Algorithm for Solving the Curriculum Based Course Timetabling Problem (Track 3)

Haris Gavranovic and Una Benlic

School of Science and Technology,
Bistrik 7, 71000 Sarajevo, Bosnia and Herzegovina

Abstract: This paper provides a brief description of a method for solving the course timetabling problem by using the local search approach with a set of neighborhood structures.

1 Algorithm

The algorithm that we implemented for the Curriculum Based Course Timetabling problem consists of two parts:

- Obtaining an initial solution for which all the hard constraints are satisfied. This solution is generated by two heuristics which are going to be described in the paper.
- Applying local search on a set of neighborhoods. During this process hard constraints are never violated.

1.1 Initial Solution Generation

As previously mentioned, we used two heuristics in the process of generating a feasible solution.

The first heuristic, called Least Saturation degree First, schedules first those events with the least number of valid periods currently available. This will give priority to those events that have very few periods available, perhaps because a large number of their conflicting events have been scheduled, or maybe because the number of suitable periods was limited to begin with. In either case such events may be difficult or impossible to schedule later on in the process.

The second heuristics is used to choose in which timeslot the lecture should be placed. The lecture is inserted into the feasible timeslot with most parallel lectures.

However, it might happen that there is no valid period in which to schedule an event. Then we must consider unscheduling some of the events already scheduled in order to make room for the critical event. Each period of the timetable is examined and will fall into one of the following three cases:

Case 1: The current event could not be scheduled in the period even if there were no conflicting events already in that period (unavailability constraint conflict). Therefore the period is no longer considered.

Case 2: The current event could be scheduled into the period if a number of already existing events were to be removed from the period. If this is the case the cost of scheduling in this period is the number of events that would have to be bumped from the timetable.

Case 3: The current event could be scheduled into the period if a number of already existing events were removed. However one of these events has already been bumped by the current event and will not be permitted to do so a second time. Therefore this period is no longer considered. This is to avoid the possibility of looping.

From the set of periods that fall into case 2 we select the one which involves bumping the least number of events.

1.2 Optimization of the feasible solution

Having obtained a feasible solution we are ready to optimize it. In order to do this we applied local search on a set of neighborhoods so that we can explore neighbors that are distant from the current solution.

We used the following neighborhood structures:

- Select a course at random and find another course at random which can swap timeslots.
- Move highest penalty course from a random X% selection of the courses to a random feasible timeslot.
- Select two timeslots at random and simply swap all the courses in one timeslot with all the courses in another timeslot.
- Move to courses to a random feasible timeslot.

Let *N* be the number of neighborhood structures and let *Sol* be our current solution. Let f(Sol) be the measure of the cost for solution Sol. At the beginning of the optimization phase, the variable *BestSolution* and *Sol* are set to the initial solution obtained at the beginning.

During an algorithm iteration we apply each neighborhood structure *m* on *Sol* (m can take values from 1 to N) to obtain $SolTemp_m$. We select the best solution among $SolTemp_m$. That solution becomes BestSolution if f(BestSolution)<f($SolTemp_m$). If f(BestSolution)<f($SolTemp_m$), $SolTemp_m$ is accepted with a certain probability in order to escape local optima. Given a new solution *SolNew*, old solution *BestSolution*, and a the current temperature temp, the new solution is accepted if a generated number [0, 1] is less than e^{δ} where δ = f(SolNew)–f(SolBest)/temp. The temperature temp is decreased every 25th iteration. However, it is reheated every 2000 iterations.

1.3 Pseudo code

```
Set the initial solution Sol by applying constructive heuristics
BestSolution ← Sol
CoolingRate = 0.93
temp = 33

While(not termination criteria)
      For m=0 to N //N is the number of neighborhood structures
         Apply neighborhood structure m on Sol to obtain tempSol_m
      End for;
      Identify best solution among f(TempSol_m) and set SolNew to this solution
      IF(f(SolNew)<f(BestSolution))
           SolutionBest = SolNew
      ELSE
           δ = f(SolNew) - f(BestSolution)/temp;
           Generate RandNum, a random number in [0,1];
           IF (RandNum < e^-δ)
              BestSolution ←SolNew;
              Sol ←SolNew;
           End IF;
      End IF;
   IF(CurrentIteration%25=0)
        temp = temp*CoolingRate;
   End IF;
```

```
    IF(CurrentIteration%2000=0)
        temp = temp+1;
    End IF;
End While;
```

REFERENCES

1. S.Abdullah, E.K.Burke, and B.McCollum: An investigation of variable neighbourhood search for university course timetabling, University of Nottingham, United Kingdom.
2. E.K.Burke, J.P.Newall and R.F.Weare: A simple guided search for the timetable problem, University of Nottingham, United Kingdom.
3. L.D.Gaspero, B.McCollum, and A.Schaerf: The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (Track 3), Queen's UniversitySARC Building, United Kingdom (2007)