Junctor (Junction Trees Optimally Recursively)

Junctor is a program for finding chordal Markov networks that are optimal with respect to a decomposable scoring criterion provided by the user.

Given a function p defined over all subsets of a vertex set V, Junctor finds a junction tree representation of a chordal Markov network on V that maximizes the score

$$\frac{\prod_i p(C_i)}{\prod_i p(S_i)}$$

where C_i are the cliques and S_i the separators in the junction tree.

Building

Junctor compiles as `C++11` (`C++0x`) and only uses standard libraries. The version used in the experiments presented in an associated paper is compiled with `GCC 4.6.4`, using

```
g++ -std=c++0x junctor.cpp -o junctor -O3
```

Running this in the directory where you extracted the package will produce a binary named 'junctor'.

Usage

Running Junctor with no arguments will produce usage instructions.

Otherwise, run

```
junctor <input file> [<maximum width>] [-flags]
```

The first argument is the input score file (see Input format below). For example, to solve the instance `bridges`, run

```
junctor bridges.score
```

This will produce one possible clique tree of some chordal Markov network that is optimal with respect to the input score.

If given, the `maximum width` argument will restrict the width of the optimal solution. For example, to find an optimal network with cliques of at most 2 vertices, run

```
junctor bridges.score 2
```

By default, Junctor will print out the cliques, the separators, and the (log) score of an optimal network, as well as a junction tree representation of it. You can control the output by giving the `-flags` argument, a dash (`-`) followed by any of the following characters:

```
v  verbose
s  print the score of an optimal solution
t  print a junction tree of the solution
m  print the adjacency matrix of the solution
d  print a .dot file of the solution
a  set all flags
```

The default flags are `-vst`. For example, to additionally output the adjacency matrix, run

```
junctor bridges.score -vstm
```

Input format

Junctor takes as an input a text file consisting of *tokens* separated by whitespaces. The first two tokens are integers, N, the number of vertices, and M, the maximum clique size. The remaining tokens are the real values $\log p(A)$ for every subset A of $V = \{1, 2, \ldots, N\}$ of size at most M, in the lexicographic order.

An example of the lexicographic order for $N = 4$, $M = 2$:

```
0000  =  {}
0001  =  {1}
0010  =  {2}
0011  =  {1,2}
0100  =  {3}
0101  =  {1,3}
0110  =  {2,3}
1000  =  {4}
1001  =  {1,4}
1010  =  {2,4}
1100  =  {3,4}
```

For examples of actual score files, see the sample files `bridges.score`, `flare.score`, and `nursery.score` included.