A Literature Review on Timetable Generation Algorithms

Tanuja Padale\textsuperscript{1}, Harshada Mane\textsuperscript{2}, Manisha Bhagat\textsuperscript{3} & Shobha Nangare\textsuperscript{4}
Department Of Computer Engineering
P.E.S Modern College Of Engineering, Pune(India).

Abstract: The problem of timetable scheduling is described as a highly constrained NP-hard problem. It is known as the timetabling problem by most researchers. Planning timetable is one of the most complex and error prone task. A lot of complex constraints need to be addressed for development of an efficient algorithm to solve this problem. There are still serious problems like generation of high cost timetable. Therefore there is a great requirement for an application distributing the course evenly and without collisions. Our aim here is to develop a simple, easily understandable, efficient and portable application, which could automatically generate good quality time tables within seconds. In this paper, we have described the different techniques that have been developed for timetable generation.

Keywords:- Hard & Soft Constraints, Simulated Annealing, Local Search, Branch & Bound, Genetic algorithm, heuristic approach, Naive Bayes, probability.

1. Introduction

Timetabling Problem is an NP-Hard problem, which is very difficult to solve by using conventional methods and the amount of computation required to find optimal solution increases exponentially with problem size. The main idea of this problem is to assign a set of lectures to rooms and time periods satisfying a number of constraints. The set of constraints are usually divided into hard constraints and soft constraints.

The hard constraints considered for this problem are:

1. No students can attend more than one lecture at a time.
2. No lecturer can teach more than one subject at a time.
3. No classroom can occupy more than one lecture at a time.
4. There must be one batch in a Lab at a time.

The soft constraints considered for this problem are:

1. No consecutive lectures of the same teacher in a class.
2. Courses must be evenly distributed
3. Same teacher must not have consecutive periods unless specified
4. Teachers daily lecture hours should be limited
5. Students should not have any free time between two classes on a day.

Hard constraints have a higher priority than soft. The objective of this problem is to satisfy the hard constraints and to minimize the violation of the soft constraints. It is therefore necessary to use efficient search methods to produce optimal or near optimal timetable that satisfy the all constraints.

2. Literature Survey

There exist various timetable generation problems such as Employee Timetabling, University Timetabling, Sports Timetabling and Examination Timetabling. A large number of diverse methods have been already proposed for solving timetabling problems. These methods come from a number of scientific disciplines like Operations Research, Artificial Intelligence, and Computational Intelligence and can be divided into four categories:

1) Sequential Methods :: Sequential Methods, that deals timetabling problems as graph problems. Generally, they order the events using domain-specific heuristics and then assign the events sequentially into valid time slots in such a way that no constraints are violated for each timeslot.

2) Constraint Based Methods :: according to which a timetabling problem is modeled 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.

3) Cluster Methods :: In which the problem is divided into a number of events sets. Each set is defined so that it satisfies all hard constraints. Then,
the sets are assigned to real time slots to satisfy the soft constraints as well.

4) Meta-heuristic methods :- Meta-heuristic methods, are 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. Meta-heuristic methods, such as genetic algorithms (GAs), simulated annealing, tabu search, and other heuristic approaches.

2.1 Simulated Annealing

Simulated annealing is a local-search meta-heuristics that takes inspiration from annealing in metallurgy. The annealing is a process of heating and controlled cooling of a material to improve its quality.

2.1.1 Local search algorithm

As a local-search algorithm, it needs an initial solution to be provided to begin. It iterates number of times using a neighbor generation method that creates a solution basing on the already existing one. The simplest local-search algorithm simply checks whether the new solution is better than the already known one and uses the better one. This type of algorithm tends to get stuck in local optimum. A set of meta-heuristic is known for changing this algorithm to avoid this disadvantage( simulated annealing is one of the examples). A number of ending criteria types are known for this type of algorithms. To name a few:

- Algorithm iterates a fixed number of time
- Solution did not improved during last n iterations
- After a number of successful iteration
- The quality of known solution satisfies the needs

2.1.2 Simulated Annealing concept

The simulated annealing allows the standard local-search algorithm to pick a worse solution. This approach allows the algorithm to escape from the local optimum. The possibility that a worse solution is accepted decreases with the time of the algorithm. This is exactly the main idea of annealing in metallurgy. High temperature give the atoms possibility to change their positions. As the temperature decreases, these changes are less likely to happen. It was empirically tested that atoms with the ability to change position find the equilibrium which increases the features of material. The local-search algorithm begins with an initial temperature that is gradually decreased. In most implementation there are a number of levels and once some criteria are satisfied, the temperature is changed. The criteria that change the temperature level are similar to the ending criteria of plain local-search algorithm. The simulated annealing usually ends when it is no longer possible to decrease the temperature.

