Happy Tents

Peter Soots

February 28, 2012

1 The Problem

The original problem is defined here, `http://drdobbs.com/184410645`. Basically, the point is to arrange campers into tents in a way that makes as many people as happy as possible. Each camper rates their preferences of being in a tent with specific campers from -12 to 12. The specific instance data is given on the website, but overall there are 16 campers and 5 tents of capacities 2, 3, 3, 4, and 4.

2 Instance Data File Specifications

2.1 Preference Table

Each line in the preference table file should contain:

<camper1> <camper2> <camper1's rating of camper2>

i.e.:

```
bob alan 4
bob sam 10
bob joe 3
alan bob 2
...
```

2.2 Tent List

Each line in the tent list file is just the capacity of a tent. So this file:

```
2
3
3
4
4
```

has one 2-man tent, two 3-man tents, and two 4-man tents.

3 Building and Running

```
$ git clone https://psoots@github.com/psoots/HappyTents.git
$ cd HappyTents
$ make
$ ./MakeHappy prefTable.txt tentList.txt
$ ./MakeHappy prefTableDayAfter.txt tentList.txt
```

4 Design Considerations

I first tried to represent the traits as a tree, (not all at once, of course), where each depth was another camper placed into some tent and once a tent was full the remaining campers would be placed into the next tent and so on until the tents were all filled at which point you can calculate the happiness of all the campers. However, one of the problems with this is that some states would be repeated. For example, Bob and Alan in the same tent is no different than Alan and Bob in the same tent with all other tents remaining as they were. Their happiness level will not change. To fix this and look at only the unique states, I adjusted the tree so that each depth is a tent with some combination (not permutation) of campers in it.

Effectively I have set the tents as the variables and the combinations of campers as the values, rather than just the campers themselves. This reduced the number of states greatly:

$$tent1: \ {}_{16}P_2 = 240$$

$$tent2: \ {}_{14}P_3 = 2184$$

$$tent3: \ {}_{11}P_3 = 990$$

$$tent4: \ {}_{8}P_4 = 1680$$

$$tent5: \ {}_{4}P_4 = 24$$

$$240 \cdot 2184 \cdot 990 \cdot 1680 \cdot 24 = \textbf{20,922,789,888,000 states}$$

$$tent1: \ {}_{16}C_2 = 120$$

$$tent2: \ {}_{14}C_3 = 364$$

$$tent3: \ {}_{11}C_3 = 165$$

$$tent4: \ {}_{8}C_4 = 70$$

$$tent5: \ {}_{4}C_4 = 1$$

$$120 \cdot 364 \cdot 165 \cdot 70 \cdot 1 = \textbf{504,504,000 states}$$

However, there is a problem with this approach, especially as I have implemented it. Namely, this single heuristic doesn't make it easy to implement other heuristics. And, if the instance size gets any bigger, other methods of searching could help reduce the time to find a solution. For example, if some heuristic were to show that some combinations typically yield a higher overall score than others, then sorting the combinations by most promising could give us a solution within the first few states. Unfortunately, as things are currently implemented, sorting the combinations before searching them could be a very expensive task.

5 Benchmarks

These are the results of running the complete search on the linuxlab.cs.pdx.edu machines. The first instance is from the data given in the problem and the second instance is that data minus 4 per each expressed preference. The best score for the first instance was 175, and 92 for the second. Both ran to completion.

Trial	**Time**
First Instance Data	
1	15:09
2	15:12
Second Instance Data	
1	15:17
2	15:23

These benchmarks would be much more interesting had I tried various sizes of instances rather than just changing the values.

6 Improvements

The primary heuristic I used took advantage of the symmetry in swapping campers within a tent, but I could also consider the symmetry of swapping campers between different tents of the same size. This would have been a little trickier to implement, but definitely worth it if instances of any greater size need to be tested.