Modelling for Combinatorial Optimisation (1DL451) and Part 1 of Constraint Programming (1DL442) Uppsala University – Autumn 2022 Initial Report for the Project by Team 1: The Interview Assignment Problem (IAP)

Andreas JONASSON and Marion DÜBENDORFER

14th October 2022

All experiments were run under Linux Ubuntu 18.04 (64 bit) on an Intel Xeon E5520 of 2.27 GHz, with 4 processors of 4 cores each, with a 70 GB RAM and an 8 MB L3 cache (a ThinLinc computer of the IT department).

A The Interview Assignment Problem (IAP)

In this problem [1], we have a number of companies (`c`) and a number of students (`s`). The task is to assign interviews with companies to the students according to their preferences (ranging from 1 to 5, 1 being highest and 5 being lowest priority). In addition, the companies have provided an interview capacity, that is a maximal number of interviews the company can hold. Each student should have at most 3 interviews.

Since the problem was initially introduced at a multi-day competition, there exist 4 stages of the problem, each stage building on top of the previous one and introducing additional or differing constraints and objectives.

In its most basic version, the constraints are given as described above and the first objective is to minimize the total sum of the students' preferences of their actually assigned interviews (thereafter referred to as *preference cost*).

In the second version, an interview with a company should only be scheduled if the student's preference for that company is 3 or lower. For each student, a *regret* is introduced, which is the difference between the sum of the preferences of the actually assigned interviews, and the three best indicated preferences. In addition, the companies have introduced a lower bound to their interview capacity, This adds the constraint that the number of interviews held with a company must be either between the lower and upper capacity bound, or zero. For each company for which no interview is scheduled, a *disappointment cost* is incurred. This results in three objectives, listed in decreasing priority:

1. Minimize the maximum regret over all students.
2. Minimize the total disappointment cost over all companies.
3. Minimize the total preference cost over all students.

For the third version of the problems, time slots are introduced. The conference is run over five days with 4 interview slots each day, resulting in 20 slots. Naturally, students can only have up to one interview during one slot. Companies however can have up to two simultaneous interviews. In addition to their preferences with respect to companies, students indicate 5 different time slots during which they are available. In addition, the interviews are held in a number of suites, and there are 12 available suites in total. The weekly rate of a suite is 200, and has to be paid as soon as an interview is scheduled in said suite (even if the suite is only used on one of the five days). Lastly, the company has to pay the conference attendance fee for each day between the first and last interview of that company. Since in the second version the problem allowed for a maximum regret of 1 and a total disappointment cost of 10 in an optimal solution, these are now imposed as constraints. The new objective is to minimize the sum of the following variables:

- The total preference cost over all students.
- The total conference attendance cost for the companies.
- The total rental cost for the suites.

B Model

For this initial report, we decided to discuss our model for the second version of the problem (as described in Section A). The aim is to further extend it such that our final model addresses the third version of the problem, possibly including some relaxations to render it slightly less complex. As the model for version 2 is far less complex than the model for version 3, and the model for version 2 performs fairly well for larger problem instances, we focused our model improvement efforts on the model for version 3. Therefore, in this initial report, the discussion of our model is a bit shorter than usual.

Our model, with the prescribed comments, has the imposed name `IAP-day2.mzn`, is uploaded, and is given in Listing 1.

