Borůvka's Algorithm

Student Number: 690065435

December 2022

Abstract

[Abstract here.]

Word Count: 1,500

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

Signature: _____

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. Each iteration of the algorithm reduces the number of trees to at most half of the previous number of trees, so it runs in logarithmic time.

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 ∞.
        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* ∞ **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

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].

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 [5]. 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 [6]. 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 [7, 8], alongside a step to remove F-heavy edges in linear time [9].

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] 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.

[6] 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.

[7] 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.

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

[9] 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.