M-Sched: A University Course Timetabler

2 authors:

Shahadat Hossain
University of Lethbridge

Minhaz Zibran
Idaho State University

M-Sched: A University Course Timetabler

Shahadat Hossain[1] and Minhaz Fahim Zibran[2]

[1] University of Lethbridge, shahadat.hossain@uleth.ca
[2] University of Calgary, mfzibran@ucalgary.ca

1 Introduction

In this paper we present a software implementation for solving the university course timetabling problem. Our timetabling implementation schedules instructors and courses taking into account the instructors' preferences on courses, days, and times. Typically, instructors have different levels of expertise in different areas. Furthermore, the instructors might have personal preferences on courses, and the times of days the courses are offered. On the other hand, for various reasons the administration may have timetabling goals conflicting with those preferences. Therefore, the "instructor-course assignment" is an important subproblem in any course timetabling exercise. Fully automated timetablers are often not useful when the constraints cannot lead to a feasible solution [8]. Hence, a high level of flexibility and user involvement is necessary to resolve such issues during the timetabling process. Courses offered by different academic departments often have interdependencies. So, flexibility to examine subdivision of events is also important [5]. As timetabling requirements widely vary over academic institutions, it is extremely hard, if not impossible, to develop a generale purpose black-box timetabler [5, 6]. Therefore, having the flexibility of dynamic customization of timetabling constraints is useful.

Our timetabling implementation focuses on course timetabling in academic departments at the University of Lethbridge. Instructors indicate which courses they would like to teach, as well as the day and time of day (morning or afternoon) they prefer to teach. Professors teach only lectures, academic assistants (lab instructors) conduct labs and tutorials, and teaching assistants (graduate students) conduct only labs. We decompose the entire problem into several smaller subproblems and solve them separately in a sequence of phases. In phase-1a, we assign lectures to professors. Phase-1b assigns labs and tutorials to academic assistants and teaching assistants. In phase-2, lectures are allotted to days. Then we allocate time-slots to lectures in phase-3. Finally, phase-4 assigns labs and tutorials to week-days and available time-slots. At each phase, the objective is to maximize a set of preferences subject to the given constraints. All architectural complexities are hidden behind a carefully designed graphical user interface. Our implementation allows the user to customize constraints as well as to generate new solutions extending the partial solutions from perviously generated timetables.

2 Software Architecture

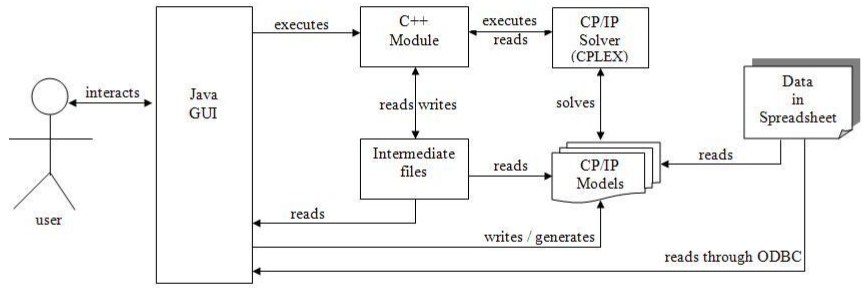

Fig. 1. software architecture of the timetabling implementation.

Our software implementation has a modular architecture as shown in Figure 1. Necessary input data are provided via MS Excel spreadsheets. For each of the subproblems we have separate constraint programming (CP) or integer programming (IP) models written in OPL [2]. ILOG's CPLEX solver is used to solve these CP/IP models. The C++ module invokes the solver to solve the appropriate CP/IP model in response to user's activity in the graphical user interface (GUI). The GUI, as shown in Figure 2, is implemented using Java following the ergonomics and usability guidelines of Human Computer Interaction (HCI) [7]. This makes the GUI user-friendly for the purpose of constraint customization (see Figure 3), as well as generation and modification of schedules.

The modular implementation using object oriented techniques makes our timetabling tool scalable and easily modifiable. For example, the CPLEX solver may be replaced by any solver without much affecting the other modules. Moreover, the software architecture conforms the MVC (model-view-controller) pattern [1] of software engineering paradigm. The GUI constitute the 'view', the C++ module along with the solver works as the 'controller', whereas the 'model' portion includes the spreadsheets and CP/IP models.

