Sampling for Beyond-Worst-Case Online Ranking*

Qingyun Chen[1], Sungjin Im[1], Benjamin Moseley[2], Chenyang Xu[3], Ruilong Zhang[4]

[1] Electrical Engineering and Computer Science, University of California at Merced
[2] Tepper School of Business, Carnegie Mellon University
[3] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
[4] Department of Computer Science and Engineering, University at Buffalo
qchen41@ucmerced.edu, sim3@ucmerced.edu, moseleyb@andrew.cmu.edu, cyxu@sei.ecnu.edu.cn, ruilongz@buffalo.edu

Abstract

The feedback arc set problem is one of the most fundamental and well-studied ranking problems where n objects are to be ordered based on their pairwise comparison. The problem enjoys several efficient approximation algorithms in the offline setting. Unfortunately, online there are strong lower bounds on the competitive ratio establishing that no algorithm can perform well in the worst case. This paper introduces a new beyond-worst-case model for online feedback arc set. In the model, a sample of the input is given to the algorithm offline before the remaining instance is revealed online. This models the case in practice where yesterday's data is available and is similar to today's online instance. This sample is drawn from a known distribution which may not be uniform. We design an online algorithm with strong theoretical guarantees. The algorithm has a small constant competitive ratio when the sample is uniform—if not, we show we can recover the same result by adding a provably minimal sample. Empirical results validate the theory and show that such algorithms can be used on temporal data to obtain strong results.

1 Introduction

Finding a global ranking of items is a common problem faced in social network analysis, Bayesian learning, information aggregation, and game design (Bar-Yehuda et al. 1994; Baweja, Jia, and Woodruff 2022). The case where pairwise preference relationships are available for constructing the global ranking is referred to as Kemeny-Young Rank Aggregation (Kenyon-Mathieu and Schudy 2007a). For example, consider users that are playing an online game. Given the features of a user's playing history, it is possible to decide a preference on which of the two users should be ranked higher. However, the ranking may be inconsistent on three users due to the numerous features. The goal is to construct a global ranking that corresponds to a ranking of all users adhering to as many of the pairwise preferences.

One of the oldest and most well-studied approaches for ranking is the Feedback Arc Set (FAS) problem in tournaments. In the FAS problem, there is a directed graph $G = (V, E)$. Nodes represent objects to be ranked. There is an edge between each pair of nodes that are directed, in which (u, v) implies v is better than u. The goal is to rank (i.e. order) the nodes in V so that the fewest number of edges are directed backward in the ordering. Equivalently, the goal is to reverse the direction of the smallest number of edges so that the graph has no directed cycles and thus admits a topological sort.

A large number of applications of FAS have led to it being well-studied theoretically and empirically. The problem is NP-Hard (Alon 2006; Charbit, Thomassé, and Yeo 2007), and prior work has focused on approximation and heuristic algorithm design. Several constant approximation algorithms are known (Ailon, Charikar, and Newman 2008; Coppersmith, Fleischer, and Rudra 2006). Moreover, several scalable algorithmic approaches have been introduced for large data sets (Baweja, Jia, and Woodruff 2022; Im and Montazer Qaem 2020; Simpson, Srinivasan, and Thomo 2016a).

While Feedback Arc Set is well understood offline and enjoys several good heuristic approaches, few results are known when items arrive over time. When items arrive over time, two possible models are the streaming and online settings. In the streaming setting, items come one at a time, and, using limited memory, the algorithm needs to construct a global ordering of the elements. The ordering can be changed as elements arrive; the challenge is that not all edges of the graph can be stored in memory. There has been a recent surge of interest in streaming algorithms (Baweja, Jia, and Woodruff 2022; Chen et al. 2021).

In the online model, nodes arrive sequentially, revealing the directed edges connected to the nodes that have appeared earlier. The objective in this model is to continually provide a reliable approximation of the nodes that have surfaced so far. To delineate the model's goals more explicitly, the following criteria must be met: (1) At every moment, an ordered list of the nodes that have arrived thus far should be maintained. (2) The relative order of any subset of nodes should remain unchanged throughout the process. (3) The model should ensure an outcome of high quality when compared to the optimal offline solution.

While this setting is natural; unfortunately, there is a $\Omega(n)$ lower bound on the competitive ratio; the proof is omitted in this version) in the worst-case online arrival model.

In the worst-case model, the problem is too difficult to

*All authors (ordered alphabetically) have equal contributions and are corresponding authors.

solve online. Fortunately, practical applications are often more forgiving than these worst-case conditions. Take online ranking as an example; we frequently have access to a history of previous rankings. This historical data can be invaluable in constructing current and future rankings.

In this paper, we model this pragmatic scenario as follows: The algorithm receives a sampling of nodes to be ranked, along with the edges among them. At the most basic level, this might involve a uniform sample, but our model can also accommodate non-uniform distributions. Subsequently, additional nodes requiring ranking will arrive online. This paper focuses on the question: can the presence of a sample effectively bypass worst-case lower bounds, thereby yielding beyond-worst-case robust results for online ranking?

1.1 Our Contributions

This paper studies beyond worst-case online models for the feedback arc set problem. We first consider the pure online case where we show in Theorem 2 that any algorithm is $\Omega(n)$ competitive. Further, we show in Theorem 1 a matching upper bound by giving a $O(n)$-competitive algorithm.

We then turn to beyond-worst-case models to overcome the strong lower bounds. In each model, the algorithm is given a (possibly non-uniform) sample S of the nodes and their pairwise edges. The remaining nodes arrive online in *adversarial order*. The sample represents data on past instances the algorithm has access to when solving a new online instance.

