P1 : Timetable Management System for an Academic Institution

Review 1 Report

Team 7 - DB4C

Aathish Sivasubrahmanian [3122 21 5001 001]

Harshida S P, [3122 21 5001 031]

Ayshwarya B, [3122 21 5001 017]

30.05.2022

FUNDAMENTALS OF PROGRAMMING AND SOFTWARE DEVELOPMENT

INTRODUCTION

Timetable planning is a complex NP-Hard problem that usually can be broken down into two distinct phases:

1. Determination of curricula and resources that can be allotted
2. Allocation of resources to those who require it, keeping in mind any hard or soft constraints that may be applied.

It is generally agreed that the second phase is where the use of computerisation is most welcome[1], as it requires manipulating several distinct variables in order to fit a varying number of constraints, which is difficult to accomplish by hand.

This report will discuss the various pre-existing methodologies and tools that allow a user to automatically generate a student/teacher timetable given valid resources to allott (such as Classrooms, Physical Labs, Faculty etc), and constraints (both soft and hard) to follow, as well as the basic knowledge that a developer needs to build such a tool.

TECHNIQUES USED

Timetable planning can be modeled as a constraint-satisfaction problem, and can be solved by many approaches, such as genetic algorithms, graph colouring and heuristic search.[2]

1. Genetic Algorithms

   The genetic algorithm is a method based on natural selection, the mechanism that drives biological evolution, for addressing both constrained and unconstrained optimization problems. A population of individual solutions is repeatedly modified by the genetic algorithm.
   Due to the inherent ambiguity in deciding the fitness function (the function determining the suitability/fitness of a particular solution produced) of the genetic algorithm, rather than relying entirely on genetic algorithms, a hybrid approach, using both genetic algorithms and constraint based reasoning is often employed. [2]

   - Hybrid Approaches
     In a hybrid approach, by using standard crossover and mutation operators, as well

as an appropriate fitness function, the genetic algorithm is employed to create viable solutions. The validity and legality of a solution are then determined using constraint-based reasoning processes, followed by an appropriate repair procedure to enhance any infeasible solutions.

2. Graph Colouring Algorithms

   The assignment of a minimal number of colours (timeslots) to the vertices (nodes) of a graph to represent events (courses) in such a way that two vertices are connected with different colours to avoid time conflicts is referred to as graph colouring. The graph's edges (constraints) express disagreement between events. As a network grows in size (number of nodes), finding an optimal solution gets more difficult since it takes exponential time, making it a hard issue. The approach entails accurately colouring the nodes of the graph and using it to build a schedule that is free of conflicts.[7]

3. Heuristic Search

   The solution process is divided into six steps, with the first four forming the computer system. Between stages, the end-user must approve that the partial solution is acceptable with manual changes. The first two phases place blocks 'together' (that is, in the same slot as each other) without specifying which slots they should go in. Stage 1 is mostly for blocks which are more challenging to schedule.
   Stage 2 introduces type blocks that occur more frequently than once a week. The precise slots are determined in Stage 3, and the remaining non-freestanding blocks are included in Stage 4. Manual stages 5 and 6 are available. Freestanding lessons are timetabled in Stage 5 and rooms are assigned at Stage 6. Final manual changes may be made as the end-user sees fit.[4]

4. Constraint Logic Programming

   If CLP is employed, Time periods and rooms can be represented in terms of integers inside given domains in a CLP language that uses finite domains. Most constraints may therefore be combined as arithmetic constraints. The overall layout of a schedule programme, like other CLP programmes used to solve Constraint Satisfaction Problems, would first specify the domains of the problem variable, then state the constraints, and then produce the values. Soft constraints may be enforced using techniques such as value

ordering heuristics and special variable heuristics,[10] or by hierarchical constraint logic programming, or by objective functions, where each soft constraint may be assigned different weights and importances[9].

PRE-EXISTING TOOLS

There are various free and paid applications that allow users to construct schedules automatically or semi-automatically (iteratively). Many tools provide a graphical interface that guides the user through the schedule creation process, allowing users to rapidly discover school resources and timetable constraints.

1. Skolaris

   Skolaris allows the end-user to input a list of faculty, the subjects they handle, and class-groups that they will teach. This information is further augmented by user input parameters such as:

   i. The latest and earliest time of starting and ending the academic day.
   ii. The minimum and maximum number of periods every day.
   iii. Busy and idle period clustering
   iv. Lunch and recess break timing
   v. Teaching load and cover

   among others.

   The timetable is then generated automatically, and the user is allowed to make manual changes if necessary.

2. Edusys

   Edusys allows the end-user to supply their system with several parameters, such as:

   i. Shift start and end times
   ii. Working days per week
   iii. Faculty count per course
   iv. Periods per day
   v. Forenoon and Afternoon period durations.

   among others. The timetable is then generated on demand automatically. The end-user

can modify the generated timetable as necessary in order to make final adjustments.

REFERENCES

1. D. de Werra, An introduction to timetabling, European Journal of Operational Research, Volume 19, Issue 2, 1985, Pages 151-162, ISSN 0377-2217, https://doi.org/10.1016/0377-2217(85)90167-5.
2. Safaai Deris, Sigeru Omatu, Hiroshi Ohta, Puteh Saad, Incorporating constraint propagation in genetic algorithm for university timetable planning, Engineering Applications of Artificial Intelligence, Volume 12, Issue 3, 1999, Pages 241-253, ISSN 0952-1976, https://doi.org/10.1016/S0952-1976(99)00007-X.
3. S. Abdullah and H. Turabieh, "Generating University Course Timetable Using Genetic Algorithms and Local Search," 2008 Third International Conference on Convergence and Hybrid Information Technology, 2008, pp. 254-260, https://doi.org/10.1109/ICCIT.2008.379
4. Wright, M. School Timetabling Using Heuristic Search. J Oper Res Soc 47, 347–357 (1996). https://doi.org/10.1057/jors.1996.34
5. https://skolaris.net/
6. https://www.edusys.co/en-in/school-management-software.html
7. Kambi, Maria & Gilbert, David. (1996). Timetabling in Constraint Logic Programming. https://www.researchgate.net/publication/2309710
8. https://egrove.olemiss.edu/cgi/viewcontent.cgi?article=1442&context=etd
9. Tacadao, Grace. (2010). A Constraint Logic Programming Approach to the Course Timetabling Problem Using ECLiPSe. https://www.researchgate.net/publication/230707225_A_Constraint_Logic_Programming_Approach_to_the_Course_Timetabling_Problem_Using_ECLiPSe
10. Rudová, H., Murray, K. (2003). University Course Timetabling with Soft Constraints. In: Burke, E., De Causmaecker, P. (eds) Practice and Theory of Automated Timetabling IV. PATAT 2002. Lecture Notes in Computer Science, vol 2740. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45157-0_21