MSc Thesis

Stochastic Minimum Spanning Tree Problem with a Single Sample

Gavin Speek

Graduation committee:	Role:
prof.dr. M.J. Uetz	Head graduation committee & Daily supervisor
dr.ir. R.P. Hoeksma	Daily supervisor
dr. J.M. Meylahn	External committee member

May 24, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering, Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Acknowledgement

This thesis is written as a finalisation of my Master's in Applied Mathematics. I very much enjoyed the process which has lead to this point, and I want to thank some people which made it possible and supported me along the way.

Firstly, I would like to express my deepest appreciation for my daily supervisors, Marc and Ruben. I very much enjoyed our meetings, during which you reminded me to think critically, and at the same time remain optimistic. Your feedback has been essential, and you have really shown me the value of writing a thesis. Next to this, I want to thank Janusz for taking place in my graduation committee and taking the time to read my report.

Finally, I want to thank my friends and family for supporting me along the way. Your interest and kindness has been invaluable. In particular, I want to thank Anouk for her mathematical insights and for being a wonderful person.

I hope you enjoy reading this report.

Gavin Speek
May 24, 2024

Abstract

We consider the minimum spanning tree problem in a setting where the edge weights are stochastic instead of deterministic, and the decision maker is given only one sample of each edge's weight distribution. Under these conditions, perhaps the only reasonable strategy is to choose the spanning tree which is minimal over the sampled weights. In general, this algorithm can, in expectation, perform arbitrarily worse than the adversary that computes the minimum spanning tree over the expected weights. For exponential weights we show that the performance of this algorithm is bounded from above by the size of the largest bond in the graph, and we show that this bound is tight. Our proof is based on an inductive argument via edge contractions, that allows us to reduce the spanning tree problem to a simpler item selection problem. We also discuss a generalization to arbitrary matroids.

Contents

Abstract **1**

Contents **2**

1 Introduction **4**

1.1 Background and motivation - Minimum Spanning Trees 4

1.2 Overview of report 5

1.3 Preliminaries 6

1.3.1 Graphs 6

1.3.2 Results related to graphs and bonds 7

1.3.3 Minimum Spanning Tree problem 8

1.3.4 Single Sample Stochastic Minimum Spanning Tree Problem 9

1.3.5 Single Sample Stochastic Minimum Item Selection Problem 10

2 Well-behaved input distributions **12**

2.1 Input distributions which are not well-behaved 12

2.2 Exponential distribution 14

2.3 SSMIS problem with exponential weights 14

3 Single Sample MST problem with exponential weights **17**

3.1 Main result and smaller results 17

3.1.1 Notation 17

3.1.2 Preliminary results 18

3.2 Lower bound 21

3.3 Upper bound 22

4 Single Sample Minimal Matroid Basis problem with exponential weights **26**

4.1 Matroids 26

4.2 SSMMB problem 27

4.3 Relative performance of SAM 28

5 Maximizing instead of minimizing **33**

5.1 Single Sample Maximum Item Selection problem 33

Bibliography **41**

Appendices **A-1**

A.1 Proof of exponential minimum A-1

Chapter 1

Introduction

In the first section of this chapter, we outline some background and motivation for why the topic of this thesis is relevant. In the subsequent sections we give an overview of the structure of this thesis, and we provide an overview of notation and concepts from graph theory that will be used in later chapters.

1.1 Background and motivation - Minimum Spanning Trees

The minimum spanning tree (MST) problem is a classic problem in optimization and graph theory, which also has many practical applications in computer networking and transportation. (Kruskal, 1956) and (Prim, 1957) are often referred to as the first efficient solutions to the MST problem. Yet, the first known solution in literature is by (Borůvka, 1926). Borůvka wanted to efficiently solve the problem of connecting the electricity network of Moravia, a city in the Czech Republic. He accomplished this by modelling it as a MST problem, and he created an algorithm to solve it efficiently.

While the classic MST problem can be solved efficiently, one might wonder what changes when uncertainty is introduced. Uncertainty is a challenge which needs to be dealt with in many situations. Measurements may be inaccurate, equipment can fail, prices can be unpredictable, or perhaps no historical data is available. Besides the interesting research which can evolve from asking such questions, efficiently solving the MST problem under uncertainty can have many real-world applications.

One paper on MSTs with uncertainty is by (Hoffmann et al., 2008). In this paper, the authors discuss an adaptation of the classic MST problem, where the edge weights lie in some uncertainty area, and the user can request updates on the actual edge weights. The goal is to find the a MST with as few updates possible. One of their main results is a deterministic algorithm which requires at most two times as many updates as the optimal update schedule, and they prove that this algorithm is the best possible under certain conditions. The authors also discuss another version of the MST problem in which the vertices are points in Euclidean space and the edge weights are the distances between the vertices. They give an algorithm which requires at most four times as many updates as the optimal update schedule, and they prove optimality among deterministic algorithms.

Another variation of MSTs with uncertainty is by (Kamousi and Suri, 2011). In this paper, the authors extend the MST problem by allowing edges to "fail", which means there is a probability that an edge cannot be chosen for the MST. In this paper, the authors prove, among other results, that finding a MST is $\#P$-Hard in this context.

The specific addition of uncertainty to the MST problem in this thesis is inspired by the paper "Sequencing Stochastic Jobs with a Single Sample", by (te Rietmole and Uetz, 2024). This paper discusses a version of the single machine scheduling problem, in which the processing times are stochastic from unknown distributions, and the scheduler is given only one sample of each job's distribution. For this restricted scheduling problem, the authors discuss the performance of perhaps the only reasonable strategy: scheduling the jobs in the order which yields the smallest makespan for the given samples. For this strategy, the authors discuss four classes of probability distributions under which this strategy performs at least as good as or better than the uniform random adversary.

This paper inspired this thesis to investigate the MST problem with the same restrictions: stochastic edge weights, and the decision maker is only given one sample of the weights. We will discuss the performance of greedy algorithm Sam, which uses the sampled weights as a proxy for the expected weights. With these restrictions, the scheduling problem and the MST problem share some similarities. Therefore, we will investigate whether the distributions for which the performance can be bounded on the scheduling problem, also yield performance bounds for Sam on our version of the MST problem. However, it is no guarantee that these distributions will also be well-behaved for the MST problem, as the problems are fundamentally different. For instance, for the scheduling problem every job needs to be considered for the schedule. For the MST problem on the other hand, a specific subset of the edges needs to be chosen to form a spanning tree. Therefore, it is no guarantee that distribution for which performance bounds are guaranteed for the scheduling problem, will also yield performance bounds for our spanning tree problem.

1.2 Overview of report

We first discuss the Single Sample Minimum Item Selection (SSMIS) problem, which can be seen as a special case of the stochastic MST problem we are interested in. For the SSMIS problem, we discuss some classes of probability distributions on which Sam is well-behaved for the scheduling problem discussed by (te Rietmole and Uetz, 2024). We show that for three out of four of these classes we cannot necessarily give a bound on the performance of Sam relative to the optimal solution. For instances with exponentially distributed weights, we can show that the performance of Sam is bounded by the number of items in the instance, and that this bound is tight. Afterwards, we extend our research to the stochastic MST problem. We will prove that on instances with exponentially distributed weights, the performance of Sam is bounded by the size of the largest bond in the graph, and we show that this bound is tight for every problem instance.

While the scheduling problem is in spirit similar to the stochastic MST problem we discuss, the results and proofs are quite different. For the scheduling problem the authors showed that Sam performs, in expectation, better than random if every pair of jobs is scheduled in the correct order with a probability of at least one half. For this proof, the authors introduce a relative optimality gap, which quantifies the performance of Sam relative to the difference between the expected value of the best and worst possible solutions. For the stochastic MST problem as discussed in this thesis, we will compare the expected performance of Sam directly to the expected value of the optimal solution. The derivation of the performance bound requires a rather involved proof by induction based on edge contractions, which makes use of properties of the exponential distribution and graph structures.

A natural generalization is to extend this result to arbitrary matroids. In Chapter 4 we extend the stochastic MST problem to general matroids, and prove that the performance of Sam on instances with exponential weights is bounded by the size of the largest co-circuit in the matroid, and that this bound is also tight. So far, the objective of the stochastic single sample problems has been to minimize the expected weight. In Chapter 5 we touch upon the main differences between minimizing and maximizing as objective. Further, we conjecture under which conditions the worst-case performance of Sam is achieved on the item selection problem with exponential weights, where the objective is to maximize the expected weight.

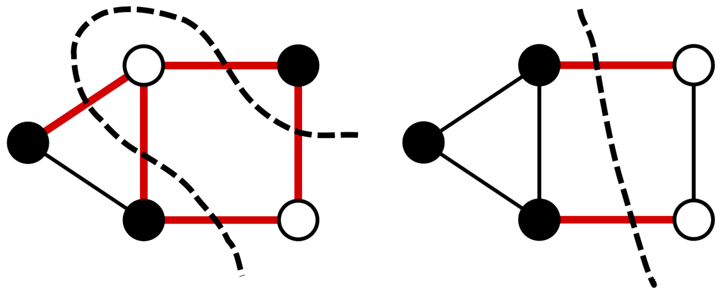
1.3 Preliminaries

1.3.1 Graphs

A graph $G = (V, E)$ consists of two sets. One set is the set of vertices, denoted by V. The other set is the edge set E, which consists of pairs of vertices. In the context of this thesis, we exclusively focus on undirected edges, meaning that the pairs of vertices in the edge set are unordered. Furthermore, we permit graphs to be *multigraphs*. Multigraphs allow the existence of multiple edges between any pair of vertices. In this case, edges are not uniquely defined by their end points. Therefore, we will not denote edges by their end points, but rather give each edge an own index. That is, we let $E = (e_1, e_2, \ldots, e_m)$, and thus $|E| = m$. Usually multigraphs allow the existence of loops, which are edges which connect a vertex to itself. Since we will be discussing the minimum spanning tree problem, and loops are never included in a spanning tree, we ignore the existence of loops.

A *cycle* is a non-empty sequence of unique edges, which starts and ends at the same vertex. In contrast to cycles, a forest is a (sub)graph which does not contain any cycle. If a forest contains all the vertices of a graph, it is a *spanning* forest. A forest which is connected is also called a *tree*. Finding a spanning forest in a disconnected graph is equivalent to finding a spanning tree in every connected component of that graph. Therefore, we will only consider connected graphs in this thesis.

A *cut* is a partition of the vertices of a graph into two disjoint subsets. A cut determines a *cut-set*, which is the set of edges which has one end point in each subset of the partition. The removal of the edges in a cut-set induces a division of the graph into two or more connected components. A specific type of cut-set is a *bond*, which is a cut-set which does not contain any other cut-set. In Figure **??**, the difference between a bond and a cut-set is illustrated.



(a) A cut-set which is not a bond

(b) A cut-set which is a bond

Figure 1.1: The right cut-set is a strict subset of the left cut-set. Therefore, the left cut-set is not a bond. The right cut-set is a bond, since it does not contain any other cut-set. https://commons.wikimedia.org/w/index.php?curid=6002348

Note that, if the edges which are in the bond would be removed from the graph, the number of connected components will increase by 1. Since bonds are not commonly used in graph theory, we will derive some results related to bonds later in this section.

By $B(G)$ we will denote a largest bond in graph G, that is, a bond for which no other bond has a larger cardinality. Note that we say ‘a’ largest bond, as there need not necessarily exist a unique largest bond. It is interesting to note that (Duarte et al., 2019) show that finding a largest bond in a graph is NP-hard.

One operation which will be useful later, is the edge contraction. Suppose we have some edge $e = (u, v)$. If we contract e, vertices u and v are merged into one vertex, and edge e is removed. In Figure 1.2, an example is shown.

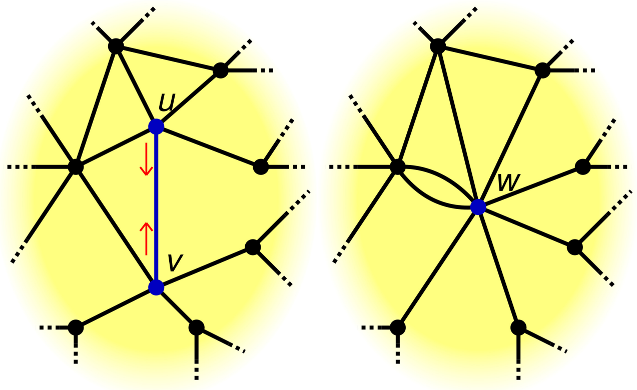


Figure 1.2: Example of an edge contraction. Vertices u and v are merged into one vertex w. By Trevorgunn - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=74615859

Notice how in this example the contraction causes two edges to become parallel to each other. If the contracted edge (u, v) has parallel edges, these parallel edges become loops incident to w in the resulting graph. Since we ignore loops in this thesis, we consider these parallel edges to be deleted as well. By G/e we denote the resulting graph which is obtained by contracting edge e from graph G. Note how this differs from $G \backslash e$, which denotes the graph obtained from G by deleting edge e.

Next, we introduce weighted graphs. Typically, edges have a deterministic weight. However, in this thesis we consider graphs with stochastic weights, which means the edge weights are governed by certain distributions. This gives rise to a stochastic weight function $w : E \to \mathbb{R}$. By $\tilde{w}(e)$ we denote one sample of weight function w for edge e.

1.3.2 Results related to graphs and bonds

In this section, we will lay out and prove some results related to graphs which will be used for the proof of the main result of Chapter 3. The first lemma relates the size of the largest bond in a graph G to the size of the largest bond in graph G/e. Recall that we defined $B(G)$ to be a largest bond in graph G.

Lemma 1.3.1. *If G is a connected graph, then $\forall\, e \in E$, we have $|B(G/e)| \leq |B(G)|$.*

Proof. We will prove this statement using a proof by contradiction.

