Term Project – Prim’s Minimum Spanning Tree Algorithm

Mostafa AlNaimi

ECE 7610
College of Electrical & Computer Engineering
Wayne State University

ABSTRACT:

The paper analyzes the runtime of the algorithm of Prim in sequential and parallel implementation for the generation of minimal span trees on undirected, edge-weighted graphs and compares the results with the theoretical runtime. The algorithm is briefly presented and implemented in the programming language C and parallelized with the MPI standard. This is followed by a description of the data used for the tests. Generic graphs of solid size are used for tests with graphs of different densities. minimum spanning tree is not an algorithm we can easily parallelize. In fact, sequential algorithms are already very effective: O(Eln(E)). It’s supposed that the parallelization was made mainly as answer to memory issues.

I. INTRODUCTION

A spanning tree is a tree of edges of a graph that contains all nodes and is circular. Minimal is called a span tree if the sum of all edge weights represents the smallest possible under all span trees of the graph.

Minimal tensioning trees are used, among other things, for the proximity of the circular travel problem or for the creation of cost-effective supply networks. They can also be used to generate mazes.

There are already publications comparing the algorithms. However, they are relatively old and do not provide the source code or use simple implementations of the algorithms. Furthermore, there is a lack of reference to real data because random generated graphs are used. This hatch should be closed. In addition, the graphs used cover a larger problem area.

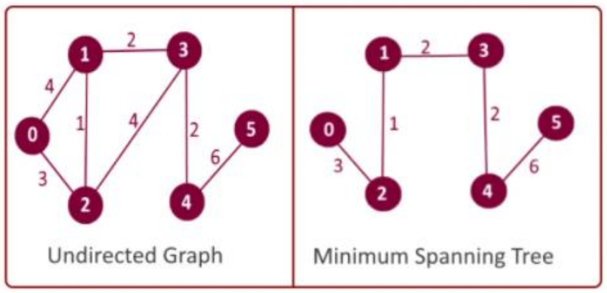


Figure 1: Example of Minimum Spanning Tree

Minimum spanning trees have direct applications in the design of networks, including computer networks, telecommunications networks, transportation networks, water supply networks, and electrical grids. Also used in:

- Approximating travelling salesman problem
- Approximating multi-terminal minimum cut problem
- Approximating minimum-cost weighted perfect Cluster Analysis
- Handwriting recognition

- Image segmentation
- Circuit design

II. PROBLEM DESCRIPTION

All methods used belong to the class of Greedy algorithms. These determine the optimal overall result from individual decisions, which promise the greatest profit at the respective time. When specifying the asymptotic runtime, E stands for the number of edges and V for the node amount of the graph

the algorithm of Prim only searches for a new edge from the previously selected nodes and thus works locally. At the beginning, any node is selected, and the following steps are processed. From the connected nodes are considered all the edges that went to an unmatched node. Of these, the edge with the lowest weight is added to the tree, which is continued until all nodes are included in the tree. An efficient implementation requires a priority queue. For every unequalled knot, this contains the edge with the lowest weight, which leads to a knot reached. Furthermore, an adjacency list is used in which the neighbors to each node are contained. For the implementation of the queue heaps are used. Using a binary heap results in an asymptotic runtime of O(E log V) and with a Fibonacci heap O(E+ Vlog V).

III. IMPLEMENTATION

For the implementation of the presented algorithm the programming language C is used. Furthermore, a parallelization is carried out using the Message Passing Interface (MPI) standard. This is a message-based programming model. This means that each process has its own storage area and is exchanged between these messages. The source code of the program is provided in the ZIP file containing the project file. The graph is represented as C-struct with an array for the edge list and values for the number of edges and nodes. The reading of a graph is done via a file in which the number of nodes and edges is the first. Then follow all edges, which are represented as three numbers. These stand for the connected knots and the edge weight. The nodes are numbered starting from zero.

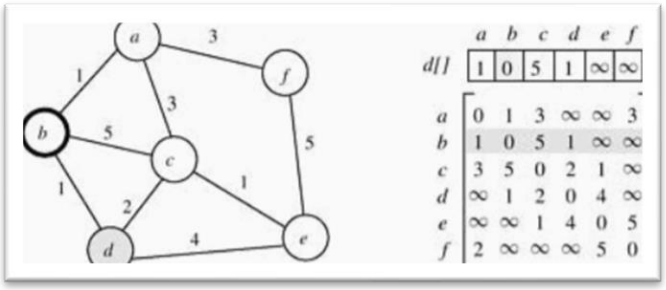
Prim's algorithm is implemented with heaps. There is a variant with a binary heap and one with a Fibonacci heap. The binary heap is implemented as an array and the Fibonacci heap with double-chained lists. In addition, an adjacency list is required to access the adjacent nodes at each edge. First, the heap is initiated with all nodes. It starts at the first node. This is removed and the heap with the edges from the adjacency list is updated to the node. After that, the minimum is removed in each step and the heap is updated with the new edges.

