A Generalized Threshold Accepting Approach for Timetabling

Martin Josef Geiger

University of Hohenheim, 70593 Stuttgart, Germany
mjgeiger@uni-hohenheim.de,
WWW home page: http://www.martingeiger.de/

Abstract. We briefly describe a generalized local search approach for the solution of timetabling problems in general, with a particular implementation for competition track 3 of the International Timetabling Competition 2007. The heuristic search procedure is based on Threshold Accepting to overcome local optima. A stochastic neighborhood is proposed and implemented, removing and reassigning events from the current solution.

1 Introduction

While the here presented solution approach is generic and may be applied to other timetabling problems without the need of bigger adaptations, the following explanations are made in the light of the specific constraints and objectives of the International Timetabling Competition 2007, track 3.

2 Solution Approach

2.1 Preprocessing

Prior to the computation of a first solution, some preprocessing is carried out.

For each given lecture L_i, events $E_{i1}, \ldots, E_{ie}$ are created which are later assigned to timeslots. The number of events e is given in the problem instances.

Second, we categorize for each lecture L_i the available rooms in three disjunct classes $\mathcal{R}_{i1}, \mathcal{R}_{i2}, \mathcal{R}_{i3}$.
$\mathcal{R}_{i1}$ refers to the rooms in which the lecture fits best, that is the rooms R_k with the minimum positive or zero value of $c_k - s_i$, c_k being the room capacity, s_i the number of students of lecture L_i.
The class $\mathcal{R}_{i2}$ stores the rooms in which lecture L_i fits, that is $s_i < c_k$, but not best, and $\mathcal{R}_{i3}$ contains the rooms in which lecture L_i does not fit. With respect to the given problem statement, events of lectures may be assigned to timeslots of rooms in $\mathcal{R}_{i3}$, this however results in a penalty.

2.2 Constructive Phase

The constructive phase tries to obtain a first feasible assignment of all events to timeslots. A simple heuristic approach is used, successively assigning all events to timeslots, one at a time, with the given pseudo-code of Algorithm 1. In the following explanations, let $\mathcal{E}$ be the set of all events, $\mathcal{E}^p$ the set of prioritized events, $\mathcal{E}^{\neg p}$ the set of non-prioritized events, and $\mathcal{E}^u$ the set events that have not been assigned during the construction phase. It is required that $\mathcal{E}^p \subseteq \mathcal{E}$, $\mathcal{E}^{\neg p} \subseteq \mathcal{E}$, $\mathcal{E}^p \cap \mathcal{E}^{\neg p} = \emptyset$, and $\mathcal{E}^p \cup \mathcal{E}^{\neg p} = \mathcal{E}$.

Algorithm 1 Constructive phase

```
1: Set E^p = ∅, E^u = ∅, loops = 0
2: repeat
3:    E^p ← E^u
4:    E^u ← ∅
5:    E^¬p ← E \ E^p
6:    while E^p ≠ ∅ do
7:       Select the most critical event E from E^p, that is the event with the smallest
         number of available timeslots
8:       if E can be assigned to at least one timeslot then
9:          Select some available timeslot T for E
10:         Assign E to the timeslot T
11:      else
12:         E^u ← E^u ∪ E
13:      end if
14:      E^p ← E^p \ E
15:   end while
16:   while E^¬p ≠ ∅ do
17:      Select the most critical event E from E^¬p, that is the event with the smallest
         number of available timeslots
18:      if E can be assigned to at least one timeslot then
19:         Select some available timeslot T for E
20:         Assign E to the timeslot T
21:      else
22:         E^u ← E^u ∪ E
23:      end if
24:      E^¬p ← E^¬p \ E
25:   end while
26:   loops ← loops + 1
27: until E^u = ∅ or loops = Maxloops
```

As given in Algorithm 1, the construction of solutions carried out in a loop until either a feasible solution is identified or a maximum number of iterations $Maxloops$ is reached. When constructing a solution, a set of events $\mathcal{E}^u$ is kept for which no timeslot has been found. When reconstructing a solution, these events are prioritized over the others. In that sense, the constructive approach is biased by its previous runs, identifying events that turn out to be difficult to assign.

The choice of timeslots for the events reflects the initial categorization of rooms. With a probability of 0.5, timeslots of rooms in $\mathcal{R}_{i1}$ are preferred over $\mathcal{R}_{i2}$ over $\mathcal{R}_{i3}$, and with a probability of 0.5, timeslots of $\mathcal{R}_{i2}$ are preferred over the ones of $\mathcal{R}_{i1}$ over $\mathcal{R}_{i3}$. Within each class, timeslots are randomly chosen with equal probability. In cases where a most-preferred class of timeslots is empty, the choice is made from the lesser preferred class and so on.

After at most a maximum number of $Maxloops$ iterations, the construction procedure returns a solution that is either feasible ($\mathcal{E}^u = \emptyset$) or not ($\mathcal{E}^u \neq \emptyset$).

2.3 Iterative Phase

The iterative procedure continues search for an optimal solution starting with the initial assignment of events to timeslots as given in Section 2.2.

In each step of the procedure, a number of events is unassigned for the timetable and reinserted in set $\mathcal{E}^u$. A reassignment phase follows. Contrary to the constructive approach, where events are selected based on whether they are critical with respect to the available timeslots, events are now randomly chosen from $\mathcal{E}^u$. The choice of the timeslot follows the logic as described in the constructive approach, prioritizing timeslots of particular room classes.

When evaluating timetables, two criteria are considered. First, the number of unassigned timeslots (distance to feasibility) hc, second, the total penalty with respect to the given soft constraints sc. Comparison of solutions implies a lexicographic ordering of the hard constraint violations hc over the penalty function sc. We therefore accept timetables minimizing the distance to feasibility independent from the soft constraint count.
In case of identical distance to feasibility hc, inferior solutions with respect to sc are accepted up to a threshold. In the current implementation, a threshold of 1% of sc has been chosen. This implies that the absolute threshold changes with the actual value of sc, approaching 0 for small values of sc.