Semi-Online Bipartite Matching

Ravi Kumar
Google, Mountain View, CA, USA
ravi.k53@gmail.com

Manish Purohit
Google, Mountain View, CA, USA
mpurohit@google.com

Aaron Schild
University of California, Berkeley, CA, USA
aschild@berkeley.edu

Zoya Svitkina
Google, Mountain View, CA, USA
zoya@cs.cornell.edu

Erik Vee
Google, Mountain View, CA, USA
erikvee@google.com

Abstract

In this paper we introduce the *semi-online* model that generalizes the classical online computational model. The semi-online model postulates that the unknown future has a predictable part and an adversarial part; these parts can be arbitrarily interleaved. An algorithm in this model operates as in the standard online model, i.e., makes an irrevocable decision at each step.

We consider bipartite matching in the semi-online model. Our main contributions are competitive algorithms for this problem and a near-matching hardness bound. The competitive ratio of the algorithms nicely interpolates between the truly offline setting (i.e., no adversarial part) and the truly online setting (i.e., no predictable part).

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Semi-Online Algorithms, Bipartite Matching

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.50

Related Version A full version of the paper is available at `https://arxiv.org/abs/1812.00134`.

Acknowledgements We thank Michael Kapralov for introducing us to the bipartite matching skeleton decomposition in [7].

1 Introduction

Modeling future uncertainty in data while ensuring that the model remains both realistic and tractable has been a formidable challenge facing the algorithms research community. One of the more popular, and reasonably realistic, such models is the online computational model. In its classical formulation, data arrives one at a time and upon each arrival, the algorithm has to make an irrevocable decision agnostic of future arrivals. Online algorithms boast a rich literature and problems such as caching, scheduling and matching, each of which abstracts common practical scenarios, have been extensively investigated [4, 20]. Competitive analysis, which measures how well an online algorithm performs compared to the best offline algorithm that knows the future, has been a linchpin in the study of online algorithms.

While online algorithms capture some aspect of the future uncertainty in the data, the notion of competitive ratio is inherently worst-case and hence the quantitative guarantees it offers are often needlessly pessimistic. A natural question that then arises is : how can we avoid modeling the worst-case scenario in online algorithms? Is there a principled way to incorporate some knowledge we have about the future? There have been a few efforts trying to address this point from different angles. One line of attack has been to consider oracles that offer some advice on the future; such oracles, for instance, could be based on machine-learning methods. This model has been recently used to improve the performance of online algorithms for reserve price optimization, caching, ski-rental, and scheduling [19, 16, 14]. Another line of attack posits a distribution on the data [5, 17, 21] or the arrival model; for instance, random arrival models have been popular in online bipartite matching and are known to beat the worst-case bounds [10, 18]. A different approach is to assume a distribution on future inputs; the field of stochastic online optimization focuses on this setting [8]. The advice complexity model, where the partial information about the future is quantified as advice bits to an online algorithm, has been studied as well in complexity theory [3].

In this work we take a different route. At a very high level, the idea is to tease the future data apart into a predictable subset and the remaining adversarial subset. As the names connote, the algorithm can be assumed to know everything about the former but nothing about the latter. Furthermore, the predictable and adversarial subsets can arrive arbitrarily interleaved yet the algorithm still has to operate as in the classical online model, i.e., make irrevocable decisions upon each arrival. Our model thus offers a natural interpolation between the traditional offline and online models; we call ours the *semi-online* model. Our goal is to study algorithms in the semi-online model and to analyze their competitive ratios; unsurprisingly, the bounds depend on the size of the adversarial subset. Ideally, the competitive ratio should approach the offline optimum bounds if the adversarial fraction is vanishing and should approach the online optimum bounds if the predictable fraction is vanishing.

Bipartite matching. As a concrete problem in the semi-online setting, we focus on bipartite matching. In the well-known online version of the problem, which is motivated by online advertising, there is a bipartite graph with an offline side that is known in advance and an online side that is revealed one node at a time together with its incident edges. In the semi-online model, the nodes in the online side are partitioned into a predicted set of size $n - d$ and an adversarial set of size d. The algorithm knows the incident edges of all the nodes in the former but nothing about the nodes in the latter. We can thus also interpret the setting as online matching with partial information and predictable uncertainty (pardon the oxymoron). In online advertising applications, there are many predictably unpredictable events. For example, during the soccer world cup games, we know the nature of web traffic will be unpredictable but nothing more, since the actual characteristics will depend on how the game progresses and which team wins.

We also consider a variant of semi-online matching in which the algorithm does not know which nodes are predictable and which are adversarial. In other words, the algorithm receives a prediction for all online nodes, but the predictions are correct only for some $n - d$ of them. We call this the *agnostic* case.

Main results. In this paper, we assume that the optimum solution on the bipartite graph, formed by the offline nodes on one side and by the predicted and adversarial nodes on the other, is a perfect matching[1]. We present two algorithms and a hardness result for the semi-online bipartite matching problem. Let $\delta = d/n$ be the fraction of adversarial nodes. The Iterative Sampling algorithm, described in Section 3, obtains a competitive ratio of $(1 - \delta + \frac{\delta^2}{2}(1 - 1/e))^2$. This algorithm "reserves" a set of offline nodes to be matched to the adversarial nodes by repeatedly selecting a random offline node that is unnecessary for matching the predictable nodes. It is easy to see that algorithms that deterministically reserve a set of offline nodes can easily be thwarted by the adversary.

The second algorithm, described in Section 4, achieves an improved competitive ratio of $(1 - \delta + \delta^2(1 - 1/e))$. This algorithm samples a maximum matching in the predicted graph by first finding a matching skeleton [7, 15] and then sampling a matching from each component in the skeleton using the dependent rounding scheme of [6]. This allows us to sample a set of offline nodes that, in expectation, has a large overlap with the set matched to adversarial nodes in the optimal solution. Surprisingly, in Section 5, we show that it is possible to sample from arbitrary set systems so that the same "large overlap" property is maintained. We prove the existence of such distributions using LP duality and believe that this result may be of independent interest.

To complement the algorithms, in Section 6 we obtain a hardness result showing that the above competitive ratios are near-optimal. In particular, no randomized algorithm can achieve a competitive ratio better than $(1 - \delta e^{-\delta}) \approx (1 - \delta + \delta^2 - \delta^3/2 + \ldots)$. Note that this expression coincides with the best offline bound (i.e, 1) and the best online bound (i.e., $1 - 1/e$) at the extremes of $\delta = 0$ and $\delta = 1$, respectively. We conjecture this to be the optimal bound for the problem.

