Traveling Salesman Problem
Presentation 1

Team PP (Parallel Programmers)
-Pragash Vijayaragavan
-Krish Rohra
Overview of the problem

● What is the Traveling Salesman Problem
  ○ It is basically the shortest Hamiltonian Cycle in a complete graph.

● So, what is a Hamiltonian Cycle
  ○ A graph cycle through a graph that visits each node exactly once.

● The Traveling Salesman problem is a complete graph problem, that is, each pair of vertices have an edge between them.
Usage: Say you want to visit each city in the United States from New York and at the end come back to New York, you would want to know the path which takes the least cost/time.
Computation Problem

Ways to solve the problem:

1. The Nearest Neighbor Algorithm: $O(n)$: Efficient but not optimal
2. Exhaustive Search: Brute force: $O(n!)$: Optimal but not efficient
3. Dynamic Programming: $O(n^2 2^n)$: Optimal but not efficient
Nearest Neighbor Algorithm

1. From the current city, we keep selecting the nearest city which has not been visited yet.
Although the complexity of the dynamic approach is $O(n^2 2^n)$. But, as you can see, each subproblem depends on the previous subproblem for computation and hence, we cannot parallelize this approach.
Exhaustive Search

- Although, exhaustive search has complexity of $O(n!)$, since we can break down the problem into independent tasks, we can efficiently get an optimal solution by parallelly handling the tasks.
- We can also impose optimizations while parallelly solving the problem, like pruning etc.
Sequential Approach

1. Traverse through the graph using Depth First Search
2. Once you get any of the hamiltonian cycle, you set the cost for that hamiltonian cycle.
3. From now on, we prune any cycles which exceed that cost.
4. Now, only if we get a Hamiltonian cycle with a lesser cost, we update the cost and the path.
5. Finally, once the whole graph is traversed we have the shortest Hamiltonian cycle in the graph, that is, the Traveling Salesman Solution.
Parallel Approach

1. Firstly, sequentially traverse through the graph using breadth-first search until a threshold depth.
2. From this point, each subgraph will be a task in the work queue.
3. The tasks in the work queue will be handled by the threads parallelly.
4. Every subgraph will be traversed using depth-first search approach, and if any hamiltonian cycles are discovered, the thread will prune any path greater to that cost.
5. At the end, each thread will have a local shortest hamiltonian cycle, which will be reduced to find the shortest Hamiltonian cycle of the graph, that is, the Traveling Salesman Solution.