A Modular Multiphase Heuristic Solver for Post Enrolment Course Timetabling

Marco Chiarandini[1], Chris Fawcett[2], and Holger H. Hoos[2,3]

[1] University of Southern Denmark,
Department of Mathematics and Computer Science,
Odense DK-5230, Denmark
[2] University of British Columbia,
Department of Computer Science,
Vancouver, BC, Canada
[3] Corresponding Author
E-mail: hoos@cs.ubc.ca
Phone: +1 604 822-1964
Fax: +1 604 822-5485

Abstract. We give a short description of the solver that ranked third in Track Two of the International Timetabling Competition 2007 (ITC2007). It implements a heuristic approach based on stochastic local search and consists of several modules that were found to be useful in different phases of the solution process. Common to all modules is the consideration of only a subset of the constraints that have to be satisfied. The solver is the result of an engineering process conducted with the aid of ParamILS, a recent tool for automated algorithm configuration. A discussion on this process and the underlying methodology is also provided. A remarkable property of our solver is the ability to consistently find feasible solutions to all of the instances from ITC2007, outperforming the other submissions by this measure.

Keywords: [post enrolment course timetabling, stochastic local search, automated algorithm configuration]

We address the Post Enrolment Course Timetabling problem as defined by Lewis et al. (2007) for the Track 2 of the International Timetabling Competition (ITC2007). Our solver builds on previous work by Chiarandini et al. (2006) and consists of several heuristic modules that have been tuned and assembled using the automated algorithm configuration procedure ParamILS (Hutter et al., 2007b). By using ParamILS throughout the development of our solver, we have been able to explore a large design space of hybrid stochastic local search algorithms and to construct heuristics in a manner similar to that described by Hutter et al. (2007a). At a high level our solver consists of two main procedures: a hard constraint solver and a soft constraint violation minimizer.

The *hard constraint solver* consists of a constructive phase followed by a stochastic local search phase. The constructive phase generates feasible partial assignments of lectures to time slots and rooms by using an approach similar to that of Arntzen and Løkketangen (2003). The heuristic assigns lectures to time slots in such a way that the five hard constraints are satisfied. Specifically, first a topological order of the precedence graph is constructed and the lectures are sorted accordingly (ties are broken randomly). Then, lectures are considered in that order and each is placed into a time slot that is feasible and that can accommodate the fewest yet unscheduled lectures. The feasibility of the insertion of a lecture in a time slot with respect to room assignment is checked by computing the exact matching of the lectures and rooms scheduled so far in that time slot. Lectures without a feasible insertion point are left in a list of unscheduled lectures.

The construction is repeated a number of times, taking advantage of the existence of multiple topological orderings to vary the insertion order. For easier problem instances, this procedure is often sufficient to obtain a complete feasible assignment. In general, however, we may be left with some unassigned lectures. These are handled in a second search phase, which uses a tabu search procedure based on the PARTIALCOL algorithm by Blöchliger and Zufferey (2008). At each iteration of this procedure, an unscheduled lecture is inserted into the best non-tabu time slot for it, and all lectures breaking any hard constraint as a result of the insertion are moved into the list of currently unscheduled lectures. The selection of the lecture to be inserted at each step is guided by evaluating the number of students attending each unscheduled lecture, which is the contribution of that lecture to the distance-to-feasibility measure of Lewis et al. (2007). After a fixed number of non-improving iterations, the best partial feasible solution is perturbed by applying the soft constraint violation minimizer. The procedure continues alternating tabu search on the unscheduled lectures and soft constraint optimization until a feasible solution is found or a given time limit is reached. On instances for which feasibility is reached relatively quickly, we additionally use a look-ahead procedure consisting of a fast soft constraint optimization, and repeat the entire procedure a number of times before selecting the assignment to exploit for the minimization of soft constraint violations.

The *soft constraint violation minimizer* is applied to the assignment returned by the hard constraint solver. It consists of local search procedures using

1-exchange, 2-exchange, swap-of-time-slots and Kempe chains neighborhoods; these have all been previously described by Chiarandini et al. (2006) and are applied in the same order as in their work. Some of these procedures include an exact matching of rooms at every change. In its final phase, the soft constraint minimizer applies simulated annealing on the 1- and 2-exchange neighborhoods with exact room reassignment.