Extensions. In Section 7, we explore variants of the semi-online matching model, including the agnostic version and fractional matchings, and present upper and lower bounds in those settings. To illustrate the generality of our semi-online model, we consider a semi-online version of the classical ski rental problem. In this version, the skier knows whether or not she'll ski on certain days while other days are uncertain. Interestingly, there is an algorithm with a competitive ratio of the same form as our hardness result for matchings, namely $1 - (1 - x)e^{-(1-x)}$, where $(1 - x)$ is a parameter analogous to δ in the matching problem. We wonder if this form plays a role in semi-online algorithms similar to what $(1 - 1/e)$ has in many online algorithms [11].

Other related work. The use of (machine learned) predictions for revenue optimization problems was first proposed in [19]. The concepts were formalized further and applied to online caching in [16] and ski rental and non-clairvoyant scheduling in [14]. Online matching with forecasts was first studied in [22]; however, that paper is on forecasting the demands rather than the structure of the graph as in our case. The problem of online matching when edges arrive in batches was considered in [15] where a $(1/2 + 2^{-O(s)})$-competitive ratio is shown, with s the number of stages. However, the batch framework differs from ours in that in our case, the nodes arrive one at a time and are arbitrarily interleaved.

[1] Our techniques extend to the case without a perfect matching; we defer the proof of the general case to the full version of the paper.

[2] Observe that an algorithm that ignores all the adversarial nodes and outputs a maximum matching in the predicted graph achieves a competitive ratio of only $1 - \delta$.

There has been a lot of work on online bipartite matching and its variants. The RANKING algorithm [13] selects a random permutation of the offline nodes and then matches each online node to its unmatched neighbor that appears earliest in the permutation. It is well-known to obtain a competitive ratio of $(1 - 1/e)$, which is best possible. For a history of the problem and significant advances, see the monograph [20]. The ski rental problem has also been extensively studied; the optimal randomized algorithm has ratio $e/(e - 1)$ [12]. The term "semi-online" has been used in scheduling when an online scheduler knows the sum of the jobs' processing times (e.g., see [1]) and in online bin-packing when a lookahead of the next few elements is available (e.g., see [2]); our use of the term is more quantitative in nature.

2 Model

We now formally define the *semi-online bipartite matching* problem. We have a bipartite graph $G = (U, V, E_G)$ where U is the set of nodes available offline and nodes in V arrive online. Further, the online set V is partitioned into the *predicted* nodes V_P and the *adversarial* nodes V_A. The *predicted graph* $H = (U, V_P, E_H)$ is the subgraph of G induced on the nodes in U and V_P. Initially, an algorithm only knows H and is unaware of edges between U and V_A. The algorithm is allowed to preprocess H before any online node actually arrives. In the online phase, at each step, one node of V is revealed with its incident edges, and has to be either irrevocably matched to some node in U or abandoned. Nodes of V are revealed in an arbitrary order[3] and the process continues until all of V has been revealed.

We note that when a node $v \in V$ is revealed, the algorithm can "recognize" it by its edges, i.e., if there is some node $v' \in V_P$ that has the same set of neighbors as v and has not been seen before, then v can be assumed to be v'. There could be multiple identical nodes in V_P, but it is not important to distinguish between them. If an online node comes that is not in V_P, then the algorithm can safely assume that it is from V_A. (In Section 7, we consider a model where the predicted graph can have errors and hence this assumption is invalid.)

We introduce a quantity δ to measure the level of knowledge that the algorithm has about the input graph G. Competitive ratios that we obtain are functions of δ. For any graph I, let $\nu(I)$ denote the size of the maximum matching in I. Then we define $\delta = \delta(G) = 1 - \frac{\nu(H)}{\nu(G)}$. Intuitively, the closer δ is to 0, the more information the predicted graph H contains and the closer the instance is to an offline problem. Conversely, δ close to 1 indicates an instance close to the pure online setting. Note that the algorithm does not necessarily know δ, but we use it in the analysis to bound the competitive ratio. For convenience, in this paper we assume that the input graph G contains a perfect matching. Let $n = |U| = |V|$ be the number of nodes on each side and $d = |V_A|$ be the number of adversarial online nodes. In this case, δ simplifies to be the fraction of online nodes that are adversarial, i.e., $\delta = \frac{|V_A|}{|V|} = \frac{d}{n}$.

3 Iterative Sampling Algorithm

In this section we give a simple polynomial time algorithm for bipartite matching in the semi-online model. We describe the algorithm in two phases: a *preprocessing phase* that finds a maximum matching M in the predicted graph H and an *online phase* that processes each node upon its arrival to find a matching in G that extends M.

[3] The arrival order can be adversarial, including interleaving the nodes in V_P and V_A.

Algorithm 1: Iterative Sampling: Preprocessing Phase.

```
Function: Preprocess(H):
    Data: Predicted graph $H$
    Result: Maximum matching in $H$, a sequence of nodes from $U$
    Let $H_0 \leftarrow H, U_0 \leftarrow U$;
    for $i = 1, 2, \ldots, d$ do
        $U_i \leftarrow \{u \in U_{i-1} \mid \nu(H_{i-1} \setminus \{u\}) = n - d\}$ ;        /* Set of nodes whose
         removal does not change the size of the maximum matching.  */
        Let $u_i$ be a uniformly random node in $U_i$;
        $H_i \leftarrow H_{i-1} \setminus \{u_i\}$;
    $M \leftarrow$ Arbitrary maximum matching in $H_d$;
    $R \leftarrow$ Uniformly random permutation of $\{u_1, \ldots, u_d\}$;
    return $M$, $R$
```

Algorithm 2: Online Phase.

```
$M, R \leftarrow$ Preprocess($H$);
for $v \in V$ arriving online do
    if $v \in V_P$ then                                          /* predicted node */
        Match $v$ to $M(v)$;
    else                                                        /* adversarial node */
        Match $v$ to the first unmatched neighbor in $R$, if one exists;  /* RANKING */
```

Preprocessing Phase

The goal of the preprocessing phase is to find a maximum matching in the predicted graph H. However, if we deterministically choose a matching, the adversary can set up the neighborhoods of V_A so that all the neighbors of V_A are used in the chosen matching, and hence the algorithm is unable to match any node from V_A. Algorithm 1 describes our algorithm to sample a (non-uniform) random maximum matching from H.

Online Phase

In the online phase nodes from V arrive one at a time and we are required to maintain a matching such that the online nodes are either matched irrevocably or dropped. In this phase, we match the nodes in V_P as per the matching M obtained from the preprocessing phase, i.e., we match $v \in V_P$ to node $M(v) \in U$ where $M(v)$ denotes the node matched to v by matching M. The adversarial nodes in V_A are matched to nodes in R that are not used by M using the RANKING algorithm [13]. Algorithm 2 describes the complete online phase of our algorithm.