We first consider a model introduced by Lattanzi et al. (Lattanzi et al. 2021) and further studied by Argue et al. (Argue et al. 2022). In this setting, the algorithm is given access to a *uniform sample* of the data. The goal is to show that the algorithm has strong performance on the remaining problem instance that arrives adversarially. In this model, we go beyond the worst-case lower bound and show that the algorithm has a *constant* competitive ratio when the sample is only a small λ faction of the items (Theorem 3). We also show that our ratio is essentially optimal (Theorem 4).

We then introduce a new online model where we are given an initial sample, which is drawn from a distribution $\mathbb{Q}$ that is not necessarily uniform. Then the goal is for the algorithm to decide on a small sampling of the remaining nodes to ensure the algorithm has strong theoretical performance. Intuitively, the initial sample is easy to obtain, yet may be skewed. We want to correct the sample by adding a minimum-sized sample as it could be costly. We show in Theorem 5 an algorithm that can achieve a constant competitive ratio by sampling a small fraction of the remaining nodes even in the worst case.We show the optimality of the extra sample in terms of their size (Theorem 6). In the full version, we further extend our new model to the related correlation clustering problem and show that our algorithm is able to establish similar results.

A summary of our theoretical results can be found in Table 1. These results give the first positive results for FAS in the online setting. We further validate the theory empirically by establishing that these algorithms have similar performance compared to offline algorithms that know the entire input, even if the data is adversarially ordered. As strong evidence of the usefulness of the algorithmic ideas developed, we show that on temporal data where the sample is constructed based on the earliest arriving nodes, that the algorithms have minimal loss over the offline setting. This temporal model well captures the situation in practice where the sample corresponds to past data.

1.2 Related Work

There are several known approximation algorithms for FAS. It is possible to achieve a $(1+\epsilon)$ approximation in polynomial time (Ailon 2012; Kenyon-Mathieu and Schudy 2007a). It has been shown that a simple deterministic greedy algorithm is 5-approximated (Coppersmith, Fleischer, and Rudra 2010), i.e., order the vertices in increasing order of their indegrees where ties are broken arbitrarily. A popular randomized greedy algorithm is Pivot (also known as Kwiksort) which achieves a 3-approximation (Ailon, Charikar, and Newman 2008). When the underlying graph is not a tournament, (Even et al. 1998) gives a $O(\log n/\log\log n)$-approximation.

Designing online algorithms models for beyond-worst-case analysis is a popular topic. Works such as (Lattanzi et al. 2020; Lykouris and Vassilvitskii 2021) have been investigating how to augment online algorithms with machine-learned predictions. Argue et al. (Argue et al. 2022), and Lattanzi et al. (Lattanzi et al. 2021) considered a similar model where the algorithm is given a random sample of the problem input and the rest of the input arrives in an adversarial order. In all of these cases, interesting algorithmic techniques and beyond-worst-case bounds emerge by allowing the algorithm to use extra information.

2 Preliminaries

This paper considers the online version of the minimum feedback arc set problem. We first state the classical offline version of the problem for completeness.

Definition 1 (Minimum Feedback Arc Set)**.** *We are given a tournament $G := (V, E)$ with $|V| = n$, where a tournament is a directed graph $G := (V, E)$ such that for each pair of vertices $i, j \in V$, either $(i, j) \in E$ or $(j, i) \in E$. The goal is to find a permutation π on V minimizing the number of backward edges with respect to π. An edge $(i, j) \in E$ is called a backward edge with respect to π if and only if π ranks j before i (denoted by $j <_\pi i$).*

In the online setting, the vertices of the input graph arrive one by one. When a vertex arrives, its edges to all previous arrivals are released to the algorithm. The algorithm has to maintain an order of all arrived vertices at all times; that is, the algorithm must make an irrevocable decision upon each vertex arrival. The algorithm knows the size of the graph prior, i.e., the number of vertices is known by the algorithm.

We show in Theorem 2 that the above online problem has a strong $\Omega(n)$ lower bound, where n is the number of vertices. We also provide an optimal algorithm for this fully online model. Theorem 2 suggests that obtaining any constant competitive ratio is impossible in the online model. Thus, we consider a semi-online model proposed by (Lattanzi et al. 2021).

Arrival Order	Upper Bound	Lower Bound
Offline	PTAS (Kenyon-Mathieu and Schudy 2007b)	NP-Hard (Alon 2006)
Adversarial	$O(n)$ (Theorem 1)	$\Omega(n)$ (Theorem 2)
Uniform Sample	$O(\frac{1}{\lambda})$ (Theorem 3)	$\Omega(\frac{1}{\lambda})$ (Theorem 4)
Extra Sample	$O(\delta(\mathbb{Q}, \mathbb{U}))$ (Theorem 5)	$\Omega(\delta(\mathbb{Q}, \mathbb{U}))$ (Theorem 6)

Table 1: The summary of our results. Offline order refers to the classical FAS problem. Adversarial order is the standard worst-case setting. Uniform sample (formally termed semi-online; see Definition 2) is the model where the algorithm has access to a uniform random sample with λn vertices, and the remaining vertices arrive in adversarial order. Extra sampling (see Definition 3) corresponds to the new sampling model where we are given a set of vertices that are sampled from a predefined distribution $\mathbb{Q}$, and the algorithm can do some extra sampling, before seeing the remaining vertices arriving adversarially. For simplicity, define $\delta(\mathbb{Q}, \mathbb{U})$ as the total variance distance between distributions $\mathbb{Q}$ and $\mathbb{U}$, where $\mathbb{U}$ is the uniform distribution.

