Can We Do Better...? Nope!

Pettie and Ramachandran's Optimal Minimum Spanning Tree Algorithm

Eli Berg, Vihan Lakshman, Sachin Padmanabhan

March 9, 2020

Stanford University

Table of contents

Introduction

Why Study Pettie-Ramachandran?

- Learn state-of-the-art algorithmic result
- See many great ideas in theoretical computer science come together
    - Fibonacci heaps, soft heaps, Method of Four Russians, decision tree complexity, other MST algorithms
- Question not can we do better but how well did we actually do?

History

Survey of MST Algorithms

- Boruvka's Algorithm (1926) - $O(m\log(n))$
- Prim's Algorithm (1930) - $O(n\log(n) + m)$
- Kruskal's Algorithm (1956) - $O(m\log(m))$
- Tarjan-Fredman Algorithm (1984) - $O(m\log^\star(m))$
- Karger-Klein-Tarjan Randomized Algorithm (1995) - $O(m)$ in expectation
- Chazelle's Algorithm (2000) - $O(m \cdot \alpha(m, n))$
- Pettie-Ramachandran Algorithm (2002) - $O(T^\star(m, n))$

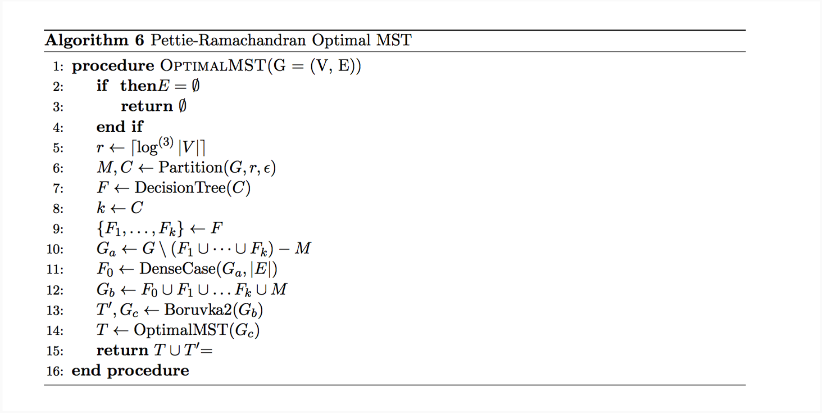
A Culmination of Decades of Work

The Pettie-Ramachandran Algorithm synthesizes many ideas from older MST Algorithms

- Contractible subgraphs (Chazelle)
- Running Boruvka's algorithm for a fixed number of steps (Karger-Klein-Tarjan, Chazelle)
- Limiting size of heap during Prim-type subroutines (Tarjan-Fredman)
- $O(m)$ time algorithm on dense graphs (Tarjan-Fredman)
- Soft heaps (Chazelle)

The Algorithm

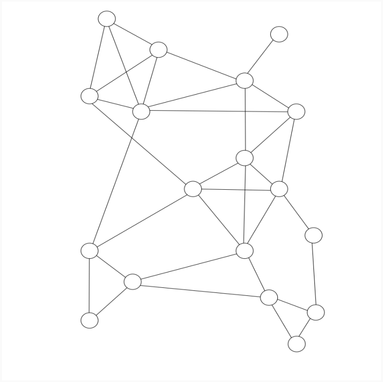
Pseudocode



Algorithm 6 Pettie-Ramachandran Optimal MST

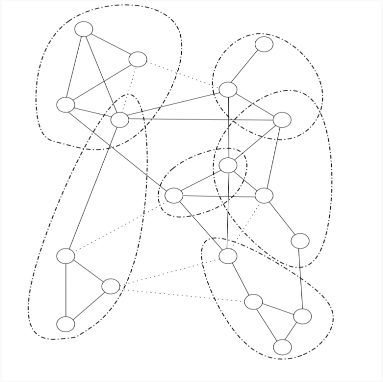
```
procedure OptimalMST(G = (V, E))
    if then$E = \emptyset$
        return $\emptyset$
    end if
    $r \leftarrow \lceil \log^{(3)} |V| \rceil$
    $M, C \leftarrow \text{Partition}(G, r, \epsilon)$
    $F \leftarrow \text{DecisionTree}(C)$
    $k \leftarrow C$
    $\{F_1, \ldots, F_k\} \leftarrow F$
    $G_a \leftarrow G \setminus (F_1 \cup \cdots \cup F_k) - M$
    $F_0 \leftarrow \text{DenseCase}(G_a, |E|)$
    $G_b \leftarrow F_0 \cup F_1 \cup \ldots F_k \cup M$
    $T', G_c \leftarrow \text{Boruvka2}(G_b)$
    $T \leftarrow \text{OptimalMST}(G_c)$
    return $T \cup T' =$
end procedure
```

