Borůvka's Minimum Spanning Tree Algorithm: Analysis and Implementation

Candidate number: 253158

word count: 1259

video presentation link: https://youtu.be/3ZHL_Ma0go0

Abstract

Borůvka algorithm is the first minimum spanning tree (MST) algorithm described in 1926 by Otakar Borůvka. It allows to find the cheapest way to connect all the nodes in a graph, having an application ranging from electrical networks engineering to machine learning. In this work I will discuss its applications, core principles, complexity, flaws, mitigations and use cases, explaining its importance to not only the field of Computer Science but many others.

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

1. Introduction

In 1926 Czech mathematician Otakar Borůvka published a paper [1] in an attempt to solve a problem of electrification of south-west Moravia [2]. Its goal was to find the most efficient way to connect multiple points on a map of the region. Once described in mathematical terms, Borůvka's algorithm became one of the first algorithmic solutions to the minimum spanning tree (MST) problem. A *minimum spanning tree* is a The basic idea behind the work is quite simple: it works by starting with a forest of trees, where each tree is a single node. The algorithm then iteratively adds the edges with the minimum weight to the forest of trees until there is only one tree remaining. This tree is the minimum spanning tree of the graph - most cost efficient way to connect all points (nodes) on the map (graph). Over time the work evolved to be a solution to many more problems than designing an electrical network finding its application in more and more areas.

1.1. Importance and uses

Borůvka's contributions were soon recognised by other scientists in the field, quoted by both Kruskal [3] and Prim [4]- papers behind famous more recent MST algorithms. An efficient way to compute best manner of node connection is a very common problem in countless industries, such as networking or image segmentation.

The Internet can be expressed as a complex graph of connected devices (nodes) that communicate with each other. Borůvka's algorithm can be used in network design and analysis, for example, to find the cheapest way to connect a set of computers. It is not the most efficient algorithm for finding minimum spanning trees. However, it is very easy to implement and parallelize [5], and it can be used in a dynamic setting where the graph is constantly changing such as the Internet.

Another common usage of Borůvka's algorithm is image fragmentation. It is a crucial step in computer vision and pattern recognition programs. MST algorithms allow to identify the areas of a picture, which are homogeneous according to a given criteria such as darkness, colour or gray level [6]. The image gets represented as a graph using graph theory tools. Pixels being vertices with a value representing its colour and edges connecting them with weights calculated based on colour difference or other factors. Such graph is then processed. An MST obtained using Borůvka's algorithm is then divided into multiple clusters by separating the graphs along the edges with largest weights (meaning largest colour difference). Those clusters represent the areas of the picture with some shared qualities that can be later used in computer vision algorithms to recognise patterns. Using MST algorithms for data clustering is popular due to its relatively low computational complexity, accuracy and ability to be parallelised.

Borůvka's algorithm was the first MST algorithm crucial in many areas requiring optimisation of problems that can be represented as a graph. It was the basis for Prim's and Kruskal MST algorithms, both commonly used in most recent research such as computer vision.

2. Main principles of the algorithm

1. **Algorithm summary:** The algorithm works by first connecting the closest nodes together creating multiple subgraphs which are then iteratively linked to each other using the connections with the least weight until only one graph remains.

2. **Algorithm input:** A graph G which is: *weighted*, meaning each graph's edge has a value correcponding to the cost of connection - the higher the value, the costlier the potential link and *undirected* - none of the edges connects a node A to node B without also connecting B to A, with the same weight.

3. **Algorithm output** - Graph MST which is the minimum spanning tree of G - it connects all the connected components of G in a manner that results in the lowest total weight.

2.1. Pseudocode representation

The following pseudocode describes Borůvka's algorithm: finding an MST of a graph by joining multiple subtrees together. In an attempt to explain the algorithm I will use an example graph to visually represent algorithm's steps.

