**Final Project Proposal**

**Team Members**

Christian Junge, Dana Katzenelson, Jonathan Friedman

**Background and Motivation**

Our project is applying parallel tempering techniques to the simulated annealing optimization method, inspired by a simulated annealing project that Christian worked on in AC290. Simulated annealing is a probabilistic technique for approximating the global optimum of a large search space by sequentially considering permutations of its elements. Between iterations, a small modification is made to elements that were just computed, and a new value is computed. The algorithm will either move forward with the new permutation or continue with the previous one with a probability that is based on the difference in their values (the energy) and a tuning parameter called temperature.[[1]](#footnote-1) Because the algorithm only considers the search parameters and results from its current iteration and the previous one, it inherently runs serially.

Parallel tempering is a method for running many simulated annealing threads in parallel, with each thread using a different temperature, and communication occurring between the threads at regular intervals. In this way, the algorithm can benefit from parallelism by using “hot” threads to be more daring in their exploration of the search space, while the “cold” ones will ideally refine the results that they receive from the hot threads with a more conservative exploration.[[2]](#footnote-2) The Parallel Tempering aspect of this project will be new to us all.

We are interested in seeing whether applying parallel tempering will make the algorithm run faster, and also whether it will allow us to find an optimum with fewer total computations. The few existing scholarly articles suggest that this should be the case. This method has been applied to this problem, so we should be able to replicate these results.3

**Objectives—Functionality and Performance**

Our project will reduce the amount of time and the number of computations necessary to approximate the global optimum of a large search space using simulated annealing. We will explore a variety of strategies for initializing the threads, the interval to communicate between the threads, and the conditions for exchanging permutations between the threads.

We will apply the algorithm to a travelling salesman problem. We will start with a small graph of 25 cities to develop our code and observe the performance of applying the strategies described above, and then scale the size of the graph to hundreds, and, if we have time, thousands of cities to demonstrate the advantages of applying parallelism to this problem.

Our intention is to start with a small problem that can be solved in just a few minutes, and move sequentially to a larger problem that might take hours without parallelism. Ideally, applying parallelism will allow the function to find an optimum with both shorter run time and fewer total computations, demonstrating a benefit from parallelism both by running operations sequentially and by leveraging communication between multiple threads with different temperatures.

**Design Overview – Technologies and use of Parallelism**

We intend to initially code the project using Python multiprocessing. If we have time and reach very large graphs, we would also like to re-implement the project in Spark to compare performance. We will be applying the understanding of the tradeoffs of parallelizing computation that we gained from class (simultaneous computation vs. communication overhead) and taking it a step further by demonstrating how communication between the threads can reduce computation further. The parallelism, as detailed above, will occur in multiple concurrent threads of the simulated annealing algorithm.

**Verification**

For small numbers of cities, the travelling salesman problem can be solved by brute force to provide a definitive correct answer. This true answer can be compared to small data runs of our algorithm to show that parallel tempering returns an estimate that is not worse than simulated annealing. We will be able to compare a serial implementation of simulated annealing with our parallel tempering version to verify that they return similar values for the optimum. We will be able to verify the benefits of parallelization by counting the total number of computations and measuring overall run time. We can verify the benefits of parallel tempering specifically by comparing implementations of the algorithm with temperature varying between threads with an implementation with constant temperature between all threads. We can also verify that we have benefitted from communication between the threads by comparing parallelized implementations with and without communication. In all cases, our metrics will be the returned optimum value, the total number of computations performed, and the overall run time.

**Schedule, Milestones, and Division of Work**

--Milestone 1, Nov 16

Code a simulated annealing function. Code a prototype parallel tempering version of the simulated annealing function.

--Milestone 2, Nov 23

Generate baseline data for comparing the implementations, with charts and numbers to show the tradeoffs and performance of the implementations described above.

--Milestone 3, Nov 30

Continue scaling up and refining strategies, potentially coming up with new implementations based on the results of the previous weeks’ experimentation.

--Final: Dec 7

Polish the final code and aggregate the results. Put together the final submission materials.

**Who does what?**

Christian suggested the project implementation and has outlined the initial proposal. He has the weakest coding skills in the group and will compensate for a slower contribution to the coding by performing more of the organization, analysis and reporting for the project.

Dana and Jonathan bring stronger Computer Science backgrounds to the project and will perform most of the coding.

1. https://en.wikipedia.org/wiki/Simulated\_annealing [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Parallel\_tempering

   3 http://www.cs.odu.edu/~yaohang/publications/AMChybridPTSA.pdf [↑](#footnote-ref-2)