Definition 2 (Semi-online Minimum Feedback Arc Set)**.** *In the semi-online setting, the vertices arrive in two phases: offline and online. In the first phase (offline phase), the algorithm is given a vertex set $S \subseteq V$ and its induced subgraph $G[S]$. The offline vertices S with $|S| := \lambda n$ are uniformly randomly sampled from V. In the online phase, the remainder of the vertices $V \setminus S$ arrives online. When a vertex arrives, its edges to all previous arrivals are released to the algorithm. Upon each vertex's arrival, the algorithm must make an irrevocable position decision. The objective is to maintain an order of arrived vertices to minimize the number of backward edges.*

For the semi-online setting, we show that a natural adaptation of the pivot algorithm proposed by (Ailon, Charikar, and Newman 2008) is essentially optimal. To make our results more robust, we extend the problem to the setting where the offline vertices may not be sampled by a uniform random distribution $\mathbb{U}$. The adversary may sample vertices by some other distributions $\mathbb{Q}$. Since distribution $\mathbb{Q}$ is input and can be arbitrary, it is impossible to get any positive result if we do not sample some extra vertices by the lower bound shown in Theorem 2. Thus, we are interested in how many extra vertices we need to sample to make the problem still admit a $O(\frac{1}{\lambda})$-competitive algorithm, i.e., recover the result obtained in the semi-online model.

Definition 3 (Extra Sampling Minimum Feedback Arc Set)**.** *In the offline phase, the algorithm is given a vertex set $S \subseteq V$ with $\mathbb{E}[|S|] := \lambda n$ and its induced subgraph $G[S]$, where the offline vertices S are sampled from V by a known distribution $\mathbb{Q}$. And the algorithm is allowed to sample some extra vertices L from V by some distribution $\mathbb{D}$, which is determined by the algorithm. In total, the algorithm can access the subgraph $G[L \cup S]$ in the offline phase. In the online phase, the remainder of the vertices $V \setminus (L \cup S)$ arrive online, and the algorithm must make an irrevocable position decision upon each vertex's arrival. The goal is to sample another set of vertices as less as possible such that the problem admits a $O(\frac{1}{\lambda})$-competitive algorithm.*

3 Online Minimum Feedback Arc Set

This section considers the online minimum feedback arc set problem under two settings: (i) Fully Online Model (Section 3.2): the adversary picks an arriving order of all vertices; (ii) Semi-online Model (Section 3.3): the algorithm is allowed to uniform sample a subset S from V, and then the adversary picks an arriving order of vertices in $V \setminus S$.

The algorithms for the two models share the same framework. The difference is that they access different vertex orders and thus obtain different competitive ratios. In the full online Model, the algorithm accesses an adversary order of all vertices while it combines a uniformly random order and adversary order in semi-online model. Our algorithm adapts the classical pivot algorithm proposed by (Ailon, Charikar, and Newman 2008). For completeness, we restate the algorithm in the full version of this paper.

The performance of the pivot algorithm heavily depends on the order of vertices. It has been shown by (Ailon, Charikar, and Newman 2008) that it is 3-approximated when the order of vertices is uniformly random. In the online setting, the algorithm does not access the whole graph and thus cannot obtain a random order of vertices. The pivot algorithm can be naturally generalized to the online setting by considering the arriving order of vertices. However, the competitive ratio is no longer a simple constant in this case. As we will see in Section 3.2, the Pivot algorithm is $O(n)$-competitive when all vertices arrive in an adversary order. In contrast, we show in Section 3.3 that the competitive ratio is improved to $O(\frac{1}{\lambda})$ when partial vertices arrive in random order.

3.1 Algorithmic Framework

For the convenience of analysis, we describe our algorithm as a binary tree construction algorithm. The binary tree construction is similar to the classical binary search tree construction. Given an order σ of all vertices, we construct a binary tree $\mathbf{T}$ iteratively. When t-th vertex (denoted by $\sigma(t)$) arrives, we check the direction of the edge between $\sigma(t)$ and $\mathbf{r}$ (the root of the tree $\mathbf{T}$). If $(\sigma(t), \mathbf{r}) \in E$, then $\sigma(t)$ goes to the left subtree; otherwise, it goes to the right subtree. We repeat the above process until we read in all vertices. The formal description can be found in Algorithm 1 and Algorithm 2. An example with vertex order $\sigma := (v_1, v_2, v_6, v_3, v_4, v_5)$ can be found in Fig. 1.

In the fully online model, Algorithm 1 takes the adversary order σ as the input; while in the semi-online model, Algorithm 1 takes $\sigma := \sigma_S + \sigma_{V \setminus S}$ as the input, where σ_S is a random order of vertices in S and $\sigma_{V \setminus S}$ is the adversary order of vertices in $V \setminus S$.