Analysis

For the sake of analysis, we construct a sequence of matchings $\{M_i^*\}_{i=0}^d$ as follows. Let M_0^* be an arbitrary perfect matching in G. For $i \geq 1$, by definition of U_i, there exists a matching M_i' in H_i of size $n - d$ that does not match node u_i. Hence, $M_i' \cup M_{i-1}^*$ is a union of disjoint paths and cycles such that u_i is an endpoint of a path P_i. Let $M_i^* = M_{i-1}^* \oplus P_i$, i.e. obtain M_i^* from M_{i-1}^* by adding and removing alternate edges from P_i. It's easy to verify that M_i^* is indeed a matching and $|M_i^*| \geq |M_{i-1}^*| - 1$. Since $|M_0^*| = n$, this yields $|M_i^*| \geq n - i$, $\forall\ 0 \leq i \leq d$. Further, by construction, M_i^* does not match any nodes in $\{u_1, \ldots, u_i\}$.

▶ **Lemma 1.** *For all $0 \leq i \leq d$, all nodes $v \in V_P$ are matched by M_i^*. Further, $|M_i^*(V_A)| \geq d - i$, i.e. at least $d - i$ adversarial nodes are matched by M_i^*.*

Proof. We prove the claim by induction. Since M_0^* is a perfect matching, the base case is trivially true. By the induction hypothesis, we assume that M_{i-1}^* matches all of V_P. Recall that M_i' also matches all of V_P and $M_i^* = M_{i-1}^* \oplus P_i$ where P_i is a maximal path in $M_i' \cup M_{i-1}^*$. Since each node $v \in V_P$ has degree 2 in $M_i' \cup M_{i-1}^*$, v cannot be an end point of P_i. Hence, all nodes $v \in V_P$ remain matched in M_i^*. Further, we have $|M_i^*(V_A)| = |M_i^*| - |M_i^*(V_P)| \geq (n - i) - (n - d) = d - i$ as desired. ◀

Equipped with the sequence of matchings M_i^*, we are now ready to prove that, in expectation, a large matching exists between the set R of nodes left unmatched by the preprocessing phase and the set V_A of adversarial nodes.

▶ **Lemma 2.** $\mathbb{E}[\nu(G[R \cup V_A])] \geq d^2/(2n)$ *where $G[R \cup V_A]$ is the graph induced by the reserved vertices R and the adversarial vertices V_A.*

Proof. We construct a sequence of sets of edges $\{N_i\}_{i=0}^d$ as follows. Let $N_0 = \emptyset$. If $M_{i-1}^*(u_i) \in V_A$, let $e_i = \{u_i, M_{i-1}^*(u_i)\}$ be the edge of M_{i-1}^* incident with u_i and let $N_i = N_{i-1} \cup \{e_i\}$. Otherwise, let $N_i = N_{i-1}$. In other words, if the node u_i chosen during the ith step is matched to an adversarial node by the matching M_{i-1}^*, add the matched edge to set N_i.

We show by induction that N_i is a matching for all $i \geq 0$. N_0 is clearly a matching. When $i > 0$, either $N_i = N_{i-1}$ (in which case we are done by the inductive hypothesis), or $N_i = N_{i-1} \cup \{e_i\}$. Let $e_i = (u_i, v_i)$ and consider any other edge $e_j = (u_j, v_j) \in N_{i-1}$. Since $u_j \notin H_{i-1}$, we have $u_j \neq u_i$. By definition, node v_i is matched in M_{i-1}^*. By construction, this implies that v_i must be matched in all previous matchings in this sequence, in particular, v_i must be matched in M_j^* (since a node $v \in V_A$ that is unmatched in M_{k-1}^* can never be matched by M_k^*). However, since $v_j = M_{j-1}^*(u_j)$, the matching $M_j^* = M_{j-1}^* \setminus \{e_j\}$ and hence v_j is not matched in M_j^*. Hence $v_i \neq v_j$. Thus we have shown that e_i does not share an endpoint with any $e_j \in N_{i-1}$ and hence N_i is a matching.

By linearity of expectation we have the following.

$$\mathbb{E}[|N_i|] = \mathbb{E}[|N_{i-1}|] + \Pr_{u_i}[M_{i-1}^*(u_i) \in V_A]$$

However, by Lemma 1, since M_{i-1}^* matches all of V_P, we must have $M_{i-1}^*(V_A) \subseteq U_i$. Hence,

$$\mathbb{E}[|N_i|] \geq \mathbb{E}[|N_{i-1}|] + \frac{|M_{i-1}^*(V_A)|}{|U_i|} \geq \mathbb{E}[|N_{i-1}|] + \frac{d - (i-1)}{n}$$

Solving the recurrence with $|N_0| = 0$ gives

$$\mathbb{E}[|N_d|] \geq \sum_{i=1}^{d} \frac{i}{n} \geq \frac{d(d+1)}{2n}$$

The lemma follows since N_d is a matching in $G[R \cup V_A]$. ◀

▶ **Theorem 3.** *There is a randomized algorithm for the semi-online bipartite matching problem with a competitive ratio of at least* $(1 - \delta + (\delta^2/2)(1 - 1/e))$ *in expectation.*

Proof. Algorithm 1 guarantees that the matching M found in the preprocessing phase matches all predicted nodes and has size $n - d = n(1 - \delta)$. Further, in the online phase, we use the RANKING [13] algorithm on the graph $G[R \cup V_A]$. Since RANKING is $(1 - 1/e)$-competitive, the expected number of adversarial nodes matched is at least $(1 - 1/e)\nu(G[R \cup V_A])$. By Lemma 2, this is at least $(1 - 1/e)(\frac{d^2}{2n}) = (\delta^2 n/2)(1 - 1/e)$.

Therefore, the total matching has expected size $n(1 - \delta + (\delta^2/2)(1 - 1/e))$ as desired. ◀

Using a more sophisticated analysis, we can show that the iterative sampling algorithm yields a tighter bound of $(1 - \delta + \delta^2/2 - \delta^3/2)$. However we omit the proof because the next section presents an algorithm with an even better guarantee.

4 Structured Sampling

In this section we give a polynomial time algorithm for the semi-online bipartite matching that yields an improved competitive ratio of $(1 - \delta + \delta^2(1 - 1/e))$. We first discuss the main ideas in Section 4.1 and then describe the algorithm and its analysis in Section 4.2.

▶ **Theorem 4.** *There is a randomized algorithm for the semi-online bipartite matching problem with a competitive ratio of at least* $(1 - \delta + \delta^2(1 - 1/e))$ *in expectation.*

4.1 Main Ideas and Intuition

As with the iterative sampling algorithm, we randomly choose a matching of size $n - d$ (according to some distribution), and define the *reserved* set R to be the set of offline nodes that are not matched. As online nodes arrive, we follow the matching for the predicted nodes; for adversarial nodes, we run the RANKING algorithm on the reserved set R.

