ITC2007: Solver Description

Tomáš Müller

Purdue University, West Lafayette IN 47907, USA
muller@unitime.org

Abstract. This paper outlines a description of a solver that was used for all three tracks of the International Timetabling Competition 2007[1].

1 Introduction

The primary objective in the construction of the search algorithm for this timetabling competition was to use the Constraint Solver Library [3]. This library contains a constraint based framework for local search based algorithms that operate over feasible, though not necessarily complete, solutions. In these solutions some variables may be left unassigned. All hard constraints on assigned variables must be satisfied however. The library is written in Java and is publicly available under GNU LGPL licence. Moreover, it has also been successfully applied to several large scale practical timetabling problems, for instance, it is used in the timetabling system that is used for university course timetabling at Purdue University [6, 7], see `http://www.unitime.org` for more details. Currently, we are also developing student sectioning and examination timetabling algorithms using this library. Another goal was to use the same algorithm for all three tracks of this competition, with only minimal changes (different problem model, neighborhoods, solver parameters, etc.).

2 Algorithm

The search algorithm consist of several phases: In the first (construction) phase, a complete solution is found using an Iterative Forward Search (IFS) algorithm [4] that is utilizing Constraint-based Statistics (CBS) [5] to prevent itself from cycling. In the next phase, a local optimum is found using a Hill Climbing (HC) algorithm. When a solution can no longer be improved, Great Deluge (GD) technique [1] is used. The GD algorithm is altered so that it allows some oscillations of the bound that is imposed on the overall solution value. Optionally, Simulated Annealing (SA) technique [2] can be used between bound oscillations of GD algorithm.

The search ends after a given time limit and the best found solution is returned.

[1] For more details see the official competition website at `http://www.cs.qub.ac.uk/itc2007`.

2.1 Construction Phase

At first, a complete solution is found using Iterative Forward Search algorithm [4]. It starts from all variables being unassigned. In each iteration, an unassigned variable (that is a lecture, a class, or an exam) is assigned with a value (assignment of a room and a time). Since it may cause some violations of hard constraints with the existing assignments, all conflicting variables are unassigned as well. This means, for instance, that if there is another class in the selected room at the selected time, such class gets unassigned.

This search ends when all variables are assigned and it is parametrized by variable and value selection criterions. First, it tries to find the worst unassigned variable. A variable is randomly selected among unassigned variables with the smallest domain size versus number of hard constraints ratio, however, some more problem-based criterion can be used as well. Then, it tries to select the best value that is to be assigned to the selected variable. A value whose assignment increases the overall solution value the least is selected among values that violate the smallest number of hard constraints (i.e., we minimize the number of conflicting variables that will need to be unassigned in order to make the problem feasible after the selected value is assigned to the selected variable). If there are more than one of such values, one is selected randomly. It is also possible to completely ignore soft constraints in this phase, a value is then selected randomly among values that are minimizing the number of violated hard constraints.

Moreover, Conflict-based Statistics [5] is used. This technique prevents repetitive assignments of the same values by memorizing conflicting assignments. Conflict-based statistics is a data structure that memorizes hard conflicts which have occurred during the search together with their frequency and assignments which caused them. More precisely, it is an array

$$CBS[V_a = v_a \ \rightarrow \ \neg\, V_b = v_b] = c_{ab}.$$

It means that the assignment $V_a = v_a$ caused c_{ab} times a hard conflict with the assignment $V_b = v_b$ in the past. Note that it does not imply that these assignments $V_a = v_a$ and $V_b = v_b$ cannot be used together in case of non-binary constraints. In the value selection criterion, each hard conflict is then weighted by its frequency, i.e., by the number of past unassignments of the current value of the conflicting variable caused by the selected assignment.

2.2 Hill Climbing Phase

Once a complete solution is found, Hill Climbing algorithm is used in order to find the local optimum. In each iteration a change in the assignment of the current solution is proposed by a random selection from a problem-specific neighborhood. The generated move is only accepted (change is made to the current solution) when it does not worsen the overall solution value (i.e., the weighted sum of violated soft constraints). Only changes that do not violate any hard constraints are considered in this phase as well as in the following phases. Also, these neighbor assignments are generated in the same way. That is, a problem specific neighborhood is selected randomly (with given probability among several neighborhoods that are created for the particular problem) and it is used to generate a random change in the current solution.

The hill climbing phase is finished after a certain number of idle iterations, i.e., when a solution is not improved during the last HC_{idle} iterations. Parameter HC_{idle} may be defined differently for each particular competition track.