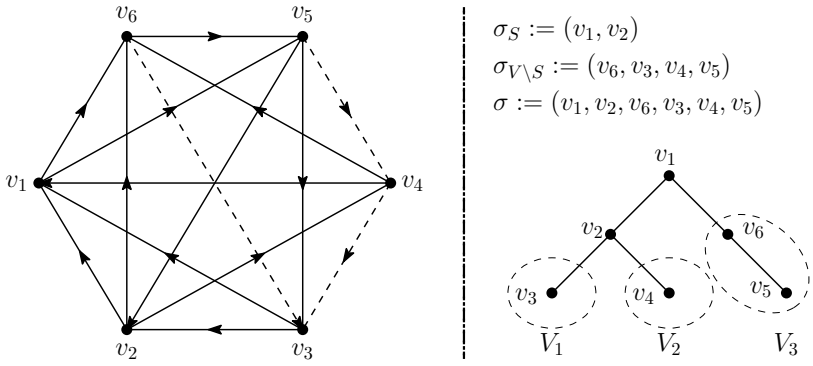


Figure 1: Illustration for the online pivot algorithm's processing. Suppose that $S := \{ v_1, v_2 \}$ and a uniform random order $\sigma_S = (v_1, v_2)$. The adversary order for $V \setminus S$ is $\sigma_{V\setminus S} = (v_6, v_3, v_4, v_5)$. Initially, the binary tree $\mathbf{T}$ is rooted by v_1, and v_2 is v_1's left child since $(v_2, v_1) \in E$. When v_6 arrives, it will be inserted as the right child of v_1 since $(v_1, v_6) \in E$. When v_3 arrives, the algorithm will first determine whether it belongs to the left or right subtree of the tree rooted at v_1. Since $(v_3, v_1) \in E$, v_3 is part of the left subtree, and v_3 will be inserted as v_2's left child since $(v_3, v_2) \in E$. After the construction, we get a binary tree shown in the right part. The inorder traversal is the algorithm's output, i.e., $\pi = (v_3, v_2, v_4, v_1, v_6, v_5)$. Thus, there are three backward edges (v_5, v_4), (v_4, v_3) and (v_6, v_3) marked by dashed-lines in the original tournament. By our notation in Section 3.3, we have $\mathcal{A} = \{ ((v_5, v_4), (v_4, v_3), (v_6, v_3)) \}$. Moreover, by the definition of the lowest common ancestor, we have $\mathsf{lca}(v_4, v_5, \mathbf{T}) = v_1$, $\mathsf{lca}(v_3, v_6, \mathbf{T}) = v_1$ and $\mathsf{lca}(v_3, v_4, \mathbf{T}) = v_2$. This implies that $\mathcal{A}(S) = \mathcal{A}$ and $\mathcal{A}(V \setminus S) = \varnothing$.

Algorithm 1: Main Algorithm for Online Minimum Feedback Arc Set

Input: An order $\sigma := (v_1, \ldots, v_n)$ of all vertices in V.
Output: A permutation $\pi : [n] \to [n]$ of all vertices in V.

$\mathbf{T} \leftarrow \varnothing$; $i \leftarrow 1$
while $i \leq n$ **do**
  Call `construct-tree`$(\mathbf{T}, \mathbf{r} \leftarrow v_1, v_i)$. ▷ Insert v_i to $\mathbf{T}$ rooted by $\mathbf{r}$.
  $i \leftarrow i + 1$.
end while
Let π be the inorder traversal of binary tree $\mathbf{T}$.
return The vertex order π.

Algorithm 2: `construct-tree`$(\mathbf{T}, \mathbf{r}, v)$

Input: A binary tree $\mathbf{T}$ rooted by $\mathbf{r}$ and a vertex v.
Output: A new binary tree $\mathbf{T}$ with v inserted.

if $\mathbf{r} = \varnothing$ **then**
  $\mathbf{r} \leftarrow v$; ▷ Set up the root $\mathbf{r}$ of $\mathbf{T}$.
end if
if $(\mathbf{r}, v) \in E$ **then** ▷ Insert to the right subtree.
  $u \leftarrow \mathbf{r}$'s right child.
  `construct-tree`$(\mathbf{T}, u, v)$.
end if
if $(v, \mathbf{r}) \in E$ **then** ▷ Insert to the left subtree.
  $\ell \leftarrow \mathbf{r}$'s left child.
  `construct-tree`$(\mathbf{T}, \ell, v)$.
end if

In the following, we present several concepts and observations which are helpful for the analysis in Section 3.2 and Section 3.3. The pivot algorithm analysis critically relies on *Bad Triangle*, which is defined in Definition 4.

Definition 4 (Bad Triangle)**.** *A bad triangle in $G := (V, E)$ is a vertex set consisting of three vertices, denoted by $\mathbf{t} := \{ v_1, v_2, v_3 \}$, such that they form a directed cycle, i.e., $(v_1, v_2) \in E$, $(v_2, v_3) \in E$ and $(v_3, v_1) \in E$. Let $\mathcal{T}(G')$ be all bad triangles in subgraph $G' \subseteq G$.*

Intuitively, the number of bad triangles provides a lower bound of the optimal solution (Observation 1) and an upper bound of the algorithm's solution (Observation 2).

Observation 1. *In any feasible solution, for any bad triangle $\mathbf{t} \in \mathcal{T}(G)$, there must exist a backward edge in $\mathbf{t}$.*

Observation 2. *Let $\mathcal{A}$ be the set of backward edges generated by Algorithm 1. Then, regardless of the vertices order, for each edge $(i, j) \in \mathcal{A}$ if and only if there exists a bad triangle $\mathbf{t} \in \mathcal{T}(G)$ such that $\{ i, j \} \subseteq \mathbf{t}$.*

3.2 Fully Online Model

Due to space limitations, we only state the two main theorems in the following and defer the proofs to the full version of this paper. The first theorem gives a $O(n)$-competitive algorithm, while the second one provides an $\Omega(n)$ lower bound to close the computational gap.

Theorem 1. *Algorithm 1 is $O(n)$-competitive when the vertices order is fully arbitrary.*

Theorem 2. *There exists an instance distribution of the online feedback arc set problem where vertices arrive in adversarial order such that any randomized algorithm has a competitive ratio $\Omega(n)$, where n is the number of vertices.*

3.3 Semi-online Minimum Feedback Arc Set

In this section, we consider the semi-online minimum feedback arc set problem. The sample S is chosen from V uniformly at random, where $|S| := \lambda n$.We mainly show the following.

