Borukva's Algorithm

December 12, 2022

0.1 Principles Of Boruvka's Algorithm

Borukva's algorithm is a greedy algorithm to find the minimum spanning tree (MST) of a given graph. It does this by first finding the minimum cost edge for each node in the graph, if two nodes share the same minimum cost edge they are connected into their tree. Once you have done that for all nodes in your graph you will have many smaller trees. Then, for all the trees that you have just made you find the minimum cost edge that is connected to another one of your trees, creating a smaller amount of larger trees. You repeat this step until you have only one tree remaining. The one remaining tree will be your MST.

The main advantage of Borvuka's algorithm over something like Kruskal's algorithm is that it can be parallelised easily, this is because each node finds its own minimum cost edge independently of all other nodes because they do not consider where the edge is going, only that it is the minimum. With this knowledge, we can give each thread a node and all its connected edges and compute the minimum cost edge of each node simultaneously. This makes the algorithm very scaleable and good for large graphs.

0.2 Pseudo Code

Borukva's Algorithm

```
$Inputs : AgraphGwithasetofverticesVandedgesE$
$Output : TheminimumspanningtreeofgraphG$
$V_T \leftarrow V$
$E_T \leftarrow 0$
while $|V_T| > 1$ do
    for $TinV_T$ do
        $M_n \leftarrow 0$
        for $NinT$ do
            $M_n \leftarrow append(minCostOutboundEdge(N))$
        end for
        $append(E_T, min(M_n))$
        $V_T \leftarrow combineTrees(V_T, E_T)$
    end for
end while
$returnV_T, E_T$
```

0.3 Time and Space Complexity

The time complexity of Borukva's algorithm is O(E log V) where E is the number of edges in your graph and V is the number of vertices in your graph. This is the same as Kruskal's algorithm However as stated earlier Borukva's algorithm has the advantage that you can utilise multiple threads to do your computations in parallel. The O(E log V) complexity comes because you have to do O(E) operations to calculate the minimal edge for each node and then connect our two trees. This doesn't have to be computed V times because each time your combine two of your trees your V gets smaller, this means that you only have to compute it log V times. In terms of space complexity, you only have to store the edges and the vertices as you can update the trees in place during the algorithm. This gives Borukva's algorithm a space complexity of O(E + V)

0.4 Limitations

A limitation of this algorithm is that your best performance of the algorithm is going to come from running the algorithm in a parallel context. This means that you have to be aware of potential race conditions of your algorithm and you have to make decisions on where you want to implement the parallelisations. This also means that not all machines will be able to take advantage of this algorithm's best performance. This also means in single-threaded contexts other algorithms such as Kruskal's algorithm are a better choice because they are simpler and easier to implement and understand.

0.5 Applications

Borukva's algorithm was originally developed to optimise electrical networks that they have represented as minimum spanning tree problems. the minimum spanning tree was needed for connecting electricity networks together in the Czech republic during the first world war and is the earliest known algorithm for computing the minimum spanning tree of a graph. a lot of its modern use and research is surrounding efficient ways of parallelising minimum spanning tree problems to get the most performance out of them on modern computers. This algorithm is useful for basically any minimum spanning tree problem. An example of where this can be used in a modern context is in computer vision, specifically in feature extraction. Minimum spanning trees can be used after finding edge points in images to connect them and identify features within the image and produce clearer and more simplified, general outputs (1) which can be used as a stage in image recognition or other machine learning environments. minimum spanning trees can also be used in clustering algorithms by using the longest edges in the MST to identify where clusters are separated. This type of clustering is also very useful as it can capture clusters of all different shapes because the factor that clusters is how close they are to the other data points in their cluster and how far they are from datapoints not in their cluster, this means that the shapes produced by MST clustering algorithms can be quite complex (2). This is also a very useful area of machine learning that MSTs can help us to solve and therefore by extension borukva's algorithm to find those MSTs. It's also to Borukva's algorithm's favour that these modern machine learning problems require modern computers that are capable of squeezing the most performance out of borukva's algorithm because they will most likely have multiple processor cores and threads that can be used in the algorithm. and so this algorithm is still continuing to change the world even after almost 100 years.

Bibliography

[1] M. Suk and O. Song, "Curvilinear feature extraction using minimum spanning trees," *Computer Vision, Graphics, and Image Processing*, vol. 26, no. 3, pp. 400–411, 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0734189X84902214

[2] P. K. Jana and A. Naik, "An efficient minimum spanning tree based clustering algorithm," in *2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS)*, 2009, pp. 1–5.