Timetabling Competition ITC2007: Solver description

Soolmaz Massoodian, Afsaneh Esteki

January 2008

Our solver is a genetic algorithm with two main stages. A local search is applied to the algorithm at the first stage and another local search at the second stage. The first stage minimizes the number of violations and the second one attempt to minimize the cost while keeping the number of violations minimized.

Introduction

The timetables represent the chromosomes. The columns in timetables represent the periods (day and timeslot) and the rows represent the rooms, therefore each location of timetable defines a room and a period. The course lectures will be placed in timetable's locations. Using this algorithm, we never have a missing or extra lecture or more than one lecture in the same room and period. So the only hard constrains to consider will be conflicts and availability.

To encode the input data, the permutation encoding is used. We assign to each course as many integer numbers as the number of its lectures, starting from 1. So the biggest integer number assigned is equal to the total number of lectures.

After assigning values to the courses, the course list is sorted, and in the initialization step, the integer numbers belonging to the courses which are more limited, will be placed in the timetable first.

The algorithm in the initialization step creates a semi-random population of chromosomes. The only thing to consider when placing the courses in the timetable is that we do not place a course lecture in a timeslot which is not available for that course. Integer numbers which start immediately after the last number assigned to a lecture will be assigned to empty locations of the timetable.

A fitness value is assigned to each timetable which shows how good the timetable is. In calculation of the fitness, the effect of a hard constraint violation is considered much stronger than one of a soft constraint violation.

Three types of fitness functions are used. The first fitness function only sees the violations of hard constraints and calculates the fitness assuming there is no soft constraint violations. This fitness is used at the first stage only. After the first stage, the main fitness function is used, which calculates the real fitness of the timetables. The third fitness function which only calculates the violations of soft constrains is being used only in our second local search (In stage 2)

At the beginning of each generation, the better half of the population is selected and is directly sent to the genetic pool in a sorted order. The other half is chosen using the tournament-3 selection method.

After the selection, crossover and then mutation are performed on a subset of population which does not contain the best chromosomes. The mutation and crossover rates are not fixed. The number of location pairs of the timetable which their values will be exchanged in mutation is also variable. The mutation operator in our algorithm does not cause any violation of the availability constraints. After crossover and before mutation, the chromosomes pool is sorted again to avoid any possibility of losing a good timetable created by the crossover operation.

Stage 1: Making violations zero

At the beginning and in the end of every 10 generations of this stage the first local search which only considers the violations of hard constraints and tries to minimize them, is applied to the best chromosome of the population. At the end of the stage one, we have at least one feasible timetable.

Stage 2: Minimizing the cost

The algorithm then tries to find a feasible timetable with the lowest possible cost. At the beginning of this stage and in the end of every 10 generations a local search is applied to the best chromosome of the population. This local search only works in the columns of the timetable, which means it only exchanges the values of two locations in the same column of the timetable, not to cause any violation of hard constraints. Since the only possible violations of hard constrains in our algorithm are conflicts and availability constraints, a feasible timetable can never become infeasible by exchanging values in any pairs of its locations which are within the same column.