Mao-Chi (Mike) Weng
CS537, Prof. Homer
12/5/2013

Randomized Linear Time Algorithm to Find Minimum Spanning Trees

Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A Minimum Spanning Tree (MST) is a spanning tree with weight less than or equal to the weight of every other spanning tree. Finding MST has been in the field of graph theory for some time. In 1926, Czech scientist Otakar Boruvka wants to solve the problem of finding an efficient coverage of Moravia with electricity. Boruvka developed the first known algorithm for finding a minimum spanning tree, Boruvka's algorithm. In this research paper, we will explain the key properties of MST, and present Boruvka's Algorithm. Finally, we will introduce and analyze the randomized algorithm to find MST along with the random sampling lemma that supports this algorithm.

Definition of the problem:

Given an undirected and weighted graph $G = (E, V, w)$, E is the set of edges, V is the set of vertices, w is the weight function. Construct a minimum spanning tree $G' = (E', V, w)$ so that G' is connected and $\Sigma_{e \in E'} w(e)$ is minimum (that is, total weight is minimum).

Minimum spanning tree has two important properties: Cycle Property and Cut Property. **Cycle Property** (Figure 1) states: *For any cycle C in a graph, the heaviest edge in C does not appear in the minimum spanning forest.* **Cut Property** (Figure 2) states: *For any proper nonempty subset X of the vertices, the lightest edge with exactly one endpoint in X belongs to the minimum spanning forest.* Below two diagrams are examples of above properties.

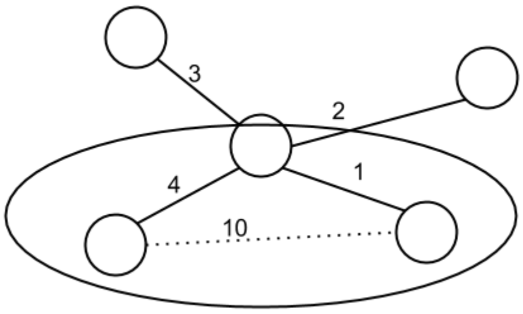

Figure 1. Cycle Property. Heaviest edge 10 in cycle does not appear in MST.

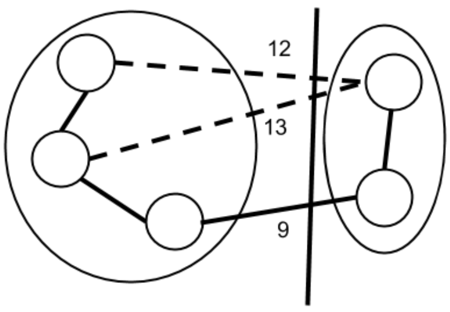

Figure 2: Cut Property. In this cut, 9 will be in the MST.

Boruvka's algorithm *(Example: Figure 3)*

```
Input: A connected graph G whose edges have distinct weights
Initialize a forest T to be a set of one-vertex trees, one for each vertex of the graph
While T has more than one component:
    For each component C of T:
        Begin with an empty set of edges S
            For each vertex v in C:
                Find the lightest edge from v to a vertex outside of C, and add it to S
            Add the lightest edge in S to T
Output: T is the minimum spanning tree of G.
```

