An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach

Anirudha Nanda, Manisha P. Pai, and Abhijeet Gole

Abstract— **This paper proposes a general solution for the School timetabling problem. Most heuristic proposed earlier approaches the problem from the students' point of view. This solution, however, works from the teachers' point of view i.e. teacher availability for a given time slot. While all the hard constraints (e.g. the availability of teachers, etc.) are resolved rigorously, the scheduling solution presented in this paper is an adaptive one, with a primary aim to solve the issue of clashes of lectures and subjects, pertaining to teachers.**

***Index Terms*— time tabling, scheduling, operational research, artificial intelligence, heuristic.**

I. Introduction

The class timetabling problem is a scheduling algorithm with great interest and implications in the fields of operational research and artificial intelligence. The problem was first studied by Gotlieb, who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students and one teacher, such that the combination of teacher and students can be chosen freely [1]. Dynamic changes in the context of timetabling problems, had started to be studied at [23]. A survey of existing approaches to dynamic scheduling can be found in [24].Because of the size of real problem, almost all effective solutions are heuristic in nature, and do not guarantee optimality. Among the well known results there are [13-16] that deal with various cases of the problem settings [17]. While setting a timetable, importance is given to effective utilization of resources such as the classroom,the teacher, etc.` This becomes a very tedious task which needs to be addressed at least once a year by every academic institute. Most institutes deal with this problem manually, i.e. a trial and error method is used to set a timetable.

The general timetabling problem comes in many different guises including nurse rostering (e.g. Cheang et al, 2003, Burke et al. 2004), sports timetabling (e.g. Easton, Nemhauser and Trick, 2004), transportation timetabling (e.g. Kwan, 2004) university timetabling (Schaerf, 1999, Burke and Petrovic, 2002, Petrovic and Burke, 2004), etc [3, 4, 5, 6]. A more indirect approach can also be seen in instruction scheduling. [12] Due to the combinatorially explosive nature of the problem, enumeration and other deterministic methods fail and heuristics is preferred [2]. Since the Timetabling problem is a common problem faced

in most walks of life, its presence with other operational research problems cannot be overlooked. For e.g., in scheduling sports timetables, consideration is not only given to scheduling a sport event, but also to reducing the cost factor, distance traveled by teams, etc. The Traveling tournament [18], where the total distance traveled by the team is minimized, further increased the academic interest in sports scheduling [19].

Manuscript received on July 1 10, 2012; revised July 30, 2012.

The authors was with the Departmnent of Computer Science, Ramnarain Ruia College, Mumbai, India (email: anirudhananda@gmail.com; manisha.ppai@gmail.com; abhijeet_gole@hotmail.com)

II. Background

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 is 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).

In a general educational timetabling problem, a set of events (e.g. courses and exams, etc) are assigned into a certain number of timeslots (time periods) subject to a set of constraints, which often makes the problem very difficult to solve in real-world circumstances [2]. In fact, large-scale timetables such as university timetables may need many hours of work spent by qualified people or team in order to produce high quality timetables with optimal constraint satisfaction [7] and optimization of timetable's objectives at the same time.

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 [8]. Some of these include Genetic Algorithms (GA) [9], Tabu Search [10], Simulated Annealing [11] and recently used Scatter Search methods.

Heuristic optimization methods are explicitly aimed at good feasible solutions that may not be optimal where complexity of problem or limited time available does not allow exact solution. Generally, two questions arise (i) How fast the solution is computed? and (ii) How close the solution is to the optimal one? Tradeoff is often required between time and quality which is taken care of by running simpler algorithms more than once, comparing results obtained with more complicated ones and effectiveness in comparing different heuristics. The empirical evaluation of heuristic method is based on analytical difficulty involved in the problem's worst case result. In its simplest form the scheduling task consists of mapping class, teacher and room combinations (which have already been pre- allocated) onto time slots.

One possible approach is as follows: We define a tuple as a particular combination of identifiers such as class, teacher and room, which is supplied as an input to the problem.[20] The problem now becomes one of mapping of tuples onto period slots such that tuples which occupy the same period slot are disjoint (have no identifiers in common). If tuples are assigned arbitrarily to periods, then in anything but the most trivial cases, a number of clashes will exist. We can use the number of clashes in a timetable as an objective measure of the quality of the schedule. Thus, we adopt the number of clashes as the cost of any given schedule. It is simple to measure the cost of a schedule. For each period of the week, we make a count of the number of occurrences of each class, teacher and room identifier. The cost of the entire timetable is the sum of each of the individual costs. This procedure is discussed in more detail in Abramson [21].

