Parallel and Sequential Implementation of Borůvka's Algorithm using OpenMP

Reshma Ram	1MS18CS140
Shradha Ravi	1MS18CS144

Contents

Overview

- The problem of finding suitable Minimum Spanning Trees (MSTs) of undirected graphs has been popular for quite some time in the field of graph theory.
- The problem arises in fields such as that of transportation, logistics, communications, image processing , wireless networks, cluster analysis etc..
- One of the algorithms that can be used to solve this problem is Borůvka's algorithm.
- It is one of the oldest greedy algorithms on record and was published by Otakar Borůvka, a Czech mathematician.
- The algorithm was first modeled in order to design an electricity distribution network but can now be used to find the MST of undirected graphs.
- By implementing both parallel and sequential forms of the algorithm, using multi-thread programming with OpenMP, we compare the results.

Introduction

Combinatorial optimization problems: finding MSTs

- Given a weighted graph, which is a graph that has certain weights or costs for each of its edges or having different length edges, find the tree that connects all nodes such that the sum of costs of all paths or the sum of all the edge lengths is minimum.
- To apply this problem to modern life is to take the nodes as cities or the edges as the communication network connections and in doing so the edge weights can be distances or construction costs.
- The important thing in this is that the tree selected should not form a cycle.

Introduction

Why is the MST problem so popular?

- Ellis Horowitz explains one of the reasons is because it accepts efficient solutions.
- Even graphs with thousands of vertices can have a solution.
- The applications of such algorithms can be found in constructing transportation networks, pipe flow networks, telephone communication networks etc..
- Looking at it in a more theoretical perspective, the MST algorithms usually take a greedy method route in finding the tree.
- This means that at each node it finds the next best edge to add to the set of edges of the tree.

Literature Review: Hybrid Architecture

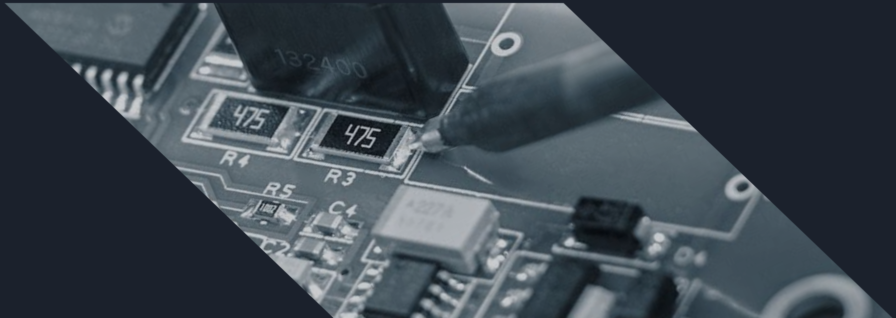
- Rolf Rabenseifner and Gerhard Wellein, in 2003, considered the usage of hybrid MPI+OpenMP programming to construct parallel applications.
- The main idea behind this was for shared memory nodes to combine both distributed memory parallelization on the node interconnect as well as the memory parallelization of each node in the system.
- Tests and simulations were run to check the latency and bandwidth issues with such programming models.
- In the end it was concluded that this hybrid architecture could only be used if the SMP model was enhanced by optimization features on the total node traffic.
- If the communication was to get bottlenecked at the master thread on top of which long messages were used, this hybrid system would fail and using either MPI or OpenMP programming would be more viable.

Literature Review: SLAM

- Previously to solve the MST problem in 2012, a Simultaneous Localization and Map Building (SLAM) technique was proposed by Songmin Jia, Xiaolin Yu and Xiuzhi Li [7].
- This technique was used for navigation by mobile robots inside buildings and houses and was based on OpenMP programming and parallelization.
- The technique used Particle Filter (PF) to find an efficient route by building solid frameworks.
- But in doing so, the programming needed an optimization as the PF provided complexities and required higher computation power.
- Using OpenMP's multi-thread programming, a parallelized algorithm was proposed to reduce the overall computation time of PF and optimize the execution of SLAM.
- From the results of the experiment, it was concluded that this was successful in reducing the PF-SLAM execution time while also prioritizing the accuracy of the navigation frameworks created by SLAM.

Literature Review: GRAPE

- Wenfei Fan , Jingbo Xu , Yinghui Wu presented a parallel system known as parallel GRAPh Engine (GRAPE) which was used for computing graphs.
- It is essentially an algorithm that takes sequential graph algorithms as input, then parallelizes and optimizes it.
- It promises accuracy under most conditions as long as the sequential algorithm is correct.
- GRAPE parallelization guarantees to terminate with correct answers under a monotonic condition.
- The system does the parallelization using a combination of ***partial evaluation and incremental computation*** while using graph optimizations such as indexing, partitioning and compression.



Implementation

Borůvka’s algorithm works as follows:

For each edge that is present:

1. From all the outgoing edges of each vertex, the edge with least weight cost is found, forming a connected component.
2. Create a directive each time an edge is chosen and select a super vertex for all the vertex in the now new tree.
3. Complete the step by adding the edge to the tree officially and remove all the other redundant edges.

Implementation

Keeping this in mind, we can say that there are two ways this algorithm can be implemented:

- Star Contraction: It uses 50/50 probability at each vertex by tossing a coin and for each bit of 0 or 1, to decide which vertex is a star and which is a satellite.
- Edge Contraction: This is the method that was just explained including the usage of the greedy method and incorporating the light edge property.