Suppose we have some bond B_2 in G/e which has a larger cardinality than the largest bond in G, which we will denote by B_1. Then, B_2 cannot a bond in G, as it would be the largest bond in G, which would be a contradiction. Therefore, B_2 is either not a cut, or not a minimal cut in G. First, suppose B_2 is not a cut in G. Since the only difference between G and G/e is the contraction of edge e (and potentially some parallel edges, which we can omit w.l.o.g.), it is implied that $B_2 \cup e$ is a bond in G. However, since we assumed $|B_1| < |B_2|$, this bond would have a larger cardinality than B_1, which is would be a contradiction.

The case that B_2 is not a cut in G yields a contradiction. Now consider the case where B_2 is not a minimal cut in G, i.e., it contains another cut. This contained cut is still a cut in G/e, thus B_2 cannot be a bond in G/e, which is also a contradiction.

Therefore, we have shown using contradictions that a larger bond cannot be formed by means of contracting. □

Next, we introduce two more lemmas. The first lemma relates the size of the largest bond in a graph G, to the size of the largest bond in graph G/e. The second lemma gives a bound on the number of edges in a graph. This bound consists of the size of the largest bond in that graph and the number of edges in a spanning tree.

Lemma 1.3.2. *Given a connected graph $G = (V, E)$ with $|V| > 2$, there always exists some edge $e \in E$ which we can contract such that the size of the largest bond $B(G)$ in graph G does not change, i.e. $B(G) = B(G/e)$.*

Proof. By the definition of a bond, the removal of the edges of the bond increases the number of connected components in the graph by exactly one. Since we assumed $|V| > 2$, one of the two connected components contains at least 2 vertices. Since this component is connected, it must therefore contain an edge, which is not part of the bond. Thus, there must exist an edge which is not part of our bond. □

Lemma 1.3.3. *Let G be a multigraph which does not allow self-loops. Furthermore, let T be a spanning tree in G, and let $B(G)$ be the largest bond in G. Then the number of edges in G can be bounded by,*

$$|E(G)| \leq |B(G)||T| \,. \tag{1.1}$$

Proof. The proof of this lemma follows from the correctness of Prim's algorithm. Prim's algorithm is initialized by partitioning G into two partitions $P_1 = \{v\}$ and $P_2 = \{V \backslash v\}$. Furthermore, an empty tree T is initialized. Note that the edges which have one end-point in P_1 and one end-point in P_2 form a bond. After initialization, the following procedure is followed. The edge $e = (v, u)$ with the lowest weight in the bond is added to T, and vertex u is removed from P_2 and added to P_1. After this update, P_1 and P_2 induce a new bond, and this process is repeated until P_2 is empty. By iteratively adding edges in this manner, a minimum weight spanning tree is obtained, according to (Korte and Vygen, 2007, p.130-131).

Prim's algorithm termites after $|T|$ iterations, and in every iteration at most $|B(G)|$ edges can be chosen. Since Prim's algorithm is always guaranteed to find a minimum weight spanning tree, every edge in G is considered at least once. Therefore, G cannot contain more than $|B(G)||T|$ edges. □

1.3.3 Minimum Spanning Tree problem

In this section, we formally introduce the classic MST problem. In the next section, we introduce the MST problem with stochastic weights. In the classic Minimum Spanning Tree problem, the objective is as follows: given a graph G with deterministic edge weights, find a spanning tree with minimal weight. There are a number of algorithms which can solve this problem efficiently. One algorithm we will highlight is Kruskal's algorithm. The pseudocode of Kruskal's algorithm is given in Algorithm 1. Kruskal's algorithm always terminates and returns a minimum spanning tree. The proof of correctness and the pseudocode follow from (Korte and Vygen, 2007).

Next, a result related to multiple spanning trees.

Lemma 1.3.4. *Suppose we have two MSTs T_1 and T_2. Furthermore, suppose we have an edge $e' = \{u, v\}$, which is not in T_1 or T_2, and let f_i be a maximum weight edge on T_i's $u-v$-path, for $i \in \{1, 2\}$. Then we have $w(f_1) = w(f_2)$.*

Proof. Assume without loss of generality that $w(f_1) \leq w(f_2)$. We introduce new weights $\overline{w}$, which we let $\overline{w}(e) = w(e)\ \forall e \in E \backslash e'$ and we let $\overline{w}(e') = \min(w(f_1), w(f_2))$. Since we assumed $w(f_1) \leq w(f_2)$, we have $\overline{w}(e') = w(f_1)$. Then T_1 is $\overline{w}$-minimal since exchanging f_1 with e' would not decrease the

Algorithm 1: Kruskal's Algorithm

```
    Input: A connected undirected graph $G$ with weights $\tilde{w}$ on edges $E$
    Output: A spanning tree $T$ of minimum weight
Sort the edges such that $\tilde{w}(e_1) \leq \tilde{w}(e_2) \leq \cdots \leq \tilde{w}(e_m)$;
Set $T$ to the set of vertices of $G$ and an empty set of edges;
for $i$ from 1 to $m$ do
    if $T \cup \{e_i\}$ contains no cycles then
        Set $T$ to $T \cup \{e_i\}$;
    end
end
```

weight. If any other exchange would decrease the weight, this would be a contradiction with respect to the assumption that T_1 is w-minimal.

Since we have $\overline{w}(T_1) = \overline{w}(T_2)$, we know that T_2 is also $\overline{w}$-minimal. Therefore, there is no edge we can exchange to obtain a cheaper tree, which implies $\overline{w}(e') \geq \overline{w}(f_2)$. Since we defined $\overline{w}(e') = \overline{w}(f_1)$, we have $\overline{w}(f_1) \geq \overline{w}(f_2)$. Thus we conclude that $w(f_1) = w(f_2)$. $\square$

Stochastic weights For graphs with deterministic weights, there is no ambiguity about whether a given spanning tree has minimal weight. However, for graphs with stochastic weights, the minimal solution for a given sample is not necessarily equal to the solution which is optimal in expectation. Therefore, it is not immediately clear which objective we should consider for our problem. More specifically, we can differentiate between two types of optima, adaptive and non-adaptive. The adaptive optimal solution is the solution which is optimal over the sampled weights. In our case, it is a spanning tree with minimal weight over the given samples. As the name suggest, this optimum can be different depending on the given sample. Let $\mathbb{T}(G)$ be the collection of all spanning trees in graph G. Then, the expected value of the adaptive optimum is given by $\mathbb{E}[\min_{T\in\mathbb{T}(G)} \tilde{w}(T)]$.

The non-adaptive optimum is the solution which is optimal in expectation. In our case, this is the solution with minimal weight over the *expected* weights. This optimum does not depend on the given samples. The expected value of the non-adaptive optimum equals $\min_{T\in\mathbb{T}(G)} \mathbb{E}[\tilde{w}(T)]$.

Clearly, $\min_{T\in\mathbb{T}(G)} \tilde{w}(T) \leq \tilde{w}(T)\ \forall T \in \mathbb{T}(G)$, which implies $\mathbb{E}[\min_{T\in\mathbb{T}(G)} \tilde{w}(T)] \leq \mathbb{E}[\tilde{w}(T)]\ \forall T \in \mathbb{T}(G)$. This in turn implies that $\mathbb{E}[\min_{T\in\mathbb{T}(G)} \tilde{w}(T)] \leq \min_{T\in\mathbb{T}(G)} \mathbb{E}[\tilde{w}(T)]$. Thus, the adaptive optimum is a stronger adversary than the non-adaptive optimum. In the next section we will explain which of these two adversaries we will use.

1.3.4 Single Sample Stochastic Minimum Spanning Tree Problem

The main problem we are concerned with in this thesis is the Single Sample Stochastic Minimum Spanning Tree problem, or SSMST in short. In this problem, the objective is to compute a spanning tree with in expectation minimal weight, while the decision maker is only given one sample of each edge's weight function. More specifically, an instance $I = (G, w)$ of the SSMST problem consists of a (connected) graph $G = (V, E)$ and some stochastic weight function $w : E \to \mathbb{R}$ from which the edge weights are drawn. By $\tilde{w}(e_i)$ we denote one sample of the weight function of edge e_i. The objective is to compute a spanning tree with in expectation minimal weight. Since only one sample of each edge's weight distribution is provided, perhaps the only reasonable strategy is to use the given weight sample as a proxy for the expected weights of the edges, similar to the strategy proposed by (te Rietmole and Uetz, 2024).

Definition 1.3.1. Algorithm Sam chooses a spanning tree by applying Kruskal's algorithm on graph G with the sampled edge weights.

Our goal is to analyze this algorithm's expected performance. To give this analysis some meaning, we will be comparing the expected weight of Sam to the expected weight of the non-adaptive optimum, which we introduced in the previous section. We will denote the non-adaptive optimum as Opt. As stated in the previous section, the non-adaptive optimum is a weaker adversary than the adaptive optimum. The decision to choose the weaker of the two adversaries will become clear in Section 1.3.5. There we will show an example for which Sam performs arbitrarily bad in comparison with the adaptive optimum.

1.3.5 Single Sample Stochastic Minimum Item Selection Problem

The SSMST problem is the main problem which will be discussed in this thesis. However, since this problem is quite involved we first consider a simpler variant, the Single Sample Minimum Item Selection, or SSMIS in short. Instead of computing a spanning tree in a graph, in the SSMIS problem the objective is to choose one item with, in expectation, the lowest weight. More specifically, an instance $I = (N, w)$ of the SSMIS problem consist of a set of m items $N = (i_1, i_2, \ldots, i_m)$, and a stochastic weight function $w : N \to \mathbb{R}$, from which weights are independently drawn for each item. By $\tilde{w}(i_j)$ we denote one weight sample of item j. Similar to the SSMST problem, the objective is to choose the item with the lowest expected weight. By the definition of the non-adaptive optimum Opt, we have $\mathbb{E}[\text{Opt}(I, w)] = \min_{i_k \in N} \mathbb{E}[\tilde{w}(i_k)]$.

It is important to note that the SSMIS problem is a special case of the SSMST problem. This can be shown via a simple reduction. Suppose we have an instance $I = (N, w)$ of the SSMIS problem with m items. Then we can reduce this instance to an instance of the SSMST problem. We can construct an instance $I' = (G, w')$ of the SSMST problem consisting of a graph G with 2 vertices and m parallel edges between these vertices, and a weight function w' such that $w(i_j) = w'(e_j)$. Since the weight functions are equal for both problems, and for both problems only one item needs to be chosen, these instances are equivalent.

For the SSMIS problem only one item needs to be chosen, in contrast to computing a whole spanning tree for the SSMST problem. This makes the calculation of the relative performance of Sam a lot simpler, since there are fewer possible combinations. Furthermore, in the SSMST problem the chance of a certain edge being included in a spanning tree not only depends on the sampled weight of that edge, but also on the structure of the graph. If an edge would form a cycle with previously chosen edges, it cannot be included anymore, even if it has a lower weight than some other edges. For the SSMIS problem we don't have to deal with such dependence relations.

This makes the SSMIS problem a nice candidate for verifying whether certain distributions are "well-behaved", by which we mean that a relative performance bound can be determined for Sam. In the context of the SSMIS problem, we define Sam to be the algorithm which chooses the item with the smallest sampled weight. If we find that the relative performance of Sam cannot be bounded for a certain distribution on the SSMIS, we also know that it cannot be bounded on the SSMST problem. However, the reverse is not necessarily true. A distribution which is well-behaved on the SSMIS problem need not necessarily be well-behaved on the SSMST problem. In the next section we discuss some classes of distributions which are not necessarily well-behaved for the SSMIS problem, and one class which is well-behaved.

Choice of adversary As mentioned in the previous section, our adversary Opt will be the non-adaptive optimum, which is a weaker adversary than the adaptive optimum. To clarify this decision, we will show an example for which Sam performs arbitrarily bad compared to the adaptive optimum.

Suppose we have an instance $I = (N, w)$ of the SSMIS problem with n items, and a weight function $w \sim \exp(\lambda)$, which draws the item weights from an exponential distribution with rate λ. Since all items have the same expected weight, the expectation of SAM on this instance is $\frac{1}{\lambda}$. It is a well-known result that the minimum of exponential random variables, is again exponentially distributed with as rate the sum of all the individual rates (Ibe, 2014, p.205). That is, $\min_i w(i) \sim \exp(n\lambda)$. Therefore, the expectation of the adaptive optimum is $\frac{1}{n\lambda}$. Hence, the expected performance of SAM is n times as large as the expected performance of the adaptive optimum. It is possible that n is arbitrarily large, and thus SAM can perform arbitrarily bad in comparison with the adaptive optimum. Therefore, we use the non-adaptive optimum as adversary, denoted by OPT.

Chapter 2

Well-behaved input distributions

In their paper, (te Rietmole and Uetz, 2024) showed four classes of processing time distributions for which Sam performs as good as or better than the uniform random adversary. We will be comparing the performance of Sam to that of the non-adaptive optimum Opt, which is a different adversary than used for the scheduling problem. Since we will be using a different adversary, the processing time distributions which perform better than the random adversary on the scheduling problem will not necessarily be weight distributions for which we can derive performance bounds on the SSMIS problem. We will call a distribution 'well-behaved' if a performance guarantee can be derived for Sam on every instance of the SSMIS problem, where all the item weights follow this distribution.

2.1 Input distributions which are not well-behaved

In this section, we will show for three out of the four distribution classes mentioned above that we cannot necessarily derive a performance guarantee for Sam on the SSMIS problem.

Lemma 2.1.1. *On the SSMIS problem, there is in general no performance guarantee if the weights are drawn from one of the following distribution classes:*

1. *Translations of identical weight distributions*
2. *Symmetric weight distributions*
3. *Identical weight distributions up to scaling*

Proof. We proof this statement by giving two examples for which no performance bound can be given. The first example covers the first two distribution classes, the second example covers the last class.

