CSCI-UA.0480-003

Parallel Computing

Lab Assignment 2

In this lab you will implement a method for solving a modified version of the traveling salesman problem in OpenMP.

What is the problem definition?

You have a group of n cities (from 0 to n-1). The traveling salesman must start from city number 0, visit each city *once*, and does not have to come back to the initial city (This is why we said it is a modified version of the original traveling salesman problem). You will be given the distance between each city (cities are fully connected to each other). You must pick the path with shortest distance. Note that there may be several paths that satisfy the above conditions, you just need to find one of them.

What is the input?

The input to your program is a text file that contains an nxn matrix, one row per line, representing the distance between any two cities. Your program will be called tsm. The command line then will be:
> tsm filename.txt
Where filename.txt is the file that contains the distance matrix.

What is output?

You program must output, to screen, the best path you found and the total distance.
Here is an example output:
Best path: 0 1 3 4 5 2
Distance: 36
This means the best path your program found is 0->1->3->4->5->2 with total distance of 36.

How will you solve this?

This problem is a combinatorial optimization problem (n! possible solutions for n cities). One of the straightforward way to solve it is through branch and bound. One way you might think about evaluating every possible route is to construct a tree that describes all of the possible routes from the first city. However, the tree may get too large. One way to reduce the size is to follow one path till the end. This path has a cost of w, for example. When you explore a second path, as soon as the cost becomes larger than w, then don't pursue it and move to another branch. If another full path turns out to be of smaller cost, then it becomes your best path and record its cost.
Feel free to optimize the above scheme in any way you want. Also you have to decide about the total number of threads needed.

How will we grade this?

The total grade is out of 20

- 5 points if your program outputs a legal solution (i.e. each city visited once) but not necessarily one of the shortest paths.
- 5 points for the report
- 10 points if your solution is within 10% of the optimal solution (more on this on the website)

We will provide you with few test files and their optimal solutions. We will test your programs with some non-distributed files.

The report

You need to write a report that includes the following:

- How did you parallelize your program?
- A graph that shows speedup over single-thread version
- Another graph showing the scalability of your code as the size of the problem (i.e. the number of cities) increases.
- Explain the above two graphs.

What to submit?

Add the source code as well as the pdf file that contains your graphs and conclusions to a zip file named: lastname.firstname.zip
Where lastname is your last name, and firstname is your first name.

How to submit?

email the above zip file to our grader (not to me, not to the mailing list!) with subject line: lab2 submission

Penalties

- If you do not follow the above protocol for submission and file name --> -1
- If your code does not compile, we will look at your source code; and out of the 20 points dedicated to the coding you may not get more than 7 (depending on how close your code is to a correct version).
- You may lose points also if your conclusions are like "As we can see from the graphs, x increasing with y". We need your explanation of what the graphs show, not your description of what we already see!
- -1 for each late day submission. If you are more than 3 days late, you will get a zero in the lab.