2.3 Grat Deluge Phase

Great Deluge algorithm [1] is using a bound B that is imposed on the overall value of the current solution that the algorithm is working with. This means that the generated change is only accepted when the value of the solution after the assignment does not exceed the bound. The bound starts at value

$$B = GD_{ub} \cdot S_{best}$$

where S_{best} is the overall value of the best solution found so far, and GD_{ub} is a problem specific parameter (upper bound coefficient). The bound is decreased after each iteration (it is multiplied by a cooling rate GD_{cr}).

$$B = B \cdot GD_{cr}$$

The search continues until the bound reaches the lower limit, that is, ${GD_{lb}}^{at} \cdot S_{best}$, where GD_{lb} is a parameter defining lower bound coefficient. When this lower limit is reached, the bound is reset back to its upper limit, that is, ${GD_{ub}}^{at} \cdot S_{best}$.

$$B < {GD_{lb}}^{at} \cdot S_{best} \;\Rightarrow\; B = {GD_{ub}}^{at} \cdot S_{best}$$

Parameter at is a counter starting from 1. It is increased by one every time after the lower limit is reached and the bound is increased. It is also reset back to 1 once a best found solution is improved. This helps the solver to wider the search when it cannot find an improvement, allowing it to get out of a deep local minima.

2.4 Simulated Annealing Phase

Simulated Annealing algorithm [2] is using a temperature T. A generated neighbor assignment is accepted when it is not worsening the overall value of the current solution or with the following probability otherwise

$$p_{accept} = e^{-T/\Delta}$$

where Δ is the increase of the overall value of the current solution when the worsening move is assigned. Temperature T starts at initial temperature SA_{it}. Temperature is cooled down (multiplied by cooling rate SA_{cr}) after each $SA_{cc} \cdot$

TL iterations (SA_{cc} is a cooling coefficient), where TL is an instance specific number (temperature length), computed as a sum of domain sizes of all variables. If the best found solution is not improved after $SA_{rc} \cdot SA_{cc} \cdot TL$ iterations (SA_{rc} is a reheat coefficient), temperature is increased to

$$T = T \cdot {SA_{cr}}^{-1.7 \cdot SA_{rc}}$$

In the case when simulated annealing is used, great deluge phase is stoped after the bound B reaches its lower limit and the control is passed to this simulated annealing phase. Similarly, the control is passed from this phase to hill climber phase just after the temperature is reheated (see Figure 1). When simulated annealing is not used, the control is never passed from great deluge phase (GD immediately continues after the bound is increased).

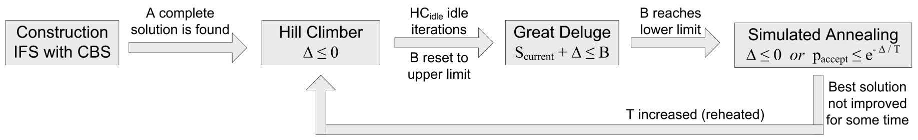
Construction IFS with CBS
A complete solution is found
Hill Climber $\Delta \leq 0$
HC_{idle} idle iterations
B reset to upper limit
Great Deluge $S_{current} + \Delta \leq B$
B reaches lower limit
Simulated Annealing $\Delta \leq 0$ or $p_{accept} \leq e^{-\Delta / T}$
Best solution not improved for some time
T increased (reheated)

Fig. 1. Algorithm schema when Simulated Annealing is used as well.

3 Competition Tracks

Individual neighborhoods, above described algorithm parameters (see Table 1) and other competition track specific aspects of the solver are discussed in this

Table 1. Solver parameters for each competition track

Parameter	Track 1	Track 2	Track 3
HC_{idle}	25,000	50,000	50,000
GD_{ub}	1.12	1.10	1.15
GD_{lb}	0.90	0.90	0.90
GD_{cr}	$1 - \frac{1}{9 \cdot 10^6}$	$1 - \frac{1}{5 \cdot 10^6}$	$1 - \frac{1}{7 \cdot 10^6}$
SA_{it}		1.5	2.5
SA_{cr}		0.97	0.82
SA_{cc}		5	7
SA_{rc}		7	7

section. As it is mentioned above, in hill climbing, great deluge, and simulated annealing phase, an assignment change (neighbor assignment) is randomly generated from the bellow described neighborhoods. Only moves that do not violate any hard constraint are generated. At first, a neighborhood is selected, it is then used to generate a change that is assigned when accepted by the currently used search strategy (HC, GD, or SA).

3.1 Track 1: Examination Timetabling

The following neighborhoods are used during the examination timetabling (all of them are selected with the same probability). All of the proposed neighborhoods try to change an assignment of an exam or to swap two exams in either period, room, or both. If a new change cannot be made, it tries to systematically search for another change that follows the proposed one (e.g., try to use one of the following periods or rooms) rather than randomly generating and checking of some other change.

Exam swap An examination is randomly selected. A new period and a room is randomly selected. If there is no (hard) conflict in assigning the selected exam into the new period and room, the new assignment is returned. If there is one conflicting examination and if it is possible to swap the selected examination with the conflicting one, such a swap is returned. Following rooms and periods are tried otherwise (both rooms and periods are considered in the order they were loaded from the input file; first all rooms are considered for the selected period, then for the next period etc.). Only periods and rooms that are valid for the selected exam are considered.