Algorithm Visualization



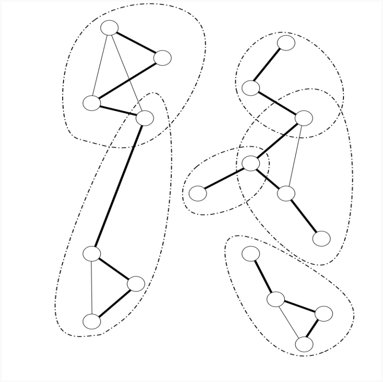
[image from Andersen, 2008]

Algorithm Visualization



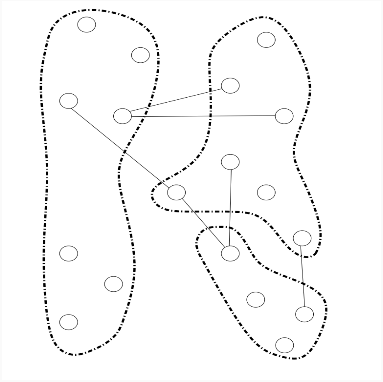
[Andersen, 2008]

Algorithm Visualization



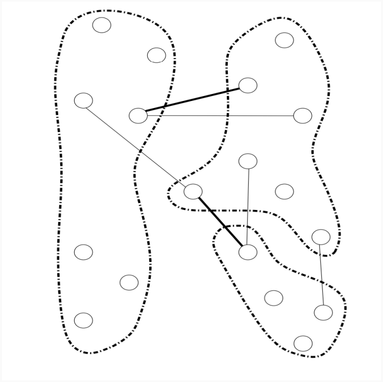
[Andersen, 2008]

Algorithm Visualization



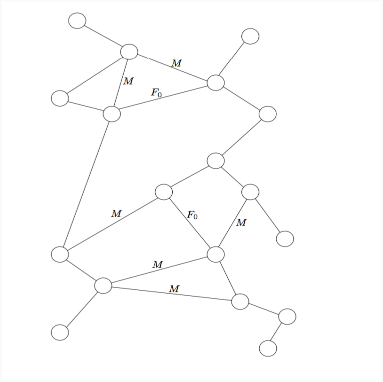
[Andersen, 2008]

Algorithm Visualization



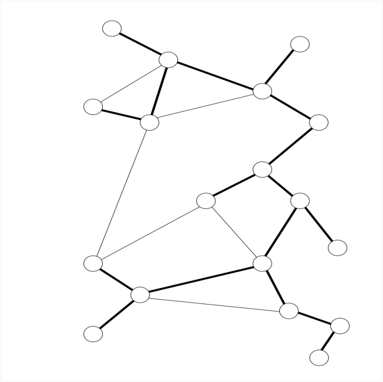
[Andersen, 2008]

Algorithm Visualization



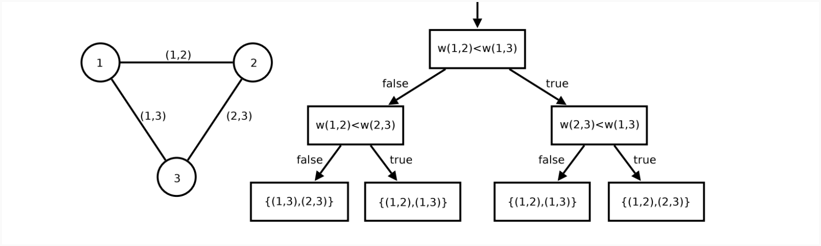
[Andersen, 2008]

Algorithm Visualization



[Andersen, 2008]

Decision Trees



[Andersen, 2008]

Decision Trees

Theorem

For partitions with maximum size $r = \lceil \log\log\log(n) \rceil$ vertices, the optimal decisions trees for Pettie-Ramachandran can be computed in $o(n)$ time.

Partition Procedure

Definition (DJP-contractible)

A subgraph is *DJP contractible* if it is the subgraph induced by running Prim's algorithm for a certain number of steps.

- Use DJP with soft heaps to quickly construct approximate MSTs of edge-disjoint subgraphs of size at most $\log^{(3)} n$
- These subgraphs are DJP contractible
- Must account for all relevant edges which may have been corrupted

Dense Case

Claim

After contracting partitions, we have at most $\frac{n}{log^{(3)}(n)}$ *vertices and can compute the MST of this new graph in* $O(m)$ *time using the Fredman-Tarjan algorithm.*

Proof of Correctness

A key lemma

Definition ($G \Uparrow M$)

Let $G \Uparrow M$ denote the graph formed by raising the weights of the edges in M by an arbitrary amount.

