Wedding Seat Assignment Problem

COMP 4418 – Assignment 3
Nicholas Mattei and Toby Walsh
Due Date: Friday 6 Nov. 2014, 11:59PM
Worth: 1/3

We want to find out if we can seat all of our friends and relatives at our sister's upcoming wedding. Mom says money is a little tight so we need to make sure all the tables that we will order will at least have some people at them. Every individual, family, or couple that received an invitation should be seated together. Additionally, each person could suggest people that they would prefer to sit with. Since this is so important to Mom we want to maximize the happiness she gets out of seeing all her friends and family sitting together happily at the wedding. For this assignment we are going to encode the problem as a constraint programming problem and find solutions so that everyone can have a fun and *happy* time.

Problem Description

Formally, there are *nGuests* attending the wedding that must be seated at *nTables*, where each table can seat at most *nSeats* guests. We are also given a list *Groups* of size *nGroups* which tells us which people must be seated together. We are additionally given a list *Pref* of length *nGuests* which tells us which people a guest would **prefer** to sit with. We use an arbitrary utility measure *PrefWeight* which tells us how much happiness each guest derives from sitting next to their preferred guests. Mom has also given us a set of guests who all must sit at different tables, otherwise they would get too drunk and cause problems.

We want to find a mapping of guests to tables such that:

1. No table has more people at it than *nSeats*.
2. Every table must have at least $\lfloor nSeats/2 \rfloor$ people at it.
3. Every person *must* be seated with their group.
4. Everyone in the set *Trouble* must be seated at different tables.

Measures of Utility

We want to investigate different functions we can use to maximize the happiness or *utility* of all the guests at the wedding. We will investigate three different measures of utility [1]. Let $AtTable(t)$ is the set of people seated at table t, $Pref(i)$ is the set of people i prefers to sit with, and $GuestAt(i)$ is the table that i is seated at.

Utilitarian Social Welfare Function: The sum of utilities of all the individuals.

$$Utilitarian = \sum_{i \in Guests} |Pref(i) \cap AtTable(GuestAt(i))| \times PrefWeight$$

Egalitarian Social Welfare Function: The utility of the worst off guest.

$$Egalitarian = \min_{i \in Guests} [|Pref(i) \cap AtTable(GuestAt(i))| \times PrefWeight]$$

Hint: Since we are only dealing with integers and not real values we can use either the G12-fd or the GeCode solvers which come bundled with MiniZinc 2.0

Suggested and Provided Resources

We recommend that you use MINIZINC which is a freely distributed modeling language and comes with a solver and IDE. It is available at `http://www.minizinc.org/`. There are a number of examples and tutorials that we have covered in class that are available from the MINIZINC webpage as well as the course website. You can use any solver you like but we will only provide support for MINIZINC.

Additionally, we have included a Python 3 script[2] which, given a properly specified input file, can check a provided solution. The command line for the script is:

```
$ python3 checkAssignment.py <Input File> <Output File>
```

In order to capture the output of your solver try:

```
$ mzn-g12cpx weddingseating.mzn base-example.dzn > base-example.out
```

and then:

```
$ python3 checkAssignment.py base-example.dzn base-example.out
```

to verify your output.

[1] For more in-depth details of welfare and happiness see H. Moulin, *Fair Division and Collective Welfare*, MIT Press, 2004.

[2] This should also run on Python 2.7+ as well.

File Formats

Your solver **must** accept input and display output in the following formats.
Example Input:

```
% Wedding Seat Assignment Data File.
%Evaluation (0 = Utilitarian, 1 = Egalitarian)
Evaluation = 0;

%Number of Tables
nTables = 3;

%Number of Seats per Table:
nSeats = 3;

%Number of Guests:
nGuests = 9;

nGroups = 3;
%Groups that Must be seated together
Groups = [
          {1,3},
          {5,6},
          {8,9}
];

%Pref Weight
PrefWeight = 3;
%Preference Graph: Row i is the person that Guest i would like to sit with.
%There must be as many rows as nGuests.
Pref = [
        {2,3,4},
        {1,6,9},
        {1,4},
        {3,9},
        {6},
        {5},
        {1,8,9},
        {7,9},
        {4}
];

%Troublemakers
Trouble = {2, 3};
```

The above data file will produce the following **Example Output:**

```
Table: Guests:
   1 : {1,3,4}
   2 : {2,5,6}
   3 : 7..9
Utilitarian Happiness: 36
Egalitarian Happiness: 0
----------
```

If we modify the above file to maximize the egalitarian welfare then we get the following **Example Output:**

```
Table: Guests:
   1 : {1,3,7}
   2 : {2,5,6}
   3 : {4,8,9}
Utilitarian Happiness: 27
Egalitarian Happiness: 3
----------
```

General and Submission Instructions

You will submit 4 files for grading for this project:

weddingseating.mzn: Your miniZinc (or other) model of the problem. This should be well documented and readable. [**65 Points**]

myexample.dzn: An example problem that is *satisfiable* that you have created. It should have at least 20 guests seated at 4 tables. It should not be trivial (ie. everyone should not necessarily want to sit with everyone else). [**10 Points**]

myoutput.out: The output of your model on your created input. The output should correspond to the output file format. [**5 Points**]

explanation.{pdf, doc, docx}: This should be a approximately 2 page write-up of your project. Specifically you should explain what the decision variables are in the model, why you chose to model the problem with these decision variables, and an explanation of each of the **constraint** lines in your model. You should be able to explain what the constraint is doing, and what it models in the real world. Discuss why you chose the modeling approach you did whether you tried any other approaches and whether or not they worked.

You must also include a section called **Testing**. This section should include a graph which shows how your model preforms (in terms of choice points for g12fd or nodes for gecode explored and the solve time) as you add more guests for both of the evaluation functions. There should be data points at 5, 10, 20, and 30 guests for each of the welfare functions. Additionally, you should write a short paragraph about how using the different utility functions, the number of guests, and any other constraints you can think of to test did (or did not) affect the solve time of your model. [**20 Points**]

- In order to submit the assignment use the command:

```
give cs4418 a3 weddingseating.mzn myexample.dzn myoutput.out explanation.pdf
```

  You can change `explanation.pdf` to have a **.docx** or **.doc** extension if needed.

- In case of late submissions, 10% will be deducted for each day late and no submissions will be accepted after 7 days.

- No extensions will be given for any of the assignments (except in case of illness or misadventure). Read the study guide carefully for the rules regarding plagiarism.

Resources

Below are links to some helpful resources regarding this assignment.

- miniZinc download along with a nice tutorial can be found at `www.minizinc.org`.
- We have provided a python script to verify the output of your solver. Note that this will not verify that the solution achieved is a Utilitarian or Egalitarian welfare optimal solution, only that the values add up consistently with the input file. You can download this at: `https://dl.dropboxusercontent.com/u/6721531/COMP4418/checkAssignment.py`.
- You can download an example data file that is *satisfiable* at `https://dl.dropboxusercontent.com/u/6721531/COMP4418/m1.dzn`.