Listing 1: A MiniZinc model for the Interview Assignment problem (IAP)

```
include "globals.mzn";

%%% ------ Parameters and derived parameters ------
% weights for the objective function
int: alpha = students * 15 * beta;
int: beta = 1;
int: gamma = sum(Disappointment);

% enum of companies:
enum Company = {AIMMS, SAS, Keelvar, Microsoft, Google, IBM,
  Cadence, Quintiq, Siemens, Cosling, COSYTEC, LocalSolver, Nside,
  UTRCI, Zoomer};
int: companies; % #of companies
int: students; % #of students
% Preference[s, c] is the preference of student s to have interview
  with company c:
array[1..students, 1..companies] of 1..5: Preference;
% Disappointment[c] is disappointment cost of company c if no
  interview is scheduled:
```

```
array[1..companies] of int: Disappointment;
% Lower[c] is lower bound for # of interviews for company c:
array[1..companies] of int: Lower;
% Upper[c] is upper bound for # of interviews for company c:
array[1..companies] of int: Upper;

% StudentInterview[s] denotes how many interviews student s has:
array[1..students] of 0..3: StudentInterviews = [
  min(sum([Preference[s,c] <= 3 | c in 1..companies]), 3) | s in
  1..students];

%%% ------ Variables and constraints ------

%% ------ Interview assignments ------

% Interview[s, c] = 1 iff student s has an interview with company
  c; 0 otherwise:
array[1..students, 1..companies] of var 0..1: Interview;

% Ensure that amount of interviews of company c is either 0 or
  inside bounds:
constraint forall(c in 1..companies)(sum(Interview[..,c]) in
  Lower[c]..Upper[c] union {0});
% Ensure that each student has the correct number of interviews:
constraint forall(s in 1..students)(sum(Interview[s,..]) =
  StudentInterviews[s]);
% Ensure that each student has interviews with companies according
  to Preference:
constraint forall(s in 1..students, c in 1..companies where
  Preference[s,c] >= 4)(Interview[s,c] = 0);

%% ------ Preference cost ------

% Total preference cost:
var 3*students..15*students: totalPreferenceCost :: add_to_output=
  sum([Preference[s,c] * Interview[s,c] | s in 1..students, c in
  1..companies]);

%% ------ Day 2: Student regret ------

% PreferenceCount[s, p] is the number of times (up to 3) student s
  expressed preference p, with 1 <= p <= 3:
array[1..students, 1..3] of 0..3: PrefCount = array2d(1..students,
  1..3, [min(count(Preference[s, ..], p), 3) | s in 1..students, p
  in 1..3]);
% IdealPrefCost[s] is for student s the sum of their three best
  preferences that are 3 or better:
array[1..students] of 0..9: IdealPrefCost = [ PrefCount[s, 1] +
                  2*min(3-PrefCount[s, 1], PrefCount[s, 2]) +
                3*min( max(3-PrefCount[s, 1]-PrefCount[s,2], 0), PrefCount[s, 3])
                | s in 1..students];
% AssignedPreference[s] is the sum of the preferences of the interviews assigned to student s:
array[1..students] of var 0..9: AssignedPreference = [ sum([Interview[s,c]*Preference[s,c] | c in 1..companies]) | s in 1..students];
% Regret[s] denotes the regret, i.e. the difference between best indicated preferences and assigned preferences:
array[1..students] of var 0..6: Regret = [AssignedPreference[s] - IdealPrefCost[s] | s in 1..students];
% The maximum regret of any student
var 0..6: maxRegret :: add_to_output = max(Regret);


%% ------ Day 2: Company disappointment ------

% IncurredDisappointment[c] denotes the actual disappointment of company c:
array[1..companies] of var 0..max(Disappointment): IncurredDisappointment;

% If no interview is scheduled for company c, a disappointment cost is incurred:
constraint forall(c in 1..companies where sum(Interview[..,c]) = 0)(IncurredDisappointment[c] = Disappointment[c]);

% Calculate the total disappointment:
var 0..sum(Disappointment): totalDisappointment :: add_to_output = sum(IncurredDisappointment);


%----Objective----
% Day 2: Objective is weighted sum of regret, total student preference cost, and total company disappointment:
var int: obj = alpha * max(Regret) + beta * totalPreferenceCost + gamma * totalDisappointment;

solve minimize obj; % minimize objective
```

Constraints Enforced by the Choice of Decision Variables. The constraint that a student can only interview with a company once is enforced by our choice of the decision variable `Interview[s, c]`, which takes value 1 if student s has an interview with company c, and 0 otherwise.

Redundant Decision Variables and Channelling Constraints. We have tried very hard but were unable to identify any (mutually or non-mutually) redundant decision variables and appropriate (one-way or two-way) channelling constraints that lead to easier modelling, or faster solving, or both.

Implied Constraints. We have not detected any implied constraints.

Symmetry-Breaking Constraints. For day 2, we have not detected any symmetries that would result in identical solutions.

Efficiency. In violation of checklist item 6, on Line 67, we use `forall(... where Interview[..,c] = 0)` with `Interview` being a 2d array of decision variables. We consider the numbers of generated constraints and variables revealed by a profiled compilation to be acceptable.

Correctness. In the problem description at `https://www.csplib.org/Problems/prob062/`, in the description for day 3, it is indicated that an optimal solution for day 2 using the provided instance allows for a maximal student regret of 1 and a total company disappointment cost of 0. As our model for day 2 produces the same optimal solution (given the correct weights for the objective function), we assume that our approach is correct.