Lemma (Key Lemma)

If M is a subset of edges of G and C is a subgraph of G that is DJP-contractible with respect to $G \Uparrow M$, then

$$MSF(G) \subseteq MSF(C) \cup MSF(G \setminus C - M_C) \cup M_C$$

where M_C denotes the edges in M that have exactly one endpoint in C.

The proof

Proof sketch.

The algorithm proceeds in the Partition procedure by finding DJP-contractible sets $C_1, \dots, C_k$, with corrupted edges. Let M_{C_i} contain the corrupted edges with exactly one endpoint in C_i. As we progress, we contract each C_i and remove the corrupted edges. By the key lemma,

$$MSF(G) \subseteq MSF(C_1) \cup MSF(G \setminus C_1 - M_{C_1}) \cup M_{C_1}$$

Inductively, assume that

$$MSF(G) \subseteq \bigcup_{i=1}^{k-1} MSF(C_i) \cup MSF\left(G \setminus \bigcup_{i=1}^{k-1} C_i - \bigcup_{i=1}^{k-1} M_{C_i}\right) \cup \bigcup_{i=1}^{k-1} M_{C_i}$$

Applying the key lemma on the middle term,

$$MSF(G) \subseteq \bigcup_{i=1}^{k} MSF(C_i) \cup MSF\left(G \setminus \bigcup_{i=1}^{k} C_i - \bigcup_{i=1}^{k} M_{C_i}\right) \cup \bigcup_{i=1}^{k} M_{C_i}$$

The proof

Proof sketch (continued).

The first term is computed correctly by the precomputed decision trees, and the second term is computed correctly by the DenseCase algorithm. This shows that the edges that are considered by the Boruvka steps are a superset of the MST edges. Boruvka steps add only guaranteed MST edges, so all edges we add in the pass must be MST edges. Inductively assuming that the recursive call, which is on a smaller input returns the correct MST, we can piggyback off the correctness of Boruvka's algorithm (cut property) to conclude that the Optimal MST algorithm is correct. □

Runtime Analysis

Proving an Optimal Bound

- Time to construct optimal decision trees: $o(n)$
- Partition Runtime: $O(m)$ for fixed soft heap error tolerance parameter ϵ
- Decision Tree Evaluation: $T^\star(m, n)$
- Dense Case: $O(m)$
- Boruvka Steps: $O(m)$

We can set up a recurrence relation and solve it to obtain a runtime bound of $O(T^\star(m, n))$. This is as good as it gets (asymptotically)!

Runtime on practical graph instances

Theorem

On graphs where $n \leq 2^{2^{16}}$, the Optimal MST algorithm will run in linear time

Proof Sketch: If $n \leq 2^{2^{16}}$, then $r \leq 4$. Therefore, the partition graphs will be planar and we can compute the MST on planar graphs in $O(m)$ time.

Runtime Analysis on Random Graphs

Preliminaries

- Karp and Tarjan's algorithm runs in expected $O(m)$ time under the assumption that edge weights are chosen randomly
- Karger's algorithm runs in expected $O(m)$ time but runs in $\Omega(n \log n)$ time with adversarially selected edge weights
- We will show that our graph runs in $O(m)$ time with high probability, *regardless of edge weights*

Definition (Erdös-Renyi random graph $G_{n,m}$)

The Erdös-Renyi random graph model, $G_{n,m}$, samples each of the $\binom{\binom{n}{2}}{2}$ graphs on n vertices and m edges with equal probability

The result

Theorem

The algorithm finds the MST in $O(m)$ time with probability

$$1 - \exp(-\Omega(m/\alpha^2))$$

for a graph drawn randomly from $G_{n,m}$. where $\alpha = \alpha(m,n)$ is the inverse Ackermann function.

Mile-high proof sketch:

1. Exhibit that if the graph G satisfies a certain condition, then the algorithm will run in $O(m)$ time
2. Show that this condition is satisfied on random graphs with high probability

The condition

Definition (Excess)

The *excess* of a subgraph H is $|E(H)| - |F(H)|$, where $F(H)$ is any spanning forest of H.

Definition ($f(G)$)

Running the Partition procedure results in components of size at most $k \leq \log^{(3)} n$. Define $f(G)$ to be the maximum excess of the subgraph induced by the intracomponent edges over any possible execution of the Partition procedure.

