristics are an emerging class of methods, which was recently proved to be very effective for a large number of combinatorial problems.

The Local Search techniques are based on the iterative exploration of a solution space. At each repetition, the algorithm for Local Search moves from one solution to one of its "neighbors", i.e., solutions that are (in some sense) close to the starting one.

One major drawback of this family of techniques is the lack of robustness on a wide variety of problems. In actual fact, in so many cases, these kinds of methods guarantee finding good results in practicable run times, while in other cases Local Search techniques are caught in the local minima.

Combination of several neighborhood structures is one of the alternative approaches [12, 13] to cope with local minima.

A set of neighborhood operators are given such that for given collection of basic neighborhoods, a new compound neighborhood is automatically created and prescribe the strategies for its exploration. This approach is based Multi - Neighborhood Search.

Definition -The Course Timetabling problem

If there are a set of x courses, C = {s_1, . . . , s_x}, a set of y periods, P = {1, . . . , y}, and a set of z rooms, R = {r_1, . . . , r_z}: each course s_i consists of c_i lectures to be scheduled in distinct time periods, and is attended by p_i pupils. Each room r_j has a capacity cp_j , expressed in terms of number of seats. There are also m groups of courses, T_1, . . . ,T_m called curricula, such that there are common pupils for any pair of courses belonging to a curriculum T_i .The output of the problem is an integer-valued x × y matrix M , such that M_{ik} = j (with $1 \leq j \leq m$) means that course s_i has a lecture in room r_j at period k, and M_{ik} = 0 means that course s_i has no class in period k. The matrix M is searched such that the hard constraints are compulsorily and the soft ones are almost satisfied and the violations are minimized.

Hard constraints: (1) Lectures (2) Room Occupancy (3) Conflicts (4) Availabilities

Soft constraints: (5) Minimum working days (6) Capacity of the room (7) Compactness of the Curriculum

By using Local Search to solve a Course Timetabling problem [14] , first the basic Local Search entities should be defined, namely the cost function, the search space and the strategy for generating the initial solution. Search space consists of all the assignment matrices for which the constraints (1) and (4) hold. States for which the hard constraints (2) and (3) do not hold are allowed, but are considerably penalized within the cost function.

The cost function is thus a weighted sum of the violations of the aforementioned hard constraints plus the violations of the soft constraints (6) – (7). The weight of constraint type (5) is the number of students without a seat, whereas the weight of constraint types (5) and (7) is fixed to 5 and 2, respectively, to reflect their relative importance in our institution. Furthermore, as sketched above, in order to give precedence to feasibility over the objectives, hard constraints are assigned the weight 100, which is a value greater than the maximum value of soft constraints violations. In detail, if we denote with fi(T) the measure of violation of the constraint (i) we choose

$$F(T) = 1000 \cdot (f2(T) + f3(T)) + f5(T) + f6(T) + f7(T). \qquad (2)$$

The initial solution is selected at random. A random matrix M is created which satisfies constrains (1) and (4). This is obtained by allotting each lecture of a course to an available period which is selected randomly, and to a random room, ignoring the fact whether the lectures are assigned to the same period [14].

Advantages: Gives good results in a few cases in practical run times.

Disadvantages: Lack of robustness in a variety of problems. Also local search techniques can get stuck at local minima.

D. Greedy Random Adaptive Search Procedure (GRASP)

GRASP is a technique that is very efficient in the combinatorial optimization field of problem solving. It has been applied to various problems like spanning tree problems etc. It is a simple but yet very efficient. It can be used to construct good quality solutions and adapt for any timetabling problem with constraints that are similar.

The typical GRASP method consists of two phases i.e. solution construction and solution improvement from which best solution that found is returned when the search procedure is terminated. The solution construction mechanism consists of two main ingredients: a dynamic constructive heuristic and randomization.

For a solution that consists of a subset of a set of elements i.e. the solution components, a solution is made step by step by adding one new element at each step. The new element is picked at random from the Restricted Candidate List. At each step the heuristic values are updated, consequently changing the values at each step depending on the choices available.

This is a dynamic construction heuristic unlike a static heuristic that assigns element scores before construction.

*Algorithm **3:** Greedy Random Adaptive Search Procedure: GRASP ( )*

```
s ←0
% s denotes a partial solution in this case a←DetermineCandidateListLength ()
% definition of the RCL length While solution not complete
```

```
do
RCLa← GenerateRestrictedCandidateList(s) x ←SelectElementAtRandom(RCLa)
s ←sU{x} UpdateGreedyFunction(s)
 % update of the heuristic values end while.
```

The above algorithm shows the Greedy randomized solution construction in which 'a' is the number of elements in the RCL and determines the strength of the heuristic bias. If a=1, it will be an extreme case in which the solution is the same as a deterministic Greedy heuristic. If a=n, the construction would be completely random, of course, as element chosen from the RCL is random.

