Interview Assignment
(Project presentation of 17th October 2022)

Andreas JONASSON and Marion DÜBENDORFER
Team 1

Uppsala University
Sweden

Course 1DL442:
Combinatorial Optimisation and Constraint Programming,
whose part 1 is Course 1DL451:
Modelling for Combinatorial Optimisation

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Outline

1. **Problem**
2. Example
3. Approach
4. Experiments
5. Conclusion

UPPSALA UNIVERSITET

Interview Assignment

Companies and students need to be matched up for interviews during a conference:

UPPSALA UNIVERSITET

Interview Assignment

Companies and students need to be matched up for interviews during a conference:

- Each student has a preference for each company (1 highest, 5 lowest).

UPPSALA UNIVERSITET

Interview Assignment

Companies and students need to be matched up for interviews during a conference:

- Each student has a preference for each company (1 highest, 5 lowest).
- Each company has a lower and upper bound for number of interviews.

UPPSALA UNIVERSITET

Interview Assignment

Companies and students need to be matched up for interviews during a conference:

- Each student has a preference for each company (1 highest, 5 lowest).
- Each company has a lower and upper bound for number of interviews.
- Each student has up to three interviews (with companies of preference 1-3).

UPPSALA UNIVERSITET

Interview Assignment: Problem

Determine which students have interviews with which companies, so that, in order of priority:

UPPSALA UNIVERSITET

Interview Assignment: Problem

Determine which students have interviews with which companies, so that, in order of priority:

Regret: The maximum difference of a student's summed assigned interview preferences and their summed ideal preferences is minimal.

UPPSALA UNIVERSITET

Interview Assignment: Problem

Determine which students have interviews with which companies, so that, in order of priority:

Regret: The maximum difference of a student's summed assigned interview preferences and their summed ideal preferences is minimal.

Disappointment: The sum of all disappointment costs (incurred when a company has zero interviews) is minimal.

UPPSALA UNIVERSITET

Interview Assignment: Problem

Determine which students have interviews with which companies, so that, in order of priority:

Regret: The maximum difference of a student's summed assigned interview preferences and their summed ideal preferences is minimal.

Disappointment: The sum of all disappointment costs (incurred when a company has zero interviews) is minimal.

Preference Cost: The sum of all students' preferences of the companies they are matched with is minimal.

UPPSALA UNIVERSITET

Outline

Simple data:

```
1  students = 3;
2  % Pref[s, c]: preference of s to interview at c
3  Preference = array2d(1..students, 1..companies,
4  [1, 2, 1, 3,
5   2, 3, 4, 3,
6   3, 4, 5, 4]);
7  companies = 4;
8  Disappointment = [10, 20, 10, 5];
9  Lower = [2, 2, 2, 2];
10 Upper = [3, 3, 2, 2];
```

Simple data:

```
1  students = 3;
2  % Pref[s, c]: preference of s to interview at c
3  Preference = array2d(1..students, 1..companies,
4  [1, 2, 1, 3,
5   2, 3, 4, 3,
6   3, 4, 5, 4]);
7  companies = 4;
8  Disappointment = [10, 20, 10, 5];
9  Lower = [2, 2, 2, 2];
10 Upper = [3, 3, 2, 2];
```

Solution:

```
Max Regret: 2, Pref Cost: 17, Dis: 10
Interview:
1, 1, 0, 1
1, 1, 0, 1
1, 0, 0, 0
```

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Parameters


```
 9 enum Company; int: companies; int: students;
10 % P[s, c] = pref. of student s to have
     interview with company c:
11 array[1..students, 1..companies] of 1..5:
     Preference;
12 % D[c] = disapp. cost of company c:
13 array[1..companies] of int: Disappointment;
14 % L[c] = lower bound for # of interviews of
     comp. c:
15 array[1..companies] of int: Lower;
16 % U[c] = upper bound for # of interviews of
     comp. c:
17 array[1..companies] of int: Upper;
```

Derived Parameters


```
20 % SI[s] = # of interviews of student s:
21 array[1..students] of 0..3: StudentInterviews =
     [
22    min(sum([Preference[s,c] <= 3
23    | c in 1..companies]), 3)
24    | s in 1..students];
25 % PC[s,p] = # of times student s expressed
     preference p, with 1 <= p <= 3:
26 array[1..students, 1..3] of 0..3: PrefCount =
27    array2d(1..students, 1..3, [
28    min(count(Preference[s, ..], p), 3)
29    | s in 1..students, p in 1..3]);
```

UPPSALA UNIVERSITET

Derived Parameters


```
30 % IPC[s] = sum of the three best preferences (3
     or better) of student s:
31 array[1..students] of 0..9: IdealPrefCost = [
     PrefCount[s, 1] +
32    2*min(3-PrefCount[s, 1], PrefCount[s, 2]) +
33    3*min( max(3-PrefCount[s, 1]-PrefCount[s,2],
        0), PrefCount[s, 3])
34    | s in 1..students];
```

UPPSALA UNIVERSITET

Decision Variables and Constraints

Automatic enforcement of the 1Interview constraint (each student interviews with a company either 0 or 1 times):

UPPSALA UNIVERSITET

Decision Variables and Constraints

Automatic enforcement of the 1Interview constraint (each student interviews with a company either 0 or 1 times):

```
40 % Interview[s, c] = 1 iff student s has an
     interview with company c, 0 otherwise:
41 array[1..students, 1..companies] of var 0..1:
     Interview;
```