The proposed algorithm aids solving the timetabling problem while giving importance to teacher availability. This algorithm uses a heuristic approach to give a general solution to school timetabling problem. It takes the user input of a number of subjects, number of teachers, subjects every teacher takes, number of days in a week for which the timetable needs to be set, number of time slots in a day and the maximum lectures a teacher can conduct in a week.

It initially uses randomly generated subject sequence to make a temporary time table. While generating this sequence, care is taken to avoid repetition of subjects over a day. After this, the teacher availability for each of the subjects allocated for the respective slot is checked. Every time a teacher is available for the subject at the allocated slot, the subject and the teacher are entered into the output data structure and marked as final. Before the allocation of this subject to the output data structure, a check is also conducted on the number of maximum lectures a teacher can conduct. If the teacher has been allocated more than the allowed maximum lectures the subject is moved into a Clash data structure.

To avoid cycling and to improve the search, this variable selection criterion can be randomized. There are several methods [22] which can be applied,

e.g.: – a random walk technique (with the given probability p a random variable is selected)

– not the worst variable, but a random selection of a variable worse enough (e.g., from the top N worst variables), or – a selection of a variable according to a probability based on the above mentioned criteria (e.g., roulette wheel selection).

A. Collision

There is a possibility that teacher availability for a subject s_i may be at a slot where another subject s_j is allocated. Under such a situation, if s_j is not present in the output data structure, s_j is moved into a Clash data (i.e. no more free time slots are available), the Clash data structure is revisited and an effort is made to allocate the subjects in it to an available time slot in the day. If, however, it is not possible to allocate any/all of the subjects in the Clash data structure, these subjects are moved to the Day_Clash data structure. When sequence for the next day is generated preference is given to the subjects under Day_Clash.

B. Variables Used

Time slots of the time tables:- ts_1, ts_2, ts_3…., ts_n
List of Subjects:- s_1,s_2,s_3, …., s_n
Teachers:- t_1,t_2,t_3, …., t_n
Batches of students:- c_1,c_2,c_3, ….., c_n
Flags indicating finalized timeslots :- tsf_1, tsf_2, tsf_3, ….., tsf_n
Data structure to hold Final Timetable:-final_tt
Count for day of week: Daycount
Number of days of the week:- n
Data structure to hold Subject-clash within the day:- clash
Each element of Clash data structure:- clash_ele
Data structure for Subject-clash across days:- Dayclash
Each element of Dayclash data structure:- day_clash_element
Subject contained in dayclash:- s_{dc}
Teacher associated with subject in dayclash:- t_{dc}
Max number of lectures of subject s_i in the week:-k
Counter for the number of subjects:- counter_sub
Random number indicating random slot allotment for subject:- rand_sub_allot
Data structure to hold randomly allotted subject:- rand_sub_seq
Data structure to hold all subjects:- init_sub

C. Assumption

This algorithm is designed to solve and generate school time tables. The following is a list of assumptions made while developing this algorithm:

- The algorithm produces optimum outputs in a five-day week.
- The number of subjects (s1, s2, …, sn) need to be finalized before the algorithm begins execution.
- Number of teachers (t1, t2, …, tn) entered before execution of the algorithm are assumed to be constant and cannot be changed during or after the algorithm has been executed.
- Any change in the above two assumptions will require a new generation of Timetable for the changed data.
- In each time table, all time-slot is filled with, a unique combination of subjects without any repetition of subjects.

- Any teacher is allowed at most 'k' number of lectures in a week. The value of k is accepted before execution of the algorithm.
- It is assumed that a teacher cannot take more than one lecture for the same class in a day.
- Timeslots ts1, ts2, … ,tsn once entered at the beginning cannot be changed throughout the execution.
- Every day in the week is assumed to have equal number of time slots.
- Classrooms for any batch id fixed throughout the day.

