CS170 Project
Zellze
Submission login: -ar
Team Members:
Nathan Wong
Nicholas Lin
Tony Duan

Approach

To approximate a solution to this variation on the Travelling Salesman Problem, we utilized ant colony optimization (ACO). The gist is that ants are sent through the network many, many times, each one laying down an amount of pheromone inversely proportional to the total length of its path. Each time an ant is at a crossroads, it picks one of the available paths from a probability distribution computed from *a priori* knowledge (in this case, the length of each path) and the amount of pheromone laid down on that path. Pheromone evaporates over time, such that (hopefully) the algorithm will converge in a reasonable amount of time.

Input Files

Our goal when generating input files was to destroy greedy algorithms primarily, although it turns out that it works pretty well on our own as well. The basic premise is as follows:

1. Choose a permutation of range(50) that will act as the optimal path
2. Make every single edge in the complete graph heavy
3. Make every edge in the optimal path have medium weight
4. For every group of three consecutive nodes in the optimal path (e.g. 1-3, 4-6, etc.), add a short edge between the first and last nodes

The reasoning behind this is that every time *any* algorithm attempts to take one of the short "bridge" edges, they will be forced to take a long one later in order to make it up. A strictly greedy algorithm will make this mistake every time (50/3 = 16 times), while a probabilistic one like ours has a tough time eliminating every such option and may still take one or two suboptimal paths.

Instructions

Directory structure is as follows in order to run the code:

```
project_directory/
    phase2.sh
    nptsp.py
    instances/
        1.in
        2.in
        ...
        495.in
    result_log  (create this)
    results/    (create this)
    compile_phase2.py
    kill_greedy.py
```

You are free to use the Python2 version of your choice (modify the shebang in nptsp.py). We used PyPy as a faster drop-in replacement for CPython. Then simply run the phase2.sh script and wait a couple of hours. After it is complete, you can use compile_phase2.py in order to parse any number of output files (will take the min if two instances of nptsp.py returned different paths for the same problem) into the correct output format.

kill_greedy.py can be run standalone anywhere; pipe its output into a file. The last line gives the optimal path through the graph.

References

http://web.science.mq.edu.au/~gilmour/publications/gilmour2005b.pdf
http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms