The KKT-Algorithm for Minimum Spanning Trees

Wolfgang Mulzer

1 The Problem

We are given a connected undirected graph $G = (V, E)$ with n vertices and m edges, together with a weight function $w : E \to \mathbb{R}$ on the edges. Without loss of generality, we assume that all weights are pairwise distinct. The goal is to compute a *minimum spanning tree* of G, i.e., a set $F \subseteq E$ of edges such that the subgraph (V, F) is acyclic and connected, and such that the total edge weight $\sum_{e \in F} w(e)$ is minimum among all acyclic connected subgraphs of G. We denote F by $\mathrm{mst}(G)$. The following fact is well known.

Fact 1.1. *Let $G = (V, E)$ be a connected undirected weighted graph such that all edge weights are pairwise distinct. Then $\mathrm{mst}(G)$ is uniquely determined.* □

The minimum spanning tree problem has been studied intensively over the last decades, and many different algorithms are known. The first algorithm is due to Borůvka. It takes $O(\min\{n^2, m \log n\})$ steps and is described in more detail below. Two further classic algorithms are Kruskal's algorithm, a greedy algorithm that runs in $O(m \log n)$ time, and the algorithm of Prim/Jarnik/Dijkstra, that requires $O(m+n \log n)$ steps if implemented with an appropriate heap structure, such as Fibonacci heaps. These algorithms have been improved significantly, and the currently fastest general deterministic algorithm is due to Chazelle. It needs $O(m\alpha(m, n))$ time, where $\alpha(m, n)$ is the slow-growing inverse Ackermann function.

If we are allowed to use randomness, there is a conceptually simple linear-time algorithm due to Karger, Klein, and Tarjan. We will describe this algorithm here.

2 Ingredients

We begin by describing some of the ingredients that the KKT algorithm borrows from previous MST-algorithms.

2.1 Borůvka's Algorithm

Borůvka's algorithm consists of an iterative application of *Borůvka phases*. In a Borůvka phase, we are given a graph $G_1 = (V_1, E_1)$, and we obtain from it a smaller graph $G_2 = (V_2, E_2)$ by *contracting* edges that are guaranteed to be in $\text{mst}(G_1)$. This is done as follows: for each vertex $v \in V_1$, we select the incident edge of minimum weight. Let F be the resulting edges. The graph G_2 contains a node for each connected component of (V_1, F). For two distinct nodes $a, b \in V_2$, the graph G_2 has an edge e between $a, b \in V_2$ if and only if G_1 has an edge between a node in component a and a node in component b. The weight of e is chosen as the minimum of all edge weights between a and b. The *corresponding edge* of e in G_1 is the edge of minimum weight between components a and b in G_1. A Borůvka phase has the following properties:

Lemma 2.1. *Let $G_1 = (V_1, E_1)$ be connected, undirected, and weighted. Apply a Borůvka phase to G_1, and let $G_2 = (V_2, E_2)$ be the result. Denote the set of contracted edges by F. Then G_2 has at most half as many vertices as G_1, i.e., $|V_2| \leq |V_1|/2$. Furthermore, the MST of G_1 is obtained by taking F and adding the corresponding edges of $\text{mst}(G_2)$ in G_1. The Borůvka phase can be implemented in $O(|E_1|)$ steps.*

Proof. Exercise. □

Borůvka's algorithm starts with $G_1 = G$ and applies Borůvka phases until there is only one node left. Let $G_1, \ldots, G_k$ be the resulting sequence of graphs, and let F_i be the edges that were contracted in G_i during the ith Borůvka phase. Let $\widehat{F}_i$ be the corresponding edges in G. The algorithm returns $\bigcup_{i=1}^{k-1} \widehat{F}_i$.

Theorem 2.2. *Borůvka's algorithm correctly computes $\text{mst}(G)$. It can be implemented in $O(\min(n^2, m \log n))$ steps.*

Proof. This follows from Lemma 2.1. The details are left as an exercise. □

2.2 MST Verification

The second ingredient is an algorithm for checking whether an acyclic subgraph of G is an MST. This algorithm also provides a witness of non-minimality. The following definition makes this notion of witness precise. Let $A \subseteq E$ be acyclic. An edge $e \in E$ is called *A-light* if either (i) $A \cup \{e\}$ is acyclic; or (ii) the unique cycle in $A \cup \{e\}$ contains an edge of weight larger than $w(e)$. Note that case (i) also applies if $e \in A$. The following lemma shows the usefulness of our definition.

Lemma 2.3. *Let $G = (V, E)$ be undirected and weighted, and let $A \subseteq E$ be such that (V, A) is an acyclic graph. Then $\text{mst}(G)$ is contained in the set of A-light edges in E. Furthermore, A is the MST of G if and only if there are no A-light edges outside of A.*

Proof. Exercise. □

The following theorem is due to Dixon, Rauch, and Tarjan. A simpler algorithm was later presented by King.

Theorem 2.4. *Let $G = (V, E)$ be undirected, connected, and weighted, and let $A \subseteq E$ be acyclic. We can find all A-light edges in G in total deterministic time $O(m + n)$.* □

3 Algorithm

The problem with Borůvka's algorithm is that although each phase at least halves the number of vertices, it does not necessarily reduce the number of edges significantly. Thus, it may be that some edges are considered during a logarithmic number of phases. To avoid this, we use random sampling in order to reduce the number of edges. We compute a minimum spanning *forest* F of the resulting graph. The random sampling may discard edges that belong to the MST. To correct for these errors, we determine the F-light edges L and recompute the MST with respect to L. The hope is that L will be small. Details follow.