D. User-Input

Time slots of the time tables:- ts_1, ts_2, ts_3…. Ts_n
List of Subjects:- s_1,s_2,s_3, …., s_n
Teachers:- t_1,t_2,t_3, …., t_n
Max number of lectures of subject s_i in the week:- k
Batches of students:- c_1,c_2,c_3, ….., c_n
Count for day of week: Daycount

III. Algorithm

```
If(daycount>n)
  end
generate()
Lbl2:For  day_clash_element 1 today clash element n
       Retrieve $s_{dc}$ from dayclash
  Retrieve $t_{dc}$ of $s_{dc}$
rehabilitate($s_{dc}$)
For($ts_1$ to $ts_n$)
  If($s_i$ exists in final_tt)
     Next iteration
  Else
     Lbl3:Retrieve $s_i$ in $ts_i$
     Retrieve $t_i$ of $s_i$
    If(availability =0 for $ts_i$)
        rehabilitate($s_i$)
    Else
         If($k_i$>0)
                 Set $s_i$ to $ts_i$ in final_tt
                 $tsf_i$=1
                 $k_i$ - -
   Else
   Lbl3
     If (tsn has been reached)
        If(clash NOT empty)
                For clash_ele$_1$ to clash_ele$_n$
            Retrieve each $s_i$ inclash_ele$_n$
            Retrieve $s_i$ with $t_i$
            rehabilitate($s_i$)
            Daycount++
        Else
            Daycount++

                Else
        Continue

generate()

For (each subject $s_i$)
  Place $s_i$ in sub_arr
  For(each dow)
          rand_seq=rand(sub_arr)
            if(dow=1)
                 init_tt (dow)=rand_seq
            else
                 curr_pos=length(rand_seq)-1
                 temp_ele=rand_seq(last)
                for (each element in rand_seq)

                  if(curr_pos==1)
                        rand_seq(curr_pos)=temp_ele
                  else

                  rand_seq(curr_pos)=rand_seq(curr_pos-1)
                  init_tt(dow)=rand_seq
End generate

rehabilitate($s_i$)
     Lbl4:Retrieve $ts_j$ such that $t_i$ availability =1

         If($tsf_j$=0)

                  If($k_i$>0)

                           Move $s_j$ to Clash
                           Set $t_i$,$s_i$ in $ts_j$ in final_tt
                           $tsf_i$=1
                           $k_j$—

                  else
                            Lbl4

                           Else if(tsfj=1)
                                Lbl4

                           Else If(all $tsf_j$=1)
                                Mark $s_j$
                               Move $s_j$ to Dayclash
End  rehabilitate
```

IV. Result

	Mon	Tue	Wed	Thurs	Fri
TS1	Mr. A/ Maths	Mr. A/ Physics	Mr. B/ History	Ms. C/ Geog	Ms. D/ Chem
TS2	Ms. D/ Chem	Mr. A/ Maths	Mr. A/ Physics	Mr. B/ History	Ms. C/ Geog
TS3	Ms. C/ Geog	Ms. D/ Chem	Mr. A/ Maths	Mr. A/ Physics	Mr. B/ History
TS4	Mr. B/ History	Ms. C/ Geog	Ms. D/ Chem	Mr. A/ Maths	Mr. A/ Physics
TS5	Mr. A/ Physics	Mr. B/ History	Ms. C/ Geog	Ms. D/ Chem	Mr. A/ Maths

Following are the results of the implementation of the algorithm mentioned above.

- The algorithm after implementation, results in the creation of a time table of batch/class of students displaying a grid of time slots.
- Each time slot is filled by a teacher and the subject that is being conducted. The output of the algorithm's implementation will be as per the above Table 1.
- The allotments of teachers to the slots will change the composition of the generated time table.

- Hence, all clashes of availability of teachers will be analyzed and the algorithm will be applied again to improve by reducing the clashes.

V. Conclusion

The intention of the algorithm to generate a time-table schedule automatically is satisfied. The algorithm incorporates a number of techniques, aimed to improve the efficiency of the search operation. It also, addresses the important hard constraint of clashes between the availability of teachers. The non-rigid soft constraints i.e. optimization objectives for the search operation are also effectively handled. Given the generality of the algorithm operation, it can further be adapted to more specific scenarios, e.g. University, examination scheduling and further be enhanced to create railway time tables. Thus, through the process of automation of the time-table problem, many an-hours of creating an effective timetable have been reduced eventually.

The most interesting future direction in the development of the algorithm lies in its extension to constraint propagation. When there is a value assigned to a variable, such assignment can be propagated to unassigned variables to prohibit all values which come into conflict with the current assignments. The information about such prohibited values can be propagated as well.

