Parallel and distributed programming

Lab 6

Finding a Hamiltonian cycle in a directed graph is a NP-complete problem and has no known polynomial-time solutions. What I try to do with my lab is to use a brute force approach, but parallelized, so that, if an efficient solution is found, the others do not matter.

My graph is saved as an int matrix, which has 0 on the cells where there is no edge between x and y, and 1 otherwise.

The recursive algorithm employed works in the following schematic manner:

1. Initialise an array with graph.size values of -1.
2. Substitute in the first position the starting vertex.
3. Enter the recursive sub-algorithm:
    a. If the current position is equal to the size of the graph, finish. We have gone through all the vertices.
    b. Iterate through all the vertices different from the current one; if we can proceed to one, add it to the current position and enter recursion with the found one. Else add -1 on the corresponding position.
        i. One can proceed if there is an enge between the previous and the current node and it wasn't already added to the solution.

The recursion is enabled using Java's ExecutorServices. The basic idea is starting parallely from all vertices of the graph, and capture the solution and the time in which it was found, if found. Then, print the fastest. Parallely working, these threads do not interact with each other. A possible improvement would be to stop the run when the first solution is found, but that cannot be achieved without breaking the contract of the executor service.

Size	Time	Found
10	1.40285E-4s	YES
30	2.38192E-4	YES
50	3.44241E-4s	YES
70	0.001198514s	YES

80	0.00128835s	YES
90	0.003330051s	YES
100	0.001860726s	YES
200	0.021277836s	YES
300	0.182086121s	YES
500	9.431802714s	YES