VI SEMESTER B.Tech. Data Science and Engineering Parallel Programming Lab (DSE 3262) –Mini Project (2024-April)

PARALLEL IMPLEMENTATION OF TSP PROBLEM PROJECT REPORT

Submitted by: Parshva Dedhia-210968156, Priyanshu Panchal-210968174, Harsh Gupta-210968130

Team Name: HAKUNAMATATA

Date of Submission: 04/04/2024

1.Abstract

In computer science and operations research, the Travelling Salesman issue (TSP) is a well-known optimisation issue. The challenge is to identify the shortest route that visits each city precisely once and returns to the beginning city, given a set of cities and the distances between them.

Put otherwise, the TSP queries a complete weighted network, in which each vertex represents a city and each edge indicates the separation between two cities, seeking the shortest Hamiltonian cycle that can be found.

The classic TSP problem is NP-hard, which means that no known algorithm can solve it in polynomial time for all occurrences. Nonetheless, a number of approximation techniques and heuristics can offer effective solutions for a large number of real-world occurrences of the issue.

Types:

PRIMS ALGORITHM

Prim's algorithm is an algorithm which helps to finds a minimum spanning tree for a weighted undirected graph.it is a greedy algorithm for minimum spanning tree for a weighted undirected graph This method always begins with a single node and proceeds through a number of neighbouring nodes to investigate every related edge that it encounters.

The spanning tree is empty when the algorithm first runs. Maintaining two sets of vertices is the goal. The vertices that are already included in the MST are in the first set, and the vertices that are not yet included are in the second set. It selects the edge with the least weight from among all the edges connecting the two sets at each step. It moves the edge's opposite endpoint to the set containing MST after selecting the edge.

In graph theory, a cut is a collection of edges that joins two groups of vertices in a graph.
The working of Prim’s algorithm can be described by using the following steps:

Step 1: *Determine an arbitrary vertex as the starting vertex of the MST.*
Step 2: *Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).*
Step 3: *Find edges connecting any tree vertex with the fringe vertices.*
Step 4: *Find the minimum among these edges.*
Step 5: *Add the chosen edge to the MST if it does not form any cycle.*
Step 6: *Return the MST and exit*

DIJKSTRAS

The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest path tree.

DIJKSTRA_OPENMP is a FORTRAN90 program which illustrates the use of the OpenMP application program interface by implementing Dijkstra's minimum graph distance algorithm.

The program is an interesting example, because it does not involve parallelization of a loop. Instead, a parallel region is defined, and the nodes of the graph are divided up among the threads. The resulting parallel algorithm naturally requires some of the more advanced OpenMP directives, including **critical**, **single** and **barrier**, in order to work correctly.

The actual graph that is analysed is very small (6 nodes), but of course the same algorithm would work for larger graphs. The point is rather to see how the OpenMP directives can be put together correctly.

BELLMAN FORD:

Imagine you have a map with different cities connected by roads, each road having a certain distance. The **Bellman–Ford algorithm** is like a guide that helps you find the shortest path from one city to all other cities, even if some roads have negative lengths. It's like a **GPS** for computers, useful for figuring out the quickest way to get from one point to another in a network. In this article, we'll take a closer look at how this algorithm works and why it's so handy in solving everyday problems.

2.Introduction

Dynamic Programming is a good approach to try when trying to get an exponential-time algorithm down to polynomial-time. Greedy algorithms are presented for finding a maximal path, for finding a maximal set of disjoint paths in a layered DAG, and for finding the largest induced subgraph of a graph that has all vertices of degree at least. These sequential algorithms can be sped up significantly using parallelism. This project is aimed to reduce the time complexity of few greedy algorithms (viz. Prism, bellman ford and Dijkstra's algorithm) by the method of dynamic parallel programming. The above algorithms have to be programmed dynamically and parallelized with the help of multicore processors. Prism algorithm is MST (minimum cost spanning tree) algorithm and Dijkstra's and bellman ford algorithm are shortest path algorithm. With the approach to dynamic programming we are going to propose the dynamic algorithm for the above listed algorithm and ultimately going for parallel implementation of these. Hence, the dynamic algorithm of these greedy algorithms has to be developed.

2.1 Significance:

Parallel and computed programming has immense significance while reducing the time complexity of an execution of the algorithms. With parallelism we can obtain the less execution time for the algorithm. Hence execution time is the major factor of performance and speed. So, to obtain good and suitable speed of processing with open MP we modified the algorithm so that it can be paralleled. Hence it enhances the performance of the algorithm immensely.

