Minimum Spanning Trees

by Hanwen Wu

Nov. 2013 - CS537 Randomized Algorithms - Prof. Steve Homer

1 Background

Members of some secret party want to shutdown some high risk communication channals, while preserving low risk channals to ensure that members are still connected in the secret network.

2 Problem Definition

Given a 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 $\sum_{e\in E'} w(e)$ is minimum (that is, total weight is minimum).

Note. We assume edges have distinct weights for simplicity.

3 Properties of MST

3.1 Property

The MST of G will have $n-1$ edges if $|V|=n$.

3.2 Tree Properties

Adding an edge to a tree will form a loop. Removing an edge of a tree will result in two disconnected subtrees.

3.3 Cut Property

Given G and some cut of G, the crossing edge with minimum weight is in the MST of G. This can be proved by contradiction. Another way of saying it is, given a subset of V, the edge with minimum weight and incident to this subset, is in the MST.

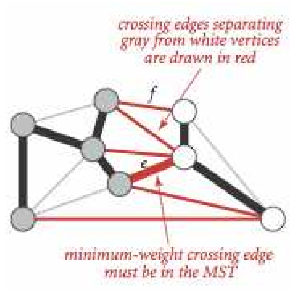
crossing edges separating gray from white vertices are drawn in red
f
e
minimum-weight crossing edge must be in the MST

Figure 1. Cut Property (*Image credits: http://algs4.cs.princeton.edu/43mst/*)

3.4 Cycle Property

Given G, and a cycle C in G. The heaviest edge on C is not in the MST of G.

4 Deterministic Algorithm - Kruskal's

The basic idea is to use cycle property.

Algorithm

1. Sort edges from lightest to heaviest as $e_1, e_2, \cdots, e_n$. Initilize an empty edge set T
2. Examine these edges from left to right
3. If it doesn't form a cycle in T, then add it to T.
4. If it does form a cycle in T, do not add it.

5 Deterministic Algorithm - Boruvka's

Lemma 1. *Let $v \in V$ be any vertex in G, the MST of G must contain edge (v, w) that is the minimum weight edge incident on v.*

Proof. Consider the cut defined by that vertex v and all other vertices. Use the Cut Property. □

Boruvka's algorithm is to concontract simultaneously the minimum weight edges incident on each vertex, and form a new contracted graph with contracted vertices. Repeat this step until a single vertex. Or in some other text, starting from n blue trees (consisting of n vertices in V), select a minimum weight edge incident to each blue tree, and color all selected edges blue. Repeat until there is only one single blue tree.

Algorithm

1. start from n vertices as n individual blue trees
2. for every blue tree, mark the edge with minimum weight incident to it as blue
3. repeat until a single blue tree
4. output all blue edges, and that is the MST

Note. The coloring phase is called Boruvka's Step/Phase.

5.1 Analysis

Each Boruvka's Step will reduce the number of blue trees by at least a factor of two, because for every blue tree, there has to be at least one chosen edge, and two blue trees may choose the same edge in the worst case. Therefore, we need $O(\log n)$ step to get to a single tree, which means the termination of the algorithm.

In every single step, we need to scan at most $O(m)$ edges. Therefore, the overall complexity is $O(m \log n)$.

6 Heavy Edges and MST Verification

We introduce some symbols. Given any forest F of G, $w_F(u, v)$ denotes the edge with maximum weight among all edges on the path in F from u to v. If there is no path, $w_F(u, v) = \infty$. $w(u, v)$ is the weight of edge (u, v) of G. Let $p(u, v) = \{e | \, e \in F, e \, \text{is on the path from} \, u \, \text{to} \, v\}$

Definition. *An edge $(u, v) \in E$ is F-heavy if $w(u, v) > w_F(u, v)$. Otherwise, it is F-light.*

Note. F-heavy edges can't be in F. Because every edge in F satisfy $w(e \in p(u, v)) \leqslant w_F(u, v)$.

Lemma 2. *Let F be any forest in G, if an edge (u, v) is F-heavy, then it doesn't lie in the MST of G. The converse is not true.*

Proof. We can prove it using contradiction. Consider to replace $p(u,v)$ with (u,v). □

6.1 MST Verification

This is not our focus, but since the randomized algorithm needs a linear time verification, we briefly introduce some backgroud.

The basic idea of verification is for every edge $e \notin F$, test if it is F-heavy. There are linear time verification algorithms.

Definition. *We are constructing a branching tree B of the MST of G based on Boruvka's Step. In every step i, each blue tree is mapped onto a node at level i in B. And each edge being chosen in step i is mapped on to an edge between level i and level $i-1$ in B, whose parent-end is the node representing the blue tree in G in step i, and whose child-end is the node representing the blue tree in G in step $i-1$, and whose weight is the same as it is in G. (If an edge is begin chosen twice, it appears twice in B.) Here, $i \geqslant 0$ where level 0 is leaf level in B.*

In short, by looking at B, we immediatly observe the combining/contracting process of blue trees, until we get the single blue tree at the end, represented by the root node of B.