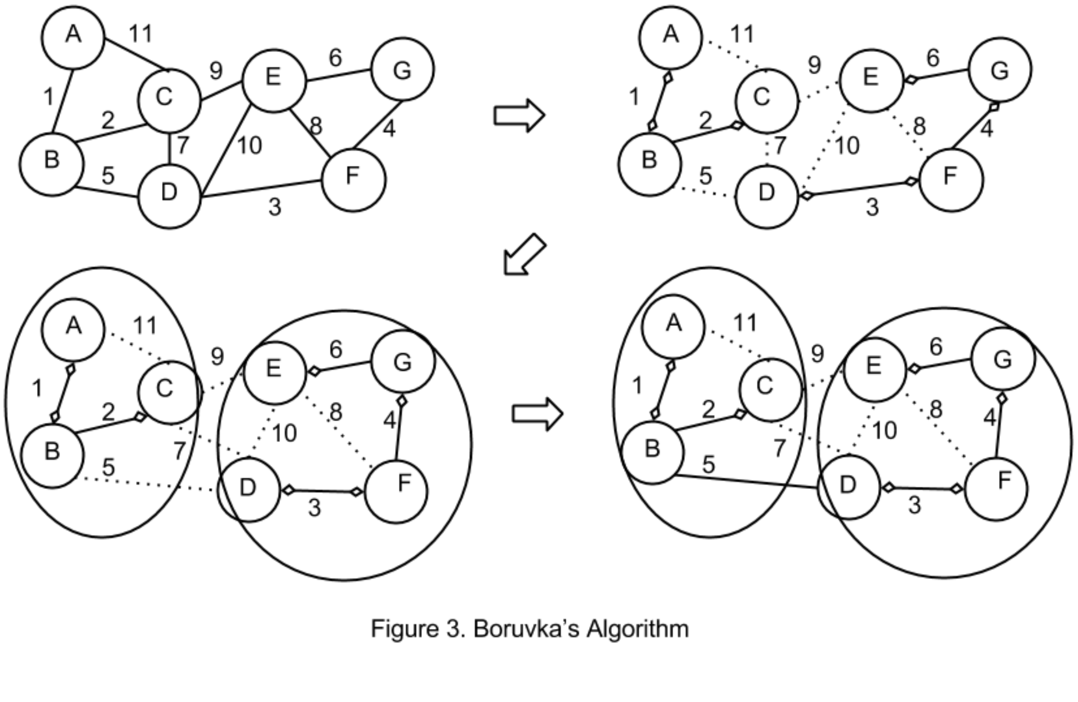

Figure 3. Boruvka's Algorithm

The running time of Boruvka's algorithm can be shown to take O(log V) iterations of the outer loop until it terminates, where each iteration runs in O(E), and therefore to run in time O(E log V), where E is the number of edges and V is the number of vertices in graph G.

There are many variations of the Boruvka's algorithm that are based on Boruvka's original algorithm. An important variation is an adaption of the original algorithm involving a technique called Boruvka Step. It is defined as follow:

Boruvka Step *(Example: Figure 4)*
Input*: A graph G with no isolated vertices*
1. For each vertex v, select the lightest edge incident on v
2. Create a contracted graph G' by replacing each component of G connected by the edges selected in step 1 with a single vertex
3. Remove all isolated vertices, self-loops, and non-minimal repetitive edges from G'
Output*: The edges selected in step 1 and the contracted graph G'*

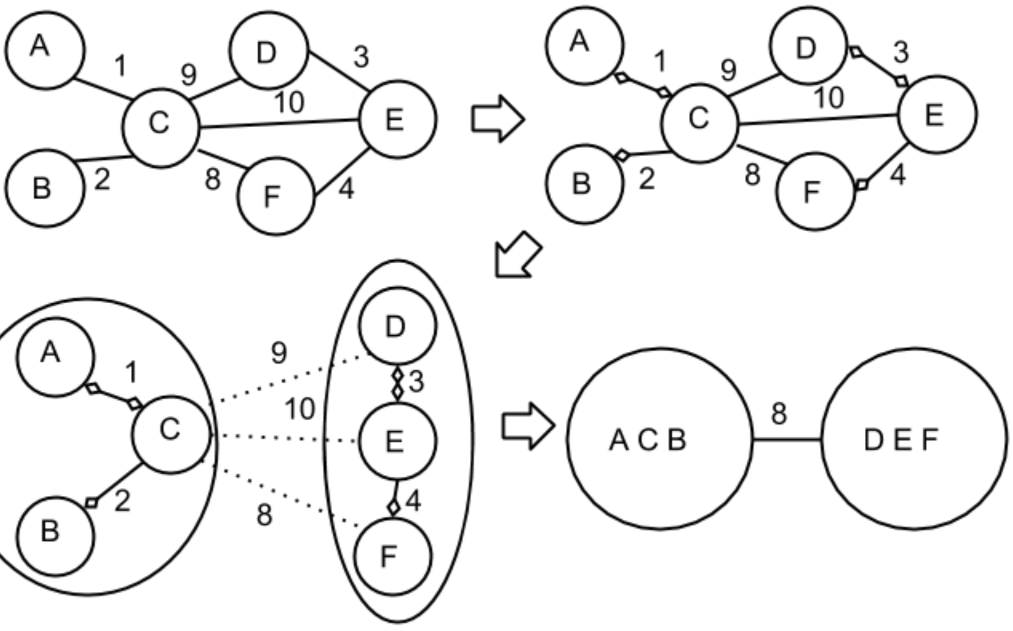

Figure 4. Boruvka Step

A Boruvka Step is equivalent to the inner loop of Boruvka's algorithm. This step runs in O(E) time, where E is the number of edges in input graph G. Observe that each edge can be selected at most twice (once by each end of vertex), the maximum number of disconnected components after step 1 is equal to half the number of vertices. Thus, a Boruvka step reduces the number of vertices in a graph by at least a factor of two and deletes at least V/2 edges where V is the number of vertices in input graph G.