File "mst_matrix.txt" includes the size of vertices plus the adjacency matrix. Here I used 1000 as infinity number. the adjacency matrix is only upper triangle if the file "mst_matrix.txt" only includes the size of vertices, the algorithm generates the adjacency matrix randomly. An example of file "mst_matrix.txt" which includes the example of the book Figure10.5. If you want to run the algorithm for 128 vertices, then you must replace the previous file with the File "mst_matrix.txt" which is in folder Matrix128.

results.txt includes the result of selected edges after prim's algorithm is applied, the total weight and Running time.

IV. RESULTS

The program was compiled and run on the WSU Grid server. First, the sequential implementation of the program was run on the 6 vertices example that is shown below.



$$\begin{array}{c|cccccc}
 & a & b & c & d & e & f \\
d[\,] & 1 & 0 & 5 & 1 & \infty & \infty \\
\hline
a & 0 & 1 & 3 & \infty & \infty & 3 \\
b & 1 & 0 & 5 & 1 & \infty & \infty \\
c & 3 & 5 & 0 & 2 & 1 & \infty \\
d & \infty & 1 & 2 & 0 & 4 & \infty \\
e & \infty & \infty & 1 & 4 & 0 & 5 \\
f & 2 & \infty & \infty & \infty & 5 & 0
\end{array}$$

Figure 2: Prim's Minimum Spanning Tree example

The execution time of the sequential algorithm took 0.000002 sec. To compare against the parallel implementation with different number of the processors, Figure 3 is shown the running time of the same example using 1, 2, 4 ,8, and 16 processors using the MPI implementation. The figure shows that the algorithm does not scall very well with small size of graphs.

In Figure 4, we are using a random graph of 128 vertices. The algorithm scales well up to 4 processors, but then it gets worse with using 16 processors. Speed up is giving by Ts/Tp. For 4 processors, speedup = 0.003911/ 0.001839 = 2.12.

In Figure 5, we are using a random graph of 256 vertices. The algorithm shows better performance/scalability in this case. The speed up for 16 processors = 0.032264/0.004776 = 6.75.