Translations of identical weight distributions & Symmetric weight distributions Suppose we have an instance $I = (N, w)$ of the SSMIS problem with two items i_1 and i_2, with $w(i_1) \sim U(-N+1, N+1)$ and $w(i_2) \sim U(-N+n, N+n)$, where $U(a,b)$ denotes the continuous uniform distribution on the domain $[a,b]$. In this instance, N is some non-negative real number. It is clear that item 1 has a lower expected weight than item 2, $\mathbb{E}[\tilde{w}(i_1)] = 1 < n = \mathbb{E}[\tilde{w}(i_2)]$. Furthermore, we have the following distribution functions, $f_1(x) = \frac{1}{2N}$ for $-N+1 < x < N+1$ and $f_2(x) = \frac{1}{2N}$ for $-N+n < x < N+n$, which we can use to calculate the expected value of Sam on this instance. For this purpose we calculate the probability that item 2 will have a smaller weight sample than item 1.

$$\begin{aligned}\mathbb{P}(\tilde{w}(i_1) > \tilde{w}(i_2)) &= \int_{-N+n}^{N+n} \int_{x}^{N+1} f_1(y) f_2(x) dy dx \\ &= \frac{1}{4N^2} \int_{-N+n}^{N+1} \int_{x}^{N+1} 1 dy dx \\ &= \frac{1}{4N^2} \int_{-N+n}^{N+1} (N+1-x) dx \\ &= \frac{1}{4N^2} ((N+1)x - \frac{x^2}{2})|_{-N+n}^{N+1} \\ &= \frac{1}{4N^2} ((N+1)^2 - \frac{(N+1)^2}{2} - (-N^2 + nN - \frac{(-N+n)^2}{2})) \\ &= \frac{1}{2} - \frac{n}{2N} + \frac{n^2}{8N^2} - \frac{n}{4N^2} + \frac{1}{8N^2} + \frac{1}{2N^3}\end{aligned}$$

Now, if we let $\lim_{N\to\infty}$, we have,

$$\mathbb{E}[\text{SAM}(I)] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot n = \frac{n}{2} + 1 \,.$$

Furthermore, we have $\mathbb{E}[\text{OPT}(I)] = 1$. Therefore, the relative performance of SAM is given by,

$$\begin{aligned}\alpha(I) &= \frac{\mathbb{E}[\text{SAM}(I)]}{\mathbb{E}[\text{OPT}(I)]} \\ &= \frac{\frac{n}{2}+1}{1} = \frac{n}{2} + 1 \,,\end{aligned}$$

which can be arbitrarily large.

Thus, there exist distributions for which SAM has a chance of a half to choose the "wrong" item, of which the expected weight can be arbitrarily large compared to the weight of the "right" item. Such examples are easy to construct when negative weights are allowed.

This also highlights one of the main differences between using random as adversary, or comparing against the non-adaptive optimum. In the above example, SAM has the same expected performance as the random adversary, since randomly choosing also yields a probability of one half of choosing the item with the lowest expected weight. Furthermore, (te Rietmole and Uetz, 2024) showed that it is sufficient for SAM to schedule any pair of jobs in the "correct" order at least half of the time, to perform at least as good as the random adversary. That is, with at least a 50% chance, SAM has to schedule the job with the larger expected weight over processing time before the other job. Since our adversary is OPT, and we need to choose a subset of the total items, choosing the "right" item half of the time can still lead to an arbitrarily bad performance ratio.

Identical weight distributions up to Scaling For weight distributions which are identical up to scaling, we give an example for which no performance guarantee for SAM can be given.

Suppose we have a normal distribution $N_1 = N(1, \sigma)$, with probability density function $g(x)$ and another normal distribution N_2 with probability density function $f(x) = cg(cx)$, for $c > 0$. Then, this second normal distribution has a mean of $1/c$. Now, if we let σ grow large, the probability that N_1 has a smaller sample than N_2 tends to one half. Furthermore, as long as we have $1 < c << \sigma$, there is a chance of one half that SAM would choose N_1, which yields an expected relative performance of $\frac{1}{2} + \frac{c}{2}$, which can be arbitrarily large. Therefore, we cannot guarantee a performance bound on this class of probability distributions. □

Note that for each class discussed we only gave one counter-example. This does not imply that there do not exist any probability distributions within these classes which are well-behaved.

2.2 Exponential distribution

While the distribution classes discussed in the previous section are not necessarily well-behaved for the SSMIS problem, we will show that the exponential distribution is well-behaved. In this section we highlight some of the properties of this distribution which make it well-behaved.

Lemma 2.2.1. *Suppose we have n exponential random variables $(X_1, \dots, X_N)$, with possibly different rate parameter λ_i. Then we have,*

$$\mathbb{P}(X_i = \min(X_1, \dots, X_n)) = \frac{\lambda_i}{\sum_{k=1}^{n} \lambda_k} . \tag{2.1}$$

Proof. The proof of this lemma is standard and can be found in appendix A.1. □

This result allows us to determine the probability that an edge or an item has the smallest weight sample, out of all weight samples.

The examples we showed in the previous section for distributions which are not well-behaved, all had in common that the variance could grow large independent of the mean. For the exponential distribution, we have that the mean equals $\frac{1}{\lambda}$, and the variance equals $\frac{1}{\lambda^2}$. Therefore, the variance is not independent of the mean. In the next section we show that the exponential distribution is well-behaved for the SSMIS problem.

2.3 SSMIS problem with exponential weights

In this section, we derive a performance bound for Sam on the SSMIS problem with exponential item weights. First we introduce some notation specific to the exponential distribution.

Recall that we defined an instance $I = (N, w)$ of the SSMIS problem to consist of an item set $N = (i_1, i_2, \dots, i_m)$ and a weight function $w : N \to \mathbb{R}$. Since we assume in this section that every item's weight function follows an exponential distribution, we denote by $\lambda_{i_k} \in \mathbb{R}_+$ the rate parameter associated with item i_k. That is, $w(i_k) \sim \exp(\lambda_{i_k})$. We denote by $\boldsymbol{\lambda}_N = (\lambda_{i_1}, \lambda_{i_2}, \dots, \lambda_{i_m})$ the vector containing all rate parameters of the items in N. Furthermore, we let Λ_N denote the collection of rate parameters $\boldsymbol{\lambda}_N$. Since this section only covers instances with exponential weights, we will denote an instance by $I = (N, \boldsymbol{\lambda}_N)$, since the exponential weight functions are uniquely defined by their rate parameter. Since the weights of the items are strictly positive under the exponential distribution, we can determine the relative performance of Sam compared to the non-adaptive optimum Opt. For this, we define a performance measure α.

Definition 2.3.1. We define $\alpha(N, \boldsymbol{\lambda}_N) = \frac{\mathbb{E}[\textsc{Sam}(N, \boldsymbol{\lambda}_N)]}{\mathbb{E}[\textsc{Opt}(N, \boldsymbol{\lambda}_N)]}$ as the relative performance of Sam compared to Opt on instance $I = (N, \boldsymbol{\lambda}_N)$.

We will derive a performance guarantee on the relative performance of Sam on the SSMIS problem with m items with exponential weights.

Lemma 2.3.1. *On the SSMIS problem with m items with exponentially distributed weights, the relative performance of Sam is no worse than m. That is, $\alpha(N, \boldsymbol{\lambda}_N) \leq m \quad \forall \boldsymbol{\lambda}_N \in \Lambda_N$. Furthermore, this bound is tight.*

Proof. First, observe that,

$$\mathbb{E}[\textsc{Sam}(N, \boldsymbol{\lambda}_N)] = \sum_{i_l \in N} \mathbb{P}(\tilde{w}(i_l) = \min_{i_k \in N} \tilde{w}(i_k))\mathbb{E}[\tilde{w}(i_l)]\,.$$

By Lemma 2.2.1 we have,

$$\begin{aligned}\sum_{i_l \in N} \mathbb{P}(\tilde{w}(i_l) = \min_{i_k \in N} \tilde{w}(i_k))\mathbb{E}[\tilde{w}(i_l)] &= \sum_{i_l \in N} \frac{\lambda_{i_l}}{\sum_{i_k \in N} \lambda_{i_k}} \frac{1}{\lambda_{i_l}} \\ &= \frac{m}{\sum_{i_k \in N} \lambda_{i_k}}\,.\end{aligned}$$

Next, we derive the expected weight of Opt. Without loss of generality we assume item 1 is the item with the lowest expected weight. This gives,

$$\mathbb{E}[\textsc{Opt}(N, \boldsymbol{\lambda}_N)] = \mathbb{E}[\tilde{w}(i_1)] = \frac{1}{\lambda_{i_1}}\,.$$

Combining these results, we can calculate the performance ratio of Sam,

$$\begin{aligned}\alpha(N, \boldsymbol{\lambda}_N) &= \frac{\mathbb{E}[\textsc{Sam}(N, \boldsymbol{\lambda}_N)]}{\mathbb{E}[\textsc{Opt}(N, \boldsymbol{\lambda}_N)]} \\ &= \frac{\frac{m}{\sum_{i_k \in N} \lambda_{i_k}}}{\frac{1}{\lambda_{i_1}}} \\ &= \frac{\lambda_{i_1} m}{\sum_{i_k \in N} \lambda_{i_k}}\,.\end{aligned}$$

Since we assumed that item 1 is the item with the lowest expected weight, we know that $\lambda_{i_1} \geq \lambda_{i_k} \forall\, i_k \in N$. Because of this, we know that $\frac{\lambda_{i_1}}{\sum_{i_k \in N} \lambda_{i_k}} \leq 1$. This gives,

$$\begin{aligned}\alpha(N, \boldsymbol{\lambda}_N) &= \frac{\lambda_{i_1} m}{\sum_{i_k \in N} \lambda_{i_k}} \\ &\leq m\,.\end{aligned}$$

Thus on the SSMIS problem with m items with exponentially distributed weights, the worst case performance ratio of Sam is m. Furthermore, this bound is tight, as this value is achieved when we take the limit, $\lambda_{i_1} \to \infty$. □

Chapter 3

Single Sample MST problem with exponential weights

In this chapter we prove the main result of this thesis; we will derive a performance bound for Sam on connected graphs with exponentially distributed edge weights. In the first section, we give the main result, and prove several small results which are needed for the proof of the main result. In the last two sections, we will derive a lower bound and an upper bound on the relative performance of Sam.

3.1 Main result and smaller results

We start this section by introducing the main result of this thesis. Recall that we denoted by $B(G) \subseteq E$ a largest bond in graph $G = (V, E)$.

Theorem 3.1.1. *For all connected graphs G with exponentially distributed edge weights, Sam is a $|B(G)|$-approximation algorithm, that is, for all connected graphs G and all $\boldsymbol{\lambda}_G \in \Lambda_G$, $\frac{\mathbb{E}[\textsc{Sam}(G,\boldsymbol{\lambda}_G)]}{\mathbb{E}[\textsc{Opt}(G,\boldsymbol{\lambda}_G)]} \leq |B(G)|$. Furthermore, for all connected graphs G and all $\epsilon > 0$, there exists a $\boldsymbol{\lambda}_G \in \Lambda_G$ such that $\alpha(G, \boldsymbol{\lambda}_G) \geq |B(G)| - \epsilon$.*

Thus, not only is the performance of Sam purely determined by the structure of the graph, but we can also find a rate vector $\boldsymbol{\lambda}_G$ such that this bound is tight.

In the remainder of this section, we will derive multiple results for the SSMST problem with exponential weights, which will we will need for the proof of the main theorem. First we introduce some notation, which follows from the notation we used earlier for the SSMIS problem.

3.1.1 Notation

Suppose we have a connected graph $G = (V, E)$, where $E = (e_1, e_2, \dots, e_m)$ and thus $|E| = m$. Each edge e_i has its associated rate parameter λ_{e_i}. It is not necessarily the case that $\lambda_{e_i} \neq \lambda_{e_j}$ if $i \neq j$. The collection of all rate parameters assigned to edges in G will be denoted by $\boldsymbol{\lambda}_G$, that is, $\boldsymbol{\lambda}_G = (\lambda_e)_{e \in E}$. The collection of all possible rate vectors $\boldsymbol{\lambda}_G$ for graph G is denoted by Λ_G. An instance of the SSMST problem is then defined by the pair $(G, \boldsymbol{\lambda}_G)$.

While this notation is well-defined for graph G, it might induce ambiguity when graph minors are considered. For the proof of the theorem we will be comparing the performance of our algorithm on graph G, to the performance on graphs which can be obtained by means of contracting edges. Therefore we will introduce some notation which will make it clear which edges and parameters are considered, and which are not.

Let H be a graph which can be obtained from G by contracting edges. Then by $\boldsymbol{\lambda}_G(H)$ we denote the rate vector $\boldsymbol{\lambda}_G$ restricted to graph H, that is, $\boldsymbol{\lambda}_G(H) = (\lambda_e)_{e \in E(H)}$, where $E(H)$ denotes the edge set of graph H. Since H is obtained from graph G by contracting edges, it is clear that $E(H) \subset E(G)$.

Next, some notation for the weight function w. For the weight of edges, we use the weight function, $w(e_i, \boldsymbol{\lambda}_G) \sim \exp(\lambda_{e_i})$, where $\lambda_{e_i} \in \boldsymbol{\lambda}_G$. A single sample of edge e_i's weight function will be denoted by $\tilde{w}(e_i, \boldsymbol{\lambda}_G)$. To denote the weight of a spanning tree, we will use the same weight function. Suppose we have some spanning tree T. Then we have,

$$\tilde{w}(T, \boldsymbol{\lambda}_G) = \sum_{e \in T} \tilde{w}(e, \boldsymbol{\lambda}_G)$$

and

$$\mathbb{E}[\tilde{w}(T, \boldsymbol{\lambda}_G)] = \mathbb{E}[\sum_{e \in T} \tilde{w}(e, \boldsymbol{\lambda}_G)] = \sum_{e \in T} \mathbb{E}[\tilde{w}(e, \boldsymbol{\lambda}_G)]\,.$$

Next, we would like to express the expected value of Opt on an instance $(G, \boldsymbol{\lambda}_G)$. Suppose T^* is a spanning tree with minimal expected weight. Then, by $\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]$ we denote the expected weight of T^*, that is, $\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] = \sum_{e \in T^*} \mathbb{E}[\tilde{w}(e, \boldsymbol{\lambda}_G)]$.