Another important concept is F-Heavy (Figure 5) and F-Light edges. It is defined as: "Let *G* be a graph with weight edges. We denote by *w(x, y)* the weight of edge *{x, y}*. If *F* is a forest in *G*, we denote by *F(x, y)* the path (if any) connecting *x* an *y* in F, and by *wF(x, y)* the maximum weight of an edge on *F(x, y)*, with the convention that *wF(x, y) = ∞* if *x* and *y* are not connected in *F*. We say that an edge *{x, y}* is *F-heavy* if *w(x, y) > wF(x, y)* and *F-light* otherwise". By this definition and the cycle property, no F-heavy edge can be in the minimum spanning forest of G. Because every edge in F satisfy the definition that *w(x, y) ≤ wF(x, y)*.

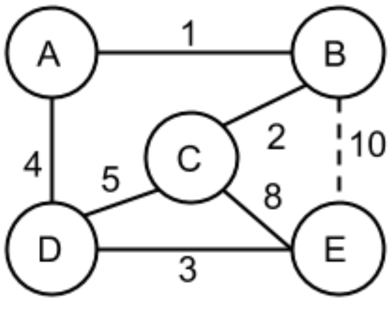

Figure 5. F-Heavy Edge. w(B, E) = 10.
wF(B, E) = 8. (Edge CE).
w(B, E) > wF(B, E). Therefore, edge BE is F-Heavy

Our randomized algorithm relies on a random-sampling step to discard edges that cannot be in the minimum spanning tree.

Lemma 1
Let H be a subgraph obtained from G by including each edge independently with probability p, and let F be the minimum spanning forest of H. The expected number of F-light edges in G is at most n/p where n is the number of vertices of G.

Proof (Akhtar, Wu [2]):
A F-light edge is included in F if it has been chosen by G(p) during construction of G(p). Therefore, among all F-light edges in G, each such edge is included in F with probability p. Suppose |F| = s. Then include on such edge in F is like flipping p-coin multiple times until we get a head. So including all s such edges in F is to flip p-coin until we get s head. Therefore the total number of F-light edges is represented by the total number of coin flips, which is a random variable with negative binomial distribution X~(s, p).

Furthermore, s is bounded by n-1 by the definition of minimum spanning forest. We can use Y~(n, p) which stochastic dominates X to bound it. Finally we get E[Y] = n/p, which is the expected number of F-light edges in G.

With Boruvka steps and random-sampling lemma we can now introduce the randomized algorithm to find minimum spanning tree.

***Algorithm**:*
0: If the graph is empty, return an empty forest. Otherwise, proceed as follows:
1: Apply two successive Boruvka steps to the graph, there by reducing the number of vertices by at least a factor of four.
2: In the contracted graph, choose a subgraph H by selecting each edge independently with probability 1/2. Apply the algorithm recursively to H, producing a minimum spanning forest F of H. Find all the F-heavy edges (both those in H and those not in H) and delete them from the contracted graph.
3: Apply the algorithm recursively to the remaining graph to compute a spanning forest F'. Return those edges contracted in Step 1 together with the edges of F'.