Algorithm 1 Borůvka's Minimum Spanning Tree Algorithm

1: **function** Boruvka($G = Graph(V, E)$) ▷ Where G - graph, V its vertices and E - edges
2: Initialise $MST = Graph(V, E')$, where V=V and E={}
3: Let $cheapest = \{\}$
4: **while** there is more than 1 subgraphs in MST **do**
5: **for** each edge ab in E **do**
6: Let a be the first node in ab
7: Let b be the second node in ab
8: Let w be the weight of ab
9: **if** a and b belong to two different subgraphs **then**
10: Let ax be the cheapest edge for the subgraph of a
11: **if** (ax is 'None') or (weight of ax > w) or (weight of ax == w) **then**
12: Set $cheapest[a\ subgraph]$ to ab
13: **end if**
14: Let bx be the cheapest edge for the subgraph of b
15: **if** (bx is 'None') or (weight of bx > w) or (weight of bx == w) **then**
16: Set $cheapest[b\ subgraph]$ to ab
17: **end if**
18: **end if**
19: **end for**
20: **for** each subtree which $cheapest$ edge is not 'None' **do**
21: add its cheapest edge to E'
22: **end for**
23: Set $cheapest$ to an empty array
24: **end while**
return MST
25: **end function**

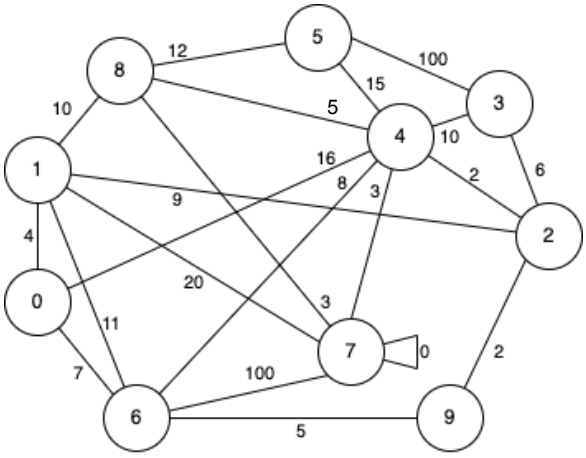

Figure 1: Initial graph: each node is its own subtree

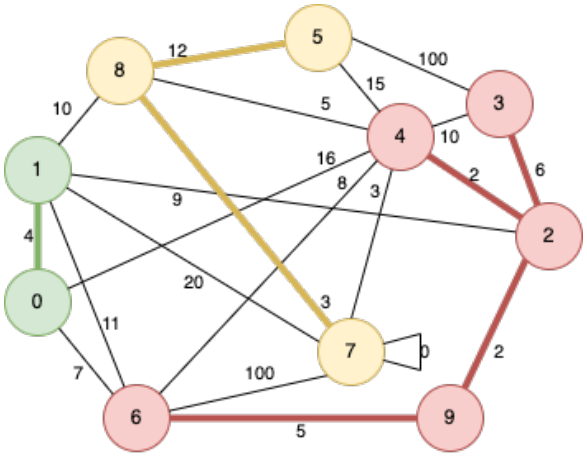

Figure 2: First iteration, each subtree connected to another subtree by the cheapest edge

The graph initially gets divided into as many subgraphs as it has nodes and the no edges yet (line 2 from Algorithm 1). Each node from Figure 1 adds a connection that is its least costly edge to the MST creating subgraphs from 2. Then, for as long as there is more than one subgraph, all nodes having edges linking them with another subgraph are examined to determine the least costly edge for each subgraph (lines 9-17) and connect it to a new subgraph (lines 20-21). Graph on Figure 3 is the result of connecting subgraphs. After that operation there is only one graph left: the MST of the input graph G. Figure 4 shows the same MST found by the python implementation of the algorithm.

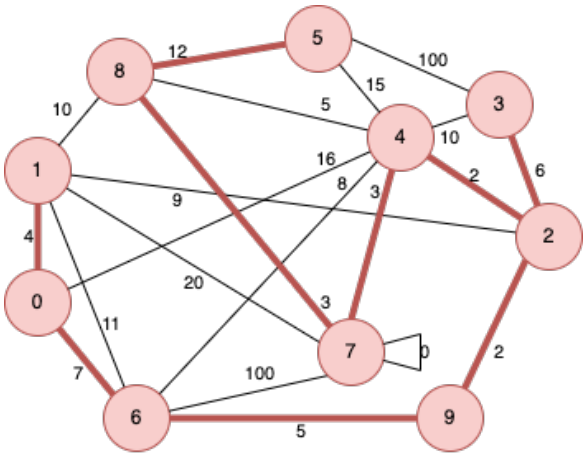

Figure 3: Second iteration of while loop (lines 4 - 24 in Algorithm 1). One subgraph left - MST of the input graph.