Theorem 3. *For the problem of semi-online minimum feedback arc set, there is a randomized algorithm that is $O(\frac{1}{\lambda})$-competitive, where λn is the number of vertices sampled in the offline phase. Moreover, any algorithm has a competitive ratio of $\Omega(\frac{1}{\lambda})$.*

Note that the vertex set S is uniformly randomly sampled from V in the current case. Recall that the input sequence of Algorithm 1 is $\sigma := \sigma_S + \sigma_{V \setminus S}$, where σ_S is a random order of vertices in S and $\sigma_{V \setminus S}$ is the adversary order of vertices in $V \setminus S$.

Notation and Analysis Framework. Let $\mathcal{A} \subseteq E$ be the set of backward edges generated by Algorithm 1, and define ALG as the objective value of the algorithm, i.e., $\mathrm{ALG} := |\mathcal{A}|$. Use $\mathbf{T}$ to denote the binary tree constructed by the algorithm. Given a tree $\mathbf{T}$ and two vertices v_1, v_2 in the tree, the *lowest common ancestor* (denoted by $\mathsf{lca}(v_1, v_2, \mathbf{T})$) of v_1 and v_2 is the lowest node in the tree $\mathbf{T}$ that has both v_1 and v_2 as descendants. Note that $\mathsf{lca}(v_1, v_2, \mathbf{T})$ is unique given the tree and two vertices. According to the property of our algorithm, we have the following observation.

Observation 3. *Let $\mathbf{T}$ be the binary tree constructed by Algorithm 1. For any backward edge $(i, j) \in \mathcal{A}$, $\{ i, j, \mathsf{lca}(i, j, \mathbf{T}) \}$ forms a bad triangle.*

Now, we split $\mathcal{A}$ into two subsets $\mathcal{A}(S)$ and $\mathcal{A}(V \setminus S)$. $\mathcal{A}(S)$ is a set of backward edges that are separated by some vertices in S, i.e., $\mathcal{A}(S) := \{ (v, v') \in \mathcal{A} \mid \mathsf{lca}(v, v', \mathbf{T}) \in S \}$, while vertices in $V \setminus S$ separate edges in $\mathcal{A}(V \setminus S)$, i.e., $\mathcal{A}(V \setminus S) := \{ (v, v') \in \mathcal{A} \mid \mathsf{lca}(v, v', \mathbf{T}) \in V \setminus S \}$. An example can be found in Fig. 1. Intuitively, for edges in $\mathcal{A}(S)$, we can charge them directly to the optimal solution by the classical analysis of the offline pivot algorithm. For edges in $\mathcal{A}(V \setminus S)$, we need more careful analysis to get rid of $\Omega(n)$ lower bound resulting from the adversary order of the vertices in $V \setminus S$. For notation consistency, define $\mathbb{E}[\mathrm{ALG}(S)] := \mathbb{E}[|\mathcal{A}(S)|]$ and $\mathbb{E}[\mathrm{ALG}(V \setminus S)] := \mathbb{E}[|\mathcal{A}(V \setminus S)|]$. Clearly, $\mathbb{E}[\mathrm{ALG}] = \mathbb{E}[\mathrm{ALG}(S)] + \mathbb{E}[\mathrm{ALG}(V \setminus S)]$. Use OPT to denote the objective value of the optimal offline solution. The two terms $\mathrm{ALG}(S)$ and $\mathrm{ALG}(V \setminus S)$ are bounded by Lemma 1 and Lemma 2 respectively.

Lemma 1. $\mathbb{E}[\mathrm{ALG}(S)] \leq 3 \cdot \mathrm{OPT}$.

Lemma 2. $\mathbb{E}[\mathrm{ALG}(V \setminus S)] \leq \frac{1}{\lambda} \cdot \mathrm{OPT}$.

Combining Lemma 1 and Lemma 2, Theorem 3 directly follows. Due to space limits, we omit the two lemma's proofs in the paper.

Remark. Our analysis leverages algorithmic and analysis ideas from previous work (Lattanzi et al. 2021; Mathieu, Sankur, and Schudy 2010) on correlation clustering. However, these two different problems lead to technical differences. The algorithms have similarities in that they recursively choose pivots, but the similarities stop there and the analysis requires considerably different techniques. In particular, the lower bounds used for the analysis are derived differently depending on the underlying problem structure and whether we include the sampled points or not. The detailed discussion can be found in the full version of the paper.

Hardness Result To complete our results, we build on the hard instance stated in Theorem 2 to give a lower bound of the semi-online model.Intuitively, we construct a large graph $G := (V, E)$ with $|V| := n$ and a vertex set $T \subseteq V$ with $|T| := \frac{1}{\lambda}$ such that the probability of each vertex in T being sampled by any algorithm is tiny. Then, by Theorem 2, any algorithm has competitive ratio $\Omega(\frac{1}{\lambda})$.

Theorem 4. *In the semi-online model, any algorithm is $\Omega(\frac{1}{\lambda})$-competitive for any $\lambda \in (0, 1)$.*

4 Extra Sampling Model

This section considers the extra sampling model. In the model, we access a predefined vertex subset S sampled from a distribution $\mathbb{Q} := \{q_v\}_{v \in V}$, where each vertex v is sampled independently with a probability of q_v and the expected number of the sampled vertices is λn. And then, the algorithm is allowed to sample some extra vertices. The goal is to make the extra sampling model still admit a $O(\frac{1}{\lambda})$-competitive algorithm. Use $\mathbb{U} := \{p_v = \lambda\}_{v \in [n]}$ to represent the uniform sampling distribution—to represent that $\mathbb{U}$ is parameterized by λ, we may use $\mathbb{U}_\lambda$. We show the following theorem.

