Robust Algorithms under Adversarial Injections*

Paritosh Garg
EPFL, Switzerland
paritosh.garg@epfl.ch

Sagar Kale
University of Vienna, Austria
sagar.kale@univie.ac.at

Lars Rohwedder
EPFL, Switzerland
lars.rohwedder@epfl.ch

Ola Svensson
EPFL, Switzerland
ola.svensson@epfl.ch

Abstract

In this paper, we study streaming and online algorithms in the context of randomness in the input. For several problems, a random order of the input sequence—as opposed to the worst-case order—appears to be a necessary evil in order to prove satisfying guarantees. However, algorithmic techniques that work under this assumption tend to be vulnerable to even small changes in the distribution. For this reason, we propose a new *adversarial injections* model, in which the input is ordered randomly, but an adversary may inject misleading elements at arbitrary positions. We believe that studying algorithms under this much weaker assumption can lead to new insights and, in particular, more robust algorithms. We investigate two classical combinatorial-optimization problems in this model: Maximum matching and cardinality constrained monotone submodular function maximization. Our main technical contribution is a novel streaming algorithm for the latter that computes a 0.55-approximation. While the algorithm itself is clean and simple, an involved analysis shows that it emulates a subdivision of the input stream which can be used to greatly limit the power of the adversary.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Adversary models

Keywords and phrases Streaming algorithm, adversary, submodular maximization, matching

* Research supported in part by the Swiss National Science Foundation project 200021-184656 "Randomness in Problem Instances and Randomized Algorithms."

1 Introduction

In the streaming model, an algorithm reads the input sequentially from the input stream while using limited memory. In particular, the algorithm is expected to use memory that is much smaller than the input size, ideally, linear in the size of a solution. We consider the most fundamental setting in which the algorithm is further restricted to only read the input stream once. In this case, the algorithm cannot remember much of the input along the way, and part of the input is irrevocably lost. Something similar happens for online algorithms: Here, the input is given to the algorithm one element at a time and the algorithm has to decide whether to take it into its solution or to discard it. This decision is irrevocable.

The most common approach to analyze the quality of an algorithm in these models is worst-case analysis. Here, an adversary has full knowledge of the algorithm's strategy and presents a carefully crafted instance to it, trying to make the ratio between the value of the algorithm's solution and that of an optimum solution (the approximation ratio; for online algorithms called the competitive ratio) as small as possible[1]. While worst-case analysis gives very robust guarantees, it is also well-known that such an analysis is often very pessimistic. Not only are good guarantees not possible for many problems, but in many cases worst-case instances appear quite artificial. Hence, the worst-case approximation/competitive ratio does not necessarily represent the quantity that we want to optimize.

One way to remedy this is to weaken the power of the adversary and a popular model to achieve that is the random-order model. Here, an adversary may pick the instance as before, but it is presented in a uniformly-random order to the algorithm. This often allows for significantly better provable guarantees. A prime example is the secretary problem: For the worst-case order it is impossible to get a bounded competitive ratio whereas for the random-order a very simple stopping rule achieves a competitive ratio of $1/e$. Unfortunately, in this model, algorithms tend to overfit and the assumption of a uniformly-random permutation of the input is a strong one. To illustrate this point, it is instructive to consider two examples of techniques that break apart when the random-order assumption is slightly weakened:

Several algorithms in the random-order model first read a small fraction of the input, say, the first 1% of the input. Such an algorithm relies on the assumption that around 1% of the elements from an optimum solution are contained in this first chunk. It computes some statistics, summaries, or initial solutions using this chunk in order to estimate certain properties of the optimum solution. Then in the remaining 99% of the input it uses this knowledge to build a good solution for the problem. For examples of such streaming algorithms, see Norouzi-Fard et al. [29] who study submodular maximization and Gamlath et al. [13] who study maximum matching. Also Guruganesh and Singla's [16] online algorithm for maximum matching for bipartite graphs is of this kind. Note that these algorithms are very sensitive to noise at the beginning of the stream.

Another common technique is to split the input into fixed parts and exploit that with high probability the elements of the optimum solution are distributed evenly among these parts, e.g., each part has at most one optimum element. These methods critically rely on the assumption that each part is representative for the whole input or that the parts are in some way homogeneous (properties of the parts are the same in expectation). Examples of such algorithms include the streaming algorithm for maximum matching [23], and the streaming algorithm for submodular maximization [1] that achieves the tight competitive ratio $1 - 1/e$ in the random-order model.

[1] We assume that the problem is a maximization problem.

The motivation of this work is to understand whether the strong assumption of uniformly-random order is necessary to allow for better algorithms. More specifically, we are motivated by the following question:

> Can we achieve the same guarantees as in the uniform-random order but by algorithms that are more robust against some distortions in the input?

In the next subsection, we describe our proposed model that is defined so as to avoid overfitting to the random-order model, and, by working in this model, our algorithms for submodular maximization and maximum matching are more robust while maintaining good guarantees.

1.1 The Adversarial Injections Model

Our model—that we call the *adversarial-injections* model—lies in between the two extremes of random-order and adversarial-order. In this model, the input elements are divided into two sets E_{NOISE} and E_{GOOD}. An adversary first picks all elements, puts each element in either E_{NOISE} or E_{GOOD}, and chooses the input order. Then the elements belonging to E_{GOOD} are permuted uniformly at random among themselves. The algorithm does not know if an element is good or noise. We judge the quality of the solution produced by an algorithm by comparing it to the best solution in E_{GOOD}.

An equivalent description of the model is as follows. First, a set of elements is picked by the adversary and is permuted randomly. Then, the adversary injects more elements at positions of his choice without knowing the random permutation of the original stream[2]. Comparing with the previous definition, the elements injected by the adversary correspond to E_{NOISE} and the elements of the original stream correspond to E_{GOOD}.

We denote by $E_{\text{OPT}} \subseteq E_{\text{GOOD}}$ the elements of a fixed optimum solution of the elements in E_{GOOD}. We can assume without loss of generality that $E_{\text{GOOD}} = E_{\text{OPT}}$, because otherwise elements in $E_{\text{GOOD}} \setminus E_{\text{OPT}}$ can be treated as those belonging to E_{NOISE} (which only strengthens the power of the adversary).

1.2 Related Models

With a similar motivation, Kesselheim, Kleinberg, Niazadeh [21] studied the robustness of algorithms for the secretary problem from a slightly different perspective: They considered the case when the order of the elements is not guaranteed to be uniformly-at-random but still contains "enough" randomness with respect to different notions such as entropy. Recently, Esfandiari, Korula, Mirrokni [8] introduced a model where the input is a combination of stochastic input that is picked from a distribution and adversarially ordered input. Our model is different in the sense that the input is a combination of randomly ordered (instead of stochastic input) and adversarially ordered elements.

Two models that are more similar to ours in the sense that the input is initially ordered in a uniformly-random order and then scrambled by an adversary in a limited way are [15]

[2] We remark that the assumption that the adversary does not know the order of the elements is important. Otherwise, the model is equivalent to the adversarial order model for "symmetric" problems such as the matching problem. To see this, let E_{OPT} correspond to an optimum matching in any hard instance under the adversarial order. Since a matching is symmetric, the adversary can inject appropriately renamed edges depending on the order of the edges (which he without this assumption knows) and obtain exactly the hard instance.

and [3]. First, in the streaming model, Guha and McGregor [15] introduced the notion of a t-bounded adversary that can disturb a uniformly-random stream but has memory to remember and delay at most t input elements at a time. Second, Bradac et al. [3] very recently introduced a new model that they used to obtain robust online algorithms for the secretary problem. Their model, called the *Byzantine* model, is very related to ours: the input is split into two sets which exactly correspond to E_{GOOD} and E_{NOISE} in the adversarial-injections model. The adversary gets to pick the elements in both of them, but an algorithm will be compared against only E_{GOOD}. Then—this is where our models differ—the adversary chooses an arrival time in $[0, 1]$ for each element in E_{NOISE}. He has no control over the arrival times of the elements in E_{GOOD}, which are chosen independently and uniformly at random in $[0, 1]$. The algorithm does not know to which set an element belongs, but it knows the timestamp of each element, as the element arrives. While the Byzantine model prevents certain kinds of overfitting (e.g., of the classical algorithm for the secretary problem), it does not tackle the issues of the two algorithmic techniques we discussed earlier: Indeed, by time $t = 0.01$, we will see around 1% of the elements from E_{OPT}. Hence, we can still compute some estimates based on them, but do not lose a lot when dismissing them. Likewise, we may partition the timeline, and thereby the input, into parts such that in each part at most one element of E_{OPT} appears.

Hence, even if our model appears very similar to the Byzantine model, there is this subtle, yet crucial, difference. The adversarial-injections model does not add the additional dimension of time, and hence, does not allow for the kind of overfitting that we discussed earlier. To further emphasize this difference, we now describe why it is strictly harder to devise algorithms in the adversarial-injections model compared to the Byzantine model. It is at least as hard as the Byzantine model, because any algorithm for the former also works for the latter. This holds because the adversarial-injections model can be thought of as the Byzantine model with additional power to the adversary and reduced power for the algorithm: The adversary gets the additional power of setting the timestamps of elements in E_{GOOD}, but not their identities, whereas the algorithm is not allowed to see the timestamp of any element.

To show that it is strictly harder, consider online bipartite matching. We show that one cannot beat 1/2 in the adversarial-injections model (for further details, see Section 2.2) whereas we observe that the $(1/2 + \delta)$-approximation algorithm [16] for bipartite graphs and its analysis generalizes to the Byzantine model as well. This turns out to be the case because the algorithm in [16] runs a greedy algorithm on the first small fraction, say 1% of the input and "augments" this solution using the remaining 99% of the input. The analysis crucially uses the fact that 99% of the optimum elements are yet to arrive in the augmentation phase. This can be simulated in the Byzantine model using timestamps in the online setting as one sees 1% of E_{OPT} in expectation.

1.3 Our Results

We consider two benchmark problems in combinatorial optimization under the adversarial-injections model in both the streaming and the online settings, namely maximum matching and monotone submodular maximization subject to a cardinality constraint. As we explain next, the study of these classic problems in our new model gives interesting insights: for many settings we can achieve more robust algorithms with similar guarantees as in the random-order model but, perhaps surprisingly, there are also natural settings where the adversarial-injection model turns out to be as hard as the adversarial order model.