We are about to test F-heavy edges using B instead of MST of G. Therefore we need this.

Lemma 3. *Let T be MST of G, let B be the branching tree of T. Let $p_T(u,v)$ be the set of edges representing the path from u to v in T. Let $p_B(u,v)$ be the set of edges representing the path from u to v in B. Then*

$$\max\left(w(e)|\, e \in p_T(u,v)\right) = \max\left(w(e)|\, e \in p_B(u,v)\right)$$

Proof. Left for the readers. Proof by contradiction. □

And the algorithm adopts Komlos's algorithm to make use the properties of B, together with some specially designed data structures to achieve linear time. And we are not going to cover this.

What we should know is we can identify F-heavy edges in $O(m+n)$ time.

7 Preliminary

7.1 Negative Binomial Distribution

Suppose X_1, X_2, $\cdots$, X_n are independent random variables with geometric distribution with parameter p. Then $X = \sum_i X_i$ stands for the number of coin flips we need to get n heads. It is a random variable with negative binomial distribution with parameter n and p. And we have

$$E[X] = \frac{n}{p}$$

7.2 Stochastic Dominance

A random variable X stochastically dominates another random variable Y if, $\forall z \in \mathbb{R}, \Pr(X > z) \geqslant \Pr(Y > z)$. And if X stochastically dominates Y we have

$$E[X] \geqslant E[Y]$$

And for negative binomial r.v. $X \sim (n_1, p)$ and $Y \sim (n_2, p)$, if $n_1 \geqslant n_2$, X stochastically dominates Y.

Proof. (Hint)

$$\Pr(X > z) = 1 - \Pr(X \leqslant z) = 1 - \sum_{i=n_1}^{z} \Pr(X = i)$$

$$\Pr(Y > z) = 1 - \Pr(Y \leqslant z) = 1 - \sum_{i=n_2}^{z} \Pr(Y = i)$$

□

7.3 Random Graph

Given a graph G, $G(p)$ is a random graph generated by individually including each edge of G with probability p. Expected number of edges is mp.

Note. $G(p)$ may not be connected.

8 Linear Time MST

In Boruvka's MST, we reduce the number of vertices in each phase. Here, we use randomness to reduce the number of edges in each phase.

And, the basic idea comes from the fact that the minimum spanning forest of $G(p)$ is actually a very good approximation of the MST of G, which we will prove in the following sections.

8.1 Random Sampling Lemma

Here, we are dealing with $G = (V, E)$, $G(p) = (V, E(p))$, and F (the MSF of $G(p)$). And we are going to prove that for such F, F-light edges in G is bound by n/p. ($|V| = n$).

Note. The bound is used for complexity analysis, but the F is used for removing F-heavy edges in the algorithm.

Note. Once $G(p)$ is fixed, F is fixed. All edges in F is F-light, all remaining edges in $G(p)$ is F-heavy, but other edges in G could be F-heavy or F-light.

A F-light edge is finally 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 finally included into F with probability p. Suppose $|F| = s$. Then include one 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 r.v. with negative binomial distribution $X \sim (s, p)$.

Furthermore, s is bounded by $n - 1$ by the definition of MSF. We can use $Y \sim (n, p)$ 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. So we have this.

Lemma 4. *Given $G, G(p)$ and the minimum spanning forest F of $G(p)$, $|V| = n$, then the number of F-light edges in G is bounded by n/p.*

8.2 The Algorithm

Algorithm

Input: $G = (V, E, w), |V| = n, |E| = m$
Output: MSF of G

1. If G is empty, return empty forest.
2. Perform three Boruvka's step on G to get G_1 with at most $n/8$ vertices. Let C be those edges being colored blue in these three steps. Note after coloring, we are contracting each components into a signle node.
3. Let $G_2 = G_1(p)$ with $p = 0.5$.
4. Recursively running this algorithm on G_2 to get the minimum spanning forest F_2 of G_2.

5. Running linear time verification algorithm on G_1 to remove F_2-heavy edges, and get G_3
6. Recursively running this algorithm on G_3 to get the minimum spanning forest F_3 of G_3
7. Return $C \cup F_3$.

8.3 Analysis

Let $T(n,m)$ denote the running time on a graph with n vertices and m edges.

1. Step 2 runs three Boruvka's steps. Coloring takes $O(m)$ time while contracting takes $O(n)$ time. Therefore this step takes $O(n+m)$ time.
2. Step 3 is a random sampling, which reads $O(n)$ vertices, and samples $O(m)$ edges. Takes $O(n+m)$ time.
3. Step 4 takes $T(n/8, m/2)$ expected running time.
4. Step 5 we use linear verification algorithm that takes $O(n+m)$ time. And G_3 has expected $n/8$ vertices and $\frac{n}{8} \times 2 = n/4$ edges by the sampling lemma.
5. Step 6 takes $T(n/8, n/4)$ time. Step 7 takes $O(n)$ time.

Therefore,

$$T(n,m) \leqslant T(n/8, m/2) + T(n/8, n/2) + c \times O(m+n)$$

Solve it and we get $O(n+m)$.