To derive to relative performance of Sam on an instance of the SSMST problem to that of Opt, we will use conditioning. Recall that Sam is defined to apply Kruskal's MST algorithm on the sampled edge weights. Therefore, we can condition on which edge will be chosen first by Sam. By $\textsc{Ch}(G, e, j, \boldsymbol{\lambda}_G)$ we denote the event where the jth edge that Sam chooses is edge e on the instance $(G, \boldsymbol{\lambda}_G)$. For $j = 1$, this denotes the event where edge e is the first edge chosen by Sam.

For a given T^*, it can occur that the first edge chosen by Sam, e_i, is not part of T^*. In this case, $T^* \cup e_i$ contains a cycle C. Then, if we would construct a minimum weight spanning tree that must include e_i, we would exchange in T^* the edge with the second largest expected weight in $T^* \cap C$ with e_i. By e_{i^*} we denote the edge with the second largest expected weight in $T^* \cap C$. In the case where $e_i \in T^*$, we define $e_{i^*} = e_i$.

3.1.2 Preliminary results

In this section we derive some results which will be needed for the proof of the main theorem. We start by introducing a lemma. The memorylessness property of the exponential distribution allows us to relate the expected value of Sam on some graph, to the expected value of Sam on a smaller graph. This result will be a key component for the proof of our theorem.

More specifically, we condition on the event that some edge e_i has the smallest weight in the given sample. Since this edge has the smallest weight in the given sample, it will be the first edge chosen by Sam. Furthermore, since we are growing a spanning tree, it makes sense that any edges that would form a cycle with e_i cannot be chosen anymore. At this point, we have only chosen one edge for our spanning tree. Therefore, the only edges that can form a cycle with e_i at this point are edges which are parallel to e_i. Since these edges and e_i cannot be chosen anymore, they can be omitted from the graph. This can be achieved by contracting e_i. Therefore, we relate the expected value of Sam on graph G to the expected value of Sam on graph G/e_i.

Lemma 3.1.2.

$$\mathbb{E}[\textsc{Sam}(G, \boldsymbol{\lambda}_G) | \textsc{Ch}(G, e_i, 1)] = \mathbb{E}[\textsc{Sam}(G/e_i, \boldsymbol{\lambda}_G(G/e_i))] + \mathbb{E}[\tilde{w}(e_i, \boldsymbol{\lambda}_G)] \tag{3.1}$$

Proof. Since we condition on the fact that edge e_i will be chosen first, it is clear that the expected weight of edge e_i is part of the expectation of SAM. Recall that $\mathbb{E}[w(e_i, \boldsymbol{\lambda}_G)] = \frac{1}{\lambda_{e_i}}$.

What is left to show, is that the expectation of SAM on graph G/e_i is the same as on graph G, ignoring the contribution of edge e_i. We will prove this by showing that the probability of another edge e_j being chosen second in graph G is equal to the probability of that same edge is chosen first in graph G/e_i. That is,

$$\mathbb{P}(\textsc{Ch}(G, e_j, 2)|\textsc{Ch}(G, e_i, 1)) = \mathbb{P}(\textsc{Ch}(G/e_i, e_j, 1))\,. \tag{3.2}$$

Proving this is sufficient, since the expected performance of SAM is uniquely determined by the probability that edges are chosen and their expected weight. Since the expected weights of the edges in both graphs are equal, it is left to show that the probability that an edge is chosen in either graph is equal.

Because of the memorylessness property of the exponential distribution, this result is easy to derive.

$$\mathbb{P}(\textsc{Ch}(G, e_j, 2)|\textsc{Ch}(G, e_i, 1)) = \frac{\mathbb{P}(\textsc{Ch}(G, e_i, 1) \cap \textsc{Ch}(G, e_j, 2))}{\mathbb{P}(\textsc{Ch}(G, e_i, 1))} \tag{3.3}$$

By $E(G/e_i)$ we denote the edge set of graph G/e_i. Then, equation 3.3 equals,

$$\begin{aligned}
&= \frac{\frac{\lambda_{e_i}}{\sum_{e_k \in E(G)} \lambda_{e_k}} \frac{\lambda_{e_j}}{\sum_{e_l \in E(G/e_i)} \lambda_{e_l}}}{\frac{\lambda_{e_i}}{\sum_{e_k \in E(G)} \lambda_{e_k}}} \\
&= \frac{\lambda_{e_j}}{\sum_{e_l \in E(G/e_i)} \lambda_{e_l}} \\
&= \mathbb{P}(\textsc{Ch}(G/e_i, e_j, 1))\,.
\end{aligned}$$

Thus the probability that some edge is chosen second after edge e_i, in graph G, is the same probability that that edge is chosen first in G/e_i. Therefore, we can conclude that $\mathbb{E}[\textsc{Sam}(G, \boldsymbol{\lambda}_G)|\textsc{Ch}(G, e_i, 1)] = \mathbb{E}[\textsc{Sam}(G/e_i, \boldsymbol{\lambda}_G(G/e_i))] + \frac{1}{\lambda_{e_i}}$. □

Now we have a result which we can use to relate the expected value of SAM on one graph, to that of the graph with one edge contracted. Since we will be comparing the expected value of SAM to that of the non-adaptive optimum OPT, it would also be useful to be able to compare the expected value of OPT on one graph, to that of the graph with one edge contracted. The next result allows us to do exactly that.

Recall how we defined OPT to represent a spanning tree with minimal expected weight, T^*. Such a tree is not necessarily unique, which is something we have to take into account. Furthermore, we have to make a case distinction on whether e_i is part of T^* or not.

Recall how we defined e_{i^*} to be equal to e_i if $e_i \in T^*$, and otherwise e_{i^*} denotes the edge with the second largest expected weight in cycle C, which is formed by taking the union of e_i and T^*.

Lemma 3.1.3.

$$\mathbb{E}[\textsc{Opt}(G/e_i, \boldsymbol{\lambda}_G(G/e_i))] = \mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] - \mathbb{E}[\tilde{w}(e_{i^*}, \boldsymbol{\lambda}_G)] \tag{3.4}$$

Proof. **Case 1.** $(\mathbf{e_i \in T^*})$
Since $e_i \in T^*$, we have $e_{i^*} = e_i$. Furthermore, T^*/e_i is an MST in G/e_i. Suppose this were not the case, then there exists some other tree T' in G/e_i which has a smaller expected weight. However, this would imply that $T' \cup e_i$ is cheaper in expectation than T^*, which would be a contradiction. Therefore, we must have that T^*/e_i is a MST in G/e_i.

This gives us,

$$\begin{aligned}\mathbb{E}[\textsc{Opt}(G/e_i, \boldsymbol{\lambda}_G(G/e_i))] &= \mathbb{E}[\tilde{w}(T^*/e_i, \boldsymbol{\lambda}_G(G/e_i))] \\ &= \mathbb{E}[\tilde{w}(T^*, \boldsymbol{\lambda}_G)] - \mathbb{E}[\tilde{w}(e_i, \boldsymbol{\lambda}_G)] \\ &= \mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] - \mathbb{E}[\tilde{w}(e_{i^*}, \boldsymbol{\lambda}_G)] \,.\end{aligned}$$

Case 2 $(\mathbf{e_i \notin T^*})$ Clearly, if e_i is part of some MST, it follows from Case 1 that the equation still holds. Therefore, we assume that e_i is not part of any MST. To obtain a minimum weight spanning tree which must include e_i, we can modify T^* by exchanging e_{i^*} with e_i. This new tree is minimal over the set of all spanning trees which include e_i. This follows from the correctness of Kruskal's algorithm. Assume for now that T^* is a unique MST in G. Suppose we start with $T = \{e_i\}$, and want to extend this to a spanning tree using Kruskal's algorithm. Note that this is equivalent to applying Kruskal's algorithm on G/e_i. Then by applying Kruskal's algorithm we obtain tree $T^* - e_{i^*} + e_i$. If another tree were to be obtained, this would be a contradiction with the assumption that T^* is the unique MST.

Now suppose T^* is not uniquely minimal, i.e. there are multiple minimum spanning trees with the same expected weight. Since the expected weights of the MSTs is equal by definition, we only need to show that if two MSTs have different e_{i^*}s, these edges must have the same expected weight. This follows directly from Lemma 1.3.4.

Therefore, we can write,

$$\begin{aligned}\mathbb{E}[\textsc{Opt}(G/e_i, \boldsymbol{\lambda}_G)] &= \mathbb{E}[\tilde{w}(T^*, \boldsymbol{\lambda}_G)] - \mathbb{E}[\tilde{w}(e_{i^*}, \boldsymbol{\lambda}_G)] \\ &= \mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] - \mathbb{E}[\tilde{w}(e_{i^*}, \boldsymbol{\lambda}_G)] \,,\end{aligned}$$

which concludes our proof. □

Lastly, a technical lemma which allows us to take the limit inside of the supremum.

Lemma 3.1.4.

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq \sup_{\boldsymbol{\lambda}_{G\setminus\overline{e}} \in \Lambda_{G\setminus\overline{e}}} \lim_{\lambda_{\overline{e}} \to \infty} \alpha(G, \boldsymbol{\lambda}_G) \tag{3.5}$$

Proof. By the definition of the supremum, we have

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq \alpha(G, \boldsymbol{\lambda}'_G) \quad \forall \, \boldsymbol{\lambda}'_G \in \Lambda_G \,.$$

In particular, we have that

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq \lim_{\lambda_{\overline{e}} \to \infty} \alpha(G, \boldsymbol{\lambda}'_G) \quad \forall \, \boldsymbol{\lambda}'_{G\setminus\overline{e}} \in \Lambda_{G\setminus\overline{e}} \,.$$

Thus we also have,

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq \sup_{\boldsymbol{\lambda}'_{G\setminus\overline{e}}} \lim_{\lambda_{\overline{e}} \to \infty} \alpha(G, \boldsymbol{\lambda}'_G) \,.$$

This concludes the proof. □

3.2 Lower bound

In this section, we derive a lower bound on the worst case relative performance of Sam on the SSMST problem with exponential weights. This results holds for any connected graph G.

Theorem 3.2.1. *For all connected graphs G with exponentially distributed edge weights, the performance bound of* Sam *can be no better than $|B(G)|$. That is,*

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq |B(G)| . \tag{3.6}$$

Proof. We prove this theorem using strong induction on the number of edges in G. First, we show that this theorem holds for connected graphs with $m = 1$ edges. Then, we assume for our induction hypothesis that the statement also holds for $m = 2$ up to k edges, and we will show that under this assumption it also holds for $m = k + 1$ edges.

For the case $m = 1$, it is clear that there is only 1 spanning tree to choose from, thus the performance bound of Sam equals 1, which is clearly (greater than or) equal to $|B(G)|$. Thus the statement holds for $m = 1$ edges. Now, we assume the statement also holds for $m = 2$ up to $m = k$ edges. From the SSMIS problem we already know that the statement is true for graphs with just 2 vertices, thus from now on we assume $|V| > 2$.

Suppose we have a graph G with $m = k + 1$ edges. We derive the following performance bound,

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) = \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\mathbb{E}[\text{Sam}(G, \boldsymbol{\lambda}_G)]}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]}$$

By conditioning we have that is equals,

$$= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\text{Ch}(G, e, 1))\mathbb{E}[\text{Sam}(G, \boldsymbol{\lambda}_G) | \text{Ch}(G, e, 1)]}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]} .$$

By Lemma 3.1.2 this equals,

$$= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\text{Ch}(G, e, 1))(\mathbb{E}[\text{Sam}(G/e, \boldsymbol{\lambda}_G(G/e))] + \frac{1}{\lambda_e})}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]} .$$

By the definition of α we this equals

$$= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\text{Ch}(G, e, 1))(\alpha(G/e, \boldsymbol{\lambda}_G(G/e))\mathbb{E}[\text{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + 1/\lambda_e)}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]} .$$

From Lemma 3.1.3 it follows that this equals,

$$\begin{aligned} &= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} (\frac{\sum_{e \in E(G)} \mathbb{P}(\text{Ch}(G, e, 1))(\alpha(G/e, \boldsymbol{\lambda}_G(G/e))\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)])}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]} \\ &\quad - \frac{\sum_{e \in E(G)} \mathbb{P}(\text{Ch}(G, e, 1))\alpha(G/e, \boldsymbol{\lambda}_G)/\lambda_{e^*} + 1/\lambda_e}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]}) \\ &= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} (\sum_{e \in E(G)} \mathbb{P}(\text{Ch}(G, e, 1))[\alpha(G/e, \boldsymbol{\lambda}_G(G/e)) \\ &\quad - \frac{\alpha(G/e, \boldsymbol{\lambda}_G)}{\lambda_{e^*}\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]} - \frac{1}{\mathbb{E}[\text{Opt}(G, \boldsymbol{\lambda}_G)]\lambda_e}]) . \end{aligned}$$

Now, let us choose one edge $\overline{e} \notin B(G)$. By Lemma 1.3.2, we know such an edge always exists. To give a lower bound on the supremum over all $\boldsymbol{\lambda}_G$, we derive the supremum over a subset of all possible $\boldsymbol{\lambda}_G$. For this, we take the limit of $\lambda_{\overline{e}}$. By lemma 3.1.4 we know this holds.