References

[1] D. Datta, Kalyanmoy Deb, Carlos M. Fonseca, "Solving Class Timetabling Problem of IIT Kanpur using Multi- Objective Evolutionary Algorithm." *KanGAL* 2005.

[2] Edmund K Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, Rong Qu, "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems." *European Journal Operational Research*, 176: 177-192, 2007.

[3] Awad and Chinneck, "Proctor Assignment" at *Carleton University* (1998).

[4] Cheang B., Li H., Lim A. and Rodrigues B. 2003, "Nurse Rostering Problems: A Bibliographic Survey." *European Journal of Operational Research*, 151(3) 447-460.

[5] Easton K., Nemhauser G. and Trick M. 2004, "Sports Scheduling." In: Leung J. (ed.) in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Chapter 52, CRC Press.*

[6] S. and Burke E.K. 2004. University Timetabling In: Leung J. (ed.) "Handbook of Scheduling: Algorithms", "Models, and Performance Analysis." Chapter 45. *CRC Press.*

[7] W. Legierski, "Constraint-based Techniques for the University Course Timetabling Problem", *CPDC, (2005*), pp.59-63.

[8] S. Abdullah, E. K. Burke and B. McCollum, "A Hybrid Evolutionary Approach to the University Course Timetabling Problem", *Proceedings of the IEEE Congress Evolutionary Computation*, Singapore, (2007).

[9] J. F. Gonçalves and J. R. De Almeida, "A Hybrid Genetic Algorithm for Assembly Line Balancing", *Journal of Heuristics*, Vol.8, (2002), pp.629-642.

[10] G. Kendall and N. M. Hussain, "A Tabu Search Hyper Heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology", *Lecture Notes in Computer Science, Springer Verlag,* vol.3616, (2005), pp.270-293.

[11] M. A. Saleh and P. Coddington, "A Comparison of Annealing techniques for Academic Course Scheduling", *Lecture Notes in Computer Science, Springer Verlag*, vol. 1408, (1998), pp.92-114.

[12] Aubin J, Ferl and J. A, "A Large Scale Timetabling Problem", *Comput. & Opr. Res.*, vol.16, no.1, pp.67-77, 1989.

[13] Dempster M. A. H, "Two algorithms for the timetable problem," *Proc, of Conference, Oxford*, July 1969, pp. 63-8.

[14] Dinkel J. J, Mote J, Venkataramanan M. A, "An efficient decision support system for academic course scheduling," *Operations Re-search* 37(6), 1989, pp. 853-864.

[15] Gotlieb C.C, "The construction of class - teacher timetables", *IFIP Congress* 62, 1962, pp.73- 77.

[16] Do Xuan Duong ,Pham Huy Dien, " Solving the Lecture Scheduling Problem by the Combination of Exchange Procedure and Tabu Search Tecniques," *Studia Informatica Universalis*, Vol.4, Number 2

[17] Easton, K., Nemhauser, G. and Trick, M, " The traveling tournament problem: description and benchmarks." In *Proc of the* 7th. *International Conference on Principles and Practice of Constraint Programming, Paphos*, 580–584, 2001

[18] K. Nurmi, D. Goossens, T. Bartsch, F. Bonomo, D, Briskorn, G. Duran, J. Kyngäs, J. Marenco, C.C. Ribeiro, F. Spieksma, S. Urrutia and R. Wolf, "A Framework for a Highly Constrained Sports Scheduling Problem," *In Proc of the International MultiConference of Engineers and Computer Science 2010 Vol III, IMECS 2010,Hong Kong.*

[19] D. Abramsom, M. Krishnamoorthy and H. Dang, "Simulated Annealing Cooling Schedules for the School Timetabling Problem" *Aisa-Pacific Journal Of Operational Research, (1999).*

[20] Abramson D. A., "Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms", *Management Science*, 37(1), 98-113, 1991.

[21] Z. Michalewicz and David B. Fogel, "How to Solve : Modern Heuristics." *Springer, 2000*.

[22] A. Elkhyari, C. Gu´eret, and N. Jussien, "Solving dynamic timetabling problems as dynamic resource constrained project scheduling problems using new constraint programming tools. In Edmund Burke and Patrick De Causmaecker, editors, *Practice And Theory of Automated Timetabling, Selected Revised Papers*," pp. 39–59. Springer- Verlag LNCS 2740, 2003.

[23] W. Kocjan, " Dynamic scheduling: State of the art report." Technical *Report* T2002:28, SICS, 2002.