Two Linear Time Algorithms for MST on Minor Closed Graph Classes

Martin Mareš*

Department of Applied Mathematics
and Institute for Theoretical Computer Science (ITI)
Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

E-mail: *mares@kam.mff.cuni.cz*

ABSTRACT: This article presents two simple deterministic algorithms for finding the Minimum Spanning Tree in $O(|V|+|E|)$ time for any proper class of graphs closed on graph minors, which includes planar graphs and graphs of bounded genus. Both algorithms require no a priori knowledge of the structure of the class except for its density; edge weights are only compared and no random access to data is needed.

1. Introduction: The MST problem

The problem of finding a minimum spanning tree of a weighted undirected graph is one of the most well-known algorithmic problems of combinatorial optimization. Since the first solution by Borůvka [1] in 1926 (see [9] for an English translation), a plethora of increasingly more efficient algorithms has been developed (for the full story, see [7] and [4]).

Assuming that edge weights are taken from an arbitrary ordered set (the only operation defined on them is comparision), the current speed record is held by the algorithms of Chazelle [2] and Pettie [10] which achieve time complexity $O(m \cdot \alpha(m,n))$ where n and m are the number of vertices and edges of the graph and $\alpha(m,n)$ is an inverse of the Ackermann's function. If the edge weights are integers whose bits can be manipulated in constant time, there exists a MST algorithm by Fredman and Willard [3] running in linear time on an unit-cost RAM. Also, there is a randomized algorithm with expected linear tme for the general case due to Karger et al. [5].

---

* Partially supported by the Project LN00A056 of the Czech Ministry of Education and by the Forschungsinstitut für Mathematik, ETH Zürich, Switzerland

Recently, Pettie and Ramachandran [11] have shown an algorithm for the pointer machine with running time bound by the size of the optimum MST decision tree. Since the decision-tree complexity is an obvious lower bound for the algorithmic time complexity of the problem, this algorithm is optimal up to a multiplicative constant and that no random access is needed to achieve optimality. However, the decision-tree complexity of the MST is still unknown and no non-trivial lower bounds are known, hence it's still open whether the MST can be found in linear time or not.

Although for general graphs it's still unresolved, there are several special cases where linear-time algorithms are known to exist. When the graph is sufficiently dense, meaning that it has at least $n \cdot \log^{(k)} n$ edges[1] for some k, Tarjan's $O(m \cdot \beta(m,n))$ algorithm [12] performs linearly. On the other end of the spectrum, the exist several $O(m+n)$ algorithms for planar graphs (e.g., by Matsui [6]), so the only problematic cases seem to be low density graphs with no special structure which could be taken advantage of.

This article narrows the gap by showing two MST algorithms which run in linear time for any proper class of graphs closed on graph minors which includes planar graphs and graphs of bounded genus. We base our time bounds on density of minor closed classes (see e.g. Nešetřil and De Mendez [8]). Our algorithms don't require any specific knowledge of class structure except for class density needed by the second algorithm. This is an obvious improvement over the previous results for planar graphs which require construction of a planar embedding.

2. Minor closed classes

We'll need a definition of a graph minor and several statements about density of minor closed classes of graphs.

Definition. *Graph H is a minor of graph G if it can be obtained from G by a sequence of edge deletions and contractions.*

Definition. *Let $\mathcal{C}$ be a class of graphs. We'll define its edge density $\varrho(\mathcal{C})$ to be an infimum of all ϱ such that $|E(G)| \leq \varrho \cdot |V(G)|$ holds for any $G \in \mathcal{C}$.*

Theorem 1 *(This seems to be well known, see e.g. Theorem 6.1 in [8]).* *A minor closed class $\mathcal{C}$ has finite edge density iff $\mathcal{C}$ is a proper class, i.e. different from the class of all graphs and the empty class.*

[1] $\log^{(k)}$ denotes binary logarithm iterated k times

Lemma 1 *(Density Lemma)*. *Let $\mathcal{C}$ be a graph class with density ϱ and $G \in \mathcal{C}$ a graph with n vertices. Then at least $n/2$ vertices of G have degree at most 4ϱ.*

Proof: Assume the contrary: let there be at least $n/2$ vertices with degree greater than 4ϱ. Then $\sum_v \deg(v) > n/2 \cdot 4\varrho = 2\varrho n$ which is in contradiction with the number of edges being $\leq \varrho n$. (The proof can also be viewed probabilistically: let X be degree of a vertex of G chosen uniformly at random. Then $\mathbf{E}X \leq 2\varrho$, hence by the Markov's inequality $\Pr[X > 4\varrho] < 1/2$, so for at least $n/2$ vertices $\deg(v) \leq 4\varrho$.)

