1. Describe your implementation. What algorithm did you chose? How did you implement it for this assignment? Treat this as though someone else is trying to re-implement what you did and provide enough details so that they could do so.

Answer:

I implemented 2 different algorithms depending on the soft constraint. For soft constraint 1, where the schedule had to be densely packed, I implemented Randomized Iterative Improvement (RII) on constructive search.

For RII, I first built an initial partial candidate solution, which was fed into the RII algorithm. For this initial solution, I relaxed the hard constraint of maximum slots assuming that the maximum number of slots will be equal to the total number of exams. This partial candidate solution was a sorted list of the exam based on two keys (heuristics). It was primarily sorted higher to lower by the number of students in each exam **(h1**), and when they matched, the sorting was done by increasing number of conflicting courses **(h2)**. The target was to have a sorted list where the starting elements in the exam list have the maximum number of students with minimum number of conflict with other courses. This partial candidate solution is not perturbed, rather it provides a guideline on sequence of selecting different solution components to build a constructive solution.

In different runs of RII, I started with a different solution component (different starting element) of my partial candidate solution and a random walk probability, and tried to pack exams as much as possible. If walk probability was above a fixed threshold, then Iterative improvement was performed by selecting the best neighbor (exam) according to the heuristics described above. If below this threshold, then both these heuristics are relaxed and a uniform random walk is performed among neighbors (exam). In both the cases, if two exams are in the same slot, then they never have the same students. In every slot maximum number of exams are tried to pack together without conflicting. The iteration stopped when all the neighbors have been explored. For RII, neighborhood, we worked on a concept of graph coloring where conflicting exams (two exams that have the same students) have the same color and non-conflicting exams have different colors – and each node (exam) is surrounded by (neighbors) colors of different node. If there is any different color node that can be selected to fit into the current slot, then one of them is selected. If there are no different color node that can be selected to fit into the particular slot, then the slot is closed and based on the two heuristics above, an exam which has not been slotted yet is selected for next slot.

At the end of run, if the solution generated does not violate the constraint of maximum number of slots, then it is selected for consideration as a candidate solution.

For Soft constraint 2, the schedule had to be as sparse as possible to lower the total student cost. Here, a greedy technique is applied with iterative best improvement to generate some initial candidate solutions with exam at each consecutive slot (a slot can have multiple non-conflicting exams). On these initial candidate solution, perturbation is performed with simple Tabu search to generate the best possible candidate solution from the initial candidate solutions.

For the initialization, I start off with a random exam. Each iteration, we see the value of the partial objective function 2 of putting a non-conflicting exam in the same slot and another exam (anyone except for the ones already selected) in the next slot. For each case I maintain two list - one list (List1) contains the best possible exams to put in the same slot which have the least value of the partial objective function 2. (There can be multiple)
And in another list (List2), I maintain the possible exams to put in the next slot which have the same least value of the partial objective function. Which of list chosen is based on the partial objective function value of list 1 and 2. If both are equal, then any can be chosen and I make a random choice between the lists. After a list is chosen, if that list contains multiple exam choices with same objective function value, then one is chosen randomly. In all cases, all of these choices are the best possible choices and any one can be chosen. This builds various schedules of various lengths. This initial schedule is fed into the Tabu search algorithm where it is perturbed using a short term memory

to escape from local optima. Two exam slots are exchanged, so the tabu tenure can be (int)(number_of_slots/2). A tabu move is not considered till the tabu tenure has expired.

2. Why did you select the search algorithm that you did? How did you choose parameter settings for it (e.g., neighborhood operator, initialization)? Cite any relevant literature or pilot experiments that you did.

Answer:

For the first soft constraint, two types of perturbations could have been done on the initial solution – switching two exams and switching two slots. The resulting effect on the length of the number of the slots – our evaluation function result remained the same at both the cases, as a result incorporating it would have been redundant. As a result, I went with constructive search from different starting point. RII was combined with constructive search as it can be proven that if the search process is run long enough, eventually an optimal solution will be generated (Chapter 2, Hoos).

For the second soft constraint, perturbation techniques yield to better evaluation function result. So here I went for a perturbative searching. Tabu search was selected in this case because the use of short term memory enables it to escape from local optima (Chapter 2, Hoos).

For first soft constraint evaluation, after running several combination of examination sequence as start points (refer to Answer 1, para 2), it was observed that if the largest exams are scheduled at the earliest it can reduce the total length of slots. Another target was to spread conflicting exams as evenly as possible, so by using the second heuristics h2, I tried to endure the spreading of conflicting exams when doing iterative improvement in RII.

For building initial solution for second soft constraint evaluation, my target was to greedily build a sequence of exams – a sequence that adds a new exam to itself when the increment in total student cost is the least. Here I chose the neighborhood operator as exchange of two slots instead of two exams because the initial experiments I did gave me sequences of exams that were in effect different sequence that have only a switch between two exam positions. As a result it felt redundant with enough random restarts, and I went for switching exam slots only (say, switching all exams between Day 1 Slot 2 and Day 1 Slot 5).

3. To what extent is your solution designed specifically for examination timetabling? What knowledge of the domain did you use in making your design decisions? What knowledge is the program using in constructing its solutions?

Answer:

The solutions are designed using the domain knowledge of examination timetabling. The ordering heuristics h1 and h2 (refer to answer 1, para 2) are selected from primary soft constraints in examination timetabling problem. The neighborhood operator in first soft constraint based evaluation tries to fit in as much exam as possible together, using conflict chart with the list of student's course number. In iterative improvement step of RII, exams which have the chance of the maximum conflict (largest number of conflicting course), are tried to spread as even as possible.
In the second soft constraint based evaluation, the initial solutions are being built by a greedy technique that employs the knowledge of conflicting exams and tries to spread exams as evenly as possible.

Based on these facts, it can be said that the solution is specifically designed and fine-tuned for examination timetable.

4. Consider the two objective functions. To what extent is your solution designed for one or the other objective function? How did the results change with that function and with the problem type (e.g., the different test problems give)?

Answer:

The solutions are designed custom to each objective function, and gives out different result for different objective function. For the given instance, using the first objective function I received a scheduled of exams taking 4 slots, where the value of the second objective function was around 165~175 in different runs. And for the same instance, using the second objective function I received a schedule of exams 7, 8, 9 or 10 slots wide and generally having a second objective function result between 35 and 50.