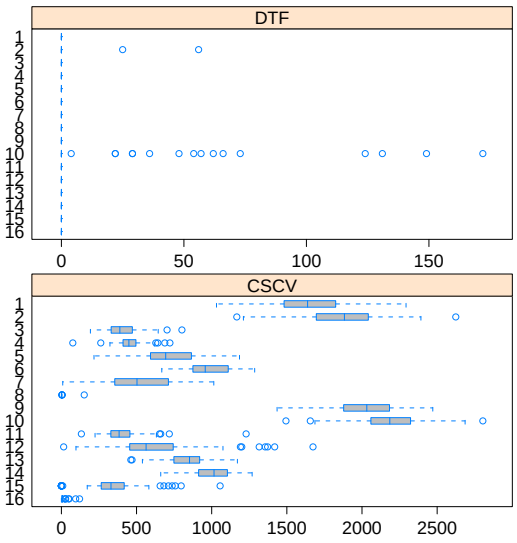
The final solver, as submitted to the competition, is the result of an engineering process that uses an automated algorithm configuration procedure to guide decision making during the construction of the algorithm, as well as in the final configuration and tuning. Generally, designers of heuristic algorithms have to make crucial choices at many levels. At a high level, the representation of candidate solutions and the overall search strategy has to be decided. At an intermediate level, concrete search procedures and search neighborhoods (including the ordering used within construction heuristics) need to be chosen; at an even more detailed level, the values of the parameters controlling the behavior of search heuristics and procedures, such as the tabu tenure and tabu acceptance level in our hard constraint solver, have to be determined. The capability of human designers to manually explore all of these choices is limited, and consequently, much can be gained by automating this aspect of algorithm development and design.

ParamILS (Hutter et al., 2007b) is a tool that performs local search in the space of parameter configurations of a given algorithm in order to achieve optimal (or near-optimal) performance on a set of problem instances. It currently supports parameters with discrete, finite domains; these parameters can be either numerical or categorical, where categorical parameters are used primarily for choices between alternative options or components, such as search heuristics. ParamILS also supports nested parameters and can hence deal with situations in which the choice of one component introduces additional parameters of this component into the configuration process, as happens often in the design of complex stochastic local search algorithms. The procedure to be configured is considered to be a black box, and ParamILS does not need any specific information about how it operates or about the problem it is solving.

Our experimental setting, in terms of problem instances, evaluation measure and time limit used, was designed to comply with the rules of the competition. Automated configuration using ParamILS was performed on a cluster of machines, each with a dual-core Intel Xeon 3.20Ghz CPU and 2GB of RAM. Each run of ParamILS used only one machine and processor core, and the cluster was used to perform multiple runs in parallel. The entire iterative process of designing and testing our solver took one month, during which a total of about 1000 CPU hours were used for ParamILS runs. In general, each tuning phase used five to ten machines in parallel for ten to twenty hours.

In Figure 1, we report the statistics of 100 runs of our solver on the 16 instances that were made public before the submission date, using one of our machines. Each run was limited to 384 seconds, the maximum runtime determined by the benchmarking tool provided for ITC2007. Our solver finds feasible

solutions in all but a small number of runs, particularly for instance comp-2007-2-10. In Figure 2, we compare the best five solvers submitted to the ITC2007 Track 2 (there were thirteen submissions overall). The figure is based on the official ITC2007 results available at http://www.cs.qub.ac.uk/itc2007/. According to an aggregate rank analysis that, compliant with the evaluation rules defined for ITC2007, takes into account both the distance-to-feasibility and the score for soft constraint violations, our solver ranks clearly behind the the winning entry by Cambazard, Hebrard, O'Sullivan and Papadopoulos. However, focusing only on the feasibility problem (left plot in the figure) we see that our solver actually performs best. This is consistent with the fact that the hard constraint solver was the primary focus of our development process and confirms the success of our approach. Making the design of the soft constraint minimizer more modular and allowing ParamILS to explore the subsequent configuration space is the subject of current and future work. Preliminary work in this direction has already improved the overall performance of the solver substantially.