Let M^* be a perfect matching in G. For a set of nodes S, let $M^*(S)$ denote the set of nodes matched to them by M^*. Call a node $u \in U$ *marked* if it is in $M^*(V_A)$, i.e., it is matched to an adversarial node by the optimal solution. We argue that the number of marked nodes in the set R chosen by our algorithm is at least d^2/n in expectation. Since RANKING finds a matching of at least a factor $(1 - 1/e)$ of optimum in expectation, this means that we find a matching of size at least $d^2/n \cdot (1 - 1/e)$ on the reserved nodes in expectation. Combining this with the matching of size $n - d$ on the predicted nodes, this gives a total of $n - d + d^2/n \cdot (1 - 1/e) = n(1 - \delta + \delta^2(1 - 1/e))$.

The crux of the proof lies in showing that R contains many marked nodes. Ideally, we would like to choose a random matching of size $n - d$ in such a way that each node of U has probability d/n of being in R. Since there are d marked nodes total, R would contain d^2/n of them in expectation. However, such a distribution over matchings does not always exist.

Instead, we use a graph decomposition to guide the sampling process. The marginal probabilities for nodes of U to be in R may differ, but nevertheless R gets the correct total number of marked nodes in expectation. H is decomposed into bipartite pairs (S_i, T_i), with $|S_i| \leq |T_i|$, so that the sets S_i partition V_P and the sets T_i partition U. This decomposition allows one to choose a random matching between S_i and T_i of size $|S_i|$ so that each node in T_i is reserved with the same probability. Letting $n_i = |T_i|$ and $d_i = |T_i| - |S_i|$, this probability is precisely d_i/n_i. Finally, we argue that the adversary can do no better than to mark d_i nodes in T_i, for each i. Hence, the expected number of nodes in R that are marked is at least $\sum_i (d_i^2/n_i)$, which we lower bound by d^2/n.

4.2 Proof of Theorem 4

We decompose the graph H into more structured pieces using a construction from [7] and utilize the key observation that the decomposition implies a *fractional matching.* Recall that a fractional matching is a function f that assigns a value in $[0,1]$ to each edge in a graph, with the property that $\sum_{e \ni v} f(e) \le 1$ for all nodes v. The quantity $\sum_{e \ni v} f(e)$ is referred to as the *fractional degree* of v. We use $\Gamma(S)$ to denote the set of neighbors of nodes in S.

▶ **Lemma 5** (Restatement of Lemma 2 from [15]). *Given a bipartite graph $H = (U, V_P, E_H)$ with $|U| \ge |V_P|$ and a maximum matching of size $|V_P|$, there exists a partition of V_P into sets $S_0, \dots, S_m$ and a partition of U into sets $T_0, \dots, T_m$ for some m such that the following holds:*
- *$\Gamma(\bigcup_{i<j} S_i) = \bigcup_{i<j} T_i$ for all j.*
- *For all $i < j$, $\frac{|S_i|}{|T_i|} > \frac{|S_j|}{|T_j|}$.*
- *There is a fractional matching in H of size $|V_P|$, where for all i, the fractional degree of each node in S_i is 1 and the fractional degree of each node in T_i is $|S_i|/|T_i|$. In this matching, nodes in S_i are only matched with nodes in T_i and vice versa.*

Further, the (S_i, T_i) pairs can be found in polynomial time.

In [7] and [15], the sets in the decomposition with $|S_i| < |T_i|$ are indexed with positive integers $i > 0$, the sets with $|S_0| = |T_0|$ get an index of 0, and the ones with $|S_i| > |T_i|$ get negative indices $i < 0$. Under our assumption that H supports a matching that matches all nodes of V_P, the decomposition does not contain sets with $|S_i| > |T_i|$, as the first such set would have $|S_i| > |\Gamma(S_i)|$, violating Hall's theorem. So we start the indices from 0.

Equipped with this decomposition, we choose a random matching between S_i and T_i such that each node in T_i is reserved[4] with the same probability. Since each (S_i, T_i) pair has a fractional matching, the dependent randomized rounding scheme of [6] allows us to do exactly that.

▶ **Lemma 6.** *Fix an index i and let S_i, T_i be defined as in Lemma 5. Then there is a distribution over matchings with size $|S_i|$ between S_i and T_i such that for all $u \in T_i$, the probability that the matching contains u is $|S_i|/|T_i|$.*

Proof. Given any bipartite graph G' and a fractional matching over G', the dependent rounding scheme of Gandhi et. al. [6] yields an integral matching such that the probability that any node $v \in G'$ is matched exactly equals its fractional degree. Since Lemma 5 guarantees a fractional matching such that the fractional degree of each node in S_i is 1 and the fractional degree of each node in T_i is $|S_i|/|T_i|$, the lemma follows. ◀

We are now ready to complete the description of our algorithm. Algorithm 3 is the preprocessing phase, while the online phase remains the same as earlier (Algorithm 2). In the preprocessing phase, we find a decomposition of the predicted graph H, and sample a matching using Lemma 6 for each component in the decomposition. In the online phase, we match all predicted online nodes using the sampled matching and use RANKING to match the adversarial online nodes.

Let $n_i = |T_i|$ and $d_i = |T_i| - |S_i|$, and let $R_i = R \cap T_i$ be the set of reserved nodes in T_i. Then Lemma 6 says that each node in T_i lands in R_i with probability d_i/n_i (although not independently). We now argue in Lemmas 7 and 8 that the adversary can do no better than to choose d_i marked nodes in each T_i.

[4] Recall that we say a node u is reserved by an algorithm if u is *not* matched in the predicted graph H.

Algorithm 3: Structured Sampling: Preprocessing Phase.

```
Function: Preprocess($H$):
    Data: Predicted graph $H$
    Result: Maximum matching in $H$, sequence of nodes from $U$
    Decompose $H$ into $\{(S_i, T_i)\}_{i=0}^m$ pairs using Lemma 5.
    $M_i \leftarrow$ Random matching between $S_i$ and $T_i$ using Lemma 6
    $M \leftarrow \bigcup_i M_i$
    Let $R_{\text{set}} \subseteq U$ be the set of nodes unmatched by $M$
    $R \leftarrow$ Uniformly random permutation of $R_{\text{set}}$
    return $M$, $R$
```

▶ **Lemma 7.** *Let $\ell_i = |M^*(V_A) \cap T_i|$. That is, let ℓ_i be the number of marked nodes in T_i. Then for all $t \geq 0$,*

$$\sum_{i \leq t} \ell_i \leq \sum_{i \leq t} d_i$$

Proof. Fix $t \leq m$, and let $U' = U - \bigcup_{i \leq t} T_i$. Since there is a perfect matching in the realized graph G, Hall's Theorem guarantees that there must be at least $|U'| - |\Gamma_H(U')|$ marked nodes in U' where $\Gamma_H(U')$ denotes the set of neighbors of U' in the predicted graph H. That is,

$$\sum_{i > t} \ell_i \geq |U'| - |\Gamma_H(U')|$$

But Lemma 5 tells us that $\Gamma(\bigcup_{i \leq t} S_i) = \bigcup_{i \leq t} T_i$, hence there is no edge between U' and $\bigcup_{i \leq t} S_i$. That is, $\Gamma_H(U') \subseteq V_P - \bigcup_{i \leq t} S_i$. Hence,

$$|\Gamma_H(U')| \leq |V_P| - \sum_{i \leq t} |S_i| = n - d - \sum_{i \leq t} (n_i - d_i)$$