For planar graphs, the bound can be easily tightened:

Lemma 2 *(Density Lemma for Planar Graphs)*. *Let G be a planar graph with n vertices. Then at least $n/2$ vertices of v have degree at most 8.*

Proof: It suffices to show that the lemma holds for triangulations (if there are any edges missing, the situation can only get better) with at least 3 vertices. Since G is planar, $\sum_v \deg(v) < 6n$. The numbers $d(v) := \deg(v) - 3$ are non-negative and $\sum_v d(v) < 3n$, hence by the same argument as in the previous proof, for at least $n/2$ vertices v it holds that $d(v) < 6$, hence $\deg(v) \leq 8$.

3. A meta-algorithm

Our two algorithms are variations on the original algorithm by Borůvka, but instead of growing a forest of MST subtrees by joining them by edges newly proven to be in the MST, we keep each subtree contracted to a single vertex. Both algorithms can be considered incarnations of the following “meta-algorithm”:

1. Start with the input graph.
2. Find some edges which are part of the MST of the current graph.
3. Contract the graph along these edges.
4. Clean up the graph (to be explained in a moment).
5. Repeat steps 2–4 until there are no edges left.

This procedure is obviously correct (due to it stopping after a finite number of contractions and the well-known fact that given any subset A of MST edges, the rest of the MST are just edges of a MST of the same graph with edges of A contracted). However, we need to decide on how will we choose the edges to be contracted and how to represent the graph in order to perform searching for these edges and all the contractions efficiently.

We'd like to make use of low density of proper minor closed classes of graphs, but unfortunately it isn't as simple as it looks, because edge contractions produce loops and parallel edges, leaving us with a multigraph which of course can have a superlinear number of edges. Loops are the easy part: they can be easily detected and removed immediately after the contraction. From a set of parallel edges, we can delete all but the lightest one, but the key problem is to avoid spending a lot of time on detecting them.

To accomplish this, we introduce a *cleanup phase* which is called occassionally during the course of the algorithm and whose purpose is to prune the graph (make it again a simple graph). Additionally, if we are using any kind of arrays to represent the graph (which is not needed, everything can be done with linked lists on the pointer machine, but arrays are often easier to handle if they're available), we also should renumber the vertices to keep the arrays bounded by the current number of vertices instead of the original one.

The cleanup phase starts by bucket-sorting all graph edges lexicographically (which brings parallel edges together), walking the edge list sequentially and deleting the unnecessary parallel edges. Then it recalculates vertex degrees, removes zero degree vertices from the vertex list and if we're using arrays, then also creating new vertex numbers, compactifying the vertex array and renumbering all vertex number occurences in the edge list. The cleanup takes time $O(m+n)$ where m and n are the number of edges and vertices *before* the cleanup took place, so we need to use this fine spice very carefully.

Without loss of generality, we'll assume that the graph is connected and that all the edge weights are distinct.

4. Algorithm 1

We'll follow the meta-algorithm, using the fact that for each vertex, the lightest incident edge belongs to the MST (it follows from the Tarjan's blue rule [12] applied to a cut formed by edges incident to the vertex). No cycles can arise since the edge weights are distinct. Also, we'll process contractions due to all vertices of the current graph at once and clean up the graph afterwards. This gives:

Algorithm 1.

1. Start with the input graph.
2. Construct a temporary graph G' on the same set of vertices. For each vertex v, G' contains the lightest edge of G incident to v.
3. Contract the graph along the edges of G': find connected components of G' and renumber all vertices of G according to the component where their cousin in G' lies.
4. Clean up the graph (as defined before).
5. Repeat steps 2–4 until there are no edges left.

Lemma 3. *For any proper minor closed class of graphs $\mathcal{C}$, Algorithm 1 finds the MST of any graph in this class in time $O(\varrho(\mathcal{C}) \cdot n)$.*

Proof: Correctness follows from the meta-algorithm (we need the properties of the graph class only for the time bound). Each pass of the algorithm takes time $O(m + n)$ and it reduces n at least by a factor of two. All graphs generated by the algorithm (after cleanup) are minors of the input graph, so they belong to $\mathcal{C}$ as well, so according to Theorem 1, $m \leq \varrho n$ holds for all of them. Therefore the total time spent by the algorithm is $O(\varrho n + \varrho n/2 + \varrho n/4 + \ldots) = O(\varrho n)$.

5. Algorithm 2

Instead of batching the contractions, we can also perform them greedily on the lightest edges adjacent to low-degree vertices and delay the cleanup until we run out of such vertices. This gives our second algorithm (t is a parameter to be chosen later):

Algorithm 2.