The second phase of Grasp if an improvement phase in which a basic local search algorithm can be used.

Two conditions are essential for GRASP to be effective which are that firstly the construction phase must have samples of the search space's most promising regions and secondly the solutions must belong to basins of attractions of the locally minimal solutions.[10]

To ensure that the first condition is satisfied we must have an effective construction heuristic and an appropriate length for the RCL. The second condition can be met by choosing a construction heuristic and local search algorithm that fits well [11].

Now for a Course time tabling problem we can use a GRASP procedure that constitutes a basic 3 step cycle that is repeated. It will first select a random class and assigns it resources. It then adds the next class and it will rank the options and so on. The expanding list of assigned lectures will be sorted with those that score the highest for the different criteria will move up the priority list, which constitutes the greedy part. Each subsequent lecture when added must then adapt to fit unless it scores more highly than the others in which case it moves up the list. In the second phase improve the list using a "local search procedure" that compares neighboring lectures and re-ranks them in pairs. This phase goes on till no further improvements are possible. At the end, a path re-linking strategy is used to spot the almost optimal solutions, which are then used to produce the final solution. "This basic cycle is repeated many times and the overall winning solution is returned as the final answer of the algorithm [8].

We should choose a time slot and room for a given class, then a restricted list is constructed by counting how many times the soft constraints are violated for each choice. When trying to insert a class into the timetable and a position does not exist, another class (or classes) previously scheduled is removed from the timetable to open a slot for the problem class. A feasible time slot is selected randomly and all conflicting lectures allocated in that time slot are removed from the timetable. If a conflicting lecture does not exist in time slot, a non-conflicting lecture is selected. This strategy is called explosion.

For the GRASP improvement phase, a local search is used on the initial timetable solution. The GRASP iteration (initial phase and improvement phase) is repeated several times generating different timetables. The final solution is the best of all generated timetables. Three different local search strategies are presented. The neighbors are generated with 2 movements: MOVE and SWAP. The first reschedules a lecture in an empty time slot. The second will exchange the time slots of two lectures. The algorithm stops if n consecutive neighbors are generated and the objective function is not reduced. The second local search method adapts from the breadth-first algorithm, in which the neighborhood is not explored extensively, only k neighbors are generated and the best neighbor is chosen for the next iteration. The generation and exploration of the neighborhood processes are identical to first method. The depth-first strategy is the simplest. The third local search is a depth-first strategy with a heuristic that will lessen the violation of soft constraints. A parallel version of the algorithm is also made.[9].

Advantages: It is faster than other algorithms. It can also be integrated with other search techniques fairly well.

Disadvantages: It requires large amount of memory because the problem instances and best solution found so far has to be stored.

III. CONCLUSION

Considering the above facts and also the experimental values given in numerous papers, TABU was found to outperform all other methods. But its complexity makes it very difficult to use it in a course timetabling problem in a short period of time. Hence we have decided to take the best aspects of GA and TABU and come up with an application oriented algorithm designed specifically for the needs of our college.

IV. ACKNOWLEDGMENT

The work reported in this paper is supported by the college through the Technical Education Quality Improvement Programme [TEQIP-II] of the MHRD, Government of India.

REFERENCES

[1] Sandeep Singh Rawat, Lakshmi Rajamani,“A timetable prediction for technical education system using Genetic Algorithm”,*Journal of Theoretical and Applied Information Technology.*

[2] Luca Di Gaspero and Andrea Schaerf ,“Tabu Search Techniques for Examination Timetabling” Dipartimento di Matematicae Informatica Università di Udine via delle Scienze 206, I-33100, Udine, Italy.

[3] George M. White and Bill,“Examination Timetables and Tabu Search with Longer-Term Memory” ,S. Xie School of Information Technology and Eng.,University of Ottawa, Ottawa, K1N 6N5, Canada.

[4] Salman Hooshmand,Mehdi Behshameh and Omid Hamidi, International Journal of Computer Science & Information Technology (IJCSIT) Salman, Dept. of Computer Eng, Hamedan University of Technology, Hamedan Iran.