Period change An examination and a new period is randomly selected. If there is no conflict in assigning the selected exam into the new period (keeping its room assignment), the new assignment is returned. The following periods are tried otherwise. First available period is returned, another neighborhood is tried if no such period can be found.

Room change An examination and a new room is randomly selected. If there is no conflict in assigning the selected exam into the new room (keeping its period assignment), the new assignment is returned. The following rooms are tried otherwise.

Period swap An examination and a new period is randomly selected. If there is just one conflict in assigning the selected exam into the new period (keeping its room assignment) and it is possible to swap these two exams (each keeping its room assignment and taking period assignment of the other exam), such a swap is returned. The following periods are tried otherwise.

Room swap An examination and a new room is randomly selected. If there is just one conflict in assigning the selected exam into the new room (keeping its period assignment) and it is possible to swap these two exams (each keeping its period assignment and taking room assignment of the other exam), such a swap is returned. The following rooms are tried otherwise.

Period and room change An examination and a new period is randomly selected. If there is no conflict in assigning the selected exam into the new period (into a randomly selected available room), the new assignment is returned. The following periods are tried otherwise.

Examination timetabling solver is not using simulated annealing phase. Generation of the above described neighbor changes takes much more time than in other tracks due to the complexity of the imposed soft constraints. When combined with the imposed time limit for each instance, various test runs indicated that the time is better spent just using great deluge approach.

3.2 Track 2: Post Enrollment based Course Timetabling

In order to be able to find a complete solution quickly, soft constraints are ignored during construction phase. Also, unlike in other tracks, it is allowed to assign an event into a time slot without assigning it into a room (e.g., in the case when there is no room available at the proposed time), or violate a precedence constraint. Both no room assignments and violated precedence constraints are treated as soft constraints. The algorithm starts with a weight of these soft constraints being one (overall solution value is the given score plus the weighted sum of violated no-room and precedence constraints), and these weights are increased by one after every 1,000 iterations during which there is no improvement in the number of violations of these constraints. This helps the solver to gradually decrease the number of these violations while it is looking for the best solution score.

The following neighborhoods are used during the post enrollment based course timetabling (all of them are selected with the same probability, except of *Precedence Swap* that is selected with $\frac{1}{10}$ probability of the others).

Time Move An event and a new time slot is selected randomly. If it is possible to reassign the event into the new time slot while keeping its room without any conflict, such an assignment is returned. Following time slots are tried otherwise, first available (if any) time slot is returned.

Room Move An event and a new room is selected randomly. If it is possible to reassign the event into the new room while keeping its current time assignment without any conflict, such an assignment is returned. Following rooms are tried otherwise, first available (if any) room is returned. Only rooms that are valid for the selected event are considered (i.e., rooms that are of sufficient size and that have all the required features).

Event Move An event is randomly selected. A new time slot and a room is randomly selected. If there is no conflict in assigning the selected event into the new time and room, such an assignment is returned. If there is exactly one event conflicting with the new assignment and it is possible to swap these events, such a swap is returned. Otherwise it tries to use one of the following time slots and rooms (first it keeps the selected time slot and picks another room, then the same with the following time slot, etc.).

Event Swap Two events are randomly selected. If it is possible to swap these two events, such a swap is retuned. Otherwise it tries to swap the times but pick a different room for these events (in a similar way as *Room Move*).

Precedence Swap This neighborhood tries to decrease the number of violated precedence constraints by reassigning an event into a different time and room. A violated precedence constraint is selected randomly, one of its events (randomly selected) is placed in a different time and room that does not violate the selected constraint. The time and room is picked in the same way as in *Event Swap* neighborhood. If no time and room can be found for the selected event, the other event is tried to be moved as well.

3.3 Track 3: Curriculum based Course Timetabling

The following neighborhoods are used during the curriculum based course timetabling (all of them are selected with the same probability, except of *Curriculum Compactness Move* that is selected with $\frac{1}{10}$ probability of the others).

Time Move A period is changed for a randomly selected lecture. First not conflicting period after a randomly selected one is used.

Room Move A room is changed for a randomly selected lecture. First not conflicting room after a randomly selected one is returned.

Lecture Move A lecture is selected randomly, a new time and room is selected for the lecture. If there is no conflict in assigning the selected lecture into the selected time and room, such an assignment is returned. If there is another lecture conflicting with the time and room and it is possible to swap these two lectures, such a swap is returned. Following times and rooms are tried otherwise. Only times that are available for the course of the selected lecture are considered.

Room Stability Move This neighborhood tries to find a change that decreases room stability penalty. A course and a room is selected randomly. It tries to assign all lectures of the course into the selected room. If there is already some other lecture in the room, it is reassinged to the room of the moving lecture.