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq \sup_{\boldsymbol{\lambda}_{G\setminus\overline{e}} \in \Lambda_{G\setminus\overline{e}}} \lim_{\lambda_{\overline{e}} \to \infty} \alpha(G, \boldsymbol{\lambda}_G)$$

We restrict our choice of $\boldsymbol{\lambda}_G$ further by assuming $\lambda_e \in o(\lambda_{\overline{e}}) \, \forall e \neq \overline{e}$. Under this assumption, we have the following,

$$\lim_{\lambda_{\overline{e}} \to \infty} \mathbb{P}(\textsc{Ch}(G, e, 1)) = \lim_{\lambda_{\overline{e}} \to \infty} \frac{\lambda_e}{\sum_{k \in E(G)} \lambda_k} = \begin{cases} 0, & \text{for } e \neq \overline{e} \\ 1, & \text{for } e = \overline{e} \end{cases}$$

Thus, by evaluating the limit, the summation over all edges can be reduced to $\overline{e}$, since all other probabilities tend to zero. Therefore, we obtain,

$$\begin{aligned}
\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq & \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \lim_{\lambda_{\overline{e}} \to \infty} \Big(\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))[\alpha(G/e, \boldsymbol{\lambda}_G(G/e)) \\
& - \frac{\alpha(G/e, \boldsymbol{\lambda}_G)}{\lambda_{e^*} \mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]} - \frac{1}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] \lambda_e}] \Big) \\
\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq & \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \lim_{\lambda_{\overline{e}} \to \infty} [\alpha(G/\overline{e}, \boldsymbol{\lambda}_G(G/\overline{e})) \\
& - \frac{\alpha(G/\overline{e}, \boldsymbol{\lambda}_G)}{\lambda_{\overline{e}} \mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]} - \frac{1}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] \lambda_{\overline{e}}}] \\
= & \sup_{\boldsymbol{\lambda}_G \in \Lambda_{G\setminus\overline{e}}} \alpha(G/\overline{e}, \boldsymbol{\lambda}_G(G/\overline{e})) \,.
\end{aligned}$$

Notice that some terms disappear in the limit. In particular, $\textsc{Opt}$ depends on $\lambda_{\overline{e}}$, but also on at least one other term which is bounded. By the induction hypothesis we have,

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_{G\setminus\overline{e}}} \alpha(G/\overline{e}, \boldsymbol{\lambda}_G(G/\overline{e})) \geq |B(G/\overline{e})| \,.$$

Since we chose $\overline{e} \notin B(G)$, we have,

$$|B(G/\overline{e})| = |B(G)| \,.$$

Thus, we have shown that $\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \geq |B(G)|$. □

3.3 Upper bound

In this section, we derive an upper bound on the worst case relative performance of $\textsc{Sam}$ on the SSMST problem with exponentially distributed weights.

Theorem 3.3.1. *For all connected graphs G with exponentially distributed edge weights, the performance bound of $\textsc{Sam}$ is no worse than $|B(G)|$. That is,*

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \leq |B(G)| \tag{3.7}$$

Proof. Similar to the proof of the lower bound, we prove this theorem using strong induction on the number of edges in G. First, we show that this theorem holds for connected graphs with $m = 1$ edges. Then, we assume for our induction hypothesis that the statement also holds for $m = 2$ up to k edges, and we will show that under this assumption it also holds for $m = k + 1$ edges.

For the case $m = 1$, it is clear that there is only 1 spanning tree to choose from, thus the performance bound of SAM equals 1, which is clearly (smaller than or) equal to $|B(G)|$. Thus the statement holds for $m = 1$ edges. Now, we assume the statement also holds for $m = 2$ up to $m = k$ edges, for some integer $k \geq 2$. From the SSMIS problem we already know that the statement is true for graphs with just 2 vertices, thus from now on we assume $|V| > 2$.

Using this induction hypothesis, we derive an upper bound for $\alpha(G, \boldsymbol{\lambda}_G)$.

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) = \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\mathbb{E}[\textsc{Sam}(G, \boldsymbol{\lambda}_G)]}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

By conditioning we have,

$$= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))\mathbb{E}[\textsc{Sam}(G, \boldsymbol{\lambda}_G)|\textsc{Ch}(G, e, 1)]}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

By lemma 3.1.2 we have,

$$= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))(\mathbb{E}[\textsc{Sam}(G/e, \boldsymbol{\lambda}_G(G/e))] + 1/\lambda_e)}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

By the definition of α,

$$= \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))(\alpha(G/e, \boldsymbol{\lambda}_G)\mathbb{E}[\textsc{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + 1/\lambda_e)}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

Next, we apply the induction hypothesis. Recall that the induction hypothesis stated that, $\sup_{\boldsymbol{\lambda}_G \in \Lambda_G(G/e)} \alpha(G/e, \boldsymbol{\lambda}_G(G/e)) \leq |B(G/e)|$.

$$\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) = \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))(\alpha(G/e, \boldsymbol{\lambda}_G)\mathbb{E}[\textsc{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + 1/\lambda_e)}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}$$

Applying the induction hypothesis gives us,

$$\leq \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))(|B(G/e)|\mathbb{E}[\textsc{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + \frac{1}{\lambda_e})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

By lemma 1.3.2 we have,

$$\leq \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))(|B(G)|\mathbb{E}[\textsc{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + \frac{1}{\lambda_e})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}$$

$$= |B(G)| \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \mathbb{P}(\textsc{Ch}(G, e, 1))(\mathbb{E}[\textsc{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + \frac{1}{|B(G)|\lambda_e})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

Writing out the probability gives,

$$= |B(G)| \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e \in E(G)} \frac{\lambda_e}{\sum_{e_k \in E(G)} \lambda_{e_k}}(\mathbb{E}[\textsc{Opt}(G/e, \boldsymbol{\lambda}_G(G/e))] + \frac{1}{|B(G)|\lambda_e})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}.$$

Next, we apply Lemma 3.1.3 to obtain the following,

$$\begin{aligned}
&= |B(G)| \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\sum_{e\in E(G)} \frac{\lambda_e}{\sum_{e\in E(G)} \lambda_e} (\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] - \frac{1}{\lambda_{e^*}} + \frac{1}{|B(G)|\lambda_e})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]} \\
&= |B(G)| \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] + \sum_{e\in E(G)} \frac{1}{\sum_{e\in E(G)} \lambda_e} (-\frac{\lambda_e}{\lambda_{e^*}} + \frac{1}{|B(G)|})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]} .
\end{aligned}$$

If we can show that,

$$\sum_{e\in E(G)} \frac{1}{\sum_{e\in E(G)} \lambda_e} [-\frac{\lambda_e}{\lambda_{e^*}} + \frac{1}{|B(G)|}] = \frac{1}{\sum_{e\in E(G)} \lambda_e} [\sum_{e\in E(G)} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}}] \leq 0 ,$$

then we are done. This is equivalent to showing that,

$$\sum_{e\in E(G)} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}} \leq 0 .$$

To show this, we will split the summation into two sums, one which covers all edges in an optimal spanning tree T^*, and one which covers the other edges.

$$\sum_{e\in E(G)} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}} = \sum_{e\in T^*} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}} + \sum_{e\notin T^*} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}}$$

For $e \in T^*$, we have $\lambda_e = \lambda_{e^*}$, thus,

$$\begin{aligned}
\sum_{e\in E(G)} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}} &= \frac{|T^*|}{|B(G)|} - |T^*| + \sum_{e\notin T^*} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}} \\
&\leq \frac{|T^*|}{|B(G)|} - |T^*| + \sum_{e\notin T^*} \frac{1}{|B(G)|} .
\end{aligned}$$

By Lemma 1.3.3, we have $|E(G)| \leq |T^*||B(G)|$. Therefore,

$$\begin{aligned}
\sum_{e\in E(G)} \frac{1}{|B(G)|} - \frac{\lambda_e}{\lambda_{e^*}} &\leq \frac{|T^*|}{|B(G)|} - |T^*| + \frac{|B(G)||T^*| - |T^*|}{|B(G)|} \\
&= \frac{|T^*|}{|B(G)|} - |T^*| + \frac{|B(G) - 1||T^*|}{|B(G)|} \\
&= \frac{|B(G)||T^*|}{|B(G)|} - |T^*| \\
&= |T^*| - |T^*| = 0
\end{aligned}$$

Thus, we conclude,

$$\begin{aligned}
\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) &\leq |B(G)| \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)] + \sum_{e\in E(G)} \frac{1}{\sum_{e\in E(G)} \lambda_e} (-\frac{\lambda_e}{\lambda_{e^*}} + \frac{1}{|B(G)|})}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda})]} \\
&\leq |B(G)| \sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \frac{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]}{\mathbb{E}[\textsc{Opt}(G, \boldsymbol{\lambda}_G)]} \\
&= |B(G)| .
\end{aligned}$$

This concludes our proof that $\sup_{\boldsymbol{\lambda}_G \in \Lambda_G} \alpha(G, \boldsymbol{\lambda}_G) \leq |B(G)|$. $\square$

Theorem 3.1.1 follows directly from Theorem 3.2.1 and Theorem 3.3.1.

Chapter 4

Single Sample Minimal Matroid Basis problem with exponential weights

In this chapter we generalise the main result of of the previous chapter to matroids in general. In the first section we introduce concepts and notation related to matroids which we will use in this chapter. Afterwards, we will show that the lemmas we used in the previous chapter not only hold for graphic matroids, but any matroid in general. This allows us to reuse the proof we used to prove the main result of the previous chapter.

4.1 Matroids

Matroids have a wide range of applications within mathematics and they are often used to study independence-related properties in various contexts.

A matroid $M = (E, \mathcal{I})$ is a pair consisting of a ground set E and independence sets $\mathcal{I}$, which is a family of subsets of E. $\mathcal{I}$ has the following properties.

1. $\varnothing \in \mathcal{I}$
2. If $A \in \mathcal{I}$ and $A' \subseteq A \subseteq E$, then $A' \in \mathcal{I}$
3. If $A \in \mathcal{I}$ and $B \in \mathcal{I}$ and $|A| > |B|$, then there exists an $x \in A \backslash B$ such that $B \cup \{x\}$ is in $\mathcal{I}$.

A *basis* of a matroid is a maximal independent set, which implies it is not contained in any other independent set. Note that, all bases in a matroid have the same cardinality. A *circuit* of matroid is a minimal dependent set. That is, removing any one element from the circuit makes it independent. A *co-circuit* is a minimal set having non-empty intersection with every basis of the matroid.

One example of a matroid is the graphic matroid. Suppose we have some undirected connected graph $G = (V, E)$. Then the graphic matroid $M = (E, \mathcal{I})$ has as ground set the set of edges of G, and the independent sets are the forests in G. In a graphic matroid, a basis is a spanning tree. Therefore, finding a minimum spanning tree in a graph, is equivalent to finding a minimal weight basis in a graphic matroid. Furthermore, circuits in a graphic matroid are equivalent to cycles in the associated graph, and co-circuits are equivalent to bonds in the associated graph.

Next, we introduce connected matroids.

Definition 4.1.1. A matroid $M = (E, \mathcal{I})$ is *connected* if for every pair of distinct elements of E, there is a circuit containing both, (Oxley, 2011).

In terms of graphic matroids, this notation of connectedness is equivalent to a graph being 2-

vertex-connected.

Similar to graphs, contractions are also well-defined on matroids. Given a matroid $M = (E, \mathcal{I})$ and some independent set $L \in \mathcal{I}$. Then, the matroid M/L, which is obtained by contracting all the elements in L from M, has as ground set $E \setminus L$ and the independent sets are the sets $K \subseteq I \backslash L$ such that $K \cup L$ is an independent set in M.

Next, we introduce weights. Similar to graphs, we have a weight function $w : E \to \mathbb{R}$, which draws weight from some distribution and assigns it to the matroid elements. By $\tilde{w}(e)$ we denote one sample of the weight function of $e \in E$.

An algorithm for efficiently solving the problem of finding a minimal weight basis is, again, Kruskal's algorithm. Kruskal's algorithm for matroids is similar to the graph variant. First, the elements are sorted by weight and an empty element set B is initialized. Then, iteratively elements are added to B if the addition of that element does not make B a dependent set. This procedure is guaranteed to terminate and find a minimum weight basis, (Jungnickel, 2012, p. 136).

Next, we introduce algorithm Sam for matroids.

Definition 4.1.2. For SSMMB problem, Sam chooses a basis by applying Kruskal's algorithm for matroids on the sampled weights.

Notice how this definition of Sam is equivalent to the definition of Sam on graphs if we consider graphic matroids.

If we can prove the lemmas which we used on graphs also hold for general matroids, we can use a similar derivation for the performance of Sam on matroids. Lemma 1.3.4 gives a result for weighted graph and we will show that this result holds for general weighted matroids as well. For this lemma, we assume the weights are deterministic.

Lemma 4.1.1. *Given a matroid $M = (E, \mathcal{I})$, suppose we have two minimum weight bases B_1 and B_2. Furthermore, suppose we have an element $e' \in E$, which is not in B_1 or B_2. Then $B_i \cup e'$ contains a circuit C_i, for $i \in \{1, 2\}$. Clearly e' is a maximum weight element on this circuit. By f_i we denote the maximum weight element from B_i on circuit C_i. Then we have $w(f_1) = w(f_2)$.*

Proof. Assume without loss of generality that $w(f_1) \leq w(f_2)$. We introduce new weights $\overline{w}$, for which we let $\overline{w}(e) = w(e)\ \forall e \in E \backslash e'$ and we let $\overline{w}(e') = \min(w(f_1), w(f_2))$. Since we assumed $w(f_1) \leq w(f_2)$, we have $\overline{w}(e') = w(f_1)$. Then B_1 is $\overline{w}$-minimal since exchanging f_1 with e' would not decrease the weight. If any other exchange would decrease the weight, this would be a contradiction with respect to the assumption that B_1 is w-minimal.

