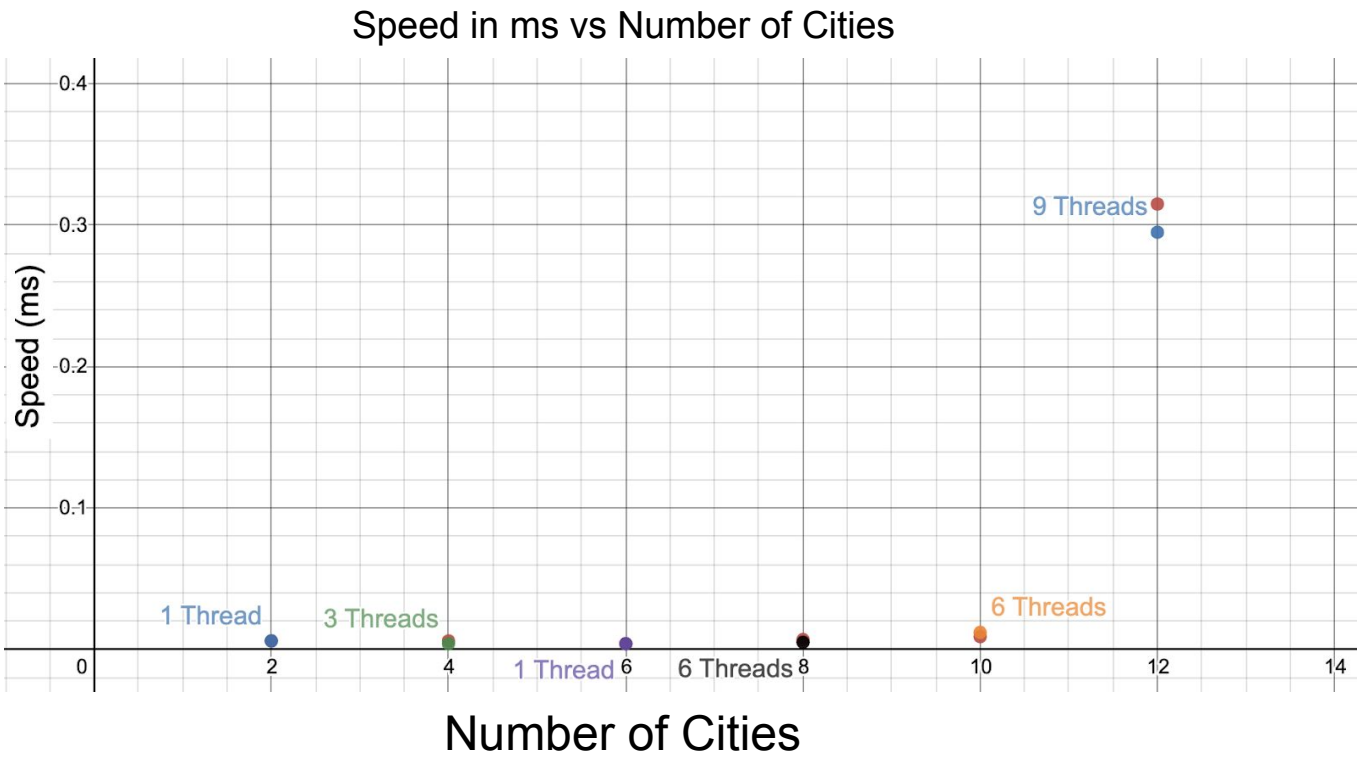
Parallel Lab 2 Analysis
Ben Kaplan
Traveling Salesman OPENMP




| Number of Cities | Optimal Number of Threads | Speed |
|---|---|---|
| 2 | 1 | 0m0.006s |
| 4 | 3 | 0m0.004s |
| 6 | 1 | 0m0.004s |
| 8 | 6 | 0m0.005s |
| 10 | 6 | 0m0.012s |
| 12 | 9 | 0m0.295s |

## Analysis

As depicted in the graph it is clear that utilizing more than one thread as problem size increases will decrease the time taken to run the algorithm and therefore increase the speed of it. There is one obvious outlier to this which can also be seen in the above chart. In all tests six cities always saw a maximum speedup when running with one thread and is and outlier in the data. From the graph and the table we can conclude that utilizing more threads will lead to an increased speed as problem size increases if we disregard the results from the six cities test. We can also conclude that given larger and larger problem sizes, we will experience greater and more noticeable speedup.

## Notes

At no point did the program run out of memory or space leading to more conclusive results. Additionally, I ran into the issue when running the program with the same number of threads as cities or a larger number of threads than cities. This would lead to a segmentation fault. To avoid this issue, if the user runs the program with a greater or equal number of threads as cities, the program will default to running with the number of cities minus one threads. Also, when running the program with a problem size of two cities, there is no reason to go into the main function call, the program simply outputs city zero to one and the weight of that path.