The maximum matching problem. We first discuss the (unweighted) *maximum*

matching problem. Given a graph $G = (V, E)$, a matching M is a subset of edges such that every vertex has at most one incident edge in M. A matching of maximum cardinality is called a maximum matching, whereas a *maximal* matching is one in which no edge can be added without breaking the property of it being a matching. The goal in the maximum matching problem is to compute a matching of maximum cardinality. Note that a maximal matching is 1/2-approximate. Work on maximum matching has led to several important concepts and new techniques in theoretical computer science [26, 24, 6, 19]. The combination of streaming and random-order model was first studied by Konrad, Magniez and Mathieu [23], where edges of the input graph arrive in the stream. We allow a streaming algorithm to have memory $O(n \operatorname{polylog}(n))$, which is called the semi-streaming setting. This is usually significantly less than the input size, which can be as large as $O(n^2)$. This memory usage is justified, because even storing a solution can take $\Omega(n \log(n))$ space ($\Omega(\log(n))$ for each edge identity). The question that Konrad et al. answered affirmatively was whether the trivial 1/2-approximation algorithm that computes a maximal matching can be improved in the random-order model. Since then, there has been some work on improving the constant [14, 9]. The state-of-the-art is an approximation ratio of $6/11 \approx 0.545$ proved by Farhadi, Hajiaghayi, Mah, Rao, and Rossi [9]. We show that beating the ratio of 1/2 is possible also in the adversarial-injections model by building on the techniques developed for the random-order model.

▶ **Theorem 1.** *There exists an absolute constant $\gamma > 0$ such that there is a semi-streaming algorithm for maximum matching under adversarial-injections with an approximation ratio of $1/2 + \gamma$ in expectation.*

We note that beating 1/2 in adversarial-order streams is a major open problem. In this regard, our algorithm can be viewed as a natural first step towards understanding this question.

Now we move our attention to the online setting, where the maximum matching problem was first studied in the seminal work of Karp, Vazirani, and Vazirani [20]. They gave a tight $(1 - 1/e)$-competitive algorithm for the so-called one-sided vertex arrival model which is an important special case of the edge-arrival model considered here. Since then, the online matching problem has received significant attention (see e.g. [4, 7, 11, 17, 14]). Unlike the adversarial streaming setting, there is a recent hardness result due to [14] in the adversarial online setting that the trivial ratio of 1/2 cannot be improved. We also know by [16] that one can beat 1/2 for bipartite graphs in the random-order online setting. Hence, one might hope at least for bipartite graphs to use existing techniques to beat 1/2 in the online adversarial-injections setting and get a result analogous to Theorem 1. But surprisingly so, this is not the case. We observe that the construction used in proving Theorem 3 in [14] also implies that there does not exist an algorithm with a competitive ratio of $1/2 + \varepsilon$ for any $\varepsilon > 0$ in the adversarial-injections model.

Maximizing a monotone submodular function subject to a cardinality constraint. In this problem, we are given a ground set E of n elements and a monotone submodular set function $f : 2^E \to \mathbb{R}_{\geqslant 0}$. A function is said to be submodular, if for any $S, T \subseteq E$ it holds that $f(S) + f(T) \geqslant f(S \cup T) + f(S \cap T)$. It is monotone if $f(S) \leqslant f(T)$ for all $S \subseteq T \subseteq E$. The problem we consider is to find a set $S \subseteq E$ with $|S| \leqslant k$ that maximizes $f(S)$. We assume that access to f is via an oracle.

In the offline setting, a simple greedy algorithm that iteratively picks the element with the largest marginal contribution to f with respect to the current solution is $(1 - 1/e)$-approximate [28]. This is tight: Any algorithm that achieves an approximation ratio of

Table 1 : Comparison of different models for the two studied problems. Here, $\gamma > 0$ is a fixed absolute constant and $\varepsilon > 0$ is any constant.

Maximum matching

	Random order	Adversarial Injections	Adversarial order
Streaming	$\geqslant 6/11$ [9]	$\geqslant 1/2 + \gamma$	$\leqslant 1 - 1/e + \varepsilon$ [18]
Online	$\geqslant 1/2$ (folklore)	$\leqslant 1/2$	$\leqslant 1/2$ [14]

Submodular function maximization

	Random order	Adversarial Injections	Adversarial order
Streaming	$\geqslant 1 - 1/e - \varepsilon$ [1] $\leqslant 1 - 1/e + \varepsilon$ [25]	$\geqslant 0.55$	$\geqslant 1/2 - \varepsilon$ [2] $\leqslant 1/2$ [12]

better than $(1 - 1/e)$ must make $\Omega(n^k)$ oracle calls [27], which is enough to brute-force over all k-size subsets. Even for maximum coverage (which is a special family of monotone submodular functions), it is NP-hard to get an approximation algorithm with ratio better than $1 - 1/e$ [10].

In the random-order online setting, this problem is called the *submodular secretary* problem, and an exponential time $1/e$-approximation and polynomial-time $(1 - 1/e)/e$-approximation algorithms are the state-of-the-art [22]. In the adversarial online setting, it is impossible to get any bounded approximation ratio for even the very special case of picking a maximum weight element. In this case, $|E_{\text{OPT}}| = 1$ and adversarial and adversarial-injections models coincide; hence the same hardness holds. In light of this negative result, we focus on adversarial-injections in the streaming setting. Note that to store a solution we only need the space for k element identities. We think of k to be much smaller than n. Hence, it is natural to ask, whether the number of elements in memory can be independent of n.

For streaming algorithms in the adversarial order setting, the problem was first studied by Chakrabarti and Kale [5] where they gave a 1/4-approximation algorithm. This was subsequently improved to $1/2 - \varepsilon$ by Badanidiyuru et al. [2]. Later, Norouzi-Fard et al. [29] observed that in the random-order model this ratio can be improved to beyond 1/2. Finally, Agrawal et al. [1] obtained a tight $(1 - 1/e)$-approximation guarantee in the random-order model.

The algorithm of Agrawal et al. [1] involves as a crucial step a partitioning the stream in order to isolate the elements of the optimum solution. As discussed earlier, this approach does not work under adversarial-injections. However, we note that the algorithm and analysis by Norouzi-Fard et al. [29] can be easily modified to work under adversarial-injections as well. Their algorithm, however, has an approximation ratio of $1/2 + 8 \cdot 10^{-14}$. In this paper, we remedy this weak guarantee.

▶ **Theorem 2.** *There exists a* 0.55*-approximation algorithm that stores a number of elements that is independent of* n *for maximizing a monotone submodular function with a cardinality constraint* k *under adversarial-injections in the streaming setting.*

We summarize and compare our results with random-order and adversarial-order models for the problems we study in Table 1. It is interesting to see that in terms of beating 1/2, our model in the streaming setting agrees with the random-order model and in the online setting agrees with the adversarial-order model.

2 Matching

In this section, we consider the problem of maximum unweighted matching under adversarial injections in both streaming and online settings where the edges of the input graph arrive one after another.

2.1 Streaming Setting

We show that the trivial approximation ratio of 1/2 can be improved upon. We provide a robust version of existing techniques and prove a statement about robustness of the greedy algorithm to achieve this.

First, let us introduce some notation which we will use throughout this section. We denote the input graph by $G = (V, E)$, and let M^* be a maximum matching. For any matching M, the union $M \cup M^*$ is a collection of vertex-disjoint paths and cycles. When M is clear from the context, a path of length $i \geqslant 3$ in $M \cup M^*$ which starts and ends with an edge of M^* is called an i-augmenting path. Notice that an i-augmenting path alternates between edges of M^* and M and that we can increase the size of M by one by taking all edges from M^* and removing all edges from M along this path. We say that an edge in M is 3-augmentable if it belongs to some 3-augmenting path. Otherwise, we say it is non-3-augmentable. Also, let $M^* = E_{\text{OPT}}$; as described in the introduction, this is without loss of generality.

As a subroutine for our algorithm we need the following procedure.

▶ **Lemma 3** (Lemma 3.1 in [13]). *There exists a streaming algorithm* 3-Aug-Paths *with the following properties:*

1. *The algorithm is initialized with a matching M and a parameter $\beta > 0$. Then a set E of edges is given to the algorithm one edge at a time.*
2. *If $M \cup E$ contains at least $\beta|M|$ vertex disjoint 3-augmenting paths, the algorithm returns a set A of at least $(\beta^2/32)|M|$ vertex disjoint 3-augmenting paths. The algorithm uses space $O(|M|)$.*

2.1.1 The Algorithm

We now describe our algorithm Match. It runs two algorithms in parallel and selects the better of the two outputs. The first algorithm simply constructs a maximal matching greedily by updating the variable M_1. The second algorithm also constructs a matching $M_2^{(1)}$ greedily, but it stops once $M_2^{(1)}$ has $|M^*|(1/2 - \varepsilon)$ edges. We call this Phase 1. Then, it finds 3-augmentations using the 3-Aug-Paths algorithm given by Lemma 3. Finally, it augments the paths found to obtain a matching M_2. The constant β used in 3-Aug-Paths is optimized for the analysis and will be specified there.

Notice that here we assumed that the algorithm knows $|M^*|$. This assumption can be removed using geometric guessing at a loss of an arbitrary small factor in the approximation ratio. We refer the reader to Appendix A.1 for details.

2.1.2 Overview of the Analysis

We discuss only the intuition here and refer the reader to Appendix A.2 for a formal proof. Consider the first portion of the stream until we have seen a small constant fraction of the elements in E_{OPT}. If the greedy matching up to this point is already close to a 1/2-approximation, this is good for the second algorithm as we are able to augment the matching using the remaining edges of M^*. The other case is good for the first algorithm: We will

show that the greedy matching formed so far must contain a significant fraction of the edges in M^* which we have seen so far. If this happens, the first algorithm outputs a matching of size a constant fraction more than $|M^*|/2$.