Since we have $\overline{w}(B_1) = \overline{w}(B_2)$, we know that T_2 is also $\overline{w}$-minimal. Therefore, there is no element we can exchange to obtain a cheaper basis, which implies $\overline{w}(e') \geq \overline{w}(f_2)$. Since we defined $\overline{w}(e') = \overline{w}(f_1)$, we have $\overline{w}(f_1) \geq \overline{w}(f_2)$. Thus we conclude that $w(f_1) = w(f_2)$. □

4.2 SSMMB problem

For the SSMST problem, the solution space consists of the bases of graphic matroids. Since we would like to extend this result to general matroids, we introduce a problem similar to the SSMST problem, where the solution space consists of the bases of general matroids.

In the Single Sample Minimum Matroid Basis (SSMMB) problem, we are given a matroid with stochastic element weights. The objective is to find a basis with minimal expected weight, while only one sample of the element weights is provided. An instance $I = (M, w)$ of the SSMMB problem consists of a connected matroid $M = (E, \mathcal{I})$ and a stochastic weight function $w : E \to \mathbb{R}$, from which the element weights are sampled. By $\tilde{w}(e)$ we denote a weight sample of element $e \in E$.

Next, we introduce our algorithm of choice Sam for the SSMMB problem.

Definition 4.2.1. On the SSMMB problem, algorithm Sam uses Kruskal's algorithm for matroids to compute a minimum weight basis on the sampled weights.

Notice that this definition of Sam yields the same expected performance on an instance $I = (M, w)$ of the SSMMB problem, where M is a graphic matroid, as on an instance $I' = (G, w)$ of the SSMST problem, where G is the graph associated with matroid M. Furthermore, we would like to evaluate the performance of Sam on the SSMMB problem. Since the SSMST problem is a special case of the SSMMB problem, we can use the performance bound we derived in Chapter 3 as an upper bound on the worst-case performance of Sam.

Exponential weights and notation

In this section, we introduce notation for the SSMMB problem with exponential weights, which is similar to the notation used in Chapter 3. An instance $I = (M, \boldsymbol{\lambda}_E)$ of the SSMMB problem with exponential weights consist of a connected matroid $M = (E, \mathcal{I})$, where each element $e_i \in E$ has an associated rate parameter λ_{e_i}. By $\boldsymbol{\lambda}_E = (\lambda_{e_1}, \ldots, \lambda_{e_m})$ we denote the vector containing all rate parameters of the elements in E, and by Λ_E we denote the family of all possible vectors $\boldsymbol{\lambda}_E$.

The objective is to find the basis with the lowest expected weight. That is, if $\mathbb{B}$ denotes the set of all bases in M, then we want to find a basis B^* for which we have,

$$\mathbb{E}[\tilde{w}(B^*)] = \sum_{e \in B^*} \mathbb{E}[\tilde{w}(e)] \leq \mathbb{E}[\tilde{w}(B)] \quad \forall B \in \mathbb{B}\,.$$

4.3 Relative performance of Sam

We have shown in Chapter 3 that the worst-case relative performance of Sam can be no better than the size of the largest bond in the graph. Furthermore, a bond in a given graph is equivalent to a co-circuit in the associated graphic matroid. Therefore, we know that the worst-case relative performance of Sam on the SSMMB problem can be no better than the size of the largest co-circuit. Therefore, if we can show that the worst-case relative performance of Sam can be no worse than the size of the largest co-circuit, we have a tight bound.

We will derive an upper bound for the performance of Sam on the SSMMB problem in the same manner as we did for the SSMST problem. In order to use the same derivation, we have to prove that all the results which hold for bonds in graphs, also hold for co-circuits in matroids.

For graphs, we restricted ourselves to connected graphs. However, there is no concept for matroids which directly corresponds to the notion of connectedness for graphs, Oxley, 2011, Ch. 4. Therefore, we restrict ourselves to connected matroids, which we defined earlier in this chapter. In terms of graphical matroids, this notion is equivalent to the notion of a graph being 2-vertex-connected.

In the remainder of this section, we will show that the lemmas we used in Chapter 3 also hold for general matroids. Afterwards, we derive an upper bound on the worst-case performance of Sam on the SSMMB problem with exponential weights.

We first prove the matroid equivalent of Lemma 1.3.1. Recall that this lemma stated that the largest bond in G/e cannot be larger than the largest bond in G. What we need to prove, is that the largest co-circuit in M/e cannot be larger than the largest co-circuit in M. We define $CC(M)$ to be a largest co-circuit in matroid $M = (E, \mathcal{I})$.

Lemma 4.3.1. *If $e \in E$ is an element of M. Then, $|CC(M/e)| \leq |CC(M)|$.*

Proof. Suppose to the contrary that $|CC(M/e)| > |CC(M)|$. This implies that $CC(M/e)$ is not a co-circuit in M, but is a co-circuit in M/e. If $CC(M/e)$ is not a co-circuit in M, it is implied that either it is not minimal, or it does not have a non-trivial intersection with every basis.

If $CC(M/e)$ is not minimal in M, there is at least one element in $CC(M/e)$ which needs to be removed such that it is minimal. Since the contraction of e does not alter $CC(M/e)$, it cannot be the case that $CC(M/e)$ is minimal in M/e.

If $CC(M/e)$ does not have a non-trivial intersection with every basis in M, then $CC(M/e) \cup e$ would have a non-trivial intersection with every basis in M. However, this would imply the existence of a larger co-circuit than $CC(M)$, which is also a contradiction. □

The next lemma for which we need a matroid equivalent is Lemma 1.3.2, which states that, if we have a connected graph with more than 2 vertices, there always exists an edge we can contract such that the size of the largest bond does not decrease. The equivalent for connected matroids of this lemma is given as follows.

Lemma 4.3.2. *Given a connected matroid M, there always exists an element $e \in E$ which can be contracted, such that the size of the largest co-circuit does not decrease.*

Proof. By the definition of contraction, the set of bases in M/e is a subset of the set of bases in M. Furthermore, since a co-circuit is defined to have a non-trivial intersection with every basis, and defined to be minimal, it is clear that the size of the largest co-circuit cannot increase via contractions, as the number of bases is non-increasing after contracting. □

The next lemma we need for matroids is Lemma 1.3.3.

Lemma 4.3.3. *Given a connected matroid $M = (E, \mathcal{I})$, the cardinality of E can be bounded by $|CC(M)||B|$, where $|CC(M)|$ is the size of the largest co-circuit and $|B|$ is the cardinality of a basis.*

Proof. Given a basis B, we know it has a non-trivial intersection with every co-circuit. In particular this holds for the largest co-circuit $CC(M)$. In the extreme case, B intersects $|B|$ co-circuits, each of which can have a cardinality of at most $|CC(M)|$. Since we assumed the matroid is connected, every element is contained in a co-circuit (Corollary 4.3.4 in Oxley, 2011). Therefore, $|E| \leq |CC(M)||B|$. □

What is left to show, is that Lemma 3.1.2 and Lemma 3.1.3 hold for matroids.

Lemma 4.3.4.

$$\mathbb{E}[\textsc{Sam}(M, \boldsymbol{\lambda}_M) | \textsc{Ch}(M, e_i, 1)] = \mathbb{E}[\textsc{Sam}(M/e_i, \boldsymbol{\lambda}_M(M/e_i))] + \mathbb{E}[\tilde{w}(e_i, \boldsymbol{\lambda}_M)] \tag{4.1}$$

Proof. Given that Sam will choose some element e_i, we know that any element e_j for which holds that $e_j \cup e_i$ is dependent, cannot be chosen anymore. Therefore, the elements that can still be chosen, are the same elements that can be chosen in M/e_i. Furthermore, in both cases the probability of choosing a certain element are equal because of the memorylessness property of the exponential distribution, just like we argued in the proof of Lemma 3.1.2. Thus, the expected values only differ by the expected value of e_i. □

Lemma 4.3.5.

$$\mathbb{E}[\textit{Opt}(M/e_i, \boldsymbol{\lambda}_M(M/e_i))] = \mathbb{E}[\textit{Opt}(M, \boldsymbol{\lambda}_M)] - \mathbb{E}[\tilde{w}(e_{i^*}, \boldsymbol{\lambda}_M)] \tag{4.2}$$

Proof. If e_i is part of a minimal weight basis, it is not hard to see that this equation holds, since $e_{i^*} = e_i$ in this case.

Now suppose e_i is not part of any minimal weight basis, and suppose B^* is the basis obtained by applying Kruskal's algorithm for matroids on matroid M. We denote by B' the basis obtained by applying Kruskal's algorithm on matroid M/e. Therefore, $\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)] = \mathbb{E}[\tilde{w}(B^*)]$, and $\mathbb{E}[\textsc{Opt}(M/e, \boldsymbol{\lambda}_M(M/e))] = \mathbb{E}[\tilde{w}(B')]$. We will argue that B^* and B' differ by only one element.

To obtain a basis in M which must include e_i, we can modify B^* by exchanging some element e_{i^*} with e_i. It follows from the correctness of Kruskal's algorithm for matroids that this basis is minimal over all bases which include e_i. By construction this basis is therefore also a minimum weight basis on M/e_i, if e_i is also contracted from this basis. Therefore, we have,

$$\mathbb{E}[\textsc{Opt}(M/e_i, \boldsymbol{\lambda}_M(M/e_i))] = \mathbb{E}[\tilde{w}(B^*)] - \mathbb{E}[\tilde{w}(e_{i^*}, \boldsymbol{\lambda}_M)]\,.$$

It follows from Lemma 4.1.1 that this reasoning also holds in case B^* is not the unique minimum weight basis in M. ☐

Now we have all results required to derive an upper bound on the relative performance of Sam on the SSMMB problem.

Theorem 4.3.6. *The relative performance of* Sam *on the SSMMB problem with exponential weights can be no worse than the size of the largest co-circuit. That is,*

$$\alpha(M, \boldsymbol{\lambda}_M) = \frac{\mathbb{E}[\textit{Sam}(M, \boldsymbol{\lambda}_M)]}{\mathbb{E}[\textit{Opt}(M, \boldsymbol{\lambda}_M)]} \leq |CC(M)|\,. \tag{4.3}$$

The derivation of this proof is similar to that of Theorem 3.3.1, but we will write it out for completeness. This proof will be by strong induction. For matroids $M = (E, \mathcal{I})$ with $|E| = 1$, it is clear that $\alpha(M, \boldsymbol{\lambda}_M) = 1 \quad \forall \boldsymbol{\lambda}_M \in \Lambda_M$, since there is only one choice. This is the base case for our induction. Then, our induction hypothesis is as follows. We assume the theorem holds for connected matroids with $|E| = k$. Then we will prove that it also holds for matroids with $|E| = k+1$ elements.

Proof. Suppose we have a matroid $M = (E, \mathcal{I})$ with $|E| = k+1$, and which is connected. Then we can derive worst-case relative performance of Sam.

$$\begin{aligned}\sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \alpha(M, \boldsymbol{\lambda}_M) &= \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\mathbb{E}[\textsc{Sam}(M, \boldsymbol{\lambda}_M)]}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} \\ &= \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \mathbb{P}(\textsc{Ch}(M, e, 1))\mathbb{E}[\textsc{Sam}(M, \boldsymbol{\lambda}_M) | \textsc{Ch}(M, e, 1)]}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]}\,. \end{aligned} \tag{4.4}$$

By Lemma 4.3.4 , equation 4.4 equals,

$$= \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \mathbb{P}(\textsc{Ch}(M, e, 1))(\mathbb{E}[\textsc{Sam}(M/e, \boldsymbol{\lambda}_M(M/e))] + 1/\lambda_e)}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]}\,. \tag{4.5}$$

By the definition of α, equation 4.5 equals

$$= \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \mathbb{P}(\textsc{Ch}(M, e, 1))(\alpha(M/e, \boldsymbol{\lambda}_M(M/e))\mathbb{E}[\textsc{Opt}(M/e, \boldsymbol{\lambda}_M(M/e))] + 1/\lambda_e)}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]}\,.$$

Applying the induction hypothesis gives us,

$$\sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \alpha(M, \boldsymbol{\lambda}_M) \leq \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \mathbb{P}(\textsc{Ch}(M, e, 1))(|CC(M/e)|\mathbb{E}[\textsc{Opt}(M/e, \boldsymbol{\lambda}_M(M/e))] + \frac{1}{\lambda_e})}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} .$$

By Lemma 4.3.2 we have,

$$\begin{aligned}\sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \alpha(M, \boldsymbol{\lambda}_M) &\leq \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \mathbb{P}(\textsc{Ch}(M, e, 1))(|CC(M)|\mathbb{E}[\textsc{Opt}(M/e, \boldsymbol{\lambda}_M(M/e))] + \frac{1}{\lambda_e})}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} \\ &= |CC(M)| \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \mathbb{P}(\textsc{Ch}(M, e, 1))(\mathbb{E}[\textsc{Opt}(M/e, \boldsymbol{\lambda}_M(M/e))] + \frac{1}{|CC(M)|\lambda_e})}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} .\end{aligned}$$

Writing out the probability gives,

$$\sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \alpha(M, \boldsymbol{\lambda}_M) = |CC(M)| \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \frac{\lambda_e}{\sum_{e_k \in E} \lambda_{e_k}}(\mathbb{E}[\textsc{Opt}(M/e, \boldsymbol{\lambda}_M(M/e))] + \frac{1}{|CC(M)|\lambda_e})}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} .$$

Next, we apply Lemma 4.3.5 to obtain,

$$\begin{aligned}\sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \alpha(M, \boldsymbol{\lambda}_M) &= |CC(M)| \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\sum_{e \in E} \frac{\lambda_e}{\sum_{e \in E} \lambda_e}(\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)] - \frac{1}{\lambda_{e^*}} + \frac{1}{|CC(M)|\lambda_e})}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} \\ &= |CC(M)| \sup_{\boldsymbol{\lambda}_M \in \Lambda_M} \frac{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)] + \sum_{e \in E} \frac{1}{\sum_{e \in E} \lambda_e}(-\frac{\lambda_e}{\lambda_{e^*}} + \frac{1}{|CC(M)|})}{\mathbb{E}[\textsc{Opt}(M, \boldsymbol{\lambda}_M)]} .\end{aligned}$$