2.2 Objective:

Objective is that we are dividing the sequential algorithm into parallel algorithm, for this threading concept is used. Program divided into number of threads and each thread is executed independent of other Thread. As number of threads executed simultaneously, time required to execute that program reduces. Main objective of this project is to save the time required to execute the programs.

3.Problem Description

The travelling salesman problem (TSP) is a popular mathematics problem that seeks for the most feasible path possible given a set of the points and cost that must all be visited. In computer science, the problem seeks the most efficient route to travel between various nodes, for data. In terms of input, the problem takes a list of physical locations or system nodes, along with distance information. Algorithms and equations work on the process of identifying the most efficient paths possible between the locations. Computer programs can do this through the process of elimination or through a process called heuristics that provides probability outcomes for this type of equation.

4.Drawbacks

The simplest attitude to solve the problem of travelling salesman is to try all the possible routes/all possible combinations of cities to visit while summing distances between the cities in particular route and finally find the route with shortest total distance. This very straight-forward solution is called brute-force or exhaustive search. However, this technique has a shortcoming of immense importance running time of Θ((n-1)!), i.e. it needs to generate (n-1)! Permutations of n cities and calculate the total distance of it. And as n grows, the factorial (n-1)! becomes larger than all polynomials and exponential functions (but 1. This enormous growing of possible routes means enormous growing of time needed for solving TSP even for contemporary computers.

Due to its time complexity Θ(N!) this algorithm is not very suitable for the purpose of this work and so if there exists other algorithm with better time complexity but still exact (thus returning the route of the same quality as brute-force algorithm, i.e. optimal route), it should be chosen for the implementation rather than brute force.

5.Advantages

Prim’sAlgorithm:

Prim's algorithm is an algorithm which helps to finds a minimum spanning tree for a weighted undirected graph.it is a greedy algorithm for minimum spanning tree for a weighted undirected graph. it finds a subset of the edges that forms a tree which include every vertex, where the total weight of all the edges in the tree is minimized. This algorithm performs by constructing tree, one vertex at a time from a starting vertex. At each step adding the least possible connection from the tree to another vertex.

The time complexity of Prim's algorithm is O ((V + E) l o g V) and depends on the data structures used for the graph and edges weight, which can be perform using a priority queue.

The advantage of Prim's algorithm is its complexity, which is better than Kruskal's algorithm. Therefore, Prim's algorithm is helpful when dealing with dense graphs that have lots of edges.

Dijkstra’s Algorithm:

It guarantees the shortest path from the source vertex of all the other vertices when the graph has non-negative edge weights. It provides an optimal solution to the single source shortest path problem.

Bellman Ford:

It is slow compared to Dijkstra algorithm but is more versatile, as it is capable of handling graphs for negative edges too. Negative edge weights can be found in several applications of graphs, Hence the usefulness of this algorithm.

6.NLP Role:

Natural Language Processing (NLP) concepts are applied to the Traveling Salesman Problem (TSP) by representing cities as text, utilizing Named Entity Recognition (NER) for city identification, employing language models for route optimization, computing semantic similarity between cities, incorporating text-based constraints, preprocessing textual data, and enhancing human-computer interaction. These methods aid in tasks such as extracting constraints from textual data, grouping similar cities, and designing user interfaces. NLP enhances TSP solutions by leveraging text processing techniques to improve data representation, constraint handling, and user interaction, offering additional insights and functionalities in tackling this optimization problem.

NLP techniques indirectly support Bellman-Ford and Dijkstra's algorithms by preprocessing textual data for graph representation, aiding Named Entity Recognition (NER) for node and edge identification, analyzing semantic context for edge weights, handling textual constraints influencing pathfinding, and enhancing graph visualization and interaction. Textual data preprocessing facilitates structured graph construction, while NER assists in identifying entities crucial for graph representation. Semantic analysis ensures accurate interpretation of edge weights derived from textual descriptions. Incorporating textual constraints into pathfinding adjusts algorithms' calculations accordingly. Graph visualization and interaction benefit from NLP by providing natural language descriptions and feedback on path results, enhancing user understanding and engagement.