Theorem 5. *Given any predefined distribution $\mathbb{Q}$ and any parameter $c \in (0, 1)$, there exists an algorithm that samples at most $\delta(\mathbb{Q}, \mathbb{U}_\lambda)$ extra vertices in expectation and achieves a competitive ratio of $O(\frac{1}{c \cdot \lambda})$ with probability at least $1 - e^{-(1-c)^2/2}$, where $\delta(\mathbb{Q}, \mathbb{U}_\lambda)$ is the total variance distance between the two distributions.*

Letting the parameter c be a constant, Theorem 5 implies an algorithm which achieves $O(\frac{1}{\lambda})$-competitive with a constant probability. To close the computational gap of our problem, we further show the hardness results.

Theorem 6. *Any $O(\frac{1}{\lambda})$-competitive algorithm must sample $\Omega(\delta(\mathbb{Q}, \mathbb{U}_\lambda))$ vertices in expectation.*

Algorithmic Intuition. Based on the result stated in Theorem 3, we know that if the first λn arrivals satisfy the uniform random distribution, then Algorithm 1 is a $O(\frac{1}{\lambda})$-competitive algorithm. Thus, the main idea is to resample some vertices according to some distribution $\mathbb{D}$ such that the probability of each vertex being sampled is λ conditioned on whether it is in the predefined sample. Intuitively, we use $\mathbb{D}$ to "rescale" the probability of each vertex being sampled. Based on the distribution $\mathbb{D}$, if we resample a vertex set L with $|L| := \lambda n$ vertices, then Algorithm 1 is a $O(\frac{1}{\lambda})$-competitive algorithm when the input sequence is $\sigma_L + \sigma_{V \setminus L}$. Note that $\mathbb{D}$ may not be uniform, and thus, we are not able to ensure that L satisfies the cardinality constraint. But we can guarantee that $|L|$ is $O(\lambda n)$ with high probability using a concentration bound.

In summary, our algorithm mainly contains the following steps:

- Given a predefined distribution $\mathbb{Q}$ and a vertex set S, construct a distribution $\mathbb{D} := (p_1, \ldots, p_v)$.
- Sample a subset L by distribution $\mathbb{D}$, i.e., for each $v \in V$, independently add v to L with probability p_v.

Algorithm 3: Over-&Under-sampling Algorithm

Input: The distribution $\mathbb{Q} := (q_1, \ldots, q_n)$ and the sampled vertex set S;
Output: A sampling distribution $\mathbb{D} := (p_1, \ldots, p_n)$.

for each vertex $v \in V$ **do**
  if $q_v < \lambda$ **then** ▷ Oversampling
    if $v \in S$ **then** $p_v \leftarrow 1$;
    else $p_v \leftarrow \frac{\lambda - q_v}{1 - q_v}$.
    end if
  else $(q_v \geq \lambda)$ ▷ Undersampling
    if $v \in S$ **then** $p_v \leftarrow \frac{\lambda}{q_v}$.
    else $p_v \leftarrow 0$.
    end if
  end if
end for
return $\mathbb{D}$.

- Run Algorithm 1 with the input sequence $\sigma_L + \sigma_{V \setminus L}$, where $\sigma_{V \setminus L}$ is a vertex order given by the adversary.

The crucial part of the algorithm is how to define the distribution $\mathbb{D}$. The main idea is to *oversample* and *undersample* vertices based on distribution $\mathbb{Q}$ and the sampled vertex set S. Intuitively, if the sampled probability of a vertex in $\mathbb{Q}$ is larger than λ, we want to decrease (undersample) its probability in the constructed probability distribution; otherwise, we want to oversample. We construct a distribution $\mathbb{D}$ to sample each vertex with probability λ conditioned on $\mathbb{Q}$ to achieve this goal. To this end, we must carefully set parameters and distinguish several cases. The formal description can be found in Algorithm 3.

Note that when the total variance distance $\delta(\mathbb{Q}, \mathbb{U})$ is 0, i.e., the predefined distribution is exactly the uniform random distribution, the constructed distribution $\mathbb{D}$ is as follows: for each vertex $v \in S$, $p_v = 1$; otherwise, $p_v = 0$, which means that the vertex set sampled by $\mathbb{D}$ is still set S and no extra vertex is added. Then by Theorem 3, we know that Algorithm 1 is a $O(\frac{1}{\lambda})$-competitive algorithm. For an arbitrary distribution $\mathbb{Q}$, the proof of Theorem 5 consists of two parts: (i) showing that the algorithm is $O(\frac{1}{c \cdot \lambda})$-competitive with a high probability, (ii) proving that the expected number of extra samples is $\delta(\mathbb{Q}, \mathbb{U})$. Due to space limitations, we defer the proofs to the full version of this paper. We remark that the total variance distance $\delta(\mathbb{Q}, \mathbb{U})$ is at most $\lambda(1 - \lambda)n$.

Upper Bound of $\delta(\mathbb{Q}, \mathbb{U})$. Note that $\sum_{v \in V} q_v = \lambda n$ since the expected size of the predefined sample set is λn. By some math, we see that $\sum_{v \in V : q_v < \lambda} (\lambda - q_v)$ is at most $\lambda(1 - \lambda)n$. Moreover, when the total variance distance is $\lambda(1 - \lambda)n$, the distribution $\mathbb{Q}$ is as follows:

$$\mathbb{Q} = (\underbrace{1, \ldots, 1}_{\lambda n}, \underbrace{0, \ldots, 0}_{(1-\lambda)n})$$

5 Experiments

This section validates the empirical performance of our sampling models on real datasets. We design two experiments. First, we investigate the empirical performance of the semi-online pivot algorithm (Algorithm 1) when we vary the size of the uniform sample. Second, we consider the extra sampling model and observe the algorithm's performance with and without extra samples.