Further, $|U'| = |U| - |\bigcup_{i \leq t} T_i| = n - \sum_{i \leq t} n_i$. Putting this together,

$$\begin{aligned}\sum_{i > t} \ell_i &\geq |U'| - |\Gamma_H(U')| \\ &\geq n - \sum_{i \leq t} n_i - \left(n - d - \sum_{i \leq t} (n_i - d_i)\right) \\ &= d - \sum_{i \leq t} d_i\end{aligned}$$

Recalling that $\sum_{i \leq m} \ell_i = d$, we see that $\sum_{i \leq t} \ell_i = d - \sum_{i > t} \ell_i \leq \sum_{i \leq t} d_i$, as desired. ◀

▶ **Lemma 8.** *Let $0 < a_0 \leq a_1 \leq \ldots \leq a_m$ be a non-decreasing sequence of positive numbers, and $\ell_0, \ldots, \ell_m$ and $k_0, \ldots, k_m$ be non-negative integers, such that $\sum_{i=0}^m \ell_i = \sum_{i=0}^m k_i$ and for all $t \leq m$, $\sum_{i \leq t} \ell_i \leq \sum_{i \leq t} k_i$. Then*

$$\sum_{i=0}^m \ell_i a_i \geq \sum_{i=0}^m k_i a_i.$$

Proof. We claim that for any fixed sequence $k_0, \ldots, k_m$, the minimum of the left-hand side ($\sum_i \ell_i a_i$) is attained when $\ell_i = k_i$ for all i. Suppose for contradiction that $\{\ell_i\}$ is the lexicographically-largest minimum-attaining assignment that is not equal to $\{k_i\}$ and let j be the smallest index with $\ell_j \neq k_j$. It must be that $\ell_j < k_j$ to satisfy $\sum_{i \le j} \ell_i \le \sum_{i \le j} k_i$. Also, $\sum_{i=0}^m \ell_i = \sum_{i=0}^m k_i$ implies that $j < m$ and that there must be an index $j' > j$ such that $\ell_{j'} > k_{j'}$. Let j' be the lowest such index.

Let $\ell'_i = \ell_i$ for all $i \notin \{j, j'\}$. Set $\ell'_j = \ell_j + 1$ and $\ell'_{j'} = \ell_{j'} - 1$. Notice that we still have $\sum_{i \le t} \ell'_i \le \sum_{i \le t} k_i$ for all t and $\sum_{i=0}^m \ell'_i = \sum_{i=0}^m k_i$, and $\{\ell'_i\}$ is lexicographically larger than $\{\ell_i\}$. In addition,

$$\sum_i \ell'_i a_i = \sum_i \ell_i a_i + a_j - a_{j'} \le \sum_i \ell_i a_i,$$

which is a contradiction. ◀

We need one last technical observation before the proof of the main result.

▶ **Lemma 9.** *Let d_i, n_i be positive numbers with $\sum_i d_i = d$ and $\sum_i n_i = n$. Then*

$$\sum_i \frac{d_i^2}{n_i} \ge \frac{d^2}{n}$$

Proof. We invoke Cauchy-Schwartz, with vectors u and v defined by $u_i = \frac{d_i}{\sqrt{n_i}}$ and $v_i = \sqrt{n_i}$. Since $||u||^2 \ge |u \cdot v|^2 / ||v||^2$, the result follows. ◀

▶ **Theorem 10.** *Choose reserved set R according to Algorithm 2. Then the expected number of marked nodes in R is at least $\delta^2 n$. That is, $|R \cap M^*(V_A)| \ge \delta^2 n$ in expectation.*

Proof. As in Lemma 7, let $\ell_i = |M^*(V_A) \cap T_i|$. Again, we have $\sum_{i \le t} \ell_i \le \sum_{i \le t} d_i$ for all t and $\sum_{i \le m} \ell_i = d = \sum_{i \le m} d_i$. For each i, the node $u \in T_i$ is chosen to be in R with probability d_i/n_i, with the d_i/n_i forming an increasing sequence. So the expected size of $|R \cap M^*(V_A)|$ is given by

$$\begin{aligned} \sum_i \frac{d_i}{n_i} \ell_i &\ge \sum_i \frac{d_i}{n_i} d_i && \text{by Lemma 8} \\ &\ge \frac{d^2}{n} && \text{by Lemma 9} \end{aligned}$$

Since $\delta = d/n$, the theorem follows. ◀

Proof of Theorem 4. The size of the matching, restricted to non-adversarial nodes, is $\sum_i (n_i - d_i) = n - d = n - \delta n$. Further, by Theorem 10, we have reserved at least $\delta^2 n$ nodes that can be matched to the adversarial nodes. RANKING will match at least a $(1 - 1/e)$ fraction of these in expectation. So in expectation, the total matching has size at least $n - \delta n + \delta^2 n(1 - 1/e) = n(1 - \delta + \delta^2(1 - 1/e))$ as desired. ◀

5 Sampling From Arbitrary Set Systems

In Section 4, we used graph decomposition to sample a matching in the predicted graph such that, in expectation, there is a large overlap between the set of reserved (unmatched) nodes and the unknown set of marked nodes chosen by the adversary. In this section we prove the existence of probability distributions on sets, with this “large overlap” property, in settings more general than just bipartite graphs.

Let U be a universe of n elements and let $\mathcal{S}$ denote a family of subsets of U with equal sizes, i.e., $|S| = d, \forall S \in \mathcal{S}$. Suppose an adversary chooses a set $T \in \mathcal{S}$, which is unknown to us. Our goal is to find a probability distribution over $\mathcal{S}$ such that the expected intersection size of T and a set sampled from this distribution is maximized. We prove in Theorem 11 that for any such set system, one can always guarantee that the expected intersection size is at least $\frac{d^2}{n}$.

The connection to matchings is as follows. Let U, the set of offline nodes in the matching problem, also be the universe of elements. $\mathcal{S}$ is a collection of all maximal subsets R of U such that there is a perfect matching between $U \setminus R$ and V_P. All these subsets have size $d = |V_A| = \delta n$. Notice that $M^*(V_A)$ is one of the sets in $\mathcal{S}$, although of course we don't know which. What we would like is a distribution such that sampling a set R from it satisfies $\mathbb{E}[|R \cap M^*(V_A)|] \geq d^2/n = \delta^2 n$.