The algorithm gets an undirected and weighted graph $G = (V, E)$ as input. We do not necessarily require that G is connected. The goal is to compute a *minimum spanning forest* for G, i.e., a minimum spanning tree for each connected component of G. We denote it by $\mathrm{msf}(G)$.

Initially, the algorithm performs three Borůvka phases on G. This reduces the number of edges at least by a factor of 8. Let $G_1 = (V_1, E_1)$ be the resulting graph, and $F_1 \subseteq E$ the corresponding set of contracted edges. We pick a random subset $R \subseteq E_1$ with $|R| = |E_1|/2$, and we recursively compute $\mathrm{msf}(V_1, R)$. Let F_2 be the result. We use Theorem 2.4 to find the set $L \subseteq E_1$ of F_2-light edges in G_1. We recurse again to compute $\mathrm{msf}(V_1, L)$. Let F_3 be the corresponding edges in G. We return $F_1 \cup F_3$ as the MSF of G.

4 Analysis

The correctness of the algorithm follows from Lemmas 2.1 and 2.3. Let $T(m, n)$ denote the expected running time on a graph with m edges and n vertices. By Lemma 2.1, the initial three Borůvka phases take $O(n + m)$ steps. The time to find the first random sample also requires $O(n + m)$ time. Since the three Borůvka phases reduce the number of vertices at least by a factor of 8, and since the sample contains at most $m/2$ edges, the first recursive call needs $T(n/8, m/2)$ steps. By Theorem 2.4, it takes $O(n + m)$ time to find the F_2-light edges. Finally, the second recursive call takes time $T(n/8, |L|)$. This gives the following recursion for T:

$$T(n, m) = O(n + m) + T(n/8, m/2) + T(n/8, |L|).$$

The following lemma bounds the expected size of L.

Lemma 4.1. *Let $G = (V, E)$ be undirected and weighted, and $R \subseteq V$ random with $|R| = |V|/2$. Let $F = \mathrm{msf}(V, R)$. The expected number of F-light edges in G is at most $2|V|$.*

Using Lemma 4.1, we can use induction and linearity of expectation to show that $T(n, m) = O(n + m)$. It remains to prove Lemma 4.1. The following proof is due to Timothy M. Chan.

Proof of Lemma 4.1. Using linearity of expectation, we have

$$\mathbf{E}[\#\text{ of } F\text{-light edges}] = \sum_{e \in E} \Pr[\text{e is } F\text{-light}].$$

Observing that $e \in E$ is F-light if and only if $e \in \mathrm{msf}(V, R \cup \{e\})$ (Exercise) and distinguishing the choices for R, this becomes

$$\mathbf{E}[\#\text{ of } F\text{-light edges}] = \sum_{e \in E} \sum_{\substack{S \subseteq E \\ |S| = |E|/2}} \Pr[R = S][e \in \mathrm{msf}(S \cup \{e\})].$$

Here, the notation $[P]$ gives 1 if the condition P is true and 0 if the condition P is false. Now we change the order of summation and distinguish whether $e \in S$. We get

$$\mathbf{E}[\#\text{ of } F\text{-light edges}] = \sum_{\substack{S \subseteq E \\ |S| = |E|/2}} \sum_{e \in S} \Pr[R = S][e \in \mathrm{msf}(S \cup \{e\})] + \sum_{\substack{S \subseteq E \\ |S| = |E|/2}} \sum_{e \in E \setminus S} \Pr[R = S][e \in \mathrm{msf}(S \cup \{e\})].$$

We bound the sums individually. For the first sum, we have

$$\begin{aligned} \sum_{\substack{S \subseteq E \\ |S| = |E|/2}} \sum_{e \in S} \Pr[R = S][e \in \mathrm{msf}(S \cup \{e\})] &= \sum_{\substack{S \subseteq E \\ |S| = |E|/2}} \Pr[R = S] \sum_{e \in S} [e \in \mathrm{msf}(S \cup \{e\})] \\ &= \sum_{\substack{S \subseteq E \\ |S| = |E|/2}} \Pr[R = S] |\mathrm{msf}(S)| \\ &\leq |V| - 1, \end{aligned}$$

since $\mathrm{msf}(S)$ has at most $|V| - 1$ edges, for any choice of $S \subseteq E$.

We rewrite the second sum as

$$\sum_{\substack{S \subseteq E \\ |S| = \frac{|E|}{2}}} \sum_{e \in E \setminus S} \Pr[R = S][e \in \mathrm{msf}(S \cup \{e\})] = \frac{|E|}{2} \sum_{\substack{S \subseteq E \\ |S| = \frac{|E|}{2}}} \sum_{e \in E \setminus S} \frac{2 \Pr[R = S]}{|E|} [e \in \mathrm{msf}(S \cup \{e\})].$$

This sum can be interpreted as follows: pick a random subset $S \subseteq E$ with $|S| = |E|/2$, and then pick a random edge $e \in E \setminus S$. What is the probability that $e \in \mathrm{msf}(S \cup \{e\})$? An equivalent formulation is that we first pick a random subset $S' \subseteq E$ with $|S'| = |E|/2 + 1$, and then we pick a random edge $e' \in S'$. Now we want the probability that $e' \in \mathrm{msf}(S')$. This second formulation gives an easy estimate for the sum: since $\mathrm{msf}(S')$ has at most $|V|-1$ edges, it is at most

$$\frac{|V| - 1}{|S'|} \leq \frac{|V|}{|E|/2}.$$

Hence,

$$\mathbf{E}[\#\text{ of } F\text{-light edges}] \leq |V| - 1 + \frac{|E|}{2} \cdot \frac{|V|}{|E|/2} < 2|V|,$$

as claimed. □