A local search for the timetabling problem

Olivia Rossi-Doria[1], Christian Blum[2],Joshua Knowles[2], Michael Sampels[2], Krzysztof Socha[2], and Ben Paechter[1]

[1] School of Computing, Napier University,
10 Colinton Road, Edinburgh, EH10 5DT, Scotland
{o.rossi-doria, B.Paechter}@napier.ac.uk

[2] IRIDIA, Université Libre de Bruxelles, CP 194/6,
Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium
{cblum, jknowles, msampels, krzysztof.socha}@ulb.ac.be

Abstract. This work is part of the Metaheuristic Network, a European Commission project that seeks to empirically compare the performance of various metaheuristics on different combinatorial optimization problems. In this paper we define a representation, a neighbourhood structure and a local search for a university timetabling problem. Our motivation is to provide a common search landscape for the metaheuristics that we aim to compare, allowing us to make a fair and meaningful analysis of the relative performance of these methods on a timetabling problem.

1 The problem

The aim of the Metaheuristics Network[⋆] — a European Commission project undertaken jointly by five European institutes — is to compare and analyse the performance of various metaheuristics on different combinatorial optimization problems including timetabling.
The timetabling problem considered here is a reduction of a typical university timetabling problem. A problem instance generator produces problem instances with different characteristics for different values of given parameters. The problem consists of a set of events E to be scheduled in 45 timeslots (5 days of 9 hours each), a set of rooms R in which events can take place, a set of students S who attend the events, and a set of features F satisfied by rooms and required by events. Each student attends a number of events and each room has a size. A feasible timetable is one in which all events have been assigned a timeslot and a room so that the following hard constraints are satisfied:

- no student attends more than one event at the same time;
- the room is big enough for all the attending students and satisfies all the features required by the event;
- only one event is in each room at any timeslot.

In addition, a candidate timetable is penalised equally for each occurrence of the following soft constraint violations:

- a student has a class in the last slot of the day;
- a student has more than two classes in a row;
- a student has a single class on a day.

[⋆] http://www.metaheuristics.net/

2 The solution representation

The solution representation chosen is a direct representation. A solution consists of an ordered list of length $|E|$ where the positions correspond to the events (position i corresponds to event i for $i = 1, ..., |E|$). An integer number between 1 and 45 (representing a timeslot) in position i indicates the timeslot to which event i is assigned. For example the list

$$3 \ \ 27 \ \ 43 \ \ \ldots \ \ 10$$

means that event 1 is assigned to timeslot 3, event 2 is assigned to timeslot 27, and so on. The room assignments are not part of the explicit representation; instead we use a matching algorithm to generate them. For every timeslot there is a list of events taking place in it, and a list of possible rooms to which these events can be assigned. The matching algorithm gives a maximum cardinality matching between these two sets using a deterministic network flow algorithm. If there are still unplaced events left, it takes them in label order and puts each one into the room of correct type and size which is occupied by the fewest events. If two or more rooms are tied, it takes the one with the smallest label. In this way each event-timeslot assignment corresponds uniquely to one timetable, *i.e.* a complete assignment of timeslots and rooms to all the events.

3 The neighbourhood structure

The solution representation described above allows us to define a neighbourhood using simple moves involving only timeslots and events. The room assignment are taken care of by the matching algorithm. We considered three types of move on a timetable involving one, two and three events respectively:

- a move of type 1 moves one event from a timeslot to a different one;
- a move of type 2 swaps two events in two different timeslots;
- a move of type 3 permutes three events in three distinct timeslots in one of the two possible ways.

Each type of move defines a neighbourhood denoted N_1, N_2, N_3, respectively. The complete neighbourhood N is then defined as the union of these three neighbourhoods: $N = N_1 \cup N_2 \cup N_3$.

4 The local search

Since the neighbourhood is large we use a first improvement local search which works as follows. Consider in order moves of type 1, 2 and 3 for every event involved in constraint violations. Initially ignore soft constraint violations. Then, if the timetable reaches feasibility, consider soft constraints as well, but never go back from a feasible to an infeasible solution. For each potential move re-apply the matching algorithm to the affected timeslots and delta-evaluate the result. The local search is ended if no improvement is found after trying all possible moves on each event of the timetable. A more detailed description of the algorithm follows:

```
1. Ev-count ← 0;
   Generate a circular randomly-ordered list of the events;
   Initialize a pointer to the left of the first event in the list;
2. Move the pointer to the next event;
   Ev-count ← Ev-count + 1;
   if ( Ev-count = $|E|$ ) {
     Ev-count ← 0;
     goto 3.; }
   (a) if ( current event NOT involved in hard constraint violation
       (hcv) )
         { goto 2.; }
   (b) if ( $\nexists$ an untried move for this event ) { goto 2.; }
   (c) Calculate next move (first in $N_1$, then $N_2$ then $N_3$)** and gen-
       erate resulting potential timetable;
   (d) Apply the matching algorithm to the timeslots affected by the
       move and delta-evaluate the result;
   (e) if ( move reduces hcvs ) {
         Make the move;
         Ev-count ← 0;
         goto to 2.;}
   (f) else goto 2.(b);
3. if ( $\exists$ any hcv remaining ) END LOCAL SEARCH;
4. Move the pointer to the next event;
   Ev-count ← Ev-count + 1;
   if ( Ev-count = $|E|$ ) END LOCAL SEARCH;
   (a) if ( current event NOT involved in soft constraint violation (scv)
       )
         { goto 4.; }
   (b) if ( $\nexists$ an untried move for this event ) { goto 4.; }
   (c) Calculate next move (first in $N_1$, then $N_2$ then $N_3$)** and gen-
       erate resulting potential timetable;
   (d) Apply the matching algorithm to the timeslots affected by the
       move and delta-evaluate the result;
   (e) if ( move reduces scvs without introducing a hcv ) {
         Make the move;
         Ev-count ← 0;
         goto 4.; }
   (f) else goto 4.(b);
```

** That is, for the event being considered, potential moves are calculated in strict order. First, we try to move the event to the next timeslot, then the next, then the next etc. If this search through N_1 fails then we move through the N_2 neighbourhood, by trying to swap the event with the next one in the list, then the next one, and so on. The same then occurs for N_3.

The described local search may be very wide and deep. Preliminary observations suggest that when used within the context of some metaheuristic technique it may be more efficient to reduce the depth and width, allowing the metaheuristic a greater proportion of CPU time. This can be achieved in a number of ways, some of them stochastic, so that the local search is parameterized. Parameters might include the following: a restriction of the neighbourhood size by excluding N_3; a time limit; a probability associated with each move being performed; a probability associated with each event being considered; and a maximum number of steps. We will present results regarding which values of these parameters are best for different contexts.

Acknowledgments: This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential program of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.