5.1 Setup

The experiments are conducted on a machine running Ubuntu 18.04 with an i7-7800X CPU and 48 GB memory. The experimental results are averaged over 10 runs.

Datasets. Following the experimental setting introduced in (Simpson, Srinivasan, and Thomo 2016b), we use social network datasets to test the performance of the feedback arc set algorithms. The experiments consider three directed temporal network datasets with different sizes: CollegeMsg[1] ($|V| = 1,899$, $|E| = 59,835$), MathOverflow[2] ($|V| = 24,818$, $|E| = 506,550$), and RedditHyperlink[3] ($|V| = 55,863$, $|E| = 858,490$). Note that the networks are not complete graphs. Since the datasets are temporal and each vertex has a timestamp, we obtain complete graphs by adding all the missing edges and letting each of them point from the earlier released vertex to the other.

Arrival Orders. The experiments investigate three arrival orders of vertices in the graphs. First of all, we consider the chronological order given by the data to simulate the performance of our sampling models in practice. Then we use the bad triangle decreasing order and the indegree decreasing order to approximate the adversarial order. In the bad triangle decreasing order, we compute the number of bad triangles that each vertex belongs to and let the vertex which is contained in more bad triangles arrive first, while the indegree decreasing order lets the vertex with a larger indegree arrive first.

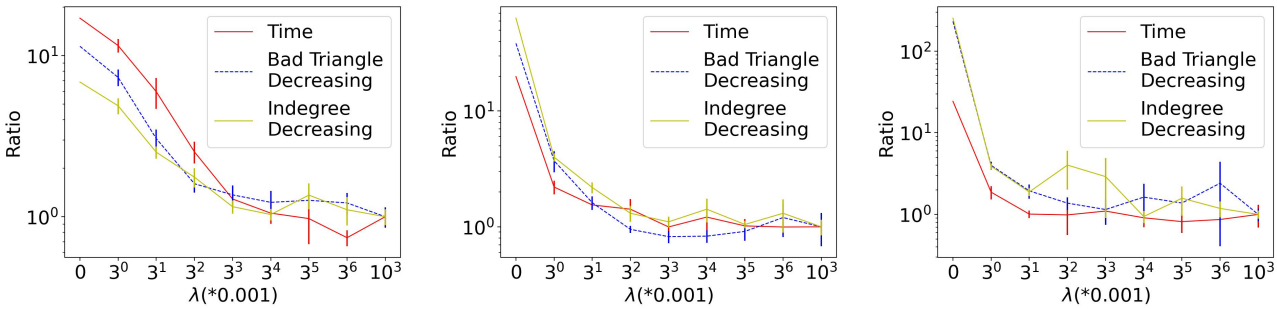
5.2 Power of Uniform Sampling

This experiment tests the performance of the pivot algorithm with different uniform sampling fractions λ's to show the power of uniform sampling. We consider $\lambda \in \{0, 0.001, 0.003, 0.027, 0.081, 0.243, 0.729, 1\}$ and use the ratio of the algorithm's objective to the objective obtained by the offline pivot algorithm to illustrate the performance of the algorithm. For each dataset, we try the aforementioned three arrival orders. The results are shown in Fig. 2.

5.3 Power of Extra Sampling

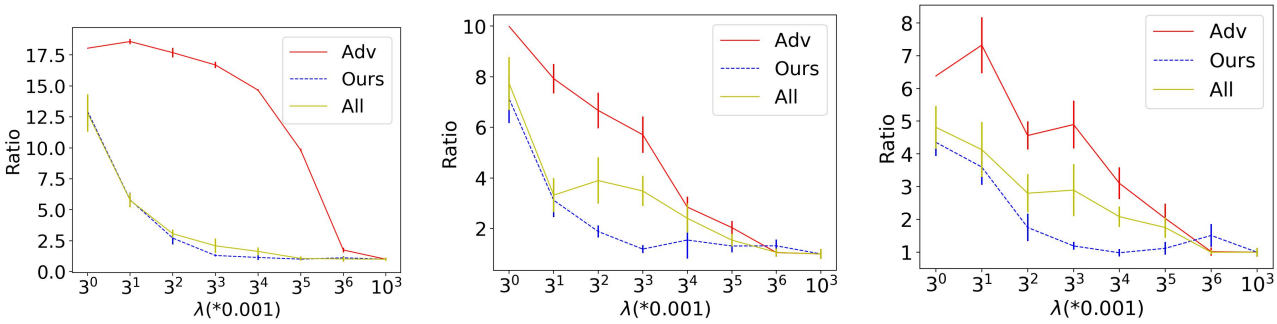
This experiment investigates the extra sampling model. To show the robustness of our extra-sampling algorithm, we let the predefined distribution be the most unfavorable one for the algorithm. To approximate such a sampling distribution under each arrival order, we let the predefined sample set always be the first λn arriving vertices; that is, the sampling probability of a vertex is 1 if the vertex is in the first λn vertices and 0 otherwise. Denote such a predefined sample set by Adv to imply that it is an adversarial-like sampling. Algorithm 3 then constructs a new sample Ours by discarding

[1] https://snap.stanford.edu/data/CollegeMsg.html
[2] https://snap.stanford.edu/data/sx-mathoverflow.html
[3] https://snap.stanford.edu/data/soc-RedditHyperlinks.html



(a) CollegeMsg (b) MathOverflow (c) RedditHyperlink

Figure 2: The experimental results on the three datasets when the sampling fraction λ varies. The x-axes and y-axes represent the sampling rate and the algorithm's objective relative to that of the pivot on the offline input, and they are both on the log scale.



