CMSC 502: Assignment 2
Due Date: Mon Nov 13th, 2018; Total points: 35

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 Assignment-2. In your report, compare the performances of these different implementations in terms of both execution times and accuracy.

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 a dynamic programming (DP) 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 serial TSP solution.
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.
5) At the block level, in essence, we are following the Euclidean TSP approach using the nearest neighbors. This problem is solved by computing the minimum spanning tree from the nearest neighbor graph and has performance guarantees within 2OPT.
   For a proper parallelization of the MST concept, we will use Boruvka's nearest neighbor algorithm which takes O(log n) steps, where n is the number of blocks (i.e., n x K points, where each block will have K points and $n=b^2$). Check hand-out for implementing the MST idea.
6) Using the MPI Cartesian coordinates, we will first solve the TSP-merge problem along each row. Since we have b columns, this will take O(log b) time (not considering the time taken for merging two TSPs; that's an exercise for you).
   Note that all rows of the 2-D grid can merge their TSPs in parallel in O(log b) steps. Once this is completed (use MPI_Barrier to check for completion), we will just have to stitch b TSPs along the column which takes another O(log b) steps. Hence total number of steps for stitching the blocks will be 2log b steps multiplied by the individual worst-case times of stitching two TSPs together. This could have been implemented in a 1-D scenario (taking O(log n) steps), however, the 2-D block partition scheme provides improvements due to the 2log b steps from an implementation perspective.
7) We should not get any inversions using this approach 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 algorithm 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.