A technical challenge and novelty comes from the fact that the two events above are not independent of the random order of E_{OPT}. Hence, when conditioning on one event, we can no longer assume that the order of E_{OPT} is uniformly at random. We get around this by showing that the greedy algorithm is robust to small changes in streams. The intuition is that in the first part of the stream the greedy solution either is large for all permutations of E_{OPT} or it is small for all permutations. Hence, these are not random events depending on the order, but two cases in which we can assume a uniform distribution.

2.2 Online Setting

Since we can improve 1/2 for the streaming setting, it is natural to hope that the existing techniques (e.g., the approach of the previous subsection) can be applied in the online setting as well. Surprisingly, this is not the case. In other words, the competitive ratio of 1/2 is optimal even for bipartite graphs. The technique from the previous subsection breaks apart, because the algorithm constructs several candidate solutions in parallel by guessing $|M^*|$. This is not a problem for a streaming algorithm, however, an online algorithm can only build one solution.

For a formal proof, we rely on the bipartite construction used in the proof of Theorem 3 from [14]. The authors show that there is no (randomized) algorithm with a competitive ratio of $1/2 + \varepsilon$ for any $\varepsilon > 0$. More precisely, they show that not even a good fractional matching can be constructed online. For fractional matchings, randomization does not help and therefore we can assume the algorithm is deterministic. The original proof is with respect to adversarial order, but it is not hard to see that it transfers to adversarial injections.

The authors construct a bipartite instance that arrives in (up to) N rounds. In round i, a matching of size i arrives. The algorithm does not know whether the current round is the last one or not. Hence, it has to maintain a good approximation after each round. This forces the algorithm to take edges that do not belong to the optimal matching and eventually leads to a competitive ratio of 1/2. The same construction works in our model: The edges from the optimal matching arrive in the last round and their internal order does not affect the proof. In fact, the construction works for any order of the elements within a round. Thus, an algorithm cannot exploit the fact that their order is randomized and therefore also cannot do better than 1/2.

3 Submodular Maximization

In this section, we consider the problem of submodular maximization subject to a cardinality constraint. The algorithm has query access to a monotone, submodular function $f : 2^E \to \mathbb{R}$ over a ground set E. Moreover, f is normalized with $f(\emptyset) = 0$. The goal is to compute a set S of size at most k that maximizes $f(S)$. We present a 0.55-approximate streaming algorithm in the adversarial-injections model which only needs the memory to store $(O(k))^k$ many elements. In particular, this number is independent of the length of the stream.

3.1 Notation

For $e \in E$ and $S \subseteq E$ we write $S + e$ for the set $S \cup \{e\}$ and $f(e \mid S)$ for $f(S + e) - f(S)$. Similarly, for $A, B \subseteq E$ let $f(A \mid B) := f(A \cup B) - f(B)$. An equivalent definition of

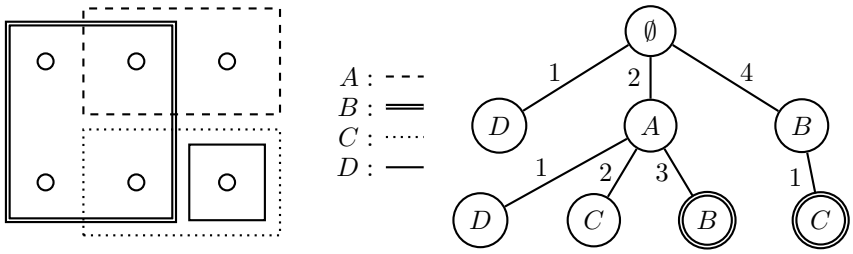

Figure 1 In this example, function f counts the dots covered by a set of rectangles. On the right, the tree for stream $\sigma = (A, B, C, D)$ and $k = 2$ is depicted. The labels on the edges correspond to the increase in f. The maximal leaves are highlighted.

submodularity to the one given in the introduction states that for any two sets $S \subseteq T \subseteq E$, and $e \in E \setminus T$ it holds that $f(e \mid S) \geqslant f(e \mid T)$.

We denote by σ the stream of elements E, by $-\infty$ and ∞ the start and end of the stream. For elements a and b, we write $\sigma[a, b]$ for the interval including a and b and $\sigma(a, b)$ for the interval excluding them. Moreover, we may assume that $f(\emptyset) = 0$, since otherwise, we may replace the submodular function by $f' : 2^E \to \mathbb{R}_{\geqslant 0}, T \mapsto f(T) - f(\emptyset)$.

Denote the permutation of E_{OPT} by π. Let o_i^π be the i'th element of E_{OPT} in the stream according to the order given by π. Let $O_0^\pi = \emptyset$ and $O_i^\pi = \{o_1^\pi, \ldots, o_i^\pi\}$ for all i; hence, $E_{\text{OPT}} = O_k^\pi$ for any π. Finally, let $\text{OPT} = f(O_k^\pi)$.

3.2 The Algorithm

For simplicity we present an algorithm with the assumption that it knows the value OPT. Moreover, for the set of increases in f, that is $I = \{f(e \mid S) : e \in E, S \subseteq E\}$, we assume that $|I| \leqslant O(k)$. These two assumptions can be made at a marginal cost in the approximation ratio and an insignificant increase in memory. This follows from standard techniques. We refer the reader to Appendix B.1 and Appendix B.2 for details.

As a central data-structure, the algorithm maintains a rooted tree T of height at most k. Every node except for the root stores a single element from E. The structure resembles a prefix tree: Each node is associated with the solution, where the elements on the path from the root to it is selected. The nodes can have at most $|I|$ children, that is, one for each increase. The basic idea is that for some partial solution $S \subseteq E$ (corresponding to a node) and two elements e, e' with $f(e \mid S) = f(e' \mid S)$ we only consider one of the solutions $S \cup \{e\}$ and $S \cup \{e'\}$. More precisely, the algorithm starts with a tree consisting only of the root. When it reads an element e from the stream, it adds e as a child to every node where (1) the distance of the node to the root is smaller than k and (2) the node does not have a child with increase $f(e \mid S)$, where S is the partial solution corresponding to this particular node.

Because of (1), the solutions are always of cardinality at most k. When the stream is read completely, the algorithm selects the best solution among all leaves. An example of the algorithm's behavior is given in Figure 1.

3.3 Overview of the Analysis

For analyzing the algorithm, we will use a sophisticated strategy to select one of the leaves and only compare this leaf to the optimum. We emphasize that this selection does not have to be computed by the algorithm. In particular, it does not need to be computable by a streaming algorithm and it can rely on knowledge of E_{OPT} and E_{NOISE}, which the algorithm

does not have. Since the algorithm always takes the best leaf, we only need to give a lower bound for one of them. Before we describe this strategy, we analyze the tree algorithm in two educational corner cases.

The first one shows that by a careful selection of a leaf the algorithm appears to take elements based on the location of the E_{OPT}, although it does not know them. Let $r_i^\pi = \text{argmax}_{e \in \sigma(-\infty, o_1^\pi]} f(e)$, that is, the most valuable element until the arrival of the first element from E_{OPT}. Here argmax breaks ties in favor of the first element in σ. We do not know when o_1^π arrives, but we know that the algorithm will have created a node (with the root as its parent) for r_1^π by then. We define iteratively $R_i^\pi = \{r_1^\pi, \dots, r_i^\pi\}$ and $r_{i+1}^\pi = \text{argmax}_{e \in \sigma(r_i^\pi, o_{i+1}^\pi]} f(e \mid R_i^\pi)$ for all i. Again, we can be sure that r_{i+1}^π, which yields the best increase for R_i^π until the arrival of o_{i+1}^π, is a appended to the path $r_1^\pi \to \cdots \to r_i^\pi$.

This selection is inspired by the following idea. Suppose we could partition the stream into k intervals such that in each exactly one elements from E_{OPT} appears. Then a sensible approach would be to start with an empty solution and greedily add the element that yields the maximal increase to our partial solution in each interval. Clearly one such partition would be $\sigma(o_i^\pi, o_{i+1}^\pi]$, $i = 1, \dots, k$. We note that while the selection above is similar, it does not completely capture this. Although r_{i+1}^π is an element that arrives before o_{i+1}^π, we cannot be certain that it arrives after o_i^π. We only know that it arrives after r_i^π.

Next, we prove that the solution R_k^π is a 1/2-approximation. This already shows that the tree algorithm is 1/2-approximate even in the adversarial order model. By definition of R_i^π and r_i^π, we have

$$f(R_k^\pi) = \sum_{i=1}^{k} f(r_i^\pi \mid R_{i-1}^\pi) \geqslant \sum_{i=1}^{k} f(o_i^\pi \mid R_{i-1}^\pi)$$
$$= \sum_{i=1}^{k} [f(o_i^\pi \mid R_{i-1}^\pi) - f(o_i^\pi \mid R_k^\pi)] + \sum_{i=1}^{k} f(o_i^\pi \mid R_k^\pi).$$

Notice that due to submodularity the term $f(o_i^\pi \mid R_{i-1}^\pi) - f(o_i^\pi \mid R_k^\pi)$ is always non-negative. Moreover, if $o_i^\pi = r_i^\pi \in R_k^\pi$, it collapses to $f(o_i^\pi \mid R_{i-1}^\pi)$. Thus, we can bound the right term of the equation and thereby $f(R_k^\pi)$ with

$$f(R_k^\pi) \geqslant \sum_{\substack{i=1 \\ r_i^\pi = o_i^\pi}}^{k} f(o_i^\pi \mid R_{i-1}^\pi) + \sum_{i=1}^{k} f(o_i^\pi \mid R_k^\pi).$$

From submodularity and monotonicity of f it follows that

$$\sum_{i=1}^{k} f(o_i^\pi \mid R_k^\pi) \geqslant f(O_k^\pi \mid R_k^\pi) = f(O_k^\pi \cup R_k^\pi) - f(R_k^\pi) \geqslant f(O_k^\pi) - f(R_k^\pi).$$

Hence, we conclude that

$$2f(R_k^\pi) \geqslant f(O_k^\pi) + \sum_{\substack{i=1 \\ r_i^\pi = o_i^\pi}}^{k} f(o_i^\pi \mid R_{i-1}^\pi).$$

This shows that R_k^π is 1/2-approximate, because $O_k^\pi = E_{\text{OPT}}$. Indeed, if a significant value of the elements in E_{OPT} are taken, then R_k^π is even better than 1/2-approximate.