UPPSALA UNIVERSITET

Interview Constraints


```
43 % Amount of interviews of company c is 0 or
     inside bounds:
44 constraint forall(c in
     1..companies)(sum(Interview[..,c]) in
     Lower[c]..Upper[c] union {0});
45 % Each student has the correct number of
     interviews:
46 constraint forall(s in
     1..students)(sum(Interview[s,..]) =
     StudentInterviews[s]);
47 % Each student has interviews with companies
     according to Preference:
48 constraint forall(s in 1..students, c in
     1..companies where Preference[s,c] >=
     4)(Interview[s,c] = 0);
```

UPPSALA UNIVERSITET

Student Preference Cost


```
52 % Total preference cost:
53 var 3*s..15*s: totalPreferenceCost =
54   sum([Preference[s,c] * Interview[s,c]
55   | s in 1..students, c in 1..companies]);
```

Student Regret

```
% AP[s] = 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] = the regret of student s:
array[1..students] of var 0..6: Regret =
  [AssignedPreference[s] - IdealPrefCost[s]
  | s in 1..students];

% The maximum regret over all students:
var 0..6: maxRegret = max(Regret);
```

UPPSALA UNIVERSITET

Company Disappointment


```
75 % ID[c] = the actual disappointment of company
     c:
76 array[1..companies] of var
     0..max(Disappointment):
     IncurredDisappointment;
77
78 % If no interview is scheduled for company c, a
     disappointment cost is incurred:
79 constraint forall(c in 1..companies where
     sum(Interview[..,c]) = 0)
80    (IncurredDisappointment[c] =
        Disappointment[c]);
81
82 % Total disappointment:
83 var 0..sum(Disappointment): totalDisappointment=
84    sum(IncurredDisappointment);
```

UPPSALA UNIVERSITET

Objective

We model the objective function (the sum of the student preference cost, student regret, and company disappointment is to be minimised) using weights in order to respect the prioritisation of objectives:

Objective

We model the objective function (the sum of the student preference cost, student regret, and company disappointment is to be minimised) using weights in order to respect the prioritisation of objectives:

```
88 % Objective is weighted sum of max regret,
     total student preference cost, and total
     company disappointment:
89 var int: obj = alpha * maxRegret + beta *
     totalPreferenceCost + gamma *
     totalDisappointment;
90
91 solve minimize obj; % minimize objective
```

UPPSALA UNIVERSITET

Implied Constraints

We did not yet derive any useful implied constraints.

UPPSALA UNIVERSITET

Symmetry-Breaking Constraints

We did not detect any symmetries in the problem or model.

UPPSALA UNIVERSITET

Output

In order to display the individual objective values (and not only the weighted sum), we make use of the `:: add_to_output` annotation for the variables `totalPreferenceCost`, `maxRegret`, and `totalDisappointment`.

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.

UPPSALA UNIVERSITET

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.

UPPSALA UNIVERSITET

Correctness

The objective values of the maximal regret and company disappointment for instance `day2_037` reported in the experiments that were proven minimal before timing out correspond to the optimal values reported in the original problem description.

Correctness

The objective values of the maximal regret and company disappointment for instance `day2_037` reported in the experiments that were proven minimal before timing out correspond to the optimal values reported in the original problem description.

As there is no indication on the optimal preference cost in the problem description, we will need to cross-check this value with Team 11.

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Experiments

Several instances are supplied at CSPlib.org. There are instances with 37, 100, 200 and 400 students, all with 15 companies.
The 37 instance has an optimal solution with 1 maximum regret and 0 company disappointment

UPPSALA UNIVERSITET

Experiments

Several instances are supplied at CSPlib.org. There are instances with 37, 100, 200 and 400 students, all with 15 companies.
The 37 instance has an optimal solution with 1 maximum regret and 0 company disappointment

Results within 300 seconds on an Linux Ubuntu 16.04 (64 bit) on an Intel Xeon E5520 of 2.27 GHz, with 4 processors of 4 cores each, with a 24 GB RAM and an 8 MB L2 cache (a ThinLinc computer of the IT department):

Experiments

Backend	CP-SAT		Gecode	
instance	reg, pref, dis, obj	time	reg, pref, dis, obj	time
day2-037	5, 207, 5, 5068147	t/o	6, 208, 45, 6152833	t/o
day2-100	1, 414, 0, **7386414**	85444	–, –, –, –	t/o
day2-200	3, 1048, 0, 88867048	t/o	–, –, –, –	t/o
day2-400	6, 2132, 0, 688466132	t/o	6, 2246, 15, 688753106	t/o

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

Backend	Gurobi	
instance	reg, pref, dis, obj	time
day2-037	1, 171, 0, **1011936**	**620**
day2-100	1, 414, 0, **7386414**	**792**
day2-200	1, 820, 0, **29622820**	**1130**
day2-400	1, 1673, 0, **114745673**	**2001**

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Conclusion

Insights:

- Interesting to build model for real-world problem from scratch

UPPSALA UNIVERSITET

Conclusion

Insights:

- Interesting to build model for real-world problem from scratch
- Both systematic and local search seem to perform fairly well on different instances given our model

UPPSALA UNIVERSITET

Conclusion

Insights:

- Interesting to build model for real-world problem from scratch
- Both systematic and local search seem to perform fairly well on different instances given our model

Future work for the final project report:

- Model third version of problem, introducing time and location constraints

UPPSALA UNIVERSITET

Conclusion

Insights:

- Interesting to build model for real-world problem from scratch
- Both systematic and local search seem to perform fairly well on different instances given our model

Future work for the final project report:

- Model third version of problem, introducing time and location constraints
- Potentially introduce problem relaxations to decrease difficulty