Minimum Spanning Forest

DD2440 - Advanced Algorithms

Maximilian Auer
maue@kth.se

Nils Balzar Ekenbäck
nilsek@kth.se

Valdemar Gezelius
vgez@kth.se

Veronica Hage
vhage@kth.se

December 2018

Kattis submission ID:
fixed: **3499402**
varying: **3506791 / 3534947**

1 Introduction

In the provided paper, ”Sublinear-time Algorithms”[1], an algorithm is presented on how to approximate a Minimum Spanning Tree (MST). This algorithm utilizes the edge weights in order to approximate the number of connected components, by iterating from 1 to max weight $W-1$. We want to modify this to suit an approximation for a Minimum Spanning Forest (MSF). This is done by subtracting the number of trees in the forest times the maximum edge weights from the original MST formula. Thus eliminating the edges that should have connected the forest into an MST.

1.1 Approximating a Minimum Spanning Tree

Approximating an MST is based on approximating the connected components at different weights. It can be summarized as follows:

$$MST = n - W + \sum_{i=1}^{W-1} \hat{c}^{(i)} \qquad (1)$$

Where $\hat{c}^{(i)}$ represents the approximated number of components $c^{(i)}$ with up to edge weight i. The summarized components will be connected by edges of W, hence it is subtracted from the MST weight calculation.[1]

1.2 Modifying for Minimum Spanning Forest

In order to tweak the algorithm to work with an MSF we use the fact that the connected components of weight 1 to $W-1$ are bound together into a tree by edges of weight W. A forest consists of multiple trees and we can therefore change our approximation formula to

$$MSF = n - W|T| + \sum_{i=1}^{W-1} \hat{c}^{(i)} \qquad (2)$$

Where $|T|$ is the number of trees in the forest. To calculate this number we approximate the number of components $c^{(W)}$ as $\hat{c}^{(W)}$. This gives us that $|T| = c^{(W)} \approx \hat{c}^{(W)}$.

1.3 Fixed and Varying Samples Sizes

For making as close an approximation as possible the right number of samples S is needed. For the fixed version of the problem, S is set to a fixed value. For the varying version of the problem, given that the approximation is given by $1+\varepsilon$ we want to make the sample size dependant on ε in order to have a varying sample size.

By utilizing the running time of $\mathcal{O}((\frac{W}{\varepsilon})^2 \log n)$ we can make S dependant on ε with $S = (\frac{W}{\varepsilon})^2 \log n$ to ensure us we have a big enough sample size to make an approximation that satisfies

$$(1-\varepsilon) \cdot MSF \leq \widetilde{MSF} \leq (1+\varepsilon) \cdot MSF \qquad (3)$$

2 The Algorithm

2.1 Implementation

The program was written in Python. For every weight $w = \{1, ..., W-1\}$ a set of randomly sampled S nodes are iterated (see section 1.3 for S). BFS is run using a queue. A random node is added to the queue, followed by it's neighbours (of the same weight or less). Each node in the queue is evaluated as the connected component is explored. A set of visited nodes was used to avoid loops in the BFS. In the varying parameters version the visited set was also used to store already discovered nodes' values to save computation time.
When the whole component or the limit of X nodes had been explored the BFS ended. If the whole component had been explored a component counter was added to. Regarding the choice of X with probability $\Pr[X \geq k] = \frac{1}{k}$; $X \geq k$ is equivalent to $\frac{1}{X} \leq \frac{1}{k}$. When Y is chosen uniformly from $[0,1]$, then $\Pr[Y \leq p] = p$. [1] Given this information we set our $X = \frac{1}{rand(0,1)}$.

[1]Information given on Canvas page for project MSF

When the connected components for weights $w = \{1, ..., W-1\}$ had been approximated and summed up the approximation for the last weight W was run. This would be, as stated earlier in 1.2, an approximation of the number of trees in the forest.

Lastly the formula for our approximated MSF weight was calculated using the formula described in 1.2.

2.2 Work progress

Our strategy was to first implement the algorithm to correctly approximate the weight of an MST according to the provided paper [1]. We then modified the algorithm to be able to approximate the weight of an MSF. The algorithm went through a few iterations before ending up in the implementation described in 2.1.

The first implementation of the algorithm that registered correct solutions on *Kattis* differed from 2.1 chiefly in that we (a) calculated connected components with weight equaling w instead of calculating connected components with weight less than or equal to w, for $w = \{1, ..., W\}$. We implemented (a) and the next tweak on the algorithm was to (b) reset the list *visited* containing visited nodes for each node in the set s instead of, as in earlier iterations, for each value of w; $w = \{1, ..., W\}$. With the tweaks (a) and (b) the code passed all test cases for the *Kattis* problem with fixed parameter.

To be able to tackle the varying parameter problem we had to adjust the sample size. From the assignment we know that the sample size should be dependant of ε, with a hint to the complexity's ϵ^2 term.

As a start for the $1 + \epsilon$ approximation we began with taking a sample size s as $s = (\frac{W}{\varepsilon})^2 \cdot \log n$. This sample size was not optimal as it also depended on n and only gave a satisfying approximation in 9 of the test cases, before exceeding the time limit on the 10th.