Recall that the elements E_{OPT} are ordered randomly in the adversarial-injections model. Hence, the worst-case in the analysis above is that R_k^π is disjoint from E_{OPT} for all realizations

of π. However, by a different analysis we can see that this case is in fact well-behaved. This is because the algorithm would select the same elements $r_1^\pi, \dots, r_k^\pi$ for every realization of π. Hence, we can safely drop the superscript π in R_i^π and r_i^π. Since for every element $o \in E_{\text{OPT}}$ there is some realization of π where $o_i^\pi = o$, yet the algorithm does not pick o_i^π, we can bound the increase of each r_i by

$$f(r_i \mid R_{i-1}) \geqslant \max_{o \in E_{\text{OPT}}} f(o \mid R_{i-1}) \geqslant \frac{1}{k} \sum_{o \in E_{\text{OPT}}} f(o \mid R_{i-1}).$$

By submodularity and monotonicity we get

$$\frac{1}{k} \sum_{o \in E_{\text{OPT}}} f(o \mid R_{i-1}) \geqslant \frac{1}{k} f(E_{\text{OPT}} \mid R_{i-1}) \geqslant \frac{1}{k}(\text{OPT} - f(R_{i-1})).$$

This is the same recurrence formula as in the classic greedy algorithm and by simple calculations we get the closed form

$$f(R_k) \geqslant \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \text{OPT} \geqslant \left(1 - \frac{1}{e}\right) \text{OPT}\,.$$

In other words, the algorithm is even $(1 - 1/e)$-approximate in this case. In our main proof we will use a more involved strategy for selecting a leaf. This is to be able to combine the two approaches discussed above.

3.4 Analysis

Let us first define the selection of the leaf we are going to analyze. The elements on the path to this leaf will be denoted by $s_1^\pi, \dots, s_k^\pi$ and we write S_i^π for $\{s_1^\pi, \dots s_i^\pi\}$. The elements are defined inductively, but as opposed to the previous section we need in addition indices $n_1, \dots, n_k$. Recall, previously we defined the $(i+1)$'th element r_{i+1}^π as the best increase in $\sigma(r_i^\pi, o_{i+1}^\pi]$. Here, we use n_{i+1} to describe the index of the element from E_{OPT} which constitutes the end of this interval. It is not necessarily o_{i+1}^π anymore. We always start with $n_1 = 1$, but based on different cases we either set $n_{i+1} = n_i + 1$ or $n_{i+1} = n_i$. We underline that n_i is independent of the realization of π. In the following, $t \in [0, 1]$ denotes a parameter that we will specify later.

The element s_i^π will be chosen from two candidates u_i^π and v_i^π. The former is the best increase of elements excluding $o_{n_i}^\pi$, that is,

$$u_i^\pi = \begin{cases} \operatorname{argmax}_{e \in \sigma(-\infty, o_{n_1}^\pi)} f(e) & \text{if } i = 1, \\ \operatorname{argmax}_{e \in \sigma(s_i^\pi, o_{n_i}^\pi)} f(e \mid S_{i-1}^\pi) & \text{otherwise.} \end{cases}$$

The latter is defined in the same way, except it includes $o_{n_i}^\pi$ in the choices, that is,

$$v_i^\pi = \begin{cases} \operatorname{argmax}_{e \in \sigma(-\infty, o_{n_1}^\pi]} f(e) & \text{if } i = 1, \\ \operatorname{argmax}_{e \in \sigma(s_{i-1}^\pi, o_{n_i}^\pi]} f(e \mid S_{i-1}^\pi) & \text{otherwise.} \end{cases}$$

We now define the choice of s_i^π and n_{i+1} based on the following two cases. Note that the cases are independent from the realization of π.

Case 1: $\mathbb{E}_\pi f(u_i^\pi \mid S_{i-1}^\pi) \geqslant t \cdot \mathbb{E}_\pi f(o_{n_i}^\pi \mid S_{i-1}^\pi)$**.** In this case, we set $s_i^\pi = u_i^\pi$ and $n_{i+1} = n_i$. Notice that this means s_i^π is chosen independently from $o_{n_i}^\pi$. In other words, we did not see $o_{n_i}^\pi$, yet. The element $o_{n_i}^\pi$ is still each of the remaining elements in E_{OPT} with equal probability. In the analysis this is beneficial, because the distribution of $o_{n_i}^\pi, \dots, o_k^\pi$ remains unchanged. This is similar to the second case in the previous section.

Case 2: $\mathbb{E}_\pi f(u_i^\pi \mid S_{i-1}^\pi) < t \cdot \mathbb{E}_\pi f(o_{n_i}^\pi \mid S_{i-1}^\pi)$. Here, set $s_i^\pi = v_i^\pi$ and $n_{i+1} = n_i + 1$. Now the distribution of $o_i^\pi, \dots, o_k^\pi$ can change. However, a considerable value of s_i^π over different π comes from taking $o_{n_i}^\pi$. As indicated by the first case in the previous section this will improve the guarantee of the algorithm.

The solution S_k^π corresponds to a leaf in the tree algorithm. Clearly, u_1^π and v_1^π are children of the root. Hence, s_1^π is also a child. Then for induction we assume s_i^π is a node, which implies u_{i+1}^π and v_{i+1}^π are also nodes: The elements u_{i+1}^π and v_{i+1}^π are the first elements after s_i^π with the respective gains ($f(u_{i+1}^\pi \mid S_i^\pi)$ and $f(v_{i+1}^\pi \mid S_i^\pi)$). Hence, s_{i+1}^π is a child of s_i^π.

In order to bound $\mathbb{E}_\pi f(S_k^\pi)$, we will study more broadly all values of $\mathbb{E}_\pi f(S_h^\pi)$ where $h \leqslant k$. To this end, we define a recursive formula $R(k,h)$ and prove that it bounds $\mathbb{E}_\pi f(S_h^\pi)/\operatorname{OPT}$ from below. Then using basic calculus we will show that $R(k,k) \geqslant 0.5506$ for all k. Initialize $R(k,0) = 0$ for all k. Then let $R(k,h)$, $h \leqslant k$, be defined by

$$R(k,h) = \min\left\{ \frac{t}{k} + \left(1 - \frac{t}{k}\right) R(k,h-1)\,,\ \ \frac{1}{k} + \left(1 - \frac{1+t}{k}\right) R(k-1,h-1)\,,\ \ \frac{1}{1+t} \right\}.$$

▶ **Lemma 4.** *For all instances of the problem and $h \leqslant k$, the solution S_h^π as defined above satisfies $\mathbb{E}_\pi f(S_h^\pi) \geqslant R(k,h)\operatorname{OPT}$.*

Proof. The proof is by induction over h. For $h = 0$, the statement holds as $R(k,0)\operatorname{OPT} = 0 = \mathbb{E}_\pi f(S_0^\pi)$. Let $h > 0$ and suppose the statement of the lemma holds with $h-1$ for all instances of the problem. Suppose we are given an instance with $k \geqslant h$. We distinguish the two cases $s_1^\pi = u_1^\pi$ and $s_1^\pi = v_1^\pi$.

First, consider $\mathbb{E}_\pi f(u_1^\pi) \geqslant t \cdot \mathbb{E}_\pi f(o_1^\pi)$, which implies that $s_1^\pi = u_1^\pi$. Note that u_1^π is the best element in $\sigma(-\infty, o_1^\pi)$, consequently, its choice is independent from the realization of π. Let us drop the superscript in u_1^π and s_1^π for clarity. We construct a new instance mimicking the subtree of s_1. Formally, our new instance still has the same k elements E_{OPT}, i.e., $k' = k$. The stream is $\sigma' = \sigma(s_1^\pi, \infty)$ and, the submodular function $f' : 2^U \to \mathbb{R}$, $f'(T) \mapsto f(T \mid s_1)$. In this instance we have $\operatorname{OPT}' = f'(E_{\text{OPT}}) = f(E_{\text{OPT}} \mid s_1) \geqslant \operatorname{OPT} - f(s_1)$. It is easy to see that the elements $s_1'^\pi, \dots, s_{h-1}'^\pi$ chosen in the new instance correspond exactly to the elements $s_2^\pi, \dots, s_h^\pi$. Hence, with the induction hypothesis we get

$$\mathbb{E}_\pi f(S_h^\pi) = f(s_1) + \mathbb{E}_\pi f(S_h^\pi \mid s_1) = f(s_1) + \mathbb{E}_\pi f'(S_{h-1}'^\pi) \geqslant f(s_1) + R(k,h-1)(\operatorname{OPT} - f(s_1)).$$

By assumption we have $f(s_1) \geqslant t \cdot \mathbb{E}_\pi f(o_i^\pi) \geqslant t \cdot \operatorname{OPT}/k$. Together with $R(k,h-1) \leqslant 1/(1+t) \leqslant 1$ we calculate

$$f(s_1) + R(k,h-1)(\operatorname{OPT} - f(s_1)) \geqslant \frac{t}{k}\operatorname{OPT} + R(k,h-1)\left(1 - \frac{t}{k}\right)\operatorname{OPT}.$$

The right-hand side is by definition at least $R(k,h)\operatorname{OPT}$.

