Borůvka's Algorithm

Student Number: 700037512

Word count: 1214

Abstract

Borůvka's Algorithm find a minimum spanning tree for a weighted undirected graph. It was the biginning of what is now a classic optimisation problem used in many fields. This report will discuss the algorithms main principles, complexity, successes, limitations and applications to explain why this particular algorithm changed the world.

I certify that all material in this report which is not my own work has been identified

Signature: Ursula Mennear

1 Introduction

In 1926 Czech mathematician Otakar Borůvka published an algorithm to find the minimum spanning tree of a graph, in an attempt to optimise an electricity network for Moravia (1). It was the first algorithm of its kind and creating a minimum spanning tree has since become a classic optimisation problem with many different algorithmic solutions (2). A minimum spanning tree of a weighted, undirected graph $G(V, E)$, where V is the number of vertexes and E the number of edges, is the edges required to connect all the vertices for the lowest total edge cost without producing a cycle (3). Figure 1 shows an example of a minimum spanning tree.

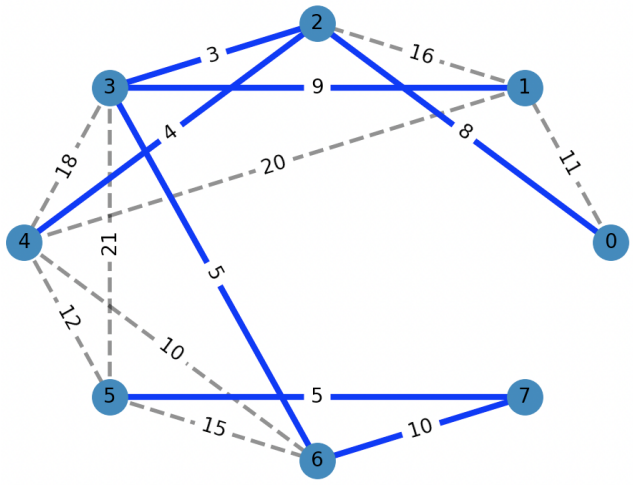

Figure 1: The minimum spanning tree created by Borůvka's Algorithm of the graph is shown in blue. It connects all nodes together and contains no cycles.

Minimum spanning trees are used in a range of applications such as designing telecommunication and other digital networks (4).

2 Main Principles

Borůvka's algorithm is a greedy algorithm, therefore, it selects the best option available at each step (5). The input of the graph must be weighted such that all the edges have a corresponding cost, this can be physical distance, computational duration or anything that can numerically distinguish the cost of each path. Each edge must be undirected which means they can be traversed in either direction (a edge `y` connecting `a` and `b` can be traversed from `a` to `b` or `b` to `a`). The algorithm begins setting all the nodes as separate components. The smallest connecting edge to every component is added to the minimum spanning tree and the nodes that have been connected merge into one component. This will create small disjoint trees where each tree signifies a component that must be connected together to form the minimum spanning tree. The process of finding the smallest connecting edge to each component continues until there is only one component containing all the nodes. The final remaining component is the minimum spanning tree. Figure 2 shows the process using a simple graph.

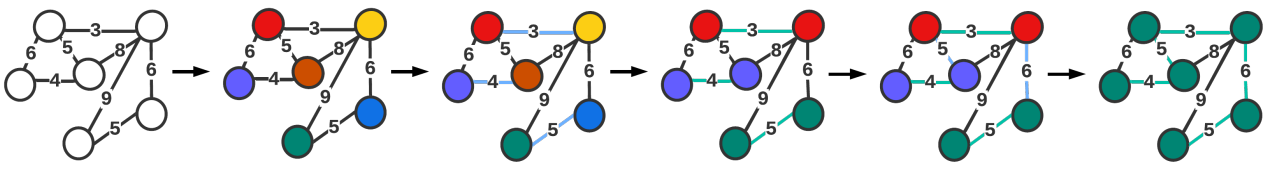

Figure 2: Visual example of Borůvka's Algorithm. Each colour represents a component and the blue edges are the smallest edge connecting two components together at each step and form the minimum spanning tree.