To improve this further we defined s as $\frac{W^2}{\varepsilon^2}$ which eliminated the use of n to define s and gave a vastly increased performance and approximation resulting in 20 approved cases.

In order to further speed up the algorithms input from *Kattis* we saved previous answers outputted from *Kattis* and reused them if the same node was sampled again. We also limited X as to not make it too big, in order to avoid exceeding the time limit. This however turned out to be a redundant feature and could be removed.

This did not however pass all the test cases. The algorithms calculations are an approximation for every iteration of 1 to W, thus we will always have a small error for every estimation of $\hat{c}^{(i)}$. To improve the approximation without the algorithm taking too long we doubled the sample size for the last iteration of the approximation of the number of trees in the forest. This way we were able to pass all the test cases

2.3 Proof of Correctness

The proof of correctness for the algorithm used to approximate the number of connected components with at most weight i, can be found in the provided paper by Czumaj and Sohler[1].

To show the proof of the MSF-formula, we first need to elaborate on this said MST formula.

The provided generalized formula (1): $MST = n - W + \sum_{i=1}^{W-1} c^{(i)}$ is a simplification of the following formula:

$$MST = 1|E_1| + 2|E_2| + ... + W|E_W|$$

Where $|E_i|$ is the minimum number of edges with at most weight i that connect the maximum amount of components.

$$|E_1| = n - (c^{(1)} - 1), |E_2| = (c^{(1)} - 1) - (c^{(2)} - 1), ..., |E_W| = (c^{(W-1)} - 1)$$

$$|E_{(2 \leq i \leq W-1)}| = (c^{(i-1)} - 1) - (c^{(i)} - 1)$$

Because all graphs which has a minimum spanning tree are connected, the formula will not take $(c^{(W)} - 1)$ into consideration in the last step, as it will always be equal to zero in the case of an MST. This is because the number of connected components in an MST always will be equal to one.

To show an example of the simplification, for the maximum edge weight $W = 4$.

$$MST = 1 \cdot (n - 1 - (c^{(1)} - 1)) + ... + 4 \cdot (c^{(3)} - 1)$$

$$= n - 1 - c^{(1)} + 1 + 2c^{(1)} - 2 - 2c^{(2)} + 2 + 3c^{(2)} - 3 - c^{(3)} + 3 + 4c^{(3)} - 4$$

$$= n - 4 + c^{(1)} + c^{(2)} + c^{(3)} = n - W + \sum_{i=1}^{W-1} \hat{c}^{(i)}$$

In the last step of the unsimplified MST-formula, it will calculate the number of edges with weight W to be $|E_W| = (c^{(W-1)} - 1)$. Since the graph of an MSF could be unconnected we instead want to calculate $|E_W| = (c^{(W-1)} - 1) - (c^(W) - 1)$, where $(c^W) - 1) = |T| - 1$, and represents the number of edges between the components we are unable to connect no matter which edges we use, i.e. the number of edges needed to connect the trees. To convert the MST-formula to be able to handle MSF-graphs we simply subtract the sum of the incorrect edges who was assumed to have weight W, resulting in:

$$MSF = MST - W(|T|) = MST - W(c^{(W)} - 1)$$

$$= n - W + \sum_{i=1}^{W-1} c^{(i)} - W(c^{(W)} - 1) = n - Wc^{(W)} + \sum_{i=1}^{W-1} c^{(i)}$$

$$= n - W|T| + \sum_{i=1}^{W-1} c^{(i)} \qquad (4)$$

Our approximation is thus given by the following:

$$\widetilde{MSF} = n - W|T| + \sum_{i=1}^{W-1} \hat{c}^{(i)} \qquad (5)$$

It is easy to see that the MSF algorithm would work as well with an any MST graphs due to the fact that $|T| - 1 = c^W - 1 = 0$ when we have a graph that is connected, i.e. a graph that has a minimum spanning tree, resulting in the same formula as (1).

As the only difference between formula (1) and (4) is that we subtract the incorrect edges, the algorithm for approximating each $\hat{c}^{(i)}$ is the same independent of which formula that is used. This means that there are no difference in the proof of correctness for the approximation algorithm and the bound for the approximation error compared to the algorithm presented by Czumaj and Sohleri[1].

2.4 Time Complexity

Run time of the original algorithm, as given by the assignment, is $(\frac{W}{\varepsilon})^2 \cdot \log n$. This complexity will however change with our algorithm as we have done a slight modification. Since we approximate the number of trees in the graph by calculating $c^{(\hat{W})}$, in other words running the algorithm 1 to W instead of 1 to $W - 1$, we increase the running time to $W \cdot (\frac{W}{\varepsilon})^2 \cdot \log n = \frac{W^3}{\varepsilon^2} \cdot \log n$.

References

[1] Artur Czumaj and Christian Sohler. Property testing. chapter Sublinear-time Algorithms, pages 41–64. Springer-Verlag, Berlin, Heidelberg, 2010.