To: Professor Vikas Chaudary

From: Chris Foytek, Daniel Quintanilla

Date: March 28, 2017

Subject: Proposal for Project: Analysis for Parallel and Single Threaded Traveling

Salesman Problem Solvers.

**Purpose**

The purpose of this proposal is to request authorization to move forward with a project that will use an already developed parallel traveling salesman problem solver to compare against a 2-opt local search traveling salesman problem solver. We intend for each of these methods to have easily identifiable comparable heuristics. Our final report will include analysis on the differences in runtime and accuracy, as well as a discussion on drawbacks and potential improvements for each method. We will also discuss tradeoffs in accuracy for an increase in speed, if there are any.

**Introduction**

The symmetric traveling salesman problem has long been one of the most studied problems in optimization. The question asked in the TSP(Traveling Salesman Problem) is best stated as “Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?” The TSP is labeled as NP-hard; which means that it is a decision problem that is at least as hard as the hardest problems in NP(nondeterministic polynomial acceptable problems). This means that as the number of cities in the map being optimized increases, the difficulty and runtime of a program attempting to solve for an exact solution also increase dramatically. To avoid these issues, developers use heuristics to find approximate solutions quickly for maps with a large number of cities.

We will use the 2-opt local search heuristic method to compare against a parallelized travelling salesman problem solver that is based on iterative hill climbing with 2-opt local search. The idea behind the 2-opt local search heuristic is to take a route in the TSP and reorder that route so that it does not cross over itself. As the 2-opt solver continues, its job to find the optimal route gets progressively easier, but we still expect the 2-opt method to be out performed by a parallelized problem solver. Our parallelized solver will also use a 2-opt heuristic, but will spawn a large number of hill climbers to find local minima in parallel. The use of the 2-opt local search heuristic in the parallelized solution provides comparable single thread performance when compared to our non-parallel solver.

**Proposed Tasks**

*Task 1. Determine Tools/Platform*

We will be using two TSP solvers. One TSP solver is a 2-opt local search that is will be written in C++. The other TSP solver will be a pre-developed parallelized random restart iterative hill climbing solver with 2-opt local search implemented by Molly O’Neil and Martin Burtscher. This parallel solver will use a CUDA GPU based implementation to permit a very high amount of parallelization.

*Task 2. Design and Implement Program*

1. The 2-opt solver - This TSP solver will utilize a swap function to find the shortest distance saved for a given route and add on to the current shortest distance saved for the TSP. As the 2 opt solver solves the shortest non over-crossing routes we should obtain the shortest route for our TSP.
2. The parallelized random restart solver - This TSP solver has been designed and implemented by Molly O’Neil, et al.1 It is freely available and open source, and we will ask the developers for permission to use the solver for comparative performance tests. In addition, we will examine the source code and possibly make changes to either our single threaded solver or this solver to produce a more fair comparison wherein one solver is not optimizing in ways that the other is not. We may also make changes to the source code for performance data logging.

*Task 3. Benchmarking and comparisons*

To benchmark our solvers, each solver will take note of several performance measures (i.e. number of cores used, runtime, accuracy of found solution compared to optimal solution, CPU cycles, etc). From these we will compare 3 main performance measurements and present them in our final report:

* Runtime - The most obvious benchmark for our problem solvers is how long the solver takes to determine an answer for the TSP given.
* Accuracy - In the project the accuracy of the answer returned from our solvers will be weighted to determine if the solvers we use are functioning and how close their returned answers are to our determined true answer.
* Speedup - As we work with an increasing number of hill climbers for our parallelized problem solver we will note the speedup between number of climbers, and determine if there is a drop off in accuracy from an increased number of climbers. We will also determine if there is a drop in accuracy and if the speedup obtained from an increased number of climbers is worth the drop in accuracy.

**Expected Outcome**

We expect the parallelized TSP to outperform the single threaded local 2-opt solver when there are more than 4 nodes in terms of runtime and accuracy. We expect the single threaded solution to suffer in performance early in tests, and have very low comparative performance with very large maps.

**Sources**

1. O'Neil, M. A., Tamir, D. & Burtscher, M. A Parallel GPU Version of the Traveling Salesman Problem. WORLDCOMP '10 (2012). http://cs.txstate.edu/~burtscher/papers/pdpta11b.pdf  
   Algorithm and implementation details for the parallelized TSP solver.