1. Start with the input graph.
2. While there exists a vertex v with $\deg(v) < 6t$, select the lightest edge e incident to v and contract it (just remove all edges incident to v and add them back to the graph with v renumbered to the other end of e). Delete all loops and isolated vertices that arise. To avoid sequential searching, keep a queue of such low-degree vertices.
3. Clean up the graph (as defined above).
4. Repeat steps 2–3 until there are no edges left.

For minor closed classes of graphs, we'll set t to the density of the class or, if the exact density is unknown, to any upper bound of the density. We'll get:

Lemma 4. *For any proper minor closed class of graphs $\mathcal{C}$, Algorithm 2 with $t \geq \varrho(\mathcal{C})$ finds the MST of any graph in this class in time $O(tn)$.*

Proof: According to Theorem 1, we know that $\mathcal{C}$ (where the input graph and all the cleaned up intermediate graphs belong) has finite density $\varrho \leq t$ and Lemma 1 tells us that after a cleanup, at least a half of the vertices has $\deg(v) \leq 4\varrho \leq 4t$. Because of this, our algorithm always stops and since it's following the meta-algorithm, it's correct.

Step 2 contributes to the total running time by $O(tn)$ — each contraction takes $O(t)$ and it removes at least one vertex, so it suffices to show that the total time spent by the cleanups is linear.

Let n_i and m_i denote the number of vertices and edges of the graph G_i at the start of step i. The time spent by a single cleanup is $O(n_2 + m_3)$ — remember that we might need to clean up holes after vertices which have gone during step 2, but we can account the holes to these vertices and bound the rest of the cleanups by $O(n_3 + m_3) = O(m_3)$ as G_3 is connected.

We know that G_3 is a multigraph with $\deg(v) \geq 6t$ for all v, while G_4 is a simple graph with $\deg(v) \leq 4t$ for at least a half of its vertices. Therefore each cleanup decreases the sum of all degrees by at least tn_3 and $n_3 \geq m_3/\varrho \geq m_3/t$, so the number of edges drops by at least $tn_3/2 \geq m_3/2$, hence $m_4 \leq m_3/2 \leq m_2/2$ and all the cleanups together take time $O(m + m/2 + m/4 + \ldots) = O(m)$ which we were to prove.

Remark. *For planar graphs, we can take advantage of having proven Lemma 2 and improve the algorithm by a constant factor by setting the parameter t to $7/3$.*

6. Conclusions

We've presented algorithms for the minimum spanning tree problem which run in deterministic linear time for any class of graphs closed on graph minors. This further reduces the classes of graphs where the complexity of finding the MST is unknown, but it still leaves the general version of the problem open.

7. References

[1] O. Borůvka. O jistém problému minimálním (About a Certain Minimal Problem). *Práce mor. přírodověd. spol. v Brně*, III:37–58, 1926. Czech with German summary.

[2] B. Chazelle. A Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity. *J. ACM*, 47:1028–1047, 2000.

[3] M. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. In *Proceedings of FOCS'90*, pages 719–725, 1990.

[4] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7:43–57, 1985.

[5] D. R. Karger, P. N. Klein, and R. E. Tarjan. Linear expected-time algorithms for connectivity problems. *J. ACM*, 42:321–328, 1995.

[6] T. Matsui. The Minimum Spanning tree Problem on a Planar Graph. *Discrete Applied Mathematics*, 58:91–94, 1995.

[7] J. Nešetřil. Some remarks on the history of MST-problem. *Archivum Mathematicum*, 33:15–22, 1997.

[8] J. Nešetřil and P. O. de Mendez. Colorings and Homomorphism of Minor Closed Classes. To appear in *Pollack-Goodman Festschrift,* Springer Verlag, 2002.

[9] J. Nešetřil, E. Milková, and H. Nešetřilová. Otakar Borůvka on Minimum Spanning Tree Problem. *Discrete Mathematics*, 233(1–3):3–36, 2001.

[10] S. Pettie. Finding minimum spanning trees in $O(m\alpha(m,n))$ time. Tech Report TR99-23, Univ. of Texas at Austin, 1999.

[11] S. Pettie and V. Ramachandran. An Optimal Minimum Spanning Tree Algorithm. In *Proceedings of ICALP'2000*, pages 49–60. Springer Verlag, 2000.

[12] R. E. Tarjan. *Data structures and network algorithms*, volume 44 of *CMBS-NSF Regional Conf. Series in Appl. Math.* SIAM, 1983.