2.1.3 Class Teacher Problem Implementation

The simulated annealing was chosen by Abramson to solve the timetabling problem. His implementation always picks the new solution if its quality is better than current one. If the solution is worse, the probability of picking equals:

\[ P (\Delta c) = e^{-\frac{\Delta c}{T}} \]

The temperature level is changed after a fixed number of successful swaps. The solution is stored in a set of lesson lists( one list for each time period). The neighbor is generated by picking a lesson from one period and moving it to another one. Both periods are picked randomly. Only the difference in schedule cost is evaluated by removing the cost of the element removed and adding the cost of added element.

The authors emphasized the importance of not swapping two lectures across two periods but simply moving one. Unlike the first approach, this one allows to change the number of lectures in particular period.

2.2 Tabu search

Tabu search is another type of local-search meta-heuristic. It improves the standard local-search to reduce the chance of being stuck in a local optimum. Unlike the Simulated Annealing, Tabu Search does not use probability so extensively. It evaluates every solution in neighborhood and always picks the best one. The algorithm keeps a tabu list(a list of moves that are not permitted), to minimize the time spent in a local minimum. After each move used, it is added to the tabu list to prevent the algorithm from going back. The tabu list is a short-term memory. After spending a tenure ( the amount of time provided as the execution parameter) the move is removed from tabu list. To assure that the optimal solution will not be prevented by the tabu list, the concept of aspiration criteria is introduced. If the solution satisfies the aspiration criteria, the solution can be used even if it is created by a prohibited move. Usually all the short time memory is not enough and algorithms may get stuck around local optimum. A diversification strategy is needed to allow the algorithm to escape from a local minimum. The long time memory is often maintained in order to be used for such a case.

2.3 Branch & Bound

A branch and bound algorithm searches the entire search space but uses bounds on the function
to be optimized and the current best solution to search through the search space. A dynamically search tree is created using unexplored subspaces which are represented as nodes. There are three major components that are used: node selection, bound calculation and branching. Branch and bound was used by Busetti to solve a timetabling problem at Purdue University with the option of users having the ability to make adjustments interactively. Another use of branch and bound was used by Friedrich, Hofsa B, and Wekeck in scheduling transit assignments and reduced the computational time from 32.5 hours (shortest path search) to 1.6 hours (branch and bound search). According to Fang branch and bound is not suitable for large timetabling problems.

2.4 Genetic Algorithm (GA)
Genetic Algorithms (GA) was invented by John Holland. 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. 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. 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 . 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.

2.5 Heuristic approach
In computer science, artificial intelligence, and mathematical optimization, a heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy or precision for speed. In a way, it can be considered a shortcut. A heuristic function, also called simply a heuristic, is a function that ranks alternatives in search algorithms at each branching point based on available information to decide which branch to follow. The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. This solution may not be the best of all the actual solutions to this problem, or it may simply approximate the exact solution. But it is still valuable because finding it does not require a prohibitively long time. Heuristics may produce results by themselves, or they may be used in conjunction with
optimization algorithms to improve their efficiency (e.g., they may be used to generate good seed values). Results about NP-Hardness in theoretical computer science make heuristics the only viable option for a variety of complex optimization problems that need to be routinely solved in real-world applications.

2.6 Naive Bayes

It is a classification technique based on Bayes Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

We are generating the timetable using naive bayes algorithm. Here we are calculating the probability of occurrence of particular subject, if probability is higher than threshold value then break the algorithm and again regenerate the new timetable, So that all subject evenly occurs.

Probability=Number of time particular subject occur/total number of subject

Existing system were generating timetable by satisfying constraints such as No lecturer can teach more than one subject at a time, No classroom can occupy more than one lecture at a time etc. But they were not considering occurrences of subject, so we are overcoming this drawback by using naive bayes algorithm.

3. Conclusion

Considering the above facts and experimental values given in numerous papers, Naive Bayes was found to outperform all other methods. This algorithm ensured all subjects occurs for same number of time.

4. Acknowledgement

A special thanks to Prof. Mrs. Milana Malvadkar for their guide and support.

5. References


[2]. Suryanti Awang, Mohamad Fairol, Marzuki Khalid, Rubiyah Yusof, "Generating University Timetable by using a Combination of Heuristic and Random Selection", 2009 IEEE.


[14]. G. Kendall and N. M. Hussain, “A Tabu Search Hyper Heuristic Approach to the Examination Timetabling Problem at the MARA University of
[15]. Do Xuan Duong, Pham Huy Dien, “Solving the Lecture Scheduling Problem by the Combination of Exchange Procedure and Tabu Search Techniques,” Studia Informatica Universalis, Vol.4, Number 2.