Inference Annotations. We did not find any inference annotations that would improve the solving times of our model.

Search Annotation. We did not find any search annotations that would improve the solving times of our model.

C Evaluation

Table 1 gives the results for various instances of the IAP for day two using our model. The time-out was 300,000 milliseconds.

We observe that Gurobi wins overall, as it finds the best solutions of all the backends while also not timing out and thus proves its solutions to be optimal. Gurobi also scales the best. For `day2-037` PicatSAT is one unit of preference cost off the optimal solution, and for `day2-200`, all backends except Gecode are close to the optimal solution, with CP-SAT finding it. For the two largest instances, Yuck finds the best solutions among the non-Gurobi backends. In summary, Gurobi scales and performs the best for all instances, while among the other backends PicatSAT performs well for the smallest two instances, while Yuck is best for the two largest. Gecode performs the worst, as it either times out or finds the least optimal solutions.

The difficulty of an instance depends on the combination of the number of students, their preferences and the capacities of the companies. Thus an instance with more students is not necessarily more difficult. For CP-SAT, `day2-100` seems simpler than `day2-037`, as it does not find the optimal solution for the latter. For Gecode it is the other way around, as it finds a solution for `day2-037` but times out for `day2-100`. While the difficulty depends on the combination of parameters and backend, in general the difficulty seems to increase with instance size. This since Gurobi takes longer time with increasing size, and the other backends are further from the optimal solution for the bigger instances.

While outshined by Gurobi, Yuck performs relatively well, especially for the two biggest instances. While a backend with systematic search wins, local search seems to perform comparatively well for our model.

No results are contradictory: all proven optima are the same. No occurrences of ‘ERR’ were generated by the experiment script.

Backend	CP-SAT		Gecode		Gurobi	
instance	`reg`, `pref`, `dis`, `obj`	time	`reg`, `pref`, `dis`, `obj`	time	`reg`, `pref`, `dis`, `obj`	time
day2-037	5, 207, 5, 5068147	t/o	6, 208, 45, 6152833	t/o	1, 171, 0, **1011936**	**620**
day2-100	1, 414, 0, **7386414**	85444	–, –, –, –	t/o	1, 414, 0, **7386414**	**792**
day2-200	3, 1048, 0, 88867048	t/o	–, –, –, –	t/o	1, 820, 0, **29622820**	**1130**
day2-400	6, 2132, 0, 688466132	t/o	6, 2246, 15, 688753106	t/o	1, 1673, 0, **114745673**	**2001**

Backend	PicatSAT		Yuck	
instance	`reg`, `pref`, `dis`, `obj`	time	`reg`, `pref`, `dis`, `obj`	time
day2-037	1, 172, 0, 1011937	t/o	2, 194, 14, 2049246	t/o
day2-100	1, 477, 0, 7386477	t/o	1, 439, 0, 7386439	t/o
day2-200	6, 1172, 0, 177733172	t/o	1, 908, 0, 29622908	t/o
day2-400	6, 2411, 0, 688466411	t/o	2, 1823, 0, 229489823	t/o

Table 1: Results for our model of the IAP for day two, which is a minimisation problem. The results for different instances are shown, where the number in the instance name denotes the number of students. The lower and upper bounds of the companies are adjusted to the number of students. The three components of the objective function: the maximum regret `reg`, the total preference cost `pref` and the total disappointment of the companies `dis`, are given for each solution. The objective value `obj` is a weighted sum of these three components, such that regret has the highest priority and preference cost the lowest. In the 'time' column, if the reported time is less than the time-out (300,000 milliseconds here), then the reported objective value was *proven* optimal; else the time-out is indicated by 't/o' and the reported objective value is either the best one found but *not* proven optimal before timing out, or '–' indicating that no feasible solution was found before timing out. Boldface indicates the best performance (time or objective value) on each row.

Feedback to the Teachers

For this project, it was very interesting to be able to choose a problem and then build a model for said problem from scratch, that is, without the help of any skeleton code. Naturally, since each group has a different problem and the problems vary in their description accuracy, difficulty, and available objective values that were proven optimal by another model/team, it was hard to estimate our progress and the quality of our model. The fact that there is a big increase in difficulty between day two and three of the IAP problem made this estimation even harder for us.

References

[1] Helmut Simonis. CSPLib problem 062: Interview assignment problem. `http://www.csplib.org/Problems/prob062`.