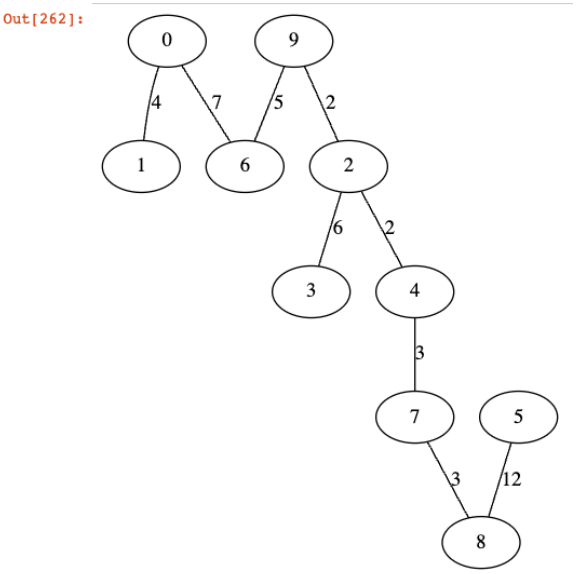

Figure 4: Same output generated by the algorithm implementation

3. Complexity analysis

3.1. Time complexity

The inner for loop (lines 5 - 19) traverses through all edges and hence runs in O(E), where E stands for number of edges in the input graph G. The second inner for loop (lines 20-22), has the same time complexity for the same reasons. The outer loop is shown to run in O(log V), where V stands for the number of vertices in G. It results in a total time complexity being O(2E log V). When simplified - **O(E log V)**. That time compplexity does not differ for the best or worse case scenario, as the algorithm will still have to traverse all edges in E no matter what, assuming no improvements. Speed of connecting subgraph determining the number of outer loop iterations does not differ either.

3.2. Space complexity

Borůvka algorithm needs table of length E - edges in the input graph and a table of length V - edges in the input graph to represent MST. Cheapest - a hashmap or a list depending on the implementation representing the cheapest connection for each subgraph. Due to the number of subgraphs being a changing variable of a small number in comparison to edges or vertices, space complexity of Borůvka algorithm can be simplified to **O(E + V)**.

4. Limitations of the algorithm and mitigations

Borůvka algorithm is not the most popular MST algorithm due to its relative complexity compared to other algorithms. For instance, in another popular solution called Prim algorithm, there is no need to add the logic of merging subgraphs. It gets even more complicated when applying methods to boost efficiency such as ”cleaning the graph” described in [7].

A counterpoint to that limitation might be that its complex nature of dividing the input graph into subgraphs allows to parallelise it achieving a great performance boost [5]. Moreover, even in a non-parallel enviroment, the previously mentioned optimisation allows for the algorithm to run in linear time by removing redundant edges after each iteration [7].

Boruvka’s algorithm is a greedy method, meaning that it makes decisions without considering the global picture. It is prone to getting stuck in local minima, leading to suboptimal solutions. In comparison, other MST algorithms such as Kruskal’s and Prim’s algorithms provide better overall solutions.

Depending on the use case, sometimes a faster suboptimal solution is beneficial to the project. Boruvka’s algorithm can fill out that niche, while other MST algorithms provide simplicity in optimisation for other cases.

5. Conclusions

Borůvka algorithm was the first MST algorithm, forever influencing the field of optimisation. It has its applications in many areas such as connecting electricity grids, networks or even image segmentation used in machine learning. Its optimised versions reach a great performance both in parallel and non-parallel settings. Despite its flaws of complexity and suboptimality, it finds modern usage cases and other algorithms inspired by it are commonly used in a wide variety of problems within Computer Science. Borůvka algorithm is undoubtedly an algorithm that has changed the world.

---

[1] O. Borvka, "Příspěvek k otázce ekonomické stavby elektrovodnỳch sítí."

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

[3] 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. [Online]. Available: http://www.jstor.org/stable/2033241

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

[5] S. Chung and A. Condon, "Parallel implementation of bouvka's minimum spanning tree algorithm," in *Proceedings of International Conference on Parallel Processing*, 1996, pp. 302–308.

[6] A. Saglam and N. A. Baykan, "Sequential image segmentation based on minimum spanning tree representation," *Pattern Recognition Letters*, vol. 87, pp. 155–162, 2017, advances in Graph-based Pattern Recognition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865516301192

[7] M. Mareš, "Two linear time algorithms for mst on minor closed graph classes," *Archivum Mathematicum*, vol. 040, no. 3, pp. 315–320, 2004. [Online]. Available: http://eudml.org/doc/249321