If we show that $\sum_{e \in E} \frac{1}{\sum_{e \in E} \lambda_e}(-\frac{\lambda_e}{\lambda_{e^*}} + \frac{1}{|CC(M)|}) \leq 0$, we have the desired result. This is equivalent to showing that,

$$\sum_{e \in E} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} \leq 0 . \tag{4.6}$$

Writing out the left hand side of equation 4.6 gives,

$$\sum_{e \in E} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} = \sum_{e \in B^*} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} + \sum_{e \notin B^*} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} .$$

If $e \in B^*$, we have $\lambda_e = \lambda_{e^*}$, which yields,

$$\begin{aligned}\sum_{e \in E} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} &= \frac{|B^*|}{|CC(M)|} - |B^*| + \sum_{e \notin B^*} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} \\ &\leq \frac{|B^*|}{|CC(M)|} - |B^*| + \sum_{e \notin B^*} \frac{1}{|CC(M)|} .\end{aligned}$$

By Lemma 1.3.3, we have that $|E| \leq |B^*||CC(M)|$. Therefore we have,

$$\begin{aligned}\sum_{e\in E} \frac{1}{|CC(M)|} - \frac{\lambda_e}{\lambda_{e^*}} &\leq \frac{|B^*|}{|CC(M)|} - |B^*| + \frac{|CC(M)||B^*| - |B^*|}{|CC(M)|} \\ &= \frac{|B^*|}{|CC(M)|} - |B^*| + \frac{|CC(M) - 1||B^*|}{|CC(M)|} \\ &= \frac{|CC(M)||B^*|}{|CC(M)|} - |B^*| \\ &= |B^*| - |B^*| = 0\,.\end{aligned}$$

Thus, we conclude,

$$\begin{aligned}\sup_{\boldsymbol{\lambda}_M\in\Lambda_M} \alpha(M,\boldsymbol{\lambda}_M) &\leq |CC(M)| \sup_{\boldsymbol{\lambda}_M\in\Lambda_M} \frac{\mathbb{E}[\textsc{Opt}(M,\boldsymbol{\lambda}_M)] + \sum_{e\in E}\frac{1}{\sum_{e\in E}\lambda_e}(-\frac{\lambda_e}{\lambda_{e^*}} + \frac{1}{|CC(M)|})}{\mathbb{E}[\textsc{Opt}(M,\boldsymbol{\lambda})]} \\ &\leq |CC(M)| \sup_{\boldsymbol{\lambda}_M\in\Lambda_M} \frac{\mathbb{E}[\textsc{Opt}(M,\boldsymbol{\lambda}_M)]}{\mathbb{E}[\textsc{Opt}(M,\boldsymbol{\lambda}_M)]} \\ &= |CC(M)|\,.\end{aligned}$$

□

Thus, we have shown the relative performance of Sam on the SSMMB problem with exponential weights is bounded by the size of the largest co-circuit in the matroid. Furthermore, we have shown that this bound is tight.

Chapter 5

Maximizing instead of minimizing

In this chapter, we consider the the single sample item selection problem. However, instead of choosing the item with the lowest cost, we want to choose the item which yields the largest profit. We give some intuition on how maximizing is fundamentally different from minimizing, and how this affects the relative performance of Sam, in this chapter the greedy algorithm which chooses the item with the largest sampled weight. Furthermore, we conjecture under which conditions the worst-case performance of Sam is achieved when the item weights are exponentially distributed.

5.1 Single Sample Maximum Item Selection problem

We start this section by introducing the counter-part of the SSMIS problem; the Single Sample Maximum Item Selection problem, or SSMaxIS for short.

An instance $I = (N, w)$ of the SSMaxIS problem consist of a set of m items $N = (i_1, i_2, \ldots, i_m)$, and a stochastic weight function $w : N \to \mathbb{R}$, from which weights are independently drawn for each item. By $\tilde{w}(i_j)$ we denote one weight sample of item j. In contrast to the SSMIS problem, the objective is to choose the item with the *highest* expected weight. By the definition of the non-adaptive optimum Opt, we have $\mathbb{E}[\textsc{Opt}(I, w)] = \max_{i_k \in N} \mathbb{E}[\tilde{w}(i_k)]$.

Definition 5.1.1. On the SSMaxIS problem, algorithm Sam chooses the item with the most expensive weight sample.

Suppose we have an instance $I = (N, w)$ of the SSMIS problem, where instead of minimizing, the objective is to choose the item with in expectation the largest weight. We will first discuss some intuition on how changing the objective from minimizing to maximizing influences the relative performance of Sam compared to Opt, which in this case is the item with the largest expected weight. For minimizing, choosing a “wrong” item, i.e., an item with larger than minimal expected weight, can have a large influence of the relative performance of Sam. The expected value of the item with minimum expected weight can be very small, the expected value of a “wrong” item can be relatively large. Specifically, the ratio between the expected weight of a “wrong” item and the optimal item can be anywhere between 1 and infinity. Even if the chance of choosing an item with a large expected weight is small, it can still have a significant impact on the relative performance of Sam. In contrast, when the objective is to maximize, choosing a “wrong” item is less detrimental. The ratio between the expected weight of a “wrong” item and the expected weight of the optimal item lies between 0 and 1 in this case. If there is a chance that a sub-optimal item will be chosen, it will have relatively little influence on the relative performance of Sam. We will give an example to show what the influence of both situations is on the relative performance α.

Example 5.1.1. Suppose we have an instance I of the SSMIS problem with two items. Item 1 has an expected value of 2 and item 2 has an expected value of 20. Furthermore the probability that item 1 gives a smaller weight sample than item 2 is 0.9. This gives us the following relative performance for SAM.

$$\begin{aligned}\alpha(I) &= \frac{\mathbb{E}[\textsc{Sam}(I)]}{\mathbb{E}[\textsc{Opt}(I)]} \\ &= \frac{0.9 \cdot 2 + 0.1 \cdot 20}{2} \\ &= \frac{3.8}{2}\end{aligned}$$

Now, suppose instead the objective is to maximize. Then, we have the following.

$$\begin{aligned}\alpha(I) &= \frac{\mathbb{E}[\textsc{Sam}(I)]}{\mathbb{E}[\textsc{Opt}(I)]} \\ &= \frac{0.9 \cdot 20 + 0.1 \cdot 2}{20} \\ &= \frac{18.2}{20}\end{aligned}$$

In the first case, an α of 1.9 is achieved, while in the second case, we have an α of $\frac{18.2}{20} \approx \frac{1}{1.1}$. This example shows that the relative performance of SAM on the maximization instance is not inversely related to the performance of SAM on a minimization instance with the same weight function.

In the remainder of this section, we investigate the worst case performance of SAM on the maximization version of the item selection problem with exponential weights. We first discuss instances with 2 and 3 items. Afterwards we conjecture how the parameters for the exponential distribution should be distributed such that a minimal α is achieved.

Lemma 5.1.1. *Given an instance of the SSMaxIS problem consisting of two items with exponentially distributed weights, the worst-case relative performance of SAM is achieved when one item's weight distribution has some parameter λ, and the other item's weight distribution has rate parameter $(1+\sqrt{2})\lambda$.*

Proof. Suppose we have an instance $J = (I, \boldsymbol{\lambda}_I)$ with 2 items, I_1 and I_2. Without loss of generality, we assume I_1 has rate λ and I_2 has rate $x\lambda$, where $x \geq 1$. With these parameters, item 1 has a larger expected weight than item 2. The expected weights are given by,

$$\mathbb{E}[\tilde{w}(I_1)] = \frac{1}{\lambda} \geq \frac{1}{x\lambda} = \mathbb{E}[\tilde{w}(I_2)]\,.$$

Since we only consider two items, it is straightforward to calculate the relative performance of SAM on this instance. Recall how we can use equation 2.2.1 to determine the probability of an item being chosen by SAM.

$$\begin{aligned}
\alpha(I, \boldsymbol{\lambda}_I) &= \frac{\mathbb{E}[\textsc{Sam}(I, \boldsymbol{\lambda}_I)]}{\mathbb{E}[\textsc{Opt}(I, \boldsymbol{\lambda}_I)]} \\
&= \frac{\mathbb{P}(\text{choose } I_1)\mathbb{E}[\tilde{w}(I_1)] + \mathbb{P}(\text{choose } I_2)\mathbb{E}[\tilde{w}(I_2)]}{\frac{1}{\lambda}} \\
&= \lambda\left(\frac{x\lambda}{\lambda + x\lambda}\frac{1}{\lambda} + \frac{\lambda}{\lambda + x\lambda}\frac{1}{x\lambda}\right) \\
&= \frac{x}{1+x} + \frac{1}{1+x}\frac{1}{x} \\
&= \frac{x^2+1}{x(1+x)}
\end{aligned}$$

.

Notice that the expression we derived for α only depends on x. Now, to find the worst case performance of SAM, we can take the derivative of this function and find its roots.

$$\frac{\mathrm{d}\alpha(I, \boldsymbol{\lambda}_I)}{\mathrm{d}x} = \frac{x^2 - 2x - 1}{x^2(x+1)^2} \tag{5.1}$$

The only root of equation 5.1 on the domain $x \geq 1$ is at $x = 1 + \sqrt{2}$. At this point, α achieves its minimum value of $2(\sqrt{2} - 1) \approx 0.83$. □

Lemma 5.1.2. *Given an instance of the SSMaxIS problem consisting of three items with exponentially distributed weights, the worst-case relative performance of* SAM *is achieved when one item's weight distribution has some parameter λ, and the other items' weight distributions have rate parameter $(1+\sqrt{2})\lambda$.*

Proof. We prove this lemma by first showing algebraically that a minimum of the relative performance of SAM, α, can only be attained in the case where one weight function has some rate parameter λ, and the other two weight function have the same rate parameter. Afterwards, we can numerically derive a minimum.

Suppose we have an instance $J = (I, \boldsymbol{\lambda}_I)$ with 3 items. Similar to the previous case, we let item 1 have rate λ, and without loss of generality, we let item 2 have rate $x\lambda$ and item 3 $y\lambda$, with $y \geq x \geq 1$. Similar to the previous lemma, we can algebraically derive the relative performance of SAM on this instance.

$$\begin{aligned}
\alpha(I, \boldsymbol{\lambda}_I) &= \frac{\mathbb{E}[\textsc{Sam}(I, \boldsymbol{\lambda}_I)]}{\mathbb{E}[\textsc{Opt}(I, \boldsymbol{\lambda}_I)]} \\
&= \lambda \mathbb{E}[\textsc{Sam}(I, \boldsymbol{\lambda}_I)] \\
&= \lambda \sum_{i=1}^{3} \mathbb{P}(\text{choose } I_i)\mathbb{E}[\tilde{w}(I_i)] \\
&= \lambda\Big(\frac{\lambda}{(1+x+y)\lambda}\frac{x\lambda}{(x+y)\lambda}\frac{1}{\lambda y} + \frac{\lambda}{(1+x+y)\lambda}\frac{y\lambda}{(x+y)\lambda}\frac{1}{\lambda x} \\
&+ \frac{x\lambda}{(1+x+y)\lambda}\frac{y\lambda}{(1+y)\lambda}\frac{1}{\lambda} + \frac{x\lambda}{(1+x+y)\lambda}\frac{\lambda}{(1+y)\lambda}\frac{1}{y\lambda} \\
&+ \frac{y\lambda}{(1+x+y)\lambda}\frac{x\lambda}{(1+x)\lambda}\frac{1}{\lambda} + \frac{y\lambda}{(1+x+y)\lambda}\frac{\lambda}{(1+x)\lambda}\frac{1}{x\lambda}\Big) \\
&= \frac{xy + \frac{x}{y}}{(1+x+y)(1+y)} + \frac{xy + \frac{y}{x}}{(1+x+y)(1+x)} + \frac{\frac{y}{x} + \frac{x}{y}}{(1+x+y)(x+y)}
\end{aligned}$$

Notice how this last expression only depends on x and y, and not on λ. To find the point at which the minimum value is attained, we use the gradient to find critical points.

$$\nabla\alpha(I) = \begin{pmatrix} \frac{\partial\alpha(I)}{\partial x} \\ \frac{\partial\alpha(I)}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{2}{(x+y)^2} - \frac{3}{(x+y+1)^2} - \frac{1}{x^2} + \frac{2}{(x+1)^2} \\ \frac{2}{(x+y)^2} - \frac{3}{(x+y+1)^2} - \frac{1}{y^2} + \frac{2}{(y+1)^2} \end{pmatrix}$$

Critical points are achieved when both $\frac{\partial\alpha(I)}{\partial x}$ and $\frac{\partial\alpha(I)}{\partial y}$ are equal to zero. Therefore, $\frac{\partial\alpha(I)}{\partial x} - \frac{\partial\alpha(I)}{\partial y} = 0$ is a necessary condition for a critical point.

$$\begin{aligned}
\frac{\partial\alpha(I)}{\partial x} - \frac{\partial\alpha(I)}{\partial y} &= 0 \\
-\frac{1}{x^2} + \frac{2}{(x+1)^2} + \frac{1}{y^2} - \frac{2}{(y+1)^2} &= 0 \\
\frac{1}{y^2} - \frac{2}{(y+1)^2} &= \frac{1}{x^2} - \frac{2}{(x+1)^2}\,. \qquad (5.2)
\end{aligned}$$

For convenience, we let $f(x) = \frac{1}{x^2} - \frac{2}{(x+1)^2}$.

Equation 5.2 is satisfied when $x = y$, but other solutions might also be possible. That is, solutions for which we have $f(x) = f(y)$ for $y > x$. We investigate this possibility by determining the slope and critical points of $f(x)$.