▶ **Theorem 11.** *For any set system $(U, \mathcal{S})$ with $|U| = n$ and $|S| = d$ for all $S \in \mathcal{S}$, there exists a probability distribution $\mathcal{D}$ over $\mathcal{S}$ such that $\forall T \in \mathcal{S}$, $\mathbb{E}_{S\sim\mathcal{D}}[|S \cap T|] \geq \dfrac{d^2}{n}$.*

As an example, consider $U = \{v, w, x, y, z\}$ and $\mathcal{S} = \{\{v, w\}, \{w, x\}, \{x, y\}, \{y, z\}\}$. Here $n = 5$ and $d = 2$, so the theorem guarantees a probability distribution on the four sets such that each of them has an expected intersection size with the selected set of at least $\frac{4}{5}$. We can set $\Pr[\{v, w\}] = \Pr[\{y, z\}] = \frac{3}{10}$ and $\Pr[\{w, x\}] = \Pr[\{x, y\}] = \frac{1}{5}$. Then the expected intersection size for the set $\{v, w\}$ is $\Pr[\{v, w\}] \cdot 2 + \Pr[\{w, x\}] \cdot 1 = \frac{4}{5}$ because the intersection size is 2 if $\{v, w\}$ is picked and 1 if $\{w, x\}$ is picked. Similarly, one can verify that the expected intersection for any set is at least $\frac{4}{5}$. However, in general, it is not trivial to find such a distribution via an explicit construction.

Theorem 11 is a generalization to Theorem 10, and we could have selected a matching and a reserved set R according to the methods used in its proof. Indeed, this gives the same competitive ratio. However, the set system generated by considering all matchings of size $n - d$ is exponentially large in general. Hence the offline portion of the algorithm would not run in polynomial time.

5.1 Proof of Theorem 11

Let $\mathcal{D}$ be a probability distribution over $\mathcal{S}$ with the probability of choosing a set S denoted by p_S. Now, for any fixed set $T \in \mathcal{S}$, the expected intersection size is given by $\mathbb{E}_{S\sim\mathcal{D}}[|S \cap T|] = \sum_{S\in\mathcal{S}} p_S \cdot |S \cap T| = \sum_{u\in T} \sum_{S\ni u} p_S$. For a given set system $(U, \mathcal{S})$, consider the following linear program and its dual.

The primal constraints exactly capture the requirement that the expected intersection size is at least $\frac{d^2}{n}$ for any choice of T. Thus, to prove the theorem, it suffices to show that the optimal primal solution has an objective value of at most 1. We show that any feasible solution to the dual linear program must have objective value at most 1 and hence the theorem follows from strong duality.

▶ **Lemma 12.** *For any set system $(U, \mathcal{S})$, the optimal solution to Dual-LP has objective value at most 1.*

$$\min \sum_{S\in\mathcal{S}} p_S$$

s.t.

$$\forall T \in \mathcal{S}, \quad \sum_{u\in T}\sum_{S\ni u} p_S \geq \frac{d^2}{n} \quad (1)$$

$$\forall S \in \mathcal{S}, \quad p_S \geq 0 \quad (2)$$

Figure 1 Primal-LP.

$$\max \sum_{T\in\mathcal{S}} q_T$$

s.t.

$$\forall S \in \mathcal{S}, \quad \sum_{u\in S}\sum_{T\ni u} q_T \leq \frac{d^2}{n} \quad (3)$$

$$\forall T \in \mathcal{S}, \quad q_T \geq 0 \quad (4)$$

Figure 2 Dual-LP.

Proof. Let $\{q_T\}_{T\in\mathcal{S}}$ denote an optimal, feasible solution to Dual-LP. For any element $u \in U$, define $w(u) = \sum_{T\ni u} q_T$ to be the total weight of all the sets that contain u. From the dual constraints, we have

$$\forall S \in \mathcal{S}, \quad \sum_{u\in S} w(u) \leq \frac{d^2}{n}$$

Since each $S \in \mathcal{S}$ has exactly d elements, we can rewrite the above as

$$\forall S \in \mathcal{S}, \quad \sum_{u\in S}\left(w(u) - \frac{d}{n}\right) \leq 0$$

Multiplying each inequality by q_S and adding over all $S \in \mathcal{S}$ yields

$$\sum_{S\in\mathcal{S}}\left(\sum_{u\in S}\left(w(u) - \frac{d}{n}\right) q_S\right) \leq 0$$

$$\sum_{u\in U}\left(\sum_{S\ni u}\left(w(u) - \frac{d}{n}\right) q_S\right) \leq 0$$

$$\sum_{u\in U} w(u)\left(w(u) - \frac{d}{n}\right) \leq 0$$

Using the fact that $y(x-y) \leq x(x-y)$ for any two real numbers x and y, we get that $\forall u \in U$, $\frac{d}{n}(w(u) - \frac{d}{n}) \leq w(u)(w(u) - \frac{d}{n})$. Thus,

$$\sum_{u\in U}\frac{d}{n}\left(w(u) - \frac{d}{n}\right) \leq 0$$

$$\sum_{u\in U} w(u) \leq d \quad (5)$$

On the other hand, we have $\sum_{u\in U} w(u) = \sum_{u\in U}\sum_{T\ni u} q_T = \sum_{T\in\mathcal{S}}\sum_{u\in T} q_T = d\sum_{T\in\mathcal{S}} q_T$. Inequality 5 then shows that $\sum_{T\in\mathcal{S}} q_T \leq 1$. ◀

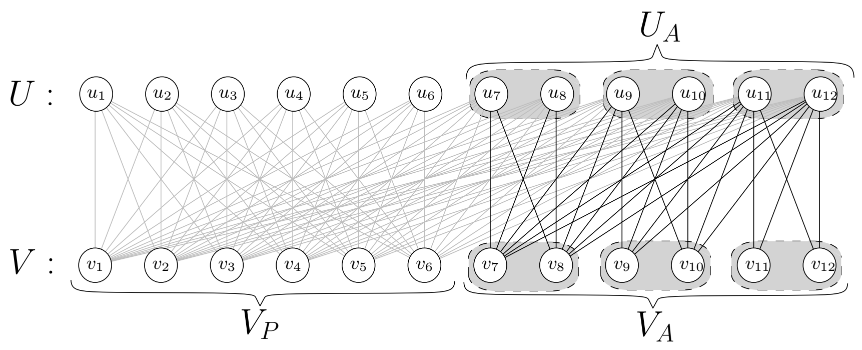

Figure 3 Hard instance for $n = 12$, $d = 6$, and $t = 1$.

6 Hardness of Semi-Online Bipartite Matching

In this section, we show that no algorithm solving the semi-online bipartite matching problem can have a competitive ratio better than $1 - \delta e^{-\delta}$. The construction is similar in spirit to the original bound for online bipartite matching of [13]. However, rather than using a graph whose adjacency matrix is upper triangular, the core hardness comes from using a block upper triangular matrix.