instance	DTF	CSCV
comp-2007-2-1.tim	0	925
comp-2007-2-2.tim	0	1156
comp-2007-2-3.tim	0	179
comp-2007-2-4.tim	0	66
comp-2007-2-5.tim	0	52
comp-2007-2-6.tim	0	536
comp-2007-2-7.tim	0	7
comp-2007-2-8.tim	0	0
comp-2007-2-9.tim	0	1480
comp-2007-2-10.tim	0	1364
comp-2007-2-11.tim	0	166
comp-2007-2-12.tim	0	1
comp-2007-2-13.tim	0	360
comp-2007-2-14.tim	0	576
comp-2007-2-15.tim	0	0
comp-2007-2-16.tim	0	0

Fig. 1. Left side: boxplots of results on the 16 public instances obtained from 100 runs of our solver; DTF indicates the distance-to-feasibility and CSCV the cost of soft constraint violations. Right side: table with the best results achieved by our solver, as submitted to the competition.

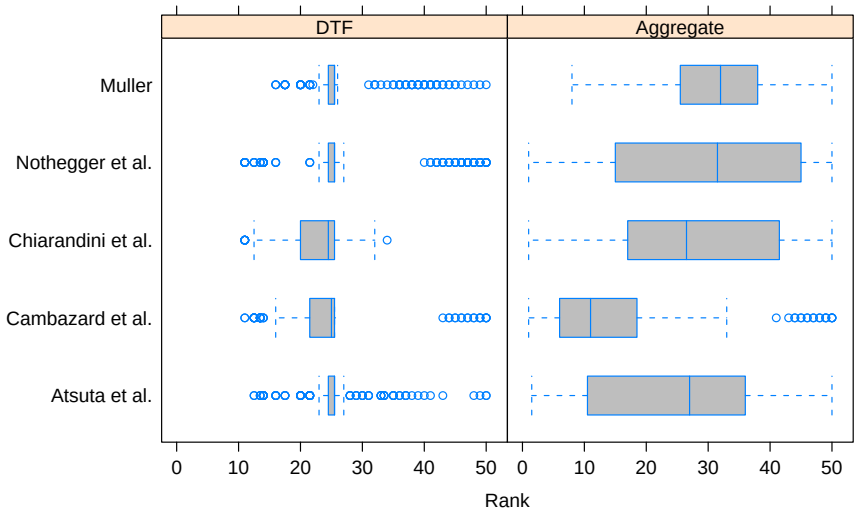


Fig. 2. Performance results of the the best five top-ranking solvers from ITC2007. Absolute values are transformed into ranks in order to remove bias due to different scales of the instances. The boxplots indicate the distribution of ranks attained by each solver in 10 runs for each of the 24 instances. Left side: Results based on distance-to-feasibility (DTF) only. Right side: Aggregate results, compliant with the rules of the competition. Note that the vertical lines inside the boxes indicate the median values of the distributions; the final ranking of the competition was determined, instead, on the basis of average values, which are the following: Cambazard et al. 13.90417; Atsuta et al. 24.42708; Chiarandini et al. 28.33958; Nothegger et al. 29.51667; Muller 31.31250.

Bibliography

Arntzen, H. and Løkketangen, A. (2003). A tabu search heuristic for a university timetabling problem. In *Proceedings of the Fifth Metaheuristics International Conferance*.

Blöchliger, I. and Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Comput. Oper. Res.*, 35(3):960–975.

Chiarandini, M., Birattari, M., Socha, K., and Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *J. of Scheduling*, 9(5):403–432.

Hutter, F., Babić, D., Hoos, H. H., and Hu (2007a). Boosting verification by automatic tuning of decision procedures. In *Formal Methods in Computer Aided Design (FMCAD'07)*.

Hutter, F., Hoos, H. H., and Stützle, T. (2007b). Automatic algorithm configuration based on local search. In *Proc. of the Twenty-Second Conference on Artifical Intelligence (AAAI '07)*, pages 1152–1157.

Lewis, R., Paechter, B., and McCollum, B. (2007). Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition. Technical report, Second International Timetabling Competition.