7. Problem and Solution:

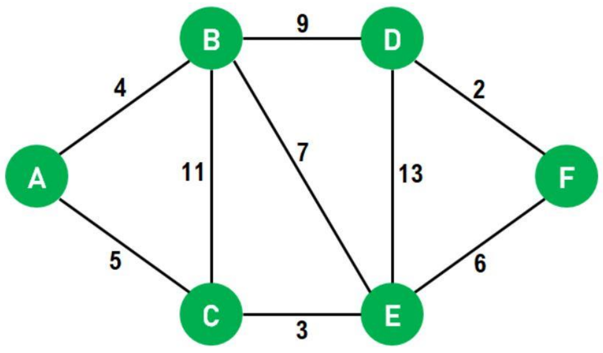


Fig1: Problem

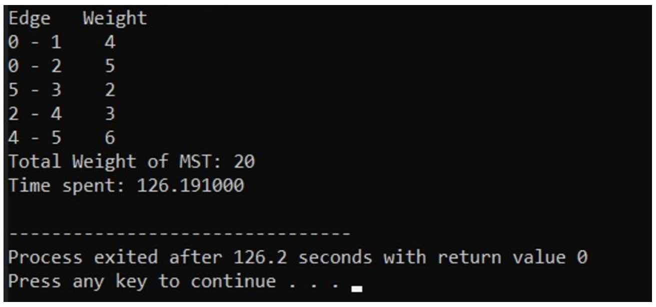
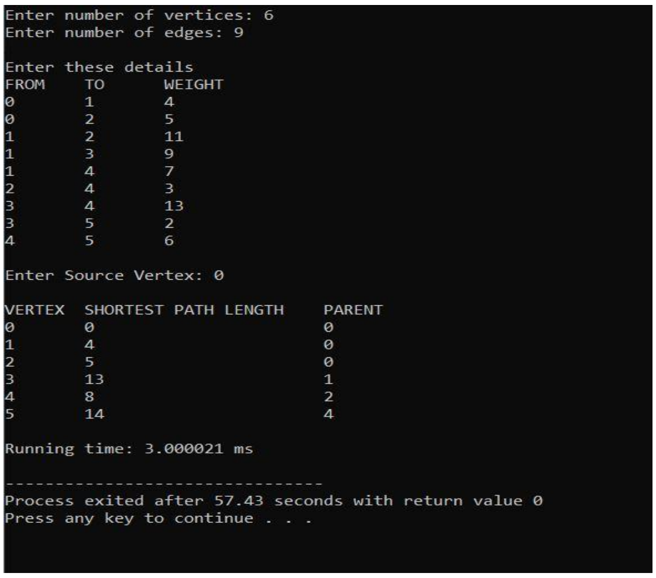
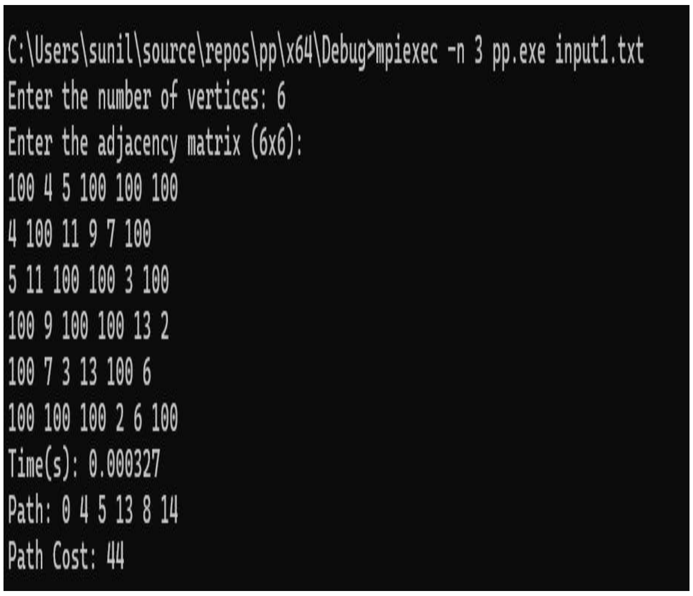

```
Edge   Weight
0 - 1    4
0 - 2    5
5 - 3    2
2 - 4    3
4 - 5    6
Total Weight of MST: 20
Time spent: 126.191000

--------------------------------
Process exited after 126.2 seconds with return value 0
Press any key to continue . . .
```