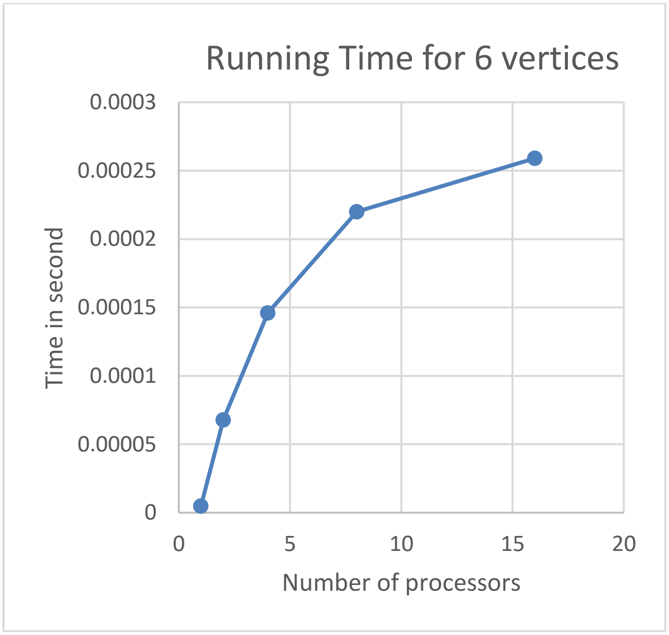


Figure 3: Parallel Running Time for 6 Vertices

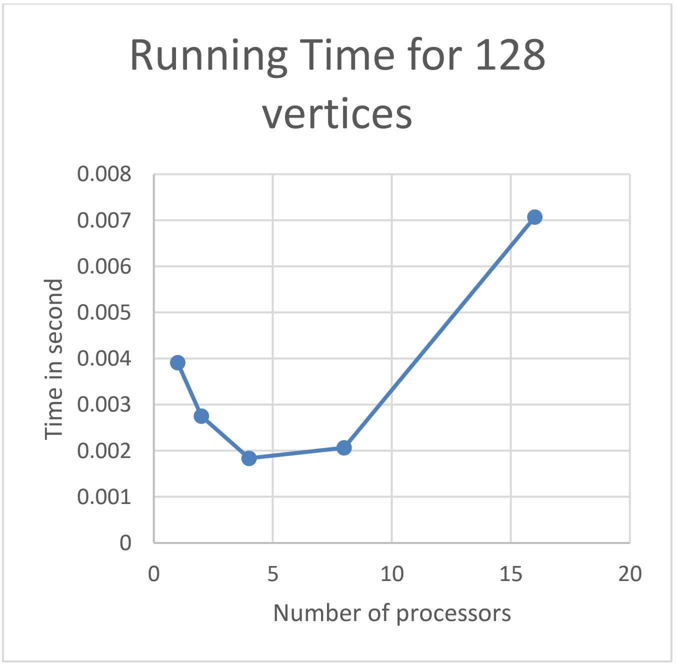


Figure 4: Parallel Running Time for 128 Vertices of Random Graph

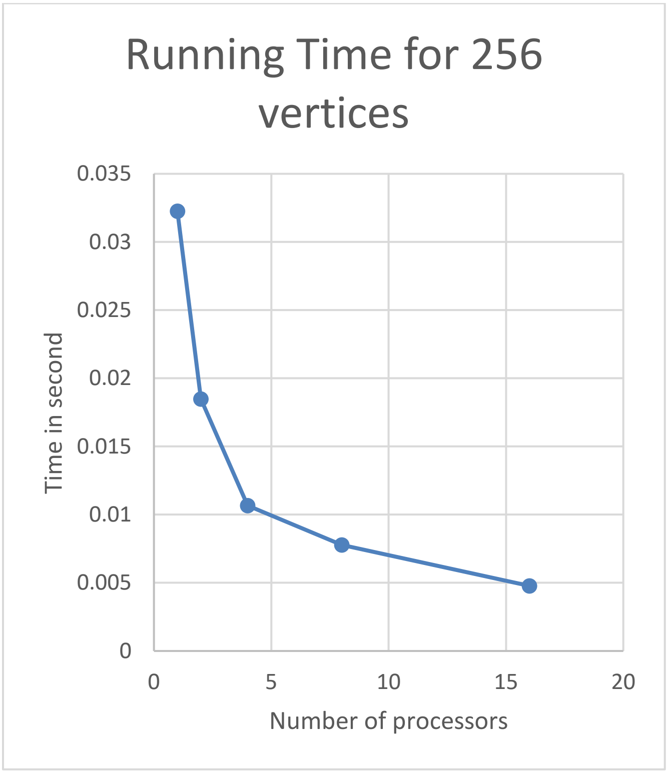


Figure 5:Parallel Running Time for 256 Vertices of Random Graph

V. CONCLUSION

Finding minimum spanning trees (MST) in various types of networks is a well-studied problem in theory and practical applications. Several efficient algorithms have been already developed for this problem. In this paper we present Prim's algorithm. Prim algorithm does not scale very well. Parallelization is useful only on dense graphs, else it does not bring so much. Adjacency matrix is a data structure that leads to serious bottleneck on sparse graph, making the benches useless.

ACKNOWLEDGMENTS

The author would like to thank Dr. Daniel Grosu, Associate Professor, Computer Science at Wayne State University for the support and guidance throughout the semester.

APPENDIX A

DEFINITIONS, ACRONYMS, ABBREVIATIONS

MST	Minimum Spanning Tree
MPI	Message Passing Ingerface
IEEE	Institute of Electrical and Electronics Engineers
ms	milliseconds
Struct	Data Structure

REFERENCES

1. Dehne, F.; Gotz, S.: Practical Parallel Algorithms for Minimum Spanning Trees. IEEE Press, S. 366, 1998.
2. Kruskal, J. B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society, 7:48–50, 1956.
3. Moret, B. M. E.; Shapiro, H. D.: An Empirical Analysis of Algorithms for Constructing a Minimum Spanning Tree. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Springer, 400-411, 1991.
4. Prim, R. C.: Shortest connection networks and some generalizations. Bell System Technical Journal, 36:1389–1401, 1957.
5. Bazlamac, ci, C.F. Hindi, K. S.: Minimum-weight spanning tree algorithms - A survey and empirical study. Computers & Operations Research, 28:767–785, 2001.
6. K. Kayser, H. Stute and M. Tacke, "Minimum spanning tree integrated optical density and lymph node metastasis in bronchial carcinoma", Analytical cellular pathology: the journal of the European Society for Analytical Cellular Pathology, vol. 5, no. 4, pp. 225-234, 1993.