6.1 Graph Construction

The constructed instance will have a perfect matching in G. Let $d = |V_A| = \delta n$ be the number of adversarial online nodes and set $t = d^{2/3}$ (it's only important that $t = o(d)$). Assume for simplicity that $\frac{t}{\delta}$ is an integer. We construct the graph as follows (refer to Figure 3 for an illustration).

- Let $U = \{u_1, \dots, u_n\}$ be the n offline nodes, $V_P = \{v_1, \dots, v_{n-d}\}$ be the $n - d$ predicted online nodes and $V_A = \{v_{n-d+1}, \dots v_n\}$ be the d adversarial online nodes.
- Let the predicted graph H be a complete bipartite graph between U and V_P.
- Pick d nodes uniformly at random from U to be neighbors of V_A. Without loss of generality, let these nodes be $U_A = \{u_{n-d+1}, \dots u_n\}$. Partition the d nodes in each of U_A and V_A in blocks of $\frac{t}{\delta}$ consecutive nodes. Let $U_A^k = \{u_{n-d+(k-1)\frac{t}{\delta}+1}, \dots, u_{n-d+(k)\frac{t}{\delta}}\}$ and $V_A^k = \{v_{n-d+(k-1)\frac{t}{\delta}+1}, \dots, v_{n-d+(k)\frac{t}{\delta}}\}$ denote the k^{th} blocks of offline and online nodes respectively. For each $j \le k$, connect all online nodes in V_A^j to all offline nodes in U_A^k. Notice that the adjacency matrix on this part of the graph looks like a block upper triangular matrix.
- Finally, the online nodes arrive in order, i.e. v_i arrives before v_j whenever $i < j$.

6.2 Analysis

After the first $n - d$ nodes have arrived, any online algorithm can do no better than guess which offline nodes to leave unmatched uniformly at random. Let $\tilde{d} \ge d$ be the number of offline nodes left unmatched by the best online algorithm after the arrival of all $n - d$ predicted nodes. So at this point, we are left with a bipartite graph with d adversarial online nodes and $\tilde{d}$ offline nodes such that each of the n total offline nodes is available with probability $\tilde{d}/n = \tilde{\delta}$.

Consider a block U_A^k of offline nodes. Since each node is available with probability $\tilde{\delta}$, in expectation $\tilde{t} = (\frac{\tilde{\delta}}{\delta})t$ nodes from the block remain available. Further, since nodes are chosen to remain available using sampling without replacement, we can obtain tight concentration around $\tilde{t}$. In particular, if t_k denotes the number of available nodes remaining in block U_A^k, by Hoeffding bounds we obtain that $\mathbb{P}r(|t_k - \tilde{t}| \geq \tilde{t}^{2/3}) \leq 2e^{-\delta \cdot \tilde{t}^{1/3}}$. Hence by a union bound over the $\frac{\delta^2 n}{t}$ blocks, we have that *every* block has $\tilde{t} \pm \tilde{t}^{2/3}$ available nodes with high probability (as $t \to \infty$). At this point, it is somewhat clearer why blocks were chosen. Had we used single edges (as in the construction of [13]), many of them would have become unavailable, making the analysis difficult.

Let G' denote the remaining graph, that is, the graph with d adversarial online nodes and $\tilde{d}$ remaining offline nodes. At this point, we'll analyze the water-filling algorithm [9] on G'. By [9], this is the best deterministic algorithm for *fractional* matching in the adversarial setting. Further, a lower bound on the performance of this algorithm provides a lower bound for any randomized algorithm for integer matchings.

▶ **Lemma 13.** *The water-filling algorithm achieves a fractional matching of total weight at most* $\delta n \cdot \left(1 - e^{-\tilde{\delta}(1+o(1))} + o(1)\right)$ *on the graph* G'.

Proof. Recall that in the water-filling algorithm, for each arriving online node, we spread its total weight of 1 across its incident edges so that the total fractional matching across the adjacent offline nodes is as even as possible. Let $B = \frac{d}{t/\delta} = \frac{\delta^2 n}{t}$ be the number of blocks.

For simplicity, let's first assume that each block has exactly $\tilde{t}$ available nodes, rather than $\tilde{t} \pm o(\tilde{t})$. By construction, each online node in the first block is connected to $B\tilde{t}$ available nodes. Every online node in the second block is connected to $(B-1)\tilde{t}$ available nodes, and so on, with every online node in the k-th block connected to $(B-k+1)\tilde{t}$ available nodes. Hence, in the water-filling algorithm, for each of the first t/δ online nodes, we will give $1/(B\tilde{t})$ weight to every available offline node. Then we will give a weight of $1/(B\tilde{t} - \tilde{t})$ to every available offline neighbor for each of the next t/δ online nodes and so on. This process continues until the weight we have given to the last available offline node is 1, at which point we cannot allocate any more weight.

Consider the weight given to last available offline node. After seeing the first $k+1$ blocks, this is

$$\frac{t}{\delta}\left(\frac{1}{\tilde{t}B} + \frac{1}{\tilde{t}(B-1)} + \ldots + \frac{1}{\tilde{t}(B-k)}\right) \geq (1/\delta)(t/\tilde{t}) \cdot \int_{B-k+1}^{B+1} \frac{1}{x} dx = (1/\delta)(t/\tilde{t}) \ln\left(\frac{B+1}{B-k+1}\right)$$

In our case, the number of available offline nodes in each block is between $\tilde{t} - \tilde{t}^{2/3}$ and $\tilde{t} + \tilde{t}^{2/3}$, w.h.p. So the amount of weight assigned to the last available node after block $k+1$ is at least

$$\ln\left(\frac{B+1}{B-k+1}\right)\left(\frac{1}{\delta}\right)\left(\frac{t}{\tilde{t} + \tilde{t}^{2/3}}\right) \geq \ln\left(\frac{B+1}{B-k+1}\right)\left(\frac{1}{\tilde{\delta}}\right)\left(\frac{1}{(1+\tilde{t}^{-1/3})}\right)$$

Note that once we have given a total weight of 1 to the last node, the water-filling algorithm will not be able to distribute any more weight. Hence, the water-filling algorithm stops after k blocks, with k being at most the smallest integer satisfying

$$\ln\left(\frac{B+1}{B-k+1}\right)\left(\frac{1}{\tilde{\delta}}\right)\left(\frac{1}{(1+\tilde{t}^{-1/3})}\right) \geq 1$$

In this case,

$$k = (B+1)(1 - e^{-\tilde{\delta}(1+\tilde{t}^{-1/3})}) = (B+1)(1 - e^{-\tilde{\delta}(1+o(1))})$$

Since each block allocates a total weight of $\frac{t}{\delta}$, the total weight of the fractional matching obtained by the water-filling algorithm is at most

$$\left(\frac{t}{\delta}\right)(B+1)\left(1-e^{-\tilde{\delta}(1+o(1))}\right) = \left(\frac{t}{\delta}\right)\left(\frac{\delta^2 n}{t}+1\right)\left(1-e^{-\tilde{\delta}(1+o(1))}\right)$$
$$= \delta n\left(1-e^{-\tilde{\delta}(1+o(1))}\right)\left(1+\frac{t}{\delta^2 n}\right)$$

Since, by construction, we have $t = o(d)$, this is $\delta n\left(1-e^{-\tilde{\delta}(1+o(1))}+o(1)\right)$ as desired. ◀

▶ **Theorem 14.** *No (randomized) algorithm for the semi-online bipartite matching problem can achieve a competitive ratio better than* $1-\delta e^{-\delta}$.