(a) Time (b) Bad Triangle Decreasing (c) Indegree Decreasing

Figure 3: The performance of different sampling sets on CollegeMsg when λ varies. The x-axe is on the log scale.

some and adding new to Adv. Further, to show the necessity of discarding some vertices in the predefined sample Adv, we also consider using all appearing in Adv and Ours, which is denoted as All. These three datasets give the same trends; thus, we show the results on one dataset in Fig. 3 and others appear in the full version of this paper.

5.4 Empirical Discussion

The results show the following trends.

- From Fig. 2, we see that on all of the datasets, even a small fraction of uniform samples can improve the performance significantly. Typically on the dataset RedditHyperlink, without any sample, the ratio is more than two hundred; while with only a sampling fraction of 0.1%, the ratio can be improved to single digits.
- From Fig. 3, we see the necessity of oversampling and undersampling. Having extra samples in addition to the predefined sample set gives a better performance (All). Discarding some according to our algorithm gives a further improvement (Ours).

6 Conclusion

The paper considers online Feedback Arc Set (FAS) in tournaments in a beyond-worst-case manner. We show that it is possible to break the pessimistic lower bound by giving a constant approximation for nodes arriving in the adversarial order, given access to a constant fraction of samples. We further investigate how to optimally exploit samples from skewed distributions and revisit the correlational clustering problem under the new model. These results take the first step towards studying various online ranking problems. We believe that the algorithmic ideas will be helpful to many other related problems in this area.

Acknowledgements

Chenyang Xu is supported by the National Key Research Project of China under Grant No. 2023YFA1009402, the National Natural Science Foundation of China (No. 62302166), the Dean's Fund of Shanghai Key Laboratory of Trustworthy Computing, ECNU, and the Key Laboratory of Interdisciplinary Research of Computation and Economics (SUFE), Ministry of Education. Qingyun Chen and Sungjin Im are supported by NSF grants CCF-1844939 and CCF-2121745. Benjamin Moseley is supported by a Google Research Award, an Infor Research Award, a Carnegie Bosch Junior Faculty Chair and NSF Grants CCF-2121744 and CCF-1845146. Ruilong Zhang is supported by NSF grant CCF-1844890.

References

Ailon, N. 2012. An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity. *J. Mach. Learn. Res.*, 13: 137–164.

Ailon, N.; Charikar, M.; and Newman, A. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5): 23:1–23:27.

Alon, N. 2006. Ranking Tournaments. *SIAM J. Discret. Math.*, 20(1): 137–142.

Argue, C.; Frieze, A.; Gupta, A.; and Seiler, C. 2022. Learning from a Sample in Online Algorithms. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Bar-Yehuda, R.; Geiger, D.; Naor, J.; and Roth, R. M. 1994. Approximation Algorithms for the Vertex Feedback Set Problem with Applications to Constraint Satisfaction and Bayesian Inference. In Sleator, D. D., ed., *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. 23-25 January 1994, Arlington, Virginia, USA*, 344–354. ACM/SIAM.

Baweja, A.; Jia, J.; and Woodruff, D. P. 2022. An Efficient Semi-Streaming PTAS for Tournament Feedback Arc Set with Few Passes. In Braverman, M., ed., *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, 16:1–16:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Charbit, P.; Thomassé, S.; and Yeo, A. 2007. The Minimum Feedback Arc Set Problem is NP-Hard for Tournaments. *Comb. Probab. Comput.*, 16(1): 1–4.

Chen, L.; Kol, G.; Paramonov, D.; Saxena, R.; Song, Z.; and Yu, H. 2021. Almost Optimal Super-Constant-Pass Streaming Lower Bounds for Reachability. *Electron. Colloquium Comput. Complex.*, TR21-027.

Coppersmith, D.; Fleischer, L.; and Rudra, A. 2006. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, 776–782.

Coppersmith, D.; Fleischer, L.; and Rudra, A. 2010. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Trans. Algorithms*, 6(3): 55:1–55:13.

Even, G.; Schieber, B.; Sudan, M.; et al. 1998. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2): 151–174.

Im, S.; and Montazer Qaem, M. 2020. Fast and Parallelizable Ranking with Outliers from Pairwise Comparisons. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, 173–188. Springer.

Kenyon-Mathieu, C.; and Schudy, W. 2007a. How to rank with few errors. In Johnson, D. S.; and Feige, U., eds., *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, 95–103. ACM.

Kenyon-Mathieu, C.; and Schudy, W. 2007b. How to rank with few errors. In *STOC*, 95–103. ACM.

Lattanzi, S.; Lavastida, T.; Moseley, B.; and Vassilvitskii, S. 2020. Online Scheduling via Learned Weights. In Chawla, S., ed., *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, 1859–1877. SIAM.

Lattanzi, S.; Moseley, B.; Vassilvitskii, S.; Wang, Y.; and Zhou, R. 2021. Robust Online Correlation Clustering. In *NeurIPS*, 4688–4698.

Lykouris, T.; and Vassilvitskii, S. 2021. Competitive Caching with Machine Learned Advice. *J. ACM*, 68(4): 24:1–24:25.

Mathieu, C.; Sankur, O.; and Schudy, W. 2010. Online Correlation Clustering. In *STACS*, volume 5 of *LIPIcs*, 573–584. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Simpson, M.; Srinivasan, V.; and Thomo, A. 2016a. Efficient Computation of Feedback Arc Set at Web-Scale. *Proc. VLDB Endow.*, 10(3): 133–144.

Simpson, M.; Srinivasan, V.; and Thomo, A. 2016b. Efficient Computation of Feedback Arc Set at Web-Scale. *Proc. VLDB Endow.*, 10(3): 133–144.