[5] Sandor Gyori,Zoltan Petres and Annamaria R Varkonyi-Koczy,“Genetic Algorithms in Timetabling. A New Approach”,Budapest University of Technology and Economics Dept. of Measurement and Information Systems Muegyetem rkp. 9., Budapest, Hungary.

[6] Rushil Raghavjee and Nelishia Pillay “An Application of Genetic Algorithms to the School Timetabling Problem”, School of Information Systems and Technology, Pietermaritzburg Campus University of KwaZulu-Natal.

[7] Majlinda Fetaji , Bekim Fetaji and Mirlinda Ebibi ,“Using Genetic Algorithm For Solving Time Tabling Multidimensional Issues and its Performance Testing”, South East European University, Contemporary Sciences and Technologies, Ilindenska bb, 1200 Tetovo, Macedonia.

[8] “A GRASP strategy for a more constrained School Timetabling Problem”.Int. J. Operational Research, 2010, 7, 152-170.

[9] “PATAT”. International timetabling competition . http://www.utwente.nl/ctit/itc2011/(2011).

[10] *Handbook of Applied Optimization* . Resende.Greedy Randomized Adaptive Search procedure”. In P.M. Pardalos and M.G.C. Resende, editors, pages 168.183. Oxford University Press, 2002.

[11] T. A. Feo and M. G. C. Resende. “Greedy randomized adaptive search procedures”. *Journal of Global Optimization*, 6:109.133, 1995.

[12] L.DiGaspero “Recolour, shake and kick are for the Examination Timetabling Problem” by E.Burke and P. DeCausmaecker, editors, Proc. Of the *4th International Conference on the Practice and Theory of Automated Timetabling*, August 2002,pages 404 – 407.

[13] L.DiGaspero “Recolour, shake and kick are for the Examination Timetabling Problem” by E.Burke and P. DeCausmaecker, editors, Proc. Of *the 4th International Conference on the Practice and Theory of Automated Timetabling*,August 2002, pages 128 – 132.

[14] A.Scharef,“A survey of Automated Timetabling-Artificial Intelligence Review”, 1999, 13(2):87-127.

[15] Ruey-Maw Chen and Hsiao-Fang Shih, “Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search” Dept. of Computer Science and Information Eng., National Chinyi University of Technology, Taichung, Taiwan.

[16] S.C. Chu and H.L. Fang ,“Genetic Algorithms vs. Tabu Search in timetable Scheduling”, IEEE, 1999 ,Electronics & Communication dept., University of South Australia, Australia.

[17] Khang Nguyen, Nguyen Dang and Khon Trieu, “Automating a Real-World University Timetabling Problem with Tabu Search Algorithm”, IEEE, 2011, Faculty of Information, University of Science, Vietnam.

[18] Amol C. Adamuthe and Rajankumar S Bichkar, “Tabu Search for Solving Personal Scheduling Problem”, IEEE, 2011, Dept. of Computer Science Eng., Rajaramnagar-Islampur MS, India.

[19] Turabieh and Salwani Abdullah, “Incorporating Tabu Search into Memetic Approach for Enrolment-based Course Timetabling Problems”, *IEEE 2nd Conference on Data Mining and Optimization*, Hamza Centre for Artificial Intelligence Technology, University Kebangsaan Malaysia, October 2009.

[20] Ms.Premalatha A Sonawane, Dr.Leena Raghu, “Hybrid Genetic Algorithm and Tabu Search Algorithm to solve class time table scheduling problem “, *Intenational Journal of Research Studies in Computer Science and Engg.*

[21] F. Glover and M. Laguna. Tabu search. Kluwer Academic Publishers, 1997.

[22] Cagdas Hakan Aladag and Gulsum Hocaoglu ,"A Tabu search Algorithm To solve a Course Timetabling Problem", *Hacettepe Journal of Mathematics and Statistics,Vol. 36 (1) ,pp. 53 64, (2007).*

[23] S.N.Sivanandam, S.N.Deepa,"Introduction to Genetic Algorithms", *Springer Berlin Heidelberg New York,2008* .

[24] Rushil Raghavjee, Nelishia Pillay, "Using Genetic Algorithms to Solve the South African School Timetabling Problem",IEEE, Second World Congress on Nature and Biologically Inspired Computing, in Kitakyushu, Fukuoka, Japan, pp.286-292, Dec. 15-17,(2010).

[25] Thamilselvan, R. and P. Balasubramanie, "Integration of Genetic Algorithm with Tabu Search for Job Shop Scheduling with Unordered Sub sequence Exchange Crossover",Journal of Computer Science, Vol.8 (5):, pp. 681-693, (2012).

JETIR