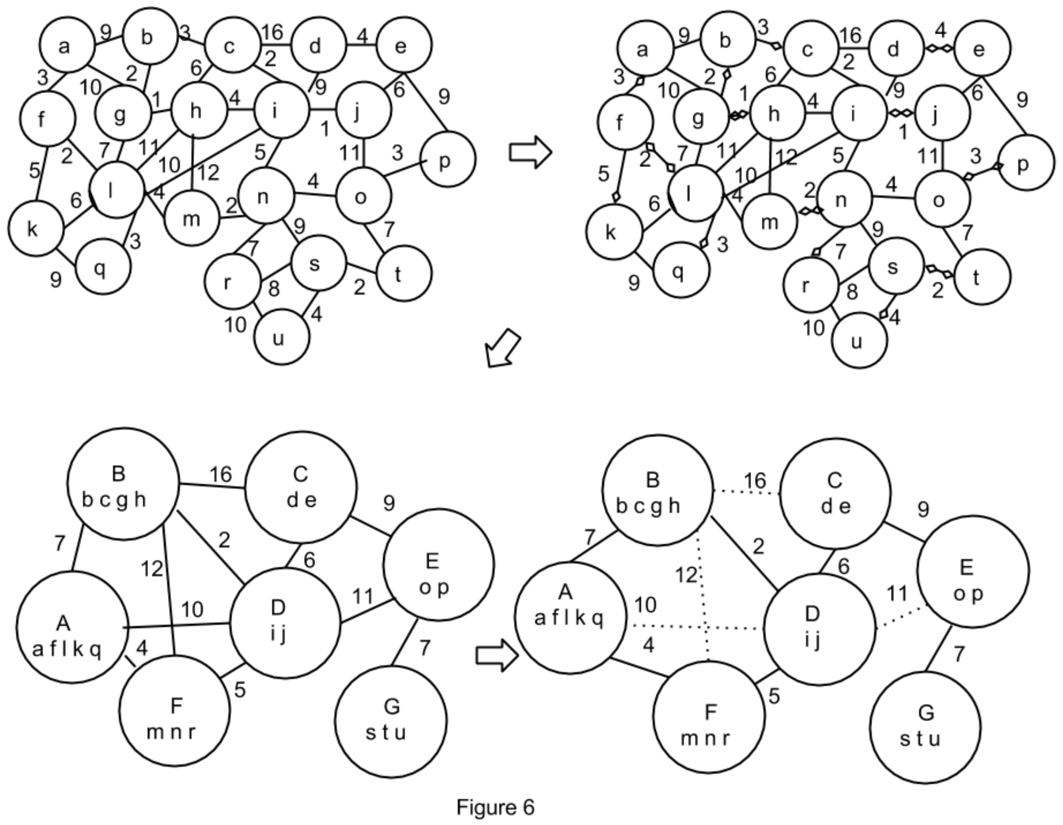

Figure 6

Figure 6 shows Step 1 to Step 2 from the randomized algorithm. Assuming the first graph is the result of the first Boruvka step. The third graph shows the result of the second Boruvka step. And the third graph shows the result of random-sampling. We can easily see the result of step 2 by removing edge AB in the third graph (edge gl in the first graph), since it is the F-heavy edge in the graph.

We can see that by Cut Property, every edge contracted during Step 1 is in the minimum spanning forest. Moreover, by Cycle Property, the edges deleted in Step 2 do not belong to the minimum spanning forest. Analysis of the algorithm states that the expected running time of this algorithm is O(V+E) with V being the number of vertices and E being the number of edges from the given graph G.

Proof (Karger, Klein, Tarjan [1], Akhtar, Wu[2]):
*Two Boruvka's steps takes O(2E) time. Random sampling reads O(V) vertices and samples O(E) edges, thus it takes O(V+E) time. The first recursive step takes T(V/4, E/2). And by random-sampling lemma it should return V/4 vertices and V/4*2 = V/2 edges. Thus, the second recursive step takes T(V/4, V/2) time. And finally, the joining takes O(V) time. Hence, the total running time is O(2E) + O(V+E) + T(V/4, E/2) + T(V/4, V/2). By master theorem, we can see that the expected running time is O(V+E).*

References

[1] David R. Karger, Philip N. Klein, Robert E. Tarjan. *A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees*. 1995
[2] Nabeel Akhtar, Hanwen Wu. *Minimum Spanning Trees*. 2013
[3] Web. *Minimum Spanning Tree History.*
http://simple.wikipedia.org/wiki/Minimum_spanning_tree#History
[4] Web. *Boruvka Algorithm.*
http://en.wikipedia.org/wiki/Bor%C5%AFvka%27s_algorithm
[5] Web. *Expected Linear Time MST Algorithm*.
http://en.wikipedia.org/wiki/Expected_linear_time_MST_algorithm
[6] Web. *Minimum Spanning Tree.*
http://en.wikipedia.org/wiki/Minimum_spanning_tree