Comparison

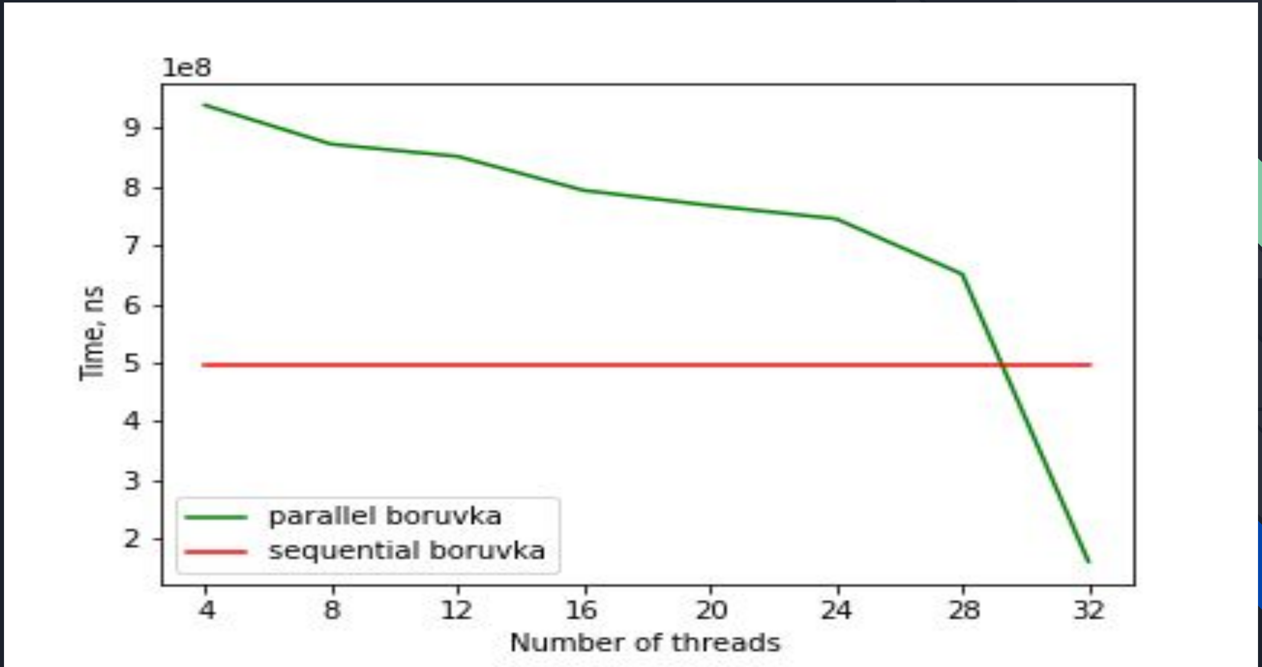
Sequential Implementation

- Each node is visited serially to find minimum weight edges for them
- Runs on a single processor
- Has constant execution time

Parallel Implementation

- Nodes are assigned to multiple processors to compute minimum edge
- Runs on multiple processors
- Slower than sequential implementation only when there are less threads used for computation

Results



1e8
9
8
7
6
5
4
3
2
Time, ns
parallel boruvka
sequential boruvka
4
8
12
16
20
24
28
32
Number of threads

Conclusion

- A fully connected tree is a spanning tree in which two disjoint subset of vertices must be connected.
- Boruvka's algorithm is a greedy algorithm similar to Prim’s and Kruskal’s algorithms with a time complexity of O(ElogV).
- Parallelizing such an algorithm helps in making randomization algorithms quicker and more efficient.
- This work provides a comparison between the sequential and parallel implementations of the Boruvka’s algorithm and also shows that the latter implementation leads to an improved performance as the number of threads are increased beyond 32.
- Although parallelising Boruvka’s algorithm improves performance for random graphs and spanning trees, sparse graphs have not seen much improvement through parallelization.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. The Design and Analysis of Computer Algorithms. Reading, Mass., Addison-Wesley, 1974.

[2] E. M. Reingold, J. Nievergelt, and N. Deo. Combinatorial Algorithms: Theory and Practice. New York, Prentice-Hall, 1977.

[3] E. Horowitz and S. Sahni. Fundamentals of Computer Algorithms. Potomac, Md., Computer Science Press, 1978.

[4] J. L. Bentley and J. H. Friedman. Fast algorithm for constructing minimal spanning trees in coordinate spaces. Tech. Rept. STAN-CS-75-529. Stanford University, January 1976.

[5] W. Chou and A. Kershenbaum. A unified algorithm for designing multidrop teleprocessing networks. Data networks-Analysis and design. Proc. DATACOMM73, IEEE 3d Data Communications Symp., pp. 148-156.

[6] H. Loberman and A. Weinberger. Formal procedures for connecting terminals with a minimum total wire length. J. ACM 4, 4 (October 1957), 428-437.

[7]Songmin Jia, Xiaolin Yin, and Xiuzhi Li, "Mobile Robot Parallel PF-SLAM Based on OpenMP", IEEE, International Conference on Robotics and Biomimetics, December 2012.

[8] R. Rabenseifner, G. Wellein: Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architectures. International Journal of High Performance Computing Applications 17(1), 49–62 (2003).

[9] Wenfei Fan, Wenyuan Yu, Jingbo Xu, Jingren Zhou, Xiaojian Luo, Qiang Yin, Ping Lu, Yang Cao, and Ruiqi Xu. 2018. Parallelizing Sequential Graph Computations. ACM Trans. Database Syst. 43, 4, Article 18 (December 2018), 39 pages.

[10] S. Prakash, Yann Hang Lee and T. Johnson, "A nonblocking algorithm for shared queues using compare-and-swap," in IEEE Transactions on Computers, vol. 43, no. 5, pp. 548-559, May 1994, doi: 10.1109/12.280802.

[11] R. L. Graham and P. Hell, "On the History of the Minimum Spanning Tree Problem," in Annals of the History of Computing, vol. 7, no. 1, pp. 43-57, Jan.-March 1985, doi: 10.1109/MAHC.1985.10011.

Thank you!