Now we turn to the case $\mathbb{E}_\pi f(u_1^\pi) < t \cdot \mathbb{E}_\pi f(o_1^\pi)$, which means $s_1^\pi = v_1^\pi$ is chosen. Similar to the previous case, we construct a new instance. After taking s_1^π, our new instance has $k' = k-1$ elements $E'_{\text{OPT}} = E_{\text{OPT}} \setminus \{o_1^\pi\}$, stream $\sigma' = \sigma(s_1, \infty)$, and submodular function $f' : 2^E \to \mathbb{R}$, $f(T) \mapsto f(T \mid s_1^\pi)$. Thus, $\operatorname{OPT}' = f'(E'_{\text{OPT}}) = f(E_{\text{OPT}} \setminus \{o_1^\pi\} \mid s_1^\pi) \geqslant \operatorname{OPT} - f(s_1^\pi \cup o_1^\pi)$. We remove o_1^π from E_{OPT}, because $s_1^\pi = v_1^\pi$ depends on it. The distribution of $o_2^\pi, \dots, o_k^\pi$ when conditioning on the value of o_1^π (and thereby the choice of s_1^π) is still a uniformly random permutation of E'_{OPT}. Like in the previous case, we can see that $S_{h-1}'^\pi = S_h^\pi \setminus \{s_1^\pi\}$ and we can apply the induction hypothesis. First, however, let us examine $\mathbb{E}_\pi f(s_1^\pi \cup o_1^\pi)$. Since we know that whenever $s_1^\pi \neq o_1^\pi$ we have $s_1^\pi = u_1^\pi$, it follows that

$$\mathbb{P}_\pi[s_1^\pi \neq o_1^\pi] \cdot \mathbb{E}_\pi[f(s_1) \mid s_1^\pi \neq o_1^\pi] \leqslant \mathbb{E}_\pi f(u_1^\pi) < t \cdot \mathbb{E}_\pi f(o_1^\pi) \leqslant t \cdot \mathbb{E}_\pi f(s_1^\pi).$$

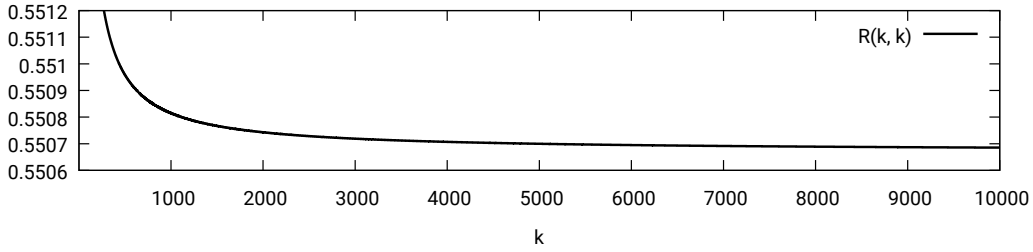

Figure 2 Values of the recurrence formula for $t = 0.8$.

Hence, we deduce

$$\mathbb{E}_\pi f(s_1^\pi \cup o_1^\pi) \leqslant \mathbb{E}_\pi f(o_1^\pi) + \mathbb{P}_\pi[s_1^\pi \neq o_1^\pi] \cdot \mathbb{E}_\pi[f(s_1) \mid s_1^\pi \neq o_1^\pi] \leqslant \mathbb{E}_\pi f(o_1^\pi) + t \cdot \mathbb{E}_\pi f(s_1^\pi).$$

We are ready to prove the bound on $\mathbb{E}_\pi f(S_h^\pi)$. By induction hypothesis, we get

$$\begin{aligned}\mathbb{E}_\pi f(S_h^\pi) &= \mathbb{E}_\pi f(s_1^\pi) + \mathbb{E}_\pi f'(S_{h-1}'^\pi) \\ &\geqslant \mathbb{E}_\pi f(s_1^\pi) + R(k-1,h-1)(\mathrm{OPT} - \mathbb{E}_\pi f(s_1^\pi \cup o_1^\pi)).\end{aligned}$$

Inserting the bound on $\mathbb{E}_\pi f(s_1^\pi \cup o_1^\pi)$ we know that the right-hand side is at least

$$\mathbb{E}_\pi f(s_1^\pi) + R(k-1,h-1)(\mathrm{OPT} - \mathbb{E}_\pi f(o_1^\pi) - t \cdot \mathbb{E}_\pi f(s_1^\pi)).$$

Using that $f(s_1^\pi) \geqslant f(o_1^\pi)$ for all π and $R(k-1,h-1) \cdot t \leqslant t/(1+t) \leqslant 1$ we bound the previous term from below by

$$\mathbb{E}_\pi f(o_1^\pi) + R(k-1,h-1)(\mathrm{OPT} - (1+t)\,\mathbb{E}_\pi f(o_1^\pi)).$$

Finally, we use that $\mathbb{E}_\pi f(o_1^\pi) \geqslant \mathrm{OPT}/k$ and $R(k-1,h-1)(1+t) \leqslant 1$ to arrive at

$$\frac{1}{k}\,\mathrm{OPT} + R(k-1,h-1)\left(\mathrm{OPT} - \frac{1+t}{k}\,\mathrm{OPT}\right) \geqslant R(k,h)\,\mathrm{OPT},$$

which concludes the proof. ◀

With $t = 0.8$ we are able to show that for sufficiently large k the minimum in the definition of $R(k,k)$ is always attained by the first term. Then, after calculating a lower bound on $R(k,k)$ for small values, we can easily derive a general bound.

▶ **Lemma 5.** *With $t = 0.8$ for all positive integers k it holds that $R(k,k) \geqslant 0.5506$.*

Figure 2 contains a diagram (generated by computer calculation), which shows that the formula tends to a value between 0.5506 and 0.5507 for $k \in \{0, \ldots, 10000\}$. The proof requires tedious and mechanical calculations and is thus deferred to Appendix B.

4 Conclusion and Open Problems

In this paper, we introduced a semi-random model called adversarial-injections with the motivation of eliminating algorithms that overfit to random-order streams while still being easier than adversarial-order streams. We studied two classical problems in combinatorial optimization in this model.

For unweighted matching, we could beat 1/2 in the streaming setting whereas we observed from [14] that we could not beat 1/2 in the online setting. This also makes our model non-trivial as there is a separation between the online and streaming setting.

For monotone submodular maximization with cardinality constraint k, we obtained a 0.55 approximation algorithm albeit with a huge memory footprint but importantly independent of n (universe size). The obvious open question is whether one can design a $(1 - 1/e)$-approximation algorithm which stores number of elements that is independent of n. Does our algorithm have an approximation ratio of $1 - 1/e$? We observed that the algorithm in [29] is a $1/2 + \varepsilon$ approximation for a very small $\varepsilon > 0$. The algorithm stores poly(k) elements. Can one design an algorithm that stores only poly(k) elements and beats 1/2 by a significant constant or, even better, gets $1 - 1/e$?

References

1 Shipra Agrawal, Mohammad Shadravan, and Cliff Stein. Submodular secretary problem with shortlists. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 1:1–1:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.1.

2 Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 671–680, 2014. doi:10.1145/2623330.2623637.

3 Domagoj Bradac, Anupam Gupta, Sahil Singla, and Goran Zuzic. Robust algorithms for the secretary problem. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 32:1–32:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.32.

4 Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, 81(5):1781–1799, 2019. doi:10.1007/s00453-018-0505-7.

5 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1-2):225–247, 2015. doi:10.1007/s10107-015-0900-7.

6 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

7 Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved bounds for randomized preemptive online matching. *Inf. Comput.*, 259(1):31–40, 2018. doi:10.1016/j.ic.2017.12.002.

8 Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Online allocation with traffic spikes: Mixing adversarial and stochastic models. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 169–186, 2015.

9 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1773–1785, 2020. doi:10.1137/1.9781611975994.108.

10 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

11 Uriel Feige. Tighter bounds for online bipartite matching. *CoRR*, abs/1812.11774, 2018. URL: http://arxiv.org/abs/1812.11774, arXiv:1812.11774.

12 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Proceedings of the Fifty-Second Annual ACM on Symposium on Theory of Computing, STOC (to appear)*, 2020.

13 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500, 2019. doi:10.1145/3293611.3331603.

14 Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 26–37, 2019. doi:10.1109/FOCS.2019.00011.

15 Sudipto Guha and Andrew McGregor. Approximate quantiles and the order of the stream. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 273–279, 2006. doi:10.1145/1142351.1142390.

16 Guru Prashanth Guruganesh and Sahil Singla. Online matroid intersection: Beating half for random arrival. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 241–253, 2017. doi:10.1007/978-3-319-59250-3_20.

17 Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2875–2886, 2019. doi:10.1137/1.9781611975482.178.

18 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013. doi:10.1137/1.9781611973105.121.

19 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986. doi:10.1007/BF02579407.

20 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 352–358, 1990. doi:10.1145/100216.100262.

21 Thomas Kesselheim, Robert D. Kleinberg, and Rad Niazadeh. Secretary problems with non-uniform arrival order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 879–888, 2015. doi:10.1145/2746539.2746602.

22 Thomas Kesselheim and Andreas Tönnis. Submodular secretary problems: Cardinality, matching, and linear constraints. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 16:1–16:22, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.16.

23 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242. 2012. doi:10.1007/978-3-642-32512-0_20.

24 László Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 565–574, 1979.

25 Andrew McGregor and Hoa T. Vu. Better streaming algorithms for the maximum coverage problem. *Theory Comput. Syst.*, 63(7):1595–1619, 2019. doi:10.1007/s00224-018-9878-x.

26 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

27 George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978. doi:10.1287/moor.3.3.177.

28 George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

29 Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 3826–3835, 2018. URL: http://proceedings.mlr.press/v80/norouzi-fard18a.html.

A Analysis of the streaming algorithm for maximum matching

We start by briefly recalling the algorithm Match. We refer the reader to Section 2.1.1 for complete details.

Match runs two algorithms in parallel and outputs the better solution. The first algorithm runs greedy whereas the second algorithm runs greedy up to certain point (Phase 1), and then starts collecting 3-augmenting paths.

Now we recall some definitions introduced in Section 2 that we will use throughout the section. Let M^* denote the maximum matching, M_1 the variable that is updated by the first algorithm and M_2 the matching produced by the second algorithm. Let $M_2^{(1)}$ denote this matching of the second algorithm just after Phase 1 ends.

A.1 Removing assumption that $|M^*|$ is known

Our algorithm Match assumed that it knew $|M^*|$. This is because the second algorithm that is run in Match needs to know $|M^*|$ as it starts collecting 3-augmenting paths when it has collected at least $|M^*|(1/2-\varepsilon)$ edges. Until this point in the stream, it has collected exactly $|M_1|$ edges. By definition, $|M_1|$ is also a lower bound on $|M^*|$. Hence at any point, we will run multiple copies of the second algorithm initialized with a guess for $|M^*|$ based on what the value of $|M_1|$ is at that point. Thus, for any fixed $\delta > 0$, we can guess the value of $|M^*|$ up to a factor of $(1+\delta)$ by running the algorithm in parallel for all powers i of $1+\delta$, that satisfy $|M_1|/(1+\delta) \leqslant (1+\delta)^i \leqslant 4|M_1|/(1-2\varepsilon)$. Notice that some copies of our algorithm may stop after some time as their $|M^*|$ estimate is no longer valid, others will continue and new ones with $|M^*|$ estimates that are already not running will start initialized with the matching M_1 at that point in the stream. This increases algorithm's space only by a factor of $O(\log \frac{4(1+\delta)}{1-2\varepsilon})$ and deteriorates the solution value by at most $O(\delta|M^*|)$.