Fig2: Prims Output



```
Enter number of vertices: 6
Enter number of edges: 9

Enter these details
FROM    TO      WEIGHT
0       1       4
0       2       5
1       2       11
1       3       9
1       4       7
2       4       3
3       4       13
3       5       2
4       5       6

Enter Source Vertex: 0

VERTEX  SHORTEST PATH LENGTH    PARENT
0       0                       0
1       4                       0
2       5                       0
3       13                      1
4       8                       2
5       14                      4

Running time: 3.000021 ms

--------------------------------
Process exited after 57.43 seconds with return value 0
Press any key to continue . . .
```

Fig3: Dijkstra's Output



```
C:\Users\sunil\source\repos\pp\x64\Debug>mpiexec -n 3 pp.exe input1.txt
Enter the number of vertices: 6
Enter the adjacency matrix (6x6):
100 4 5 100 100 100
4 100 11 9 7 100
5 11 100 100 3 100
100 9 100 100 13 2
100 7 3 13 100 6
100 100 100 2 6 100
Time(s): 0.000327
Path: 0 4 5 13 8 14
Path Cost: 44
```

Fig 4 : Bellman Ford Output

8.Conclusion *:*

After studying all the techniques to solve greedy algorithm, it is concluded that the traditional algorithms have major short comings: firstly, they are not suitable for negative edge networks; secondly, they exhibit higher computational complexity. Therefore, using developed ACO algorithm, a group of ants can effectively explore the graph and generate the optimal solution. Although, researchers have got remarkable success in designing a better algorithm in term of space and time complexity to solve shortest path problem.

9.Acknowledgement :

We would like to express our special thanks and gratitude to our professor and the project guide Dr.Girisha Surathkal and Dr. Savitha G as well as MIT who gave us this golden opportunity to work on this wonderful project on the topic “**PARALLEL IMPLEMENTATION OF TRAVELLING SALESMAN PROBLEM ”** which also helped us in doing a lot of research and we came to know about so many new things

10.References :

[1] Cao H,WangF,Fang X,TuH-L, Shi J. OpenMP parallel optimal path algorithm and its performance analysis. 2009 WRIWorldCongr Softw Eng WCSE 2009.2009;1.

[2] AwariR. Parallelization of shortest path algorithm using OpenMP and MPI. Proc Int ConfIoT Soc Mobile, Anal Cloud, I-SMAC 2017.2017;304–9.

[3] Pathare S, KulkarniP,Kardel R. Performance Analysis of Algorithm UsingOpenMP.2014;1(1):152–6.

[4] Fallis A. Data Structure and algorithms in java.Vol.53, Journal of Chemical Informationand Modeling. 2013. 1689-1699 p.

[5] Prim RC. Shortest Connection Networks And Some Generalizations. Bell SystTechJ. 1957;36(6):1389–401.

[6] Grama A, Gupta A, Karypis G, KumarV.Introduction to Parallel Computing, SecondEdition. Communication. 2003. 856p.

[7] P.Pacheco, An Introduction to Parallel Programming, Burlington, USA:Elsevier,2011.

[8] ShiY,Lu J, Shi X, Zheng J, Li J. The Key Algorithms of Promoter Data ParallelProcessing based onOpenMP.2014;154–7.

[9] OpenMPArchitectureReviewBoard,"OpenMPApplicationProgramInterface",http://openmp.org/forum/

[10] AnjaneyuluGSGN,DashoraR,VijayabarathiA,RathoreBS.Improvingtheperformanceof Approximation algorithm to solve Travelling Salesman Problem using Parallel Algorithm. 2014;337(3):334–7.

[11] Jasika N, Alispahic N, Elma A, Ilvana K, Elma L, Nosovic N. Dijkstra’s shortest path algorithm serial and parallel execution performance analysis. MIPRO, 2012 Proc 35th Int Conv Inf CommunTechnolElectron Microelectron. 2012;1811–5.

[12] Maroosi A, Muniyandi RC, Sundararajan E, Zin AM. Parallel and distributed computing models on a graphics processing unit to accelerate simulation of membrane systems. Simulation Modelling Practice andTheory.2014;47:60-78.Availablefrom, DOI:10.1016/j.simpat.2014.05.005

[13] Ang MC, Aghamohammadi A, NgKW,Sundararajan E, Mogharrebi M, LimTL. Multi-core Frameworks Investigation on a real-time object Tracking application. J Theor Appl InfTechnol.2014;70(1):163

to the conference. Failure to remove template text from your paper may result in your paper not being published.