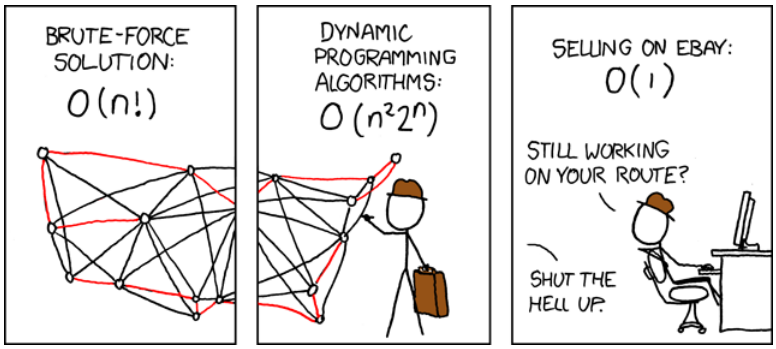
Project Assignment
Parallel Solver for Traveling Salesman Problem

Massimo Di Pierro

February 5, 2013

Your project consists of implementing a parallel solver for the Traveling Salesman Problem and writing a short paper to document your solution.

The deadline for the project is Sat Feb 9, 2013.



1 Requirements about Code

- Your code must be written in C/C++ with MPI and/or in Python with `psim.py`.
- It is recommended that you implement a parallel version of the Branch-Bound algorithm. You can use the Held-Karp which is faster but also more difficult to implement.
- The algorithm must take as input a flat text file of the CSV form:

```
0, 1, 10.9628629098
0, 3, 8.98886303215
0, 4, 6.65365965779
1, 2, 4.52195982628
1, 3, 1.70565185304
2, 3, 3.32596142881
2, 4, 4.23563784771
```

where the first and second column are indices of the vertices (from 0 to N-1) and the third column is the distance between vertices. The program should read the input, parse it, determine the total number of vertices, and build a representation of the graph. Some links may be missing from the file, this means the corresponding distance is infinity. The order of the links in the input file is not specified.

- The program should output the shortest path that visits all vertices and returns to the origin. This should be in the form of flat file containing a list of indices of vertices in the order in which they are visited, separated by a comma. The traveling salesman always start at vertex 0 (origin). For example

  ```
  0, 4, 2, 3, 1
  ```

- The program should run (20 points)
- The program should return the correct result (10)
- Each function must include a documentation comment including description of input, output, and purpose for the function. (3)
- Each call to communication functions must include a comment explaining what data is communicated. (3)
- The program should not contain memory leaks. (2)
- The program must be properly indented. (1)
- The program should not contain un-necessary code nor commented code. (1)
- Compact code is better than verbose code.

2 Requirements about Documentation

The documentation should be in the form of paper containing:

- An introduction (no formulas or code) (2 points)
- Examples of applications (2 points)
- A description of the non-parallel algorithm with pseudocode and discussion of running time (2 points)
- An overview of existing parellel implementations. (2 points)
- A description of your data representation. (2 points)
- A description of the parallel algorithm that you used (2 points)
- A description of your implementation of the algorithm (2 points)
- Formulas for Speedup, Efficiency, and Cost in the best case and in the worst case. (6 points)
- Benchmarks for your algorithm for one, two and four parallel processes (even if you run it on a single node and therefore does not scale). (2 points)
- List of references to papers you found useful or quoted.
- Appendices with your code (both serial and parallel).

The total length of the paper excluding the appendices with program should be more then 10 pages long and less then 30.

References

[1] `https://wiki.engr.illinois.edu/download/attachments/203953697/report-TSP.pdf`

[2] `http://arxiv.org/pdf/1208.3933.pdf`

[3] `http://arxiv.org/pdf/1208.3933.pdf`

[4] http://www.jot.fm/issues/issue_2003_03/column7.pdf
http://www.jot.fm/issues/issue_2003_05/column7.pdf
http://www.jot.fm/issues/issue_2003_07/column8.pdf
http://www.jot.fm/issues/issue_2003_11/column5.pdf