CMSC 502: Assignment 2
Due Date: Tue Nov 2nd, 2021; Total points: 100

Parallel TSP using MPI

Your assignment is to compare the serial TSP solution, threaded TSP solution and the MPI-based TSP Solutions from Assignment-1 with the solutions for Assignment-2. In your report, compare the performances of these different implementations in terms of both execution times and accuracy.

One change from assignment-1: update the edge weights as a sum of infection probabilities and normalized distance between the source-destination pair. To do this, divide each infection probability by the maximum infection probability across all edges; similarly, for each s-d edge, divide the distance between s and d by the maximum distance in your adjacency matrix. Then simply add up these two normalized values (equal weight to infection probability and distance).

Requirements

1) You will generate the same number of points (x,y pairs) for each block separately. Make sure the points you generate for each block are within the x,y boundaries for that block. This ensures an even distribution of points for each block.
2) You MUST use the multi-threaded EMST solution to solve the individual TSP subproblems. Instead of creating an open-TSP as in Assignment-1, we will now directly use the optimal closed loops from the EMST (see the hand-out).
3) Use an MPI Cartesian topology to set up your b^2 blocks in a 2-D grid as in assignment-1.
4) Use the TSP-merge technique discussed in the hand-out for merging two closed TSP solutions from two different blocks.
   The problem is to find an optimal sequence of TSPs (each block has a TSP) to merge.
5) Implement the stitching in two steps: (i) row-wise stitching; this is done in parallel for each row. Use MPI_Barrier() to ensure this part completes first. (ii) Implement a column-wise stitching in the last step; only a single column-wise stitch is needed.
6) We should not get any inversions using these approaches as in Assignment-1. But make sure that is the case. If we do get inversions, we need to use the same strategy as before to switch edges and resolve them.

Results and Report

I expect that you will execute timing runs. From these you can prepare plots showing speedup and accuracy (in terms of length of the tour) for parallel versions using several sizes.

Deliverables

1. Source code(s)

2. Written report describing results. I expect a few pages with graphs or tables along with paragraphs describing your results and conclusions.

3. Show theoretical analyses of your parallel algorithms in terms of run time complexity, efficiency, scalability (isoefficiency), cost and memory optimality and calculate the optimal number of processors needed for a problem of size N points.