Proof. Lemma 13 shows that after matching $n-\tilde{d}$ predicted vertices, the best randomized algorithm can match at most $\delta n\left(1-e^{-\tilde{\delta}(1+o(1))}+o(1)\right)$ of the adversarial vertices. Let M be the matching found by any randomized algorithm on the graph G. Hence, we have

$$\begin{aligned}\mathbb{E}[|M|] &\le n-\tilde{d}+\delta n\left(1-e^{-\tilde{\delta}(1+o(1))}+o(1)\right) \le n-\delta n+\delta n\left(1-e^{-\delta(1+o(1))}+o(1)\right)\\ &= n-\delta n e^{-\delta(1+o(1))}+\delta n o(1)\\ &= n\left(1-\delta e^{-\delta(1+o(1))}+\delta o(1)\right)\end{aligned}$$

Since G has a perfect matching of size n, the competitive ratio is upper bounded by $\left(1-\delta e^{-\delta}\right)$ as $n\to\infty$. ◀

7 Extensions - Imperfect Predictions and Agnosticism

In this section, we consider a more general model where we allow the predicted graph to have small random errors. We define the (d,ϵ) semi-online model as follows - We are given a predicted graph $H=(U,V,E_H)$, where $|U|=|V|=n$. As before, U are the offline nodes and V are the online[5] nodes. However, we do not explicitly separate V into predicted and adversarial nodes; all nodes are seen by the offline preprocessing stage, but some subset of these nodes will be altered adversarially.

An adversary selects up to d online nodes and may arbitrarily change their neighborhoods. In addition, we allow the realized graph G to introduce small random changes to the remaining predicted graph after the adversary has made its choices. Specifically, each edge in H not controlled by the adversary is removed independently with probability ϵ. Further, for each $u\in U, v\in V$, we add edge (u,v) (if it does not already exist in the graph) independently with probability $\epsilon|M|/n^2$, where M is a maximum matching in H. Note that in expectation, we will add fewer than $\epsilon|M|$ edges; simply adding edges with probability ϵ (instead of $\epsilon|M|/n^2$) would overwhelm the embedded matching. We call an algorithm *agnostic* if it does not know the d nodes chosen by the adversary during the preprocessing (offline) phase. There are two variants - either the algorithm knows the value of d or it does not. We show a hardness result in the former case and consider algorithms in the latter case.

We first consider agnostic algorithms to find integral matchings in this (d,ϵ) semi-online model and give a hardness result and a corresponding tight algorithm for the case when $\epsilon=0$.

[5] The algorithm does not know the arrival order of nodes in V.

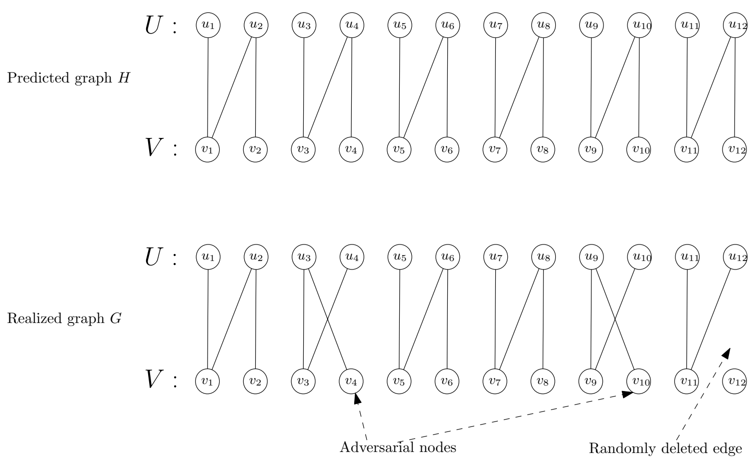
Predicted graph H
U :
V :
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12}
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12}
Realized graph G
Adversarial nodes
Randomly deleted edge

Figure 4 Hard instance for Agnostic algorithms.

▶ **Theorem 15.** *In the (d, ϵ) semi-online model with $d < n/4$, no (randomized) agnostic algorithm can find a matching of size more than $n - d - \epsilon(n - 3d) + O(\epsilon^2 n)$ in expectation, taken over the randomness of the algorithm and the randomness of the realized graph. This holds even if d is known in advance by the algorithm.*

Proof. Assume n is even. Our hard instance consists of the following predicted graph H: For each integer $i \in [0, \frac{n}{2})$, add edges (v_{2i+1}, u_{2i+1}), (v_{2i+1}, u_{2i+2}), and (v_{2i+2}, u_{2i+2}). This creates $n/2$ connected components. See Figure 4 for an illustration.

The adversary chooses d components uniformly at random. Let $\mathcal{A} = \{i_1, i_2, \ldots, i_d\} \subset [0, \frac{n}{2})$ denote the indices of the d components selected by the adversary. For each index $i \in \mathcal{A}$, the adversary then selects v_{2i+2} and changes its neighborhood so it only connects with u_{2i+1} (instead of u_{2i+2}).

For simplicity, let's first consider the case when $\epsilon = 0$. The algorithm can do no better than picking some $p \in [0, 1]$, and matching v_{2i+1} to u_{2i+1} with probability p, and matching v_{2i+1} to u_{2i+2} otherwise, for all i. The algorithm then matches v_{2i+2} to its neighbor, if possible. Now, for all $i \in \mathcal{A}$ (components selected by the adversary), this gets an expected matching of size $p + 2(1 - p) = 2 - p$. On the other hand, for all $i \notin \mathcal{A}$, the expected matching is size $2p + (1 - p) = 1 + p$. Since there are d components with an adversary and $n/2 - d$ components without, this gives a total matching of size $(2 - p)d + (1 + p)(n/2 - d) = n/2 + d + p(\frac{n}{2} - 2d)$. This is maximized when $p = 1$ (since $d < n/4$) to yield a matching of size $n - d$.

When $\epsilon > 0$, the algorithm still should set $p = 1$; if the desired edge is removed, then the algorithm