Constraint Programming Exercise 2

Apoorva Tamaskar 2349061T

March 5, 2018

Design for Solver and Modifications for Optimize

Design for Solver

For the solver class, only the name of a text file is taken as input. The text file is read and the following variables are stored:-

- The number of people.
- The number of meetings.
- The number of available slots to schedule the meetings in.

This information enables us to declare the arrays $meeting\_distance$(of dimension number of meetings * number of meetings), $meeting_distance[i][j] = k$ means that distance between i^{th} and j^{th} meeting points is k units and a matrix $attend\_matrix$ which records who are attending which meeting. The dimensions are (number of people * number of meetings). $attend_matrix[i][j] = 1$ means person i is going to attend meeting j, 0 otherwise.
As we have the values of all the distances, we can get the max of all. Call this $max\_distance$, will use this to make few instances "easier".
It can be easily proved that if the number of slots available is >= number of meetings $*(max\_distance + 1)$, then we always have a solution $(0, max\_distance, 2 * max\_distance, ...)$. We can achieve this by saying the time of i^{th} meeting is before $(i + 1)^{th}$ meeting and have to have distance of at least $max\_distance + 1$.
If not then we can add constraint in the following manner:-

- $\forall i$
  $(\forall j > i)$
  if $\exists k \ni attend\_matrix[k][i] = 1 \wedge attend\_matrix[k][j] = 1$
  then $|time[i] - time[j]| > meeting\_distance[i][j] \wedge$ break loop of k.

As we have the required constraint we just solve it. The ordering used is "inputOrderLBSearch".

Modifications for optimize

First modification we did was to give an option to take in an parameter of timelimit. This puts a limit on how long the solver should try to solve the model made.
Second we remove the first types of constraints we were adding to simplify few cases.
We use the $max\_distance$ to get an upperbound on constraint variables time array.
We have a constraint variable which is the max of the time array. We have to minimize it.
The last modification is that we keep getting the results as the solver is running and at the end just print the result we have left.

Heuristics

The speed ups we have used are:-

- We have not duplicated constraints for by adding constraint for i^{th} city to j^{th} city then adding one for j^{th} city to i^{th} city.
- We have added only 1 constraint at max for each pair, by properly choosing the order of for loops and breaking after we find even a single person common.(Thinking formally!)
- We used the simplification that we can not go through these possibly n^2 constraints and just work with $2 * n$ constraints if number of available slots > number of meetings $*(max\_distance + 1)$
- The lastly we used "inputOrderLBSearch" search order.

Experiments

For simplicity purpose will assume time limit is 1 sec, it is only applicable on Optimize

Searching order used:- inputOrderLBSearch

- Optimize and Solve details for problem01.txt:- (1.008s,26,250 nodes) and (0.156s, 1,303 nodes)

- Optimize and Solve details for problem02.txt:- (0.295s, 2,416 nodes) and (0.191s, 2,068 nodes)
- Optimize and Solve details for problem03.txt:- (0.182s, 1,055 nodes) and (0.058s, 212 nodes)
- Optimize and Solve details for problem04.txt:- (0.180s, 1,087 nodes) and (0.025s, 12 nodes)
- Optimize and Solve details for problem05.txt:- (0.205s, 1,087 nodes) and (0.035s, 23 nodes)
- Optimize and Solve details for problem06.txt:- (1.005s,16,668 nodes) and (1.558s,79,981 nodes)
- Optimize and Solve details for problem07.txt:- (1.009s, 33,414 nodes) and (3.905s, 412,760 nodes)
- Optimize and Solve details for problem08.txt:- (1.005s, 23,018 nodes) and (10.781s, 1,088,207 nodes)
- Optimize and Solve details for problem09.txt:- (1.008s, 23,806 nodes) and (1.627s, 89370 nodes)
- Optimize and Solve details for problem10.txt:- (.005s,41,846 nodes) and (0.247s, 2,860 nodes)

Searching order used:- minDomLBSearch

- Optimize and Solve details for problem01.txt:- (0.207s, 1233 nodes) and (0.019s, 7 nodes)
- Optimize and Solve details for problem02.txt:- (0.208s, 1104 nodes) and (0.154s, 891 nodes)
- Optimize and Solve details for problem03.txt:- (0.093s, 266 nodes) and (0.035s, 29 nodes)
- Optimize and Solve details for problem04.txt:- (0.067s, 309 nodes) and (0.041s, 92 nodes)
- Optimize and Solve details for problem05.txt:- (0.199s, 1,190 nodes) and (0.043s, 57 nodes)
- Optimize and Solve details for problem06.txt:- (0.190s, 906 nodes) and (0.141s, 712 nodes)
- Optimize and Solve details for problem07.txt:- (0.256s, 1,512 nodes) and (0.234s, 1,317 nodes)
- Optimize and Solve details for problem08.txt:- (1.005s, 1,7901 nodes) and (1.330s, 46,655 nodes)
- Optimize and Solve details for problem09.txt:- (0.218s, 1,089 nodes) and (0.148s, 1010 nodes)
- Optimize and Solve details for problem10.txt:- (0.888s, 21,621 nodes) and (0.109s, 574 nodes)

Alternate Model

For every person we would be given a set containing the meetings which the person has to attend. This is related to the first model in the same way an adjacency list is related to adjacency matrix.
Here we will have constraints if between the meetings corresponding to every pair in each given list. The constraint will be that the distance between i^{th} and j^{th} meeting points has to be at least $meeting\_distance[i][j]$ units.
On the first glance it does look like this model will have lots of repeated constraint, not just by an order of magnitude, but rather a whole factor of number of meetings. Hence once may think that this model will be slower and messier. A possible speed up will be use of minDomLBSearch instead of inputOrderLBSearch, as the experiments do show that inputOrderLBSearch is slower on most of them.