Min Working Days Move This neighborhood tries to find a change that decreases course minimum working days penalty. A course with a positive penalty is selected randomly, a day on which two or more lectures are taught is selected and one of lectures of that day is moved to a day that the course is not being taught on.

Curriculum Compactness Move This neighborhood tries to find a change that decreases curriculum compactness penalty. A curriculum is selected randomly, a lecture that is not adjacent to any other one of the curriculum is selected, and placed to another available period that has an adjacent lecture of the curriculum (if such placement exists and does not create any conflict). A different room may be assigned to the lecture if the current one is not available as well.

4 Results

The following results were achieved with the above described solver. The tables below present results for all three tracks of the competition, computed within given time limit, including early and late instances. The best solutions of 100 runs of each instance are presented.

Table 2. Results of Track 1 (Examination Timetabling)

Instance Number	1	2	3	4	5	6	7	8
Two Exams in a Row	42	0	1275	7533	40	3700	0	0
Two Exams in a Day	0	10	2070	3245	0	0	0	0
Period Spread	2534	0	5193	3958	1361	19900	3628	6718
Mixed Durations	100	0	0	0	0	75	0	0
Larger Exams Constraints	260	380	840	105	1440	375	460	380
Room Penalty	1150	0	0	0	0	1250	0	125
Period Penalty	270	0	190	1750	100	475	0	342
Overall Value	4356	390	9568	16591	2941	25775	4088	7565

All results have zero *Distance to Feasibility* (i.e., a complete feasible solution was found), except of one instance of Track 2 (Post Enrollment based Course Timetabling), see Table 3 for more details.

Table 3. Results of Track 2 (Post Enrollment based Course Timetabling)

Instance Number	1	2	3	4	5	6	7	8
Distance to Feasibility	0	0	0	0	0	0	0	0
More than Two in a Row	728	1093	73	111	0	8	2	0
One Class on a Day	23	21	132	283	0	0	3	0
Last Time Slot of a Day	579	1040	0	0	0	5	0	0
Overall Value	1330	2154	205	394	0	13	5	0
Instance Number	9	10	11	12	13	14	15	16
Distance to Feasibility	0	57	0	0	0	0	0	0
More than Two in a Row	881	1268	118	169	70	2	0	2
One Class on a Day	16	33	177	233	1	0	0	4
Last Time Slot of a Day	998	1139	52	51	3	0	0	0
Overall Value	1895	2440	347	453	74	2	0	6

Table 4. Results of Track 3 (Curriculum based Course Timetabling)

Instance Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Room Capacity	4	0	0	0	0	0	0	0	0	0	0	0	0	0
Minimum Working Days	0	15	10	5	180	15	0	5	35	5	0	255	10	5
Curriculum Compactness	0	28	62	30	114	26	14	34	68	4	0	76	56	48
Room Stability	1	0	0	0	4	0	0	0	0	0	0	0	0	0
Overall Value	5	43	72	35	298	41	14	39	103	9	0	331	66	53

5 Conclusion

At this point, it is hard to estimate how successful the presented approach will be. However, the comparison with other solvers could be quite valuable, especially since in the presented work the same solver was used to solve all three tracks of the competition. And also since the presented approach is built on a framework and techniques that are currently being used to solve real world, large scale course timetabling problems.

Various test runs with different solver settings and other changes to the above described neighborhoods were performed. The above described algorithm and its parameters represent the best achieved results. However, there is most likely still enough space in various optimizations of the solver behavior and parameters.

For more information about the presented approach, including source codes, please visit `http://www.unitime.org/itc2007`.

References

[1] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to record travel. *Journal of Computational Physics*, 104:86–92, 1993.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[3] Tomáš Müller. Constraint solver library. GNU Lesser General Public License, SourceForge.net. Available at `http://cpsolver.sf.net`.

[4] Tomáš Müller. *Constraint-based Timetabling.* PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2005.

[5] Tomáš Müller, Roman Barták, and Hana Rudová. Conflict-based statistics. In J. Gottlieb, D. Landa Silva, N. Musliu, and E. Soubeiga, editors, *EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics.* University of Nottingham, 2004.

[6] Keith Murray, Tomáš Müller, and Hana Rudová. Modeling and solution of a complex university course timetabling problem. In Edmund Burke and Hana Rudová, editors, *Practice And Theory of Automated Timetabling, Selected Revised Papers*, pages 189–209. Springer-Verlag LNCS 3867, 2007.

[7] H. Rudová T. Müller, R. Barták. Minimal perturbation problem in course timetabling. In Edmund Burke and Michael Trick, editors, *Practice And Theory of Automated Timetabling, Selected Revised Papers*, pages 126–146. Springer-Verlag LNCS 3616, 2005.