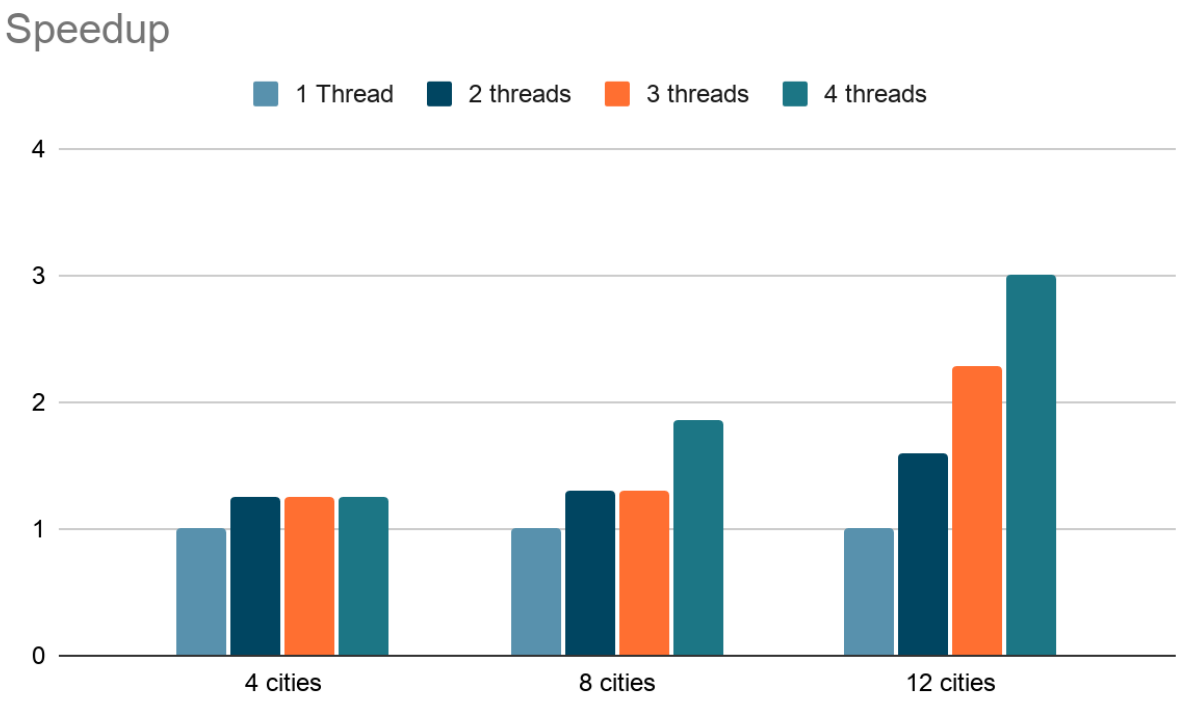
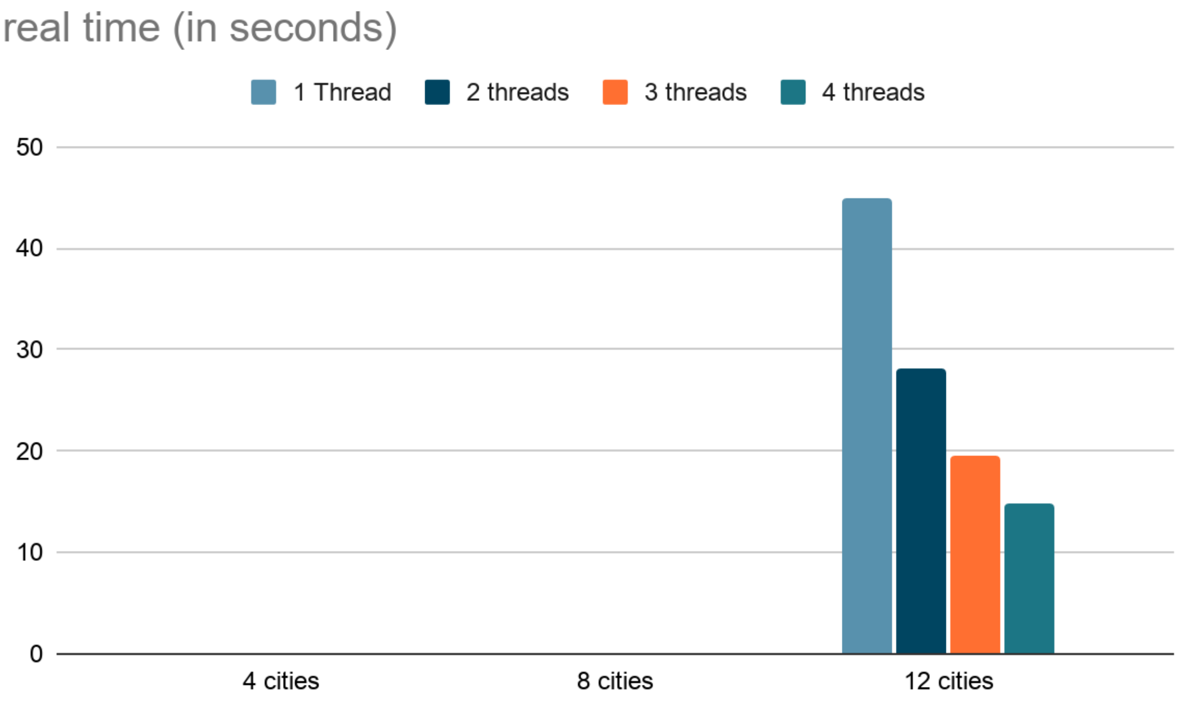
The modified travelling salesman problem is a hamiltonian path problem in a fully connected graph.