3 Conclusion

The multi-phase approach [3, 9] allows us to work on a smaller problem at a time. Besides, it enables us to exploit the problem structure of each phase and apply different solution strategies (CP or ILP) as appropriate. One of the objectives of this work is to provide the user enough flexibility so that customized schedules can be produced in a user-friendly way. In the current implementation the software permits addition or removal of constraints on the fly, loading and modifying a previously saved solution, and computing a new solution from a partial

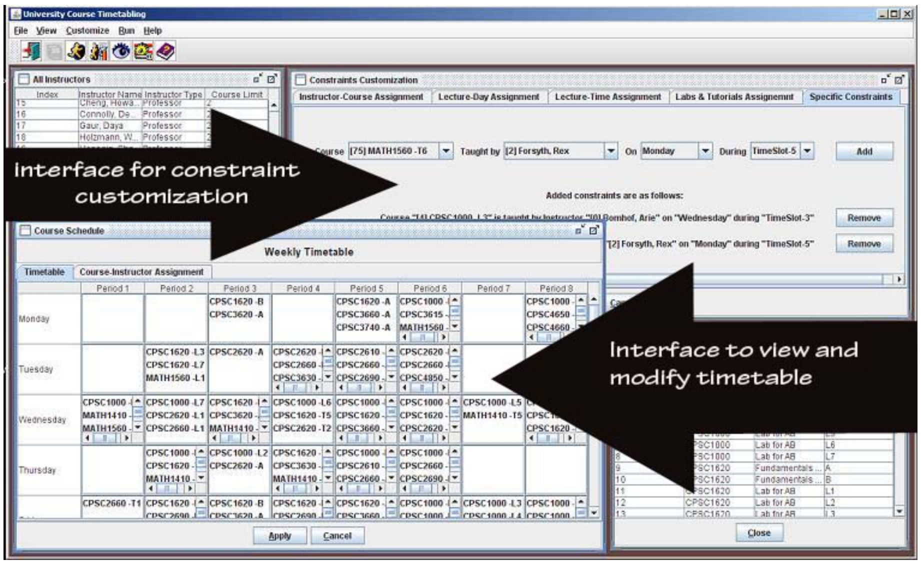

Fig. 2. A snapshot of user interface of the timetabling implementation.

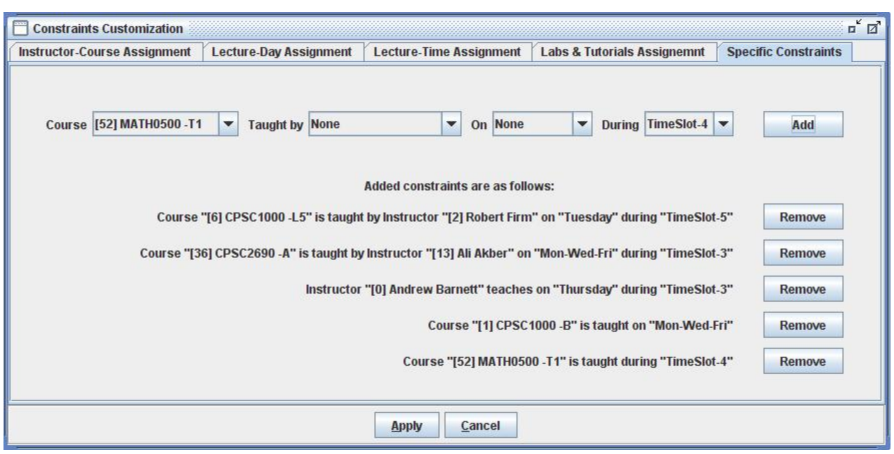

Fig. 3. GUI for constraint customization.

solution. In such cases, the new solution is attained quickly as it is not necessary to solve the problem from scratch [4]. Since the phases are solved separately, partial solutions may be generated, examined and amended. Furthermore, the graphical user interface on top of the actual computational modules makes our timetable implementation flexible and easy to use.

References

1. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns–Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. ISBN 0201633612. 21st Printing, November 2000.
2. Pascal Van Hentenryck and et al. *Constraint Programming in OPL*. International Conference on Principles and Practice of Declarative Programming (PPDP-99), Paris, France.
3. Shahadat Hossain and Minhaz Zibran. *A Multi-phase Approach to the University Course Timetabling Problem*. In the proceedings of the 6th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, The Netherlands, 2007. J.L. Hurink, W. Kern, G.F. Post, and G.J. Still (Eds). pp. 73 – 76, ISSN: 1574 - 0846.
4. T. Müller, R. Bartàk, H. Rudovà. *Minimal Perturbation Problem in Course Timetabling*. Practice and Theory of Automated Timetabling, Selected Revised Papers, pp. 126 – 146. Springer-Verlag LNCS 3616, 2005.
5. Barry McCollum. *University Timetabling: Bridging the Gap between Research and Practice*. PATAT 2006, pp. 15 – 35. ISBN 80-210-3726.
6. Keith Murray and Tomáŝ Müller. *Automated System for University Timetabling*. PATAT 2006, pp. 536-541. ISBN 80-210-3726-1.
7. Jacob Nielsen. *Usability Engineering*, pp. 115 – 163, Academic Press, 1993.
8. Sylvain Piechowiak, Jingxua Ma, and René Mandiau. *An Open Timetabling Tool*. PATAT 2004. LNCS 3616, pp. 34 – 50, 2005.
9. Minhaz Fahim Zibran. *A Multi-phase Approach to University Course Timetabling*. M.Sc. Thesis, Department of Mathematics and Computer Science, University of Lethbridge, Canada, 2007.