Borůvka's Algorithm

Student Number: 690065435

December 2022

Abstract

[Abstract here.]

Word Count: 1,500

1 Principles of Borůvka's Algorithm

Borůvka's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected, edge-weighted undirected graph. It originated from Otakar Borůvka in 1926, as a method of constructing an efficient electricity network for Moravia, a region of the Czech Republic [1] – this made it the first published algorithm that solved the minimum spanning tree problem in polynomial time [2]. It was independently rediscovered by numerous other researchers in later years, most notably by Georges Sollin in 1965, which has led to the algorithm also being known as Sollin's algorithm in parallel computing literature [3].

The algorithm uses a divide-and-conquer approach that is based on the idea of building a forest of trees, and is a hybrid of Kruskal's and Prim's algorithms to find a minimum spanning tree. At each step, it finds the cheapest edge that connects two different trees and combines the trees into a single tree. Borůvka's algorithm continues until there is only one tree left, which is the minimum spanning tree.

2 Pseudocode

Algorithm 1: Borůvka's Algorithm

```
Input  : A connected, edge-weighted, undirected graph $G = (V, E)$
Output: $T$, a minimum spanning tree of $G$
Initialise a tree $T$ = (V, E'), where E' = {}.
Initialise a list of components $N$, where $N_k$ denotes the vertices in component $k$.
for each vertex $v$ in $V$ do
    $N_v = v$.
while $|N| > 1$ do
    Initialise an empty list of minimum connecting edges, $L = \{\}$.
    for each component $C$ in $N$ do
        Initialise the cheapest edge $e$ to $\infty$.
        for each edge $e'$ in $C$ do
            if $e'$ contains an endpoint that isn't in $C$ and $e'$ is cheaper than $e$ then
                Set $e$ to $e'$.
        if $e$ is not $\infty$ then
            Add $e$ to $L$.
    for each edge $e$ in $L$ do
        if $e$ connects two different components then
            Merge the components $N_i$ and $N_j$ into a single component, $N_k$, such that
              $N_k = N_i \cup N_j$.
            Add $e$ to $E'$ in $T$.
```

3 Time and Space Complexity Analysis

Borůvka's algorithm has a time complexity of O(E log(V)), where E is the number of edges and V is the number of vertices in the graph.

The outer loop runs for log(V) iterations, as each iteration of the algorithm reduces the number of trees to at most half of the previous number of trees. Within each iteration, the algorithm performs a single pass over all the edges to find the minimum weight edge that connects two different components. This is performed in O(E) time, as the algorithm must consider each edge once. The algorithm then performs a single pass over all the edges to merge two components into one where possible, which is also performed in O(E) time. This gives a total time complexity of O(E log(V)).

The space complexity of Borůvka's algorithm is O(V), as it only stores the component information for each vertex, which contains each vertex's parent and minimum weight edge.

4 Limitations and Constraints

5 Applications

As Borůvka's algorithm finds a minimum spanning tree, it is most directly used in the design of networks, such as electrical networks, communication networks, and transportation networks [4]. In the original application of an electricity network for Moravia, the vertices represented towns, and edges represented the distances between towns. Borůvka used the assumption that it was not necessary to directly connect every town to the source of electricity – it was sufficient for a town to connect via another town that was already connected to power [1].

As Boruvka's algorithm satisfies the triangle inequality and finds a minimum spanning tree, it also acts as a two-approximation algorithm for the travelling salesman problem [5], which is NP-hard and thus not possible to solve in polynomial time [6]. Because it produces a two-approximation, the output is at most twice the cost of the optimal solution. This can be proven as the total cost of a full walk is at most twice the cost of the minimum spanning tree, and the algorithm returns a path with a cost less than the full walk, as our pre-order walk replaces two or more edges of the full walk with a single edge [5].

Algorithm 2: Two-Approximation for the Travelling Salesman Problem with MST-DFS [5]

```
1 Set a vertex as the start.
2 Construct a minimum spanning tree, $T$.
3 Create a list of vertices, $H$, that is ordered according to when they are visited in a pre-order
    tree walk of $T$, and add the start vertex at the end.
4 Return the path $H$.
```

There are several other algorithms that are more optimal for finding a minimum spanning tree depending on the input graph – Prim's algorithm is faster for dense graphs, and Kruskal's algorithm is faster for sparse graphs [7]. However, this only considers sequential implementations of the algorithms – Borůvka's algorithm has become increasingly popular because it is easy to parallelise and is therefore well-suited to distributed computing [8]. As it starts with multiple components and seeks to combine them with the shortest edge, it can be easily parallelised by assigning each component to a different processor.

The concepts behind Borůvka's algorithm have also been used to develop faster sequential algorithms. For example, the expected linear time minimum spanning tree algorithm proposed by Karger, Klein, and Tarjan involves an adaptation of Borůvka's algorithm, known as the Borůvka step [9, 10], alongside a step to remove F-heavy edges in linear time [11].

References

[1] J. Nešetřil, E. Milková, and H. Nešetřilová, Otakar Borůvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history *Discrete mathematics*, vol. 233, no. 1-3, pp. 3–36, 2001.

[2] A. Gupta and A. Bansal, "15-859e: Advanced Algorithms, Lecture #1: Deterministic MSTs." `https://www.cs.cmu.edu/~anupamg/advalgos15/lectures/lecture01.pdf`, 2015. Date Accessed: 27 December 2022.

[3] M. Sollin, Le trace de canalisation *Programming, Games, and Transportation Networks*, 1965.

[4] R. L. 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.

[5] T. Andreae and H.-J. Bandelt, Performance guarantees for approximation algorithms depending on parametrized triangle inequalities *SIAM Journal on Discrete Mathematics*, vol. 8, no. 1, pp. 1–16, 1995.

[6] M. Jünger, G. Reinelt, and G. Rinaldi, The traveling salesman problem *Handbooks in operations research and management science*, vol. 7, pp. 225–330, 1995.

[7] 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.

[8] A. Mariano, A. Proenca, and C. D. S. 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*, pp. 610–617, IEEE, 2015.

[9] B. Dixon, M. Rauch, and R. E. Tarjan, Verification and sensitivity analysis of minimum spanning trees in linear time *SIAM Journal on Computing*, vol. 21, no. 6, pp. 1184–1192, 1992.

[10] V. King, A simpler minimum spanning tree verification algorithm in *Workshop on Algorithms and Data Structures*, pp. 440–448, Springer, 1995.

[11] D. R. Karger, P. N. Klein, and R. E. Tarjan, A randomized linear-time algorithm to find minimum spanning trees *Journal of the ACM (JACM)*, vol. 42, no. 2, pp. 321–328, 1995.