Approach used :

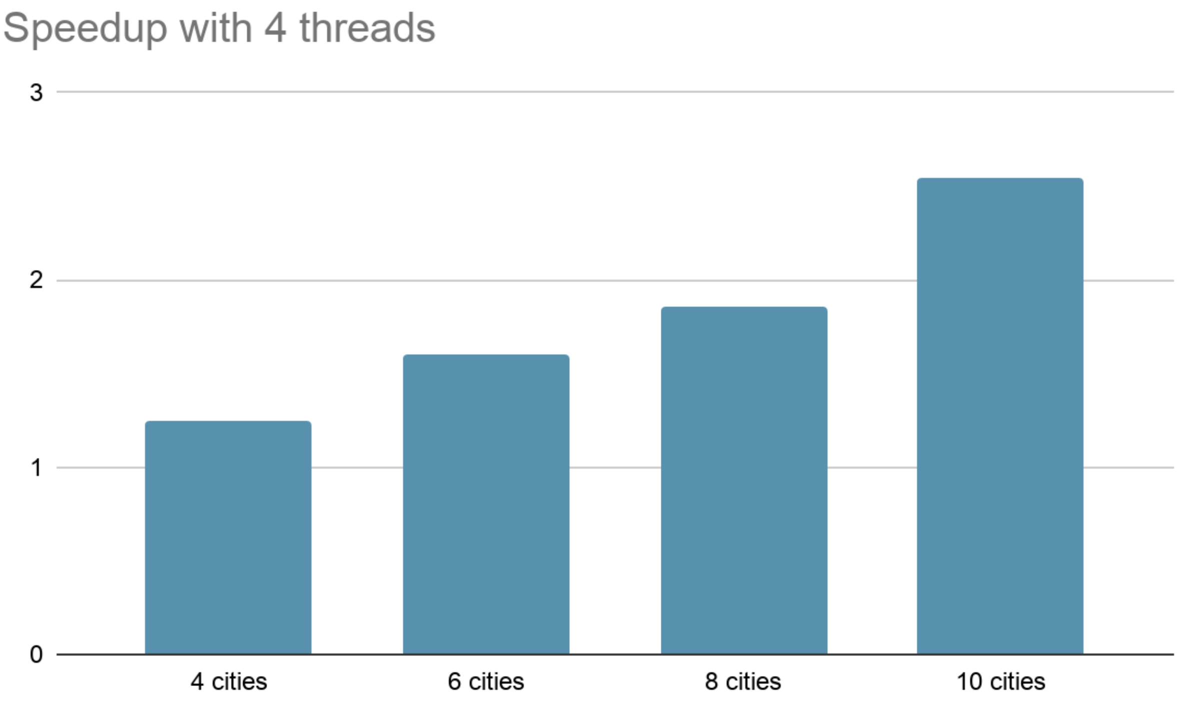
- Generate ith permutation of an array without calculating the previous permutation using factroid numbers.
- Fix 0 as the starting point. Generate all (n-1)! Permutations of array [1..n-1]. Find min cost path permutation and update.
- HIghly parallelizable, since there is no data dependency between each iteration of the for loop. (Except to update minimum cost)

Graph 1: A graph that shows speedup over single-thread version for a problem of size 4 cities, another graph of problem of size 8 cities, and a third graph for 12 cities. For each graph, the x-axis the number of threads (1, 2, 3, and 4) and y-axis is the time generated by time command (real, not user or system).





Graph 2: Fix the number of threads to four. The y-axis is the speedup relative to one thread. The x-axis is the number of cities. In an increment of 2, starting with 4, show the speedup for 4, 6, 8, 10, … x. Where x is the smallest number of cities where four threads show a speedup > 2. If x turns out to be 6 or 8, for example, then stop at that number.



- We see that the problem scales extremely well as problem size increases and number of threads given increases.

  For example :

  - With **30** threads, and 12 cities: it takes **0m4.601s**

  ```
  [at4524@crunchy6 scratch]$ time ./ptsm 12 30 ../cities12.txt
  Min cost is  52
  Min cost path is :
  0 10 2 9 5 3 8 11 6 1 7 4
  real    0m4.601s
  user    1m35.890s
  sys     0m0.066s
  ```

  - With **50** threads, and 12 cities: it takes **0m3.032s**

  ```
  [at4524@crunchy6 scratch]$ time ./ptsm 12 50 ../cities12.txt
  Min cost is  52
  Min cost path is :
  0 4 9 6 1 7 2 10 8 3 5 11
  real    0m3.032s
  user    1m44.292s
  sys     0m0.261s
  ```

  - With **100** threads, and 12 cities: it takes **0m3.032s**

  ```
  [at4524@crunchy6 scratch]$ time ./ptsm 12 100 ../cities12.txt
  Min cost is  52
  Min cost path is :
  0 1 7 6 9 2 10 8 3 5 11 4
  real    0m1.948s
  user    1m26.755s
  sys     0m0.268s
  ```

- Since there is little chance for cache coherence, due to lack of updating shared memory (except for minimum cost - handled with critical section), we have achieved good speedup. We still pay the cost of allocating and deallocating memory for each permutation calculated.