3 Pseudo code

The Pseudo code for Borůvka's algorithm is given below.

```
 1: def Borůvka's algorithm
      input: A undirected weighted graph G(V,E)
      output: The minimum spanning tree of G(V,E)

 2:   minimum_spanning_tree = []
 3:   component_size = []
 4:   for each vertex create a component containing only itself
      and add 1 to the component_size list
 5:   no_components = number of vertices
 6:   while there is more than one component:
 7:       for every edge y connecting 2 components a and b in G:
 8:           let min_edge_a be the cheapest edge for a
 9:           if y is smaller than min_edge_a:
10:               min_edge_a <- y
11:           let min_edge_b be the cheapest edge for b
12:           if y is smaller than min_edge_b:
13:               min_edge_b <- y
14:       for every vertex in G:
15:           if cheapest_edge for components is not null:
16:               join the a and the b components together
17:               add cheapest_edge to minimum_spanning_tree
18:               no_components = no_components - 1
19:   return(minimum_spanning_tree)
```

4 Complexity

4.1 Time Complexity

The time complexity is $O(m \log(n))$ where m is the number of edges and n is the number of vertices (6). The outer while loop on line 6 in the pseudo code has time complexity $O(\log(n))$ because the loop halves the number of nodes each time so iterates $\log(n)$ times. The first inner for loop beginning on line 7 of the pseudo code is independent of the outer loop and has a time complexity of m because the algorithm looks at the edges each time. The second for loop beginning on line 14 has a time complexity of n as it iterates for the number of nodes. As such when the two inner nodes are added together the complexity is:

$O(m) + O(n) = O(max(m, n)) = O(m)$

assuming m ¿ n which is the case if the graph has a cycle and therefore is not a minimum spanning tree. Hence the inner loops have a complexity of $O(m)$ which is iterated $O(\log(n))$ time giving the complexity of $O(m \log(n))$.

4.2 Space Complexity

The space complexity is $O(m + n)$ as the size of the input data is the size of the number of edges plus the number of nodes.

5 Importance

Borůvka algorithm helped advance the mathematical field of graph theory, shaping it into what it currently is in the present (7). His paper containing details of his minimum spanning tree algorithm was published 10 years before graph theory was officiated as a mathematical discipline. Graph theory has now expanded to include help solve problems such as image segmentation (8), noting interactive

chemical reactions (9) and explaining sustainable supply chain networks (10). More directly, Borůvka's algorithm was cited in more recent minimum spanning tree algorithms, such as Kruskal's Algorithm (11) then Prim's Algorithm (12). These algorithms are more commonly used than Borůvka's algorithm. Not only did the algorithm inspire others, but it has also been modified to reduce its time complexity and add parallelism (13).

6 Limitations

One limitation of the algorithm is that it is greedy which means in some cases the optimum solution is not found (14). This means earlier decisions are never reversed and if the local optimum is not the global optimum the algorithm will not alter the result to become globally optimum. However, this problem only occurs in a small fraction of graphs and the speed and reduced complexity using greedy algorithms are an acceptable trade-off in most circumstances.

It is not used as commonly as Prim's or Kruskal's algorithm as it is more difficult to understand in comparison. The ability of non subject experts to understand the process is important when implementing programs across multiple fields such as biology or medical science (15). The reason it is more difficult to understand can be turned into a strength because the structure can allow for parallelism which increases performance (13).

7 Applications

Borůvka's algorithm has possible applications in numerous areas, however, it has been optimised and commonly used in the areas discussed below.

An area which uses Borůvka's algorithm is image segmentation which can generate different scales of superpixels to identify elements or objects of a large image in real-time by turning the pixels into nodes (8). These superpixels can then be used in machine learning. Its applications include ship detection and remote sensing.

Borůvka's algorithm is also used in wireless sensor networks to effectively distribute sensors as a grid which can be used for fire detection or art preservation (16) (17).

Finally, the algorithm is used in reorganising databases so that the access time required to retrieve an object is minimised (18) (19).