The derivative of $f(x)$ is given by $f'(x) = \frac{4}{(x+1)^3} - \frac{2}{x^3}$. The only root on the domain $x \geq 1$ is at $x = 1 + \sqrt[3]{2} + \sqrt[3]{4}$. Furthermore, we note that $f'(x) < 0$ for $1 \leq x < 1 + \sqrt[3]{2} + \sqrt[3]{4}$ and $f'(x) > 0$ for $x > 1 + \sqrt[3]{2} + \sqrt[3]{4}$. Since we assumed that $x \leq y$, a minimum where $x \neq y$ can only be achieved when $1 \leq x < 1 + \sqrt[3]{2} + \sqrt[3]{4}$ and $y > 1 + \sqrt[3]{2} + \sqrt[3]{4}$. Now, it is important that critical points of α exist on this domain. However, when considering $\frac{\partial\alpha(I)}{\partial y}$, we numerically find that it has no roots on the domain $y > 1 + \sqrt[3]{2} + \sqrt[3]{4}$. Therefore a minimum value for α can only be attained in the case where $x = y$. Specifically, we numerically find that the minimum of α is achieved when $x = y \approx 2.502$. □

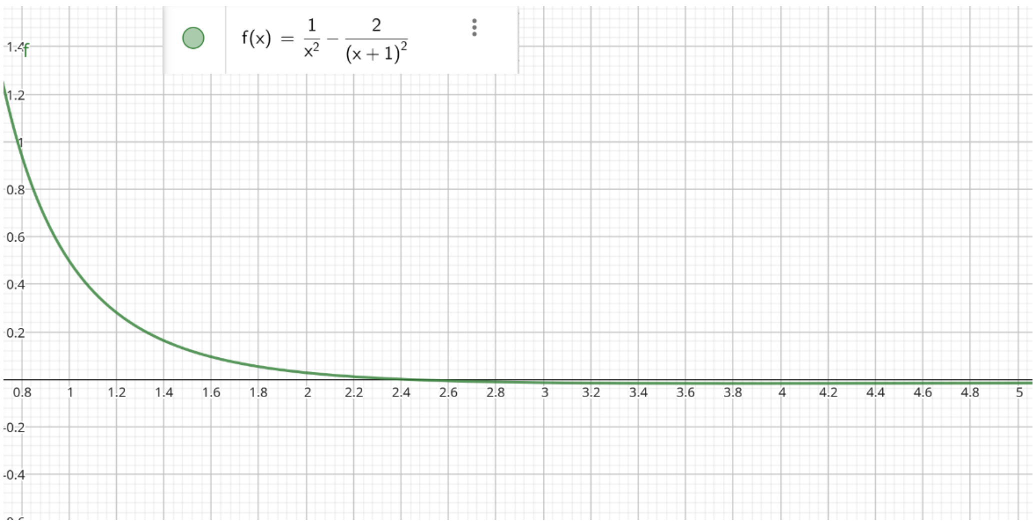


Figure 5.1: Plot of $f(x)$

For four or more items, an analysis similar to the analysis of the previous two instances becomes quite cumbersome. For n items, α will depend on $n-1$ variables. Furthermore, there is no trivial argument which proves that the worst case is achieved when all $n-1$ variables take on the same value. However, since α is a symmetric function, it intuitively makes sense that a minimum value of α is achieved when the $n-1$ variables all take on the same value. Therefore, we introduce a conjecture.

Conjecture 5.1.3. *For the maximization version of the SSMIS problem with exponential weights, the worst-case relative performance of* Sam *is achieved when an instance with m items consists of one item with rate λ, and all other $m-1$ items have a rate parameter of $x\lambda$, with $x \geq 1$.*

In the remainder of this chapter, we investigate the worst-case relative performance of Sam, assuming Conjecture 5.1.3 is true. Even if the conjecture is proven to be wrong, at least we can give a bound on the worst-case performance. That is, the worst-case relative performance of Sam can be no better than the bound we give. Thus, suppose we have an instance with m items, of which one has rate λ, and the $m-1$ other items have rate $x\lambda$, where $x > 1$. Then the probability that Sam chooses the item with rate λ, i.e. the "right" item since it has the largest expected weight, is the same probability that in a given sample, that item has the largest sampled weight. We can calculate this probability using equation 3.2.

$$\mathbb{P}(\text{choose "right"}) = (m-1)! \frac{x\lambda}{(1+(m-1)x)\lambda} \frac{x\lambda}{(1+(m-2)x)\lambda} \cdots \frac{x\lambda}{(1+x)\lambda} \frac{\lambda}{\lambda} \tag{5.3}$$

There are $(m-1)!$ combinations of orderings which result in the item with the largest expected weight having the largest sampled weight. Since all other choices are "wrong", the chance of choosing "wrong" is just one minus the chance of choosing "right".

Using this insight, we can calculate the relative performance of Sam.

$$
\begin{aligned}
\alpha(I, \boldsymbol{\lambda}_I) &= \frac{\mathbb{E}[\textsc{Sam}(I, \boldsymbol{\lambda}_I)]}{\mathbb{E}[\textsc{Opt}(I, \boldsymbol{\lambda}_I)]} \\
&= \lambda(\mathbb{P}(\text{choose "right"})\frac{1}{\lambda} + \mathbb{P}(\text{choose "wrong"})\frac{1}{x\lambda}) \\
&= \frac{(m-1)!x^{m-1} - (m-1)!x^{m-2}}{\prod_{k=1}^{m-1}(1+kx)} + \frac{1}{x}
\end{aligned}
$$

For large m, this formula is not easy to work out by hand. However, since this formula only depends on x after an m is chosen, finding minimum values for $\alpha(I, \boldsymbol{\lambda}_I)$ is numerically possible, even for large m.

In Figure 5.2 we visualised the worst-case relative performance for $m = 2$ up to 13 items, which we derived numerically.

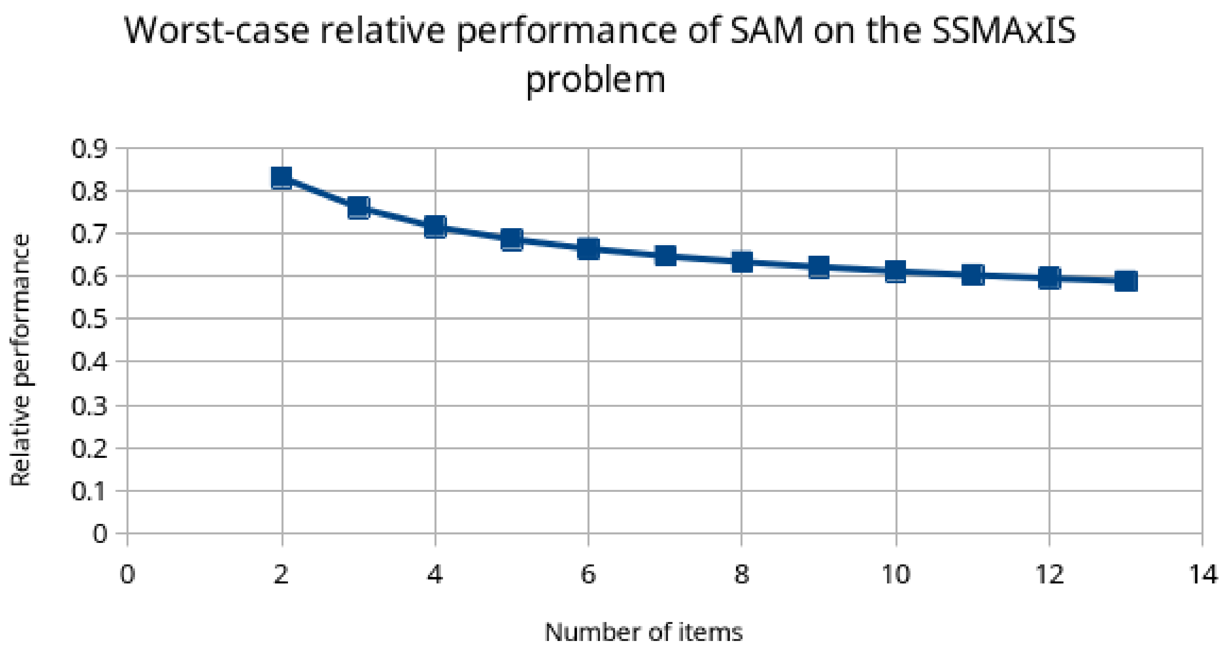


Figure 5.2: Plot showing the worst-case relative performance of Sam on the SSMaxIS problem.

Discussion and conclusion

In this thesis, we discuss an adaptation of the classic minimum spanning tree problem. In this adaptation the edges have stochastic weights instead of deterministic weights. Furthermore, only one sample of each edge's weight function is provided. The objective is to compute a spanning tree with in expectation minimal weight. We denoted this problem as the SSMST problem. Given these restrictions, perhaps the only reasonable strategy is to use the sampled weights as a proxy for the expected weights. We refer to this algorithm as SAM. The main goal of this thesis is to identify for which weight distributions the performance of SAM can be bounded.

In Chapter 2 we introduce an item selection problem as a special case of the SSMST problem. This enables us to evaluate the worst-case performance of SAM on certain classes of distributions. We give examples of distributions for which the relative performance of SAM can be unbounded. For the item selection problem with exponentially distributed weights we found that the performance of SAM can be bounded by the number of items.

In Chapter 3 we discuss the SSMST problem with exponential weights. We show that the performance of SAM is bounded by the size of the largest bond in the given graph. Furthermore, we prove that this bound is tight. This proves makes use of several results related to graphs and bonds, which we introduce and prove in Chapter 1.

In Chapter 4, we generalize the result we derived for the SSMST problem to general matroids. On the SSMST problem, the solution space consists of the bases of graphic matroids. We generalize this to a problem of which the solution space consists of the bases of general connected matroids. We show that on instances of this problem with exponential weights, the performance of SAM is bounded by the size of the largest co-circuit. The notion of connectedness for matroids is more restrictive than the notion of connectedness for graphs, as a connected graphic matroid corresponds to a 2-vertex-connected graph.

The results we derived are limited to instances where the weights are exponentially distributed. The examples for which the performance of SAM is unbounded, which we show in Chapter 2, rely mainly on the fact that non-positive weights are allowed. Therefore, for future research it would be interesting to see whether the performance of SAM can be bounded for all weight distributions with strictly positive weights.

The scheduling paper by (te Rietmole and Uetz, 2024) uses the random adversary, which chooses a scheduling uniformly at random, to compare the performance of SAM to. One could explore the performance of SAM on the SSMST problem relative to a random adversary, which picks any spanning tree uniformly at random for instance. With this adversary, the distributions which are well-behaved on the scheduling problem can be reassessed on the SSMST problem, to see whether the relative performance of SAM can be bounded compared to the performance of the random adversary. Furthermore, by comparing directly to the non-adaptive optimum, a division by zero might occur if the weight distributions allow for non-positive weights. The random adversary can be a reasonable alternative for such instances.

In Chapter 5 we discussed the stochastic item selection problem with a single sample where the objective is to choose the item with the largest expected weight, instead of the smallest weight like in the chapters beforehand. For this problem we also restricted our scope to exponential weights. The analysis of this problem is far from done. We introduced a conjecture which states that the worst-case performance of SAM is achieved when all but one of the rate parameters are equal. Besides a proof of this conjecture, it would be interesting to see how the maximization version SSMST problem behaves.

Bibliography

Borůvka, O. (1926). O jistém problému minimálním (on a certain minimal problem). Práce moravské přirodovědecké společnosti 3 (1926), 37-58.

Duarte, G. L., Lokshtanov, D., Pedrosa, L. L. C., Schouery, R. C. S., & Souza, U. S. (2019). Computing the Largest Bond of a Graph. In B. M. P. Jansen & J. A. Telle (Eds.), *14th international symposium on parameterized and exact computation (ipec 2019)* (12:1–12:15, Vol. 148). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.IPEC.2019.12
Keywords: bond, cut, maximum cut, connected cut, FPT, treewidth, clique-width.

Hoffmann, M., Erlebach, T., Krizanc, D., Mihal'ák, M., & Raman, R. (2008). Computing Minimum Spanning Trees with Uncertainty. In S. Albers & P. Weil (Eds.), *25th international symposium on theoretical aspects of computer science* (pp. 277–288, Vol. 1). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.STACS.2008.1358
Keywords: Algorithms and data structures; Current challenges: mobile and net computing.

Ibe, O. (2014). *Fundamentals of applied probability and random processes, second edition* (2nd ed.). Elsevier Science. https://doi.org/10.1016/C2013-0-19171-4

Jungnickel, D. (2012). *Graphs, networks and algorithms*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-32278-5

Kamousi, P., & Suri, S. (2011). Stochastic minimum spanning trees and related problems. *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*, 107–116.

Korte, B., & Vygen, J. (2007). *Combinatorial optimization: Theory and algorithms* (4th). Springer Publishing Company, Incorporated.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, *7*(1), 48–50. https://doi.org/10.1090/S0002-9939-1956-0078686-7

Oxley, J. (2011, February). *Matroid Theory*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198566946.001.0001

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, *36*(6), 1389–1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x

te Rietmole, P., & Uetz, M. (2024). Sequencing stochastic jobs with a single sample. In A. Basu, A. R. Mahjoub, & J. J. Salazar González (Eds.), *Combinatorial optimization* (pp. 235–247). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-60924-4_18

Appendices

A.1 Proof of exponential minimum

$$\begin{aligned} P(\min_i \tilde{w}(e_i) = \tilde{w}(e_k)) &= \int_{x=0}^{\infty} P(\tilde{w}(e_k) = x) P(\forall_{i \neq k} \tilde{w}(e_i) > x) dx \\ &= \int_{x=0}^{\infty} \lambda_k e^{-\lambda_k x} (\prod_{i=1, i \neq k}^{n} e^{-\lambda_i x}) dx \\ &= \lambda_k \int_{x=0}^{\infty} e^{-x \sum_{i=1}^{n} \lambda_i} dx \\ &= \frac{\lambda_k}{\sum_{i=1}^{n} \lambda_i} \end{aligned} \tag{4}$$