The condition

- Intuitively, $f(G)$ is the number of “bad” edges inside the components.
- If $f(G)$ is small, then the resulting recursive call will be smaller, and thus the runtime will be faster

Lemma

If $f(G) \leq m/\alpha$, then the algorithm runs in $O(m)$ time.

Proof sketch.

If the condition holds, running $\log \alpha$ Boruvka steps (which takes linear time since the number of edges decreases by a constant factor after each two Boruvka steps) reduces the total number of intracomponent edges to at most $2m/\alpha$. The resulting graph has $O(m/\alpha)$ edges, so we are guaranteed to be able to find the MST in $O(\alpha \cdot m/\alpha) = O(m)$ time. □

Tools from probability

Definition (Martingale)

A *martingale* is a sequence of random variables $Y_0, \ldots, Y_m$ such that $E[Y_i \mid Y_{i-1}] = Y_{i-1}$ for $0 < i \leq m$.

Theorem

The edge-addition martingale $f_E(G_i) = E[f(G_m)|G_i]$ for $0 \leq i \leq m$ is a martingale, where G_0 is the empty graph on n nodes and each G_i is formed from G_{i-1} by adding a random edge X_i. Furthermore, particularly for f, $|f_E(G_i) - f_E(G_{i-1})| \leq 1$ and $f_E(G_0) = o(m/\alpha)$.

The martingale reveals progressively more about $f(G_m) = f(G)$. We begin with no information about G, and the value of the martingale is just $E[f(G)]$. As we get more information in the form of X_i's, we learn more until at the end, we have complete information about G and thus $f(G)$.

Azuma's Inequality

- In expectation, the value of the martingale $f_E(G_i)$ doesn't change over consecutive time steps
- It doesn't fluctuate too much between consecutive time steps

Theorem (Azuma's Inequality)

Let $Y_0, \ldots, Y_m$ be a martingale such that $|Y_i - Y_{i-1}| \leq 1$ for $0 < i \leq m$, Then, for any $\lambda > 0$, $\Pr[|Y_m - Y_0| > \lambda\sqrt{m}] < \exp(-\lambda^2/2)$.

Finishing the analysis

We now give the proof of the following lemma, which completes the proof of the original theorem.

Lemma

If G is a random graph drawn from $G_{n,m}$, then $\Pr[f(G) \leq m/\alpha] \geq 1 - \exp(-\Omega(m/\alpha^2))$.

Proof.

By Azuma's inequality, $\Pr[|f_E(G_m) - f_E(G_0)| > \lambda\sqrt{m}] < \exp(-\lambda^2/2)$. Now, we set $\lambda = \frac{\sqrt{m}}{\alpha} - \frac{f_E(G_0)}{\sqrt{m}}$ and note that by the previous lemma, $f_E(G_0) = o(m/\alpha)$, so $\lambda = \Omega(\sqrt{m}a)$. Thus,

$$\Pr[f(G) - f_E(G_0) > m/\alpha - f_E(G_0)] = \Pr[f(G) > m/\alpha] < \exp(-\Omega(m/\alpha^2))$$

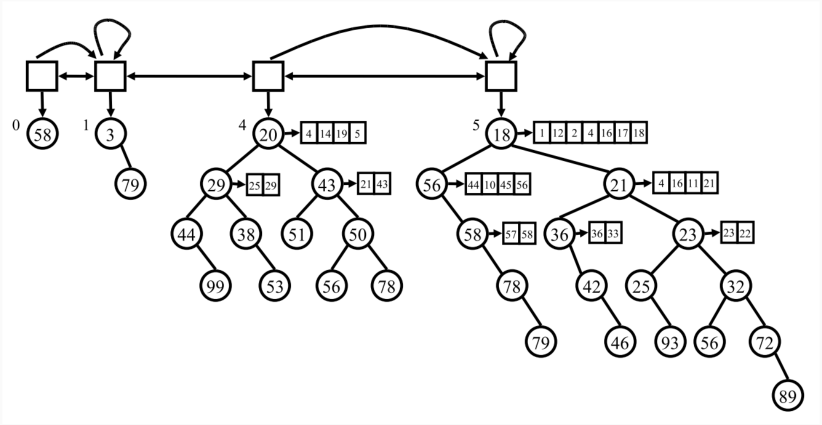
from which the result follows immediately. □

Implementation Details

Structures Implemented

- Undirected Graph
- Disjoint Set
- Disjoint Set Forest
- Fibonacci Heap
- Modified Soft Heap

Modifying the Soft Heap



[Kaplan & Zwick, 2009]