A.2 Proof of Theorem 1

In this section, we prove that Match has an approximation ratio of at least $1/2+\gamma$ in expectation for some fixed constant $\gamma > 0$.

First we prove a lemma on the robustness of the greedy algorithm to small changes in the stream which is required to deal with correlations that may arise in the case where we need to show that the greedy algorithm picks a significant fraction of edges of $|M^*|$. For more on this, we refer the reader to Section 2.1.2.

▶ **Lemma 6.** *Let σ and σ' be streams of edges in G such that σ can be transformed into σ' by deleting an edge from σ. Let M and M' be the matchings computed by the greedy algorithm on σ and σ' respectively. Then $||M|-|M'|| \leqslant 1$.*

Proof. Let $C = (M \setminus M') \cup (M' \setminus M)$, that is, the symmetric difference of M and M'. Notice that C is a collection of disjoint paths and cycles that alternate between edges of M and M'. We claim that in this collection there is at most one path of non-zero length. This implies the statement in the lemma.

We argue that if a path exists, it must contain the edge that was deleted. Hence, it is the unique path. Assume toward contradiction, that there exists a path in C, which does not contain the deleted edge. We now closely examine the first edge e of this path that arrives. Note that this is the same for σ and σ'. In both runs of the greedy algorithm, the two vertices of e are not incident to a matching edge when e arrives. This means that e should have been taken in both runs and therefore cannot be in the symmetric difference of the matchings, a contradiction.

◀

In our analysis we will use the following lemma by Konrad et al which bounds the number of edges that cannot be augmented by 3-augmenting paths if size of maximal matching is small.

▶ **Lemma 7** (Lemma 1 in [23]). *Let $\alpha > 0$, M be a maximal matching in G and M^* be a maximum matching in G with $|M| \leqslant (1/2 + \alpha)|M^*|$. Then the number of 3-augmentable edges in M is at least $(1/2 - 3\alpha)|M^*|$. In particular, the number of non-3-augmentable edges in M is at most $4\alpha|M^*|$.*

We are now ready to prove Theorem 1.

Proof of Theorem 1. Define $\varepsilon = 1/50$, $\alpha = \varepsilon$, and $\rho = \varepsilon/4$. Without loss of generality, we assume that $|M^*| > 2/\rho$. Otherwise, the algorithm can just store the whole graph as it is sparse i.e., it has linear number of edges.

Let σ_1 be the smallest prefix of the stream that contains $k = \lceil \rho \cdot |M^*| \rceil$ elements of M^*. Further, let σ_2 be the remainder of the stream. Notice, that $M_2^{(1)} \subseteq M_1$, since the first algorithm starts the same way, but continues even after reaching this threshold. Let $M_2^{(\sigma_1,1)} \subseteq M_2^{(1)}$ be a random variable corresponding to only those edges in $M_2^{(1)}$ taken during σ_1.

Case 1: For all permutations of E_{opt} it holds that $M_2^{(\sigma_1,1)} = M_2^{(1)}$. Notice that by definition, $|M_2^{(\sigma_1,1)}| = |M_2^{(1)}| \geqslant (1/2 - \varepsilon)|M^*|$. The basic idea for this case is to show that in σ_2 there are a lot of 3-augmenting paths that can be used to improve $M_2^{(1)}$ via 3-Aug-Paths. If $|M_1| \geqslant |M^*|(1/2 + \alpha)$, we are already significantly better than $1/2$. Hence, assume otherwise. From Lemma 7 it follows that the number of non-3-augmentable edges in M_1 is at most $4\alpha|M^*|$. The number of 3-augmentable edges in $M_2^{(\sigma_1,1)}$ is obviously $|M_2^{(\sigma_1,1)}|$ minus the number non-3-augmentable edges in it. The former is at least $|M^*|(1/2 - \varepsilon)$ while the latter is a subset of the non-3-augmentable edges in M_1 and hence at most $4\alpha|M^*|$. It follows that the number of 3-augmentable edges in $M_2^{(\sigma_1,1)}$ is at least $(1/2 - 4\alpha - \varepsilon)|M^*|$.

We will now restrict our attention to the subgraph of the edges in $M_2^{(\sigma_1,1)}$ and σ_2 and the 3-augmentable edges there. Recall, every 3-augmentable edge corresponds to a 3-augmenting path that has two edges from M^* and one from $M_2^{(\sigma_1,1)}$. If at least one of the edges from M^* appears in σ_1, this edge is no longer 3-augmentable when we restrict ourselves to σ_2. However, by definition only k of the edges from M^* appear in σ_1 and each of them can appear in only one 3-augmenting path. In consequence, the number of 3-augmentable edges in $M_2^{(\sigma_1,1)}$ considering only σ_2 is at least

$$\left(\frac{1}{2} - 4\alpha - \varepsilon\right)|M^*| - k \geqslant \left(\frac{1}{2} - 4\alpha - \varepsilon - \rho - \frac{1}{|M^*|}\right)|M^*| \geqslant \left(\frac{1}{2} - 4\alpha - \varepsilon - \frac{3\rho}{2}\right)|M^*|.$$

Here we use that by assumption $|M^*| > 2/\rho$. After constructing the matching $M_2^{(1)}$, the second algorithm proceeds to collect 3-augmenting paths for it using 3-Aug-Paths. We fix its parameter

$$\beta := \left(\frac{1}{2} - 4\alpha - \varepsilon - \frac{3\rho}{2}\right) / \left(\frac{1}{2} - \varepsilon\right).$$

Notice that after completing Phase 1, the second algorithm has at least $\beta|M_2^{(1)}|$ many 3-augmenting paths in the remaining instance. Hence, by Lemma 3 we are guaranteed to find $(\beta^2/32)|M_2^{(1)}|$ many 3-augmenting paths. We conclude that

$$|M_2| \geqslant \left(1 + \frac{\beta^2}{32}\right)|M_2^{(1)}| \geqslant \left(1 + \frac{\beta^2}{32}\right)(1-\varepsilon)|M^*|.$$

Using the definitions of the constants, we calculate that $\beta = (4 - 43\varepsilon)/(4 - 8\varepsilon)$.

Case 2: For at least one permutation of E_{opt} it holds that $M_2^{(\sigma_1,1)} \subsetneq M_2^{(1)}$. Let σ_1^* be the realization of the random variable σ_1 for such a permutation, i.e., we have $|M_2^{(\sigma_1^*,1)}| \leqslant (1/2 - \varepsilon)|M^*|$. We will argue that in expectation (not just for σ_1^*) the greedy algorithm will select a considerable number of elements from M^*. This directly improves its guarantee: Let $S = M^* \cap M_1$. Every edge in $M^* \setminus S$ intersects with some edge in $M_1 \setminus S$, but every edge in $M_1 \setminus S$ can only intersect with two edges in $M^* \setminus S$. This implies $2|M_1 \setminus S| \geqslant |M^* \setminus S|$ and consequently

$$|M_1| \geqslant (|M^*| + |S|)/2. \tag{1}$$

We bound $\mathbb{E}[|S|]$ from below by examining the elements of M^* in σ_1, denoted by $e_1^*, e_2^*, \ldots, e_k^*$. Let $\sigma^{(1)}, \ldots, \sigma^{(k)}$ correspond to the prefix of σ until right before $e_1^*, \ldots, e_k^*$ arrive. Further, define $M_1^{(1)}, \ldots, M_1^{(k)}$ as the value of M_1 after each prefix $\sigma^{(1)}, \ldots, \sigma^{(k)}$. Notice that $\sigma^{(k)}$ can be transformed to σ_1^* by adding and deleting at most $2k$ elements. Thus, it follows from Lemma 6 that for all $i \leqslant k$ and any σ_1,

$$|M_1^{(i)}| \leqslant |M_1^{(k)}| \leqslant (1/2 - \varepsilon)|M^*| + 2k.$$

This implies that the number of edges in M^* not intersecting with edges in $M_1^{(i)}$ is at least $|M^*| - 2|M_1^{(i)}| \geqslant 2\varepsilon|M^*| - 4k$. If e_i^* is one of these edges, then it is taken in M_1. The probability for each element in M^*, which has not arrived yet, to be e_i^* is equal. Hence, conditioning on some choice of $\sigma_1^{(i)}$ the probability that e_i^* is taken in M_1 is at least $(2\varepsilon|M^*| - 4k)/(|M^*| - (i-1)) \geqslant 2\varepsilon - 4k/|M^*|$. Thus,

$$\begin{aligned}\mathbb{E}[S] &\geqslant \sum_{i=1}^{k} \mathbb{P}[e_i^* \in M_1] \geqslant k\left(2\varepsilon - \frac{4k}{|M^*|}\right) \\ &\geqslant \rho|M^*|\left(2\varepsilon - 4\frac{\rho|M^*|+1}{|M^*|}\right) = \left(2\varepsilon\rho - 4\rho^2 - \frac{4\rho}{|M^*|}\right)|M^*|.\end{aligned}$$

With assumption $M^* > 2/\rho$, (1) we conclude that

$$\mathbb{E}[|M_1|] \geqslant \left(\frac{1}{2} + 2\varepsilon\rho - 6\rho^2\right)|M^*| = \left(\frac{1}{2} + \frac{\varepsilon^2}{8}\right)|M^*|.$$

Taking the worst of the bounds, we calculate the constant

$$\gamma = \min\left\{\varepsilon, \frac{1}{32}\left(\frac{4-43\varepsilon}{4-8\varepsilon}\right)^2(1-\varepsilon) - \varepsilon, \frac{\varepsilon^2}{8}\right\} = \frac{1}{20000}.$$

◀

B Omitted proofs for submodular function maximization

We start by briefly recalling the algorithm. We refer the reader to Section 3.2 for the complete details.