8 Conclusion

This algorithm changed the world by introducing an elegant solution to find the minimum spanning tree in a time complexity of $O(m \log(n))$. It inspired other such as Prim and Kruskal to do the same, advancing the area of graph theory. Despite some of its limitations it is still used today in various fields and has been modified to mitigate some of its previous drawbacks.

References

[1] R. Graham and P. Hell, "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[2] J. Nešetril and H. Nešetrilová, "The origins of minimal spanning tree algorithms–boruvka and jarnık," *Documenta Mathematica*, pp. 127–141, 2012.

[3] Y.-S. Myung, C.-H. Lee, and D.-W. Tcha, "On the generalized minimum spanning tree problem," *Networks*, vol. 26, no. 4, pp. 231–241, 1995. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230260407

[4] A. Nagarajan and R. Ayyanar, "Application of minimum spanning tree algorithm for network reduction of distribution systems," in *2014 North American Power Symposium (NAPS)*. IEEE, 2014, pp. 1–5.

[5] D. Rachmawati, F. Y. P. Pakpahan *et al.*, "Comparative analysis of the kruskal and boruvka algorithms in solving minimum spanning tree on complete graph," in *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*. IEEE, 2020, pp. 55–62.

[6] C. F. Bazlamaçcı and K. S. Hindi, "Minimum-weight spanning tree algorithms a survey and empirical study," *Computers Operations Research*, vol. 28, no. 8, pp. 767–785, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054800000071

[7] F. Neuman, "95 years of otakar borvka," *Mathematica Bohemica*, vol. 119, no. 1, pp. 97–99, 1994.

[8] B. Basavaprasad and S. H. Ravindra, "A survey on traditional and graph theoretical techniques for image segmentation," *International Journal of Computer Applications*, vol. 975, p. 8887, 2014.

[9] D. V. Gowda, K. Shashidhara, M. Ramesha, S. Sridhara, and S. M. Kumar, "Recent advances in graph theory and its applications," *Adv. Math. Sci. J.*, vol. 10, no. 3, pp. 1407–1412, 2021.

[10] Z. Luo, R. Dubey, T. Papadopoulos, B. Hazen, and D. Roubaud, "Explaining environmental sustainability in supply chains using graph theory," *Computational Economics*, vol. 52, no. 4, pp. 1257–1275, 2018.

[11] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[12] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[13] A. Mariano, A. Proenca, and C. Da Silva Sousa, "A generic and highly efficient parallel variant of boruvka's algorithm," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2015, pp. 610–617.

[14] J. Bang-Jensen, G. Gutin, and A. Yeo, "When the greedy algorithm fails," *Discrete Optimization*, vol. 1, no. 2, pp. 121–127, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528604000222

[15] T. J. Loftus, P. J. Tighe, T. Ozrazgat-Baslanti, J. P. Davis, M. M. Ruppert, Y. Ren, B. Shickel, R. Kamaleswaran, W. R. Hogan, J. R. Moorman *et al.*, "Ideal algorithms in healthcare: Explainable, dynamic, precise, autonomous, fair, and reproducible," *PLOS Digital Health*, vol. 1, no. 1, 2022.

[16] D.-R. Chen, L.-C. Chen, M.-Y. Chen, and M.-Y. Hsu, "A coverage-aware and energy-efficient protocol for the distributed wireless sensor networks," *Computer Communications*, vol. 137, pp. 15–31, 2019.

[17] T. Ikikardes, "Routing algorithms for safety critical wireless sensor networks," *Swiss Federal Institute of Technology Zurich, Zurich*, 2006.

[18] H. N. Gabow, J. L. Bentley, and R. E. Tarjan, "Scaling and related techniques for geometry problems," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 1984, pp. 135–143.

[19] V. S. Wietrzyk and M. A. Orgun, "Dynamic reorganization of object databases," in *Proceedings. IDEAS'99. International Database Engineering and Applications Symposium (Cat. No. PR00265)*. IEEE, 1999, pp. 110–118.