The algorithm produces a tree of height at most k (cardinality constraint) and each root to node (at height i) path corresponds to a solution (of i elements). Each node has at most $|I|$(set of all possible increments) children. At the beginning of the stream, the root of the tree stores the empty set. For each element e in the stream, we add it as a child of any node T at height less than k, if for no existing child c of T it holds that $f(c \mid S)=f(e \mid S)$ where S denotes the solution corresponding to the path from root to T. At the end of the stream, the algorithm produces the best solution among all leaves.

We will choose I to be a set of size $O(k)$. For this however, we first need to know OPT. We show below how to remove this assumption with a small increase in space.

B.1 Assumption that OPT is known

The algorithm presented in Section 3.2 uses the assumption that it knows the value OPT. We will describe in the following how to remove this assumption. We use the same trick as described in [2]. Let $\delta > 0$ be a small constant. It is easy to see that when using some value $g \leqslant \mathrm{OPT}$ instead of OPT, the algorithm produces a solution of value at least $0.5506g$. We will run the algorithm in parallel for multiple guesses of g. If $g \leqslant \mathrm{OPT} \leqslant (1+\delta)g$ for some guess, we would only loose a factor of $(1+\delta)$ in the approximation ratio. However, the range in which OPT lies cannot be bounded. Hence, we must adapt our guesses as we read the stream. To that end, we keep track of the maximum element in the stream at any point of time. Let m_i denote the maximum element after observing the first i elements of the stream σ, i.e., $m_i = \max_{j \leqslant i} f(\{\sigma_j\})$. It is easy to see that any subset of $\sigma_1, \ldots, \sigma_i$ with cardinality at most k has a value between m_i and $k \cdot m_i$. Let

$$G_i = \left\{ (1+\delta)^j : j \in \mathbb{Z}, \frac{1}{1+\delta} m_i \leqslant (1+\delta)^j \leqslant \frac{k}{\delta} m_i \right\}.$$

At any point i in the stream, we will run our algorithm in parallel for all guesses in G_i. When a new maximum element arrives, this may remove previously existing guesses, in which case we stop the execution for this guess and dismiss its result. On the other hand, new guesses may be added and we start a new execution pretending the stream begins at σ_i.

Let us consider g, the correct guess for OPT, i.e., $g \leqslant \mathrm{OPT} \leqslant (1+\delta)g$. Once g is added to G_i, it remains in the set of guesses until the end of execution: If it was removed, this would mean it exists an element of value greater than $(1+\delta)g \geqslant \mathrm{OPT}$. However, no set smaller than k can have a value larger than OPT. It remains to check that the error induced by starting the algorithm late is not significant. Let O denote the elements from the optimal solution and $O' \subseteq O$ those that arrived before execution for g was started. All elements in O' were smaller than $g\delta/k$. Hence, $f(O') \leqslant k \cdot g\delta/k = g\delta$. This implies

$$f(O \setminus O') \geqslant f(O) - f(O') \geqslant \mathrm{OPT} - \delta g \geqslant (1-\delta)\,\mathrm{OPT}\,.$$

Hence, the approximation ratio decreases by a factor of at most $(1-\delta)/(1+\delta)$ and the space increases by a factor of $\log_{1+\delta}(k(1+\delta)/\delta) = O(1/\delta \log(k/\delta))$.

B.2 Bounding the number of increases

Recall, the tree algorithm stores roughly $|I|^k$ elements. Here $I = \{f(e \mid S) \mid S \subseteq E, |S| < k, e \in E\}$ is the set of all possible increases of f. In order to achieve a reasonable memory

bound, we need to make sure that $|I|$ is small. In the following we describe a way that bounds $|I|$ by $O(k)$ and only decreases the approximation ratio marginally.

We assume that OPT is known (see previous section). Let $\delta > 0$ be a small constant. We divide the range 0 to OPT into k/δ buckets each of size $\delta \cdot \text{OPT}/k$. The idea is that I now represents the set of possible range of increases of f where each bucket corresponds to a range. We now argue that this discretization does not affect the approximation ratio much. Recall that the recursion $R(k,h)$ was defined as follows:

$$R(k,h) = \min\left(\frac{t}{k} + \left(1 - \frac{t}{k}\right) R(k,h-1), \frac{1}{k} + \left(1 - \frac{1+t}{k}\right) R(k-1,h-1), \frac{1}{1+t}\right).$$

In Section 3.4, $R(k,k) \cdot \text{OPT}$ was proven to be a lower bound on the expected value of the solution returned by the algorithm. Due to bucketing, the element that is picked now might differ in value by at most $(\text{OPT} \cdot \delta/k)$ from the value promised by the analysis. As the recursion $R(k,k)$ has depth k (the solution S_k consists of k elements), it is not hard to see that one can lower bound the expected value of the returned solution after bucketing by $R(k,k) \cdot \text{OPT} - k \cdot O(\delta/k)\,\text{OPT}$. Thus the loss incurred by discretization can be made arbitrarily small by appropriately selecting $\delta > 0$.

B.3 Analysis of recursion function

In order to prove that our algorithm is 0.55 competitive, we will lower bound the value of the recursion $R(k,h)$ where $R(k,h)$ was defined as the lower bound on the approximation ratio of the solution (we defined in Section 3.4) at height i as compared to OPT over all submodular functions, streams and opt elements. The heart of the proof lies in the fact that for a certain threshold t and large k, the complicated recursion which involves taking the minimum of three terms simplifies to a recursion (that one can solve easily) involving just the first term. We prove this in Lemma 11. For complete details on the recursion definition and the solution we analyze, we refer the reader to Section 3.4.

We first state some technical claims which would be helpful in proving Lemma 11.

▷ Claim 8. The following inequality is satisfied for all $x \in [-0.1, 0]$:

$$e^x - \frac{x^2}{2} \leqslant 1 + x \leqslant e^x - \frac{x^2}{2} - \frac{x^3}{6}.$$

Proof. We first write down the taylor expansion of e^x:

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}.$$

For $x \in [-0.1, 0]$,

$$\sum_{i=3}^{\infty} \frac{x^i}{i!} \leqslant -\frac{|x|^3}{6} + \frac{|x|^3}{24} \cdot \sum_{i=1}^{\infty} |x|^i = -\frac{|x|^3}{6} + \frac{|x|^3}{24} \cdot \frac{1}{1-|x|} \leqslant 0.$$

Hence, $e^x \leqslant 1 + x + \frac{x^2}{2}$.
Similarly,

$$\sum_{i=4}^{\infty} \frac{x^i}{i!} \geqslant \frac{|x|^4}{24} - \frac{|x|^4}{120} \cdot \sum_{i=1}^{\infty} |x|^i = \frac{|x|^4}{24} - \frac{x^4}{120} \cdot \frac{1}{1-|x|} \geqslant 0.$$

Hence, $e^x \geqslant 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$.

◀

▷ Claim 9. For $k \geqslant 999$ and $t \leqslant 1$, the following inequality is satisfied:

$$\frac{1}{2}\sum_{i=2}^{\infty}\left(\frac{t^2 \cdot e^{\frac{t}{k}}}{1.9 \cdot k}\right)^i \leqslant \frac{t^4}{3 \cdot k^2}.$$

Proof. By using formula for sum of an infinite geometric series:

$$\frac{1}{2}\sum_{i=2}^{\infty}\left(\frac{t^2 \cdot e^{\frac{t}{k}}}{1.9 \cdot k}\right)^i = \frac{t^4 \cdot e^{\frac{2 \cdot t}{k}}}{1.9^2 \cdot k^2} \cdot \frac{1}{1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{1.9 \cdot k}} \leqslant \frac{t^4}{3 \cdot k^2}.$$

◀

Now we prove a closed form expression for $R(k,h)$ when only the first rule is applied.

▷ Claim 10. For any integer $k \geqslant 1000$ and any non-negative integer $h \leqslant k$ such that $R(k,h') = t/k + (1 - t/k)R(k,h'-1)$ for all $1 \leqslant h' \leqslant h$ it holds that

$$R(k,h) = 1 - \left(1 - \frac{t}{k}\right)^h.$$

In fact, Lemma 11 will show that for large values of k (and any h) the condition and thereby this closed form holds.

Proof. We will prove the equality by induction on h for any $k \geqslant 1000$.
For $h = 0$, the equality holds as $R(k,h) = 0$. Then by induction hypothesis

$$\begin{aligned} R(k,h) &= \frac{t}{k} + \left(1 - (1 - \frac{t}{k})^{h-1}\right) \cdot (1 - \frac{t}{k}) \\ &= \frac{t}{k} + 1 - \frac{t}{k} - \left(1 - \frac{t}{k}\right)^h \\ &= 1 - \left(1 - \frac{t}{k}\right)^h. \end{aligned}$$

◀

▶ **Lemma 11.** *With $t = 0.8$ for every $k \geqslant 1000$ and $h \leqslant k$ it holds that*

$$R(k,h) = \frac{t}{k} + \left(1 - \frac{t}{k}\right) R(k,h-1).$$

Proof. We will prove by induction on k and h.
For $h = 1$, $R(k,1) = \frac{t}{k}$. For $k = 1000$, we have verified that the induction hypothesis holds by computer assisted calculation. Hence the base case holds.
Recall that $R(k,h)$ is defined as:

$$R(k,h) = \min\left(\frac{t}{k} + R(k,h-1) \cdot \left(1 - \frac{t}{k}\right), \frac{1}{k} + R(k-1,h-1) \cdot \left(1 - \frac{1+t}{k}\right), \frac{1}{1+t}\right).$$

By induction hypothesis and Claim 10, we know that $R(k,h-1) = 1 - (1 - t/k)^{h-1}$. Hence for $k \geqslant 1000$ and $h \leqslant k$, we get:

$$\begin{aligned} \frac{t}{k} + R(k,h-1) \cdot \left(1 - \frac{t}{k}\right) &= \frac{t}{k} + (1 - (1 - \frac{t}{k})^{h-1}) \cdot \left(1 - \frac{t}{k}\right) \\ &= 1 - (1 - \frac{t}{k})^h \\ &\leqslant 1 - (1 - \frac{t}{k})^k. \end{aligned}$$

As $(1-\frac{t}{k})^k$ is monotonically decreasing in k and $t = 0.8$, we get:

$$\begin{aligned}\frac{t}{k} + R(k, h-1) \cdot \left(1 - \frac{t}{k}\right) &\leqslant 1 - (1 - \frac{t}{1000})^{1000} \\ &\leqslant 0.5509 \\ &\leqslant \frac{1}{1+t}.\end{aligned}$$

Hence it suffices to prove the below for $k \geqslant 1000$ and $h \leqslant k$:

$$\frac{t}{k} + R(k, h-1) \cdot \left(1 - \frac{t}{k}\right) \leqslant \frac{1}{k} + R(k-1, h-1) \cdot \left(1 - \frac{1+t}{k}\right).$$

Or, equivalently:

$$t \leqslant 1 + R(k-1, h-1) \cdot (k-1-t) - R(k, h-1) \cdot (k-t).$$

By induction hypothesis and Claim 10, we know that $R(k, h-1) = 1 - (1-\frac{t}{k})^{h-1}$ and $R(k-1, h-1) = 1 - (1-\frac{t}{k-1})^{h-1}$.

$$\begin{aligned}t &\leqslant 1 + (1 - (1 - \frac{t}{k-1})^{h-1}) \cdot (k-1-t) - (1 - (1 - \frac{t}{k})^{h-1}) \cdot (k-t) \\ &\leqslant 1 + \left(k - 1 - t - \frac{(k-1-t)^h}{(k-1)^{h-1}}\right) - \left(k - t - \frac{(k-t)^h}{(k)^{h-1}}\right) \\ &\leqslant \frac{(k-t)^h}{(k)^{h-1}} - \frac{(k-1-t)^h}{(k-1)^{h-1}} \\ &\leqslant \underbrace{k \cdot (1 - \frac{t}{k})^h - (k-1) \cdot (1 - \frac{t}{k-1})^h}_{E_1}. \end{aligned} \tag{2}$$

E_1 is monotonically decreasing in h as shown below:

$$\begin{aligned}\left(k \cdot (1 - \frac{t}{k})^h - (k-1) \cdot (1 - \frac{t}{k-1})^h\right) - \left(k \cdot (1 - \frac{t}{k})^{h+1} - (k-1) \cdot (1 - \frac{t}{k-1})^{h+1}\right) &\geqslant 0 \\ k \cdot (1 - \frac{t}{k})^h \cdot (1 - (1 - \frac{t}{k})) - (k-1) \cdot (1 - \frac{t}{k-1})^h \cdot (1 - (1 - \frac{t}{k-1})) &\geqslant 0 \\ t \cdot (1 - \frac{t}{k})^h - t \cdot (1 - \frac{t}{k-1})^h &\geqslant 0.\end{aligned}$$

Hence we can lower bound E_1 by assuming $h = k$. We lower bound E_1 below:

$$E_1 \geqslant k \cdot (1 - \frac{t}{k})^k - (k-1) \cdot (1 - \frac{t}{k-1})^k.$$

By using Claim 8, $k \geqslant 1000$ and $t \leqslant 1$, we get:

$$E_1 \geqslant k \cdot (e^{\frac{-t}{k}} - \frac{t^2}{2 \cdot k^2})^k - (k-1) \cdot \left(\underbrace{e^{\frac{-t}{k-1}} - \frac{t^2}{2 \cdot (k-1)^2} + \frac{t^3}{6 \cdot (k-1)^3}}_{T_1} \right)^k$$
$$\geqslant k \cdot e^{-t} \cdot \underbrace{\left(1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k^2}\right)^k}_{T_2} - (k-1) \cdot T_1 \cdot e^{-t} \cdot \underbrace{\left(1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)^2} + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)^3}\right)^{k-1}}_{T_3}. \tag{3}$$

We lower bound the term T_2.

$$T_2 = \left(1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k^2}\right)^k$$
$$\geqslant 1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k} - \sum_{i=2}^{\infty} \frac{k^i}{2} \cdot \left(\frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k^2}\right)^i \text{ (By Binomial Expansion.)}$$
$$= 1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k} - \frac{1}{2} \sum_{i=2}^{\infty} \left(\frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k}\right)^i$$
$$\geqslant 1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k} - \frac{t^4}{3 \cdot k^2}. \text{ (By Claim 9, } t \leqslant 1 \text{ and } k \geqslant 1000.)$$

We now simplify and lower bound $k \cdot e^{-t} \cdot T_2$:

$$k \cdot e^{-t} \cdot T_2 \geqslant k \cdot e^{-t} \cdot (1 - \frac{t^2 \cdot e^{\frac{t}{k}}}{2 \cdot k} - \frac{t^4}{3 \cdot k^2})$$
$$= k \cdot e^{-t} - \frac{t^2 \cdot e^{\frac{t}{k}} \cdot e^{-t}}{2} - \frac{t^4 \cdot e^{-t}}{3 \cdot k}. \tag{4}$$

We now upper bound T_3.

$$T_3 = \left(1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)^2} + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)^3}\right)^{k-1}.$$

By using Binomial Expansion, we get:

$$T_3 \leqslant 1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)^3} + \sum_{i=2}^{\infty} \frac{(k-1)^i}{2} \cdot \left(\frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)^2} + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)^3}\right)^i$$
$$= 1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)^3} + \frac{1}{2} \cdot \sum_{i=2}^{\infty} \left(\frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)^2}\right)^i.$$

By using that $t \leqslant 1$ and $k \geqslant 1000$, we get:

$$\begin{aligned} T_3 &\leqslant 1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{t^3}{3 \cdot (k-1)^2} + \frac{1}{2} \cdot \sum_{i=2}^{\infty} \left(\frac{t^2 \cdot e^{\frac{t}{k-1}}}{1.9 \cdot (k-1)} \right)^i \\ &\leqslant 1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{t^3}{3 \cdot (k-1)^2} + \frac{t^4}{3 \cdot (k-1)^2} \text{ (By Claim 9, } t \leqslant 1 \text{ and } k \geqslant 1000.) \\ &\leqslant 1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{2 \cdot t^3}{3 \cdot (k-1)^2}. \end{aligned}$$

We now upper bound T_1. By using Claim 8, $k \geqslant 1000$ and $t \leqslant 1$, we get:

$$T_1 \leqslant 1 - \frac{t}{k-1} + \frac{t^3}{6 \cdot (k-1)^3}.$$

We now simplify and upper bound $(k-1) \cdot T_1 \cdot e^{-t} \cdot T_3$:

$$\begin{aligned} &(k-1) \cdot T_1 \cdot e^{-t} \cdot T_3 \\ &\leqslant (k-1) \cdot e^{-t} \cdot T_1 \cdot (1 - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} + \frac{2 \cdot t^3}{3 \cdot (k-1)^2}) \\ &= e^{-t} \cdot T_1 \cdot ((k-1) - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2} + \frac{2 \cdot t^3}{3 \cdot (k-1)}) \\ &\leqslant e^{-t} \cdot (1 - \frac{t}{k-1} + \frac{t^3}{6 \cdot (k-1)^3}) \cdot ((k-1) - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2} + \frac{2 \cdot t^3}{3 \cdot (k-1)}) \\ &\leqslant e^{-t} \cdot \left(\left((k-1) - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2} + \frac{2 \cdot t^3}{3 \cdot (k-1)} \right) + \left(-t + \frac{t^3 \cdot e^{\frac{t}{k-1}}}{2 \cdot (k-1)} \right) + \left(\frac{t^2 \cdot e^{\frac{t}{k-1}}}{3 \cdot (k-1)^2} \right) \right) \\ &\leqslant e^{-t} \cdot \left(k - 1 - t - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2} + \frac{7 \cdot t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)} + \frac{t^2 \cdot e^{\frac{t}{k-1}}}{3 \cdot (k-1)^2} \right). \end{aligned} \tag{5}$$

By replacing (4) and (5) in (3), we get:

$$\begin{aligned} E_1 &\geqslant k \cdot e^{-t} - \frac{t^2 \cdot e^{\frac{t}{k}} \cdot e^{-t}}{2} - \frac{t^4 \cdot e^{-t}}{3 \cdot k} - e^{-t} \cdot (k - 1 - t - \frac{t^2 \cdot e^{\frac{t}{k-1}}}{2} + \frac{7 \cdot t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)} + \frac{t^2 \cdot e^{\frac{t}{k-1}}}{3 \cdot (k-1)^2}) \\ &= e^{-t} \cdot (1+t) + e^{-t} \cdot (\frac{t^2 \cdot e^{\frac{t}{k-1}}}{2} - \frac{t^2 \cdot e^{\frac{t}{k}}}{2}) - e^{-t} \cdot (\frac{t^4}{3 \cdot k} + \frac{7 \cdot t^3 \cdot e^{\frac{t}{k-1}}}{6 \cdot (k-1)}) - e^{-t} \cdot \frac{t^2 \cdot e^{\frac{t}{k-1}}}{3 \cdot (k-1)^2} \\ &\geqslant e^{-t} \cdot (1+t) - e^{-t} \cdot \frac{2 \cdot t^3 \cdot e^{\frac{t}{k-1}}}{(k-1)} - e^{-t} \cdot \frac{t^2 \cdot e^{\frac{t}{k-1}}}{3 \cdot (k-1)^2}. \end{aligned} \tag{6}$$

By using (2) and (6), we get that it suffices to prove the following:

$$t \leqslant e^{-t} \cdot (1+t) - e^{-t} \cdot \frac{2 \cdot t^3 \cdot e^{\frac{t}{k-1}}}{(k-1)} - e^{-t} \cdot \frac{t^2 \cdot e^{\frac{t}{k-1}}}{3 \cdot (k-1)^2}.$$

Here, $t = 0.8$ satisfies the final inequality. ◀

▶ **Lemma 8.** *For all positive integers k, $R(k,k) \geqslant 0.5506$.*

Proof. For $k \leqslant 1000$, $R(k,k) \geqslant 0.5506$ can easily be verified with computer assistance, since it involves only a dynamic program of size 1000×1000. For $k > 1000$, by Lemma 11 and Claim 10, $R(k,k) \geqslant 1 - (1 - \frac{0.8}{k})^k \geqslant 1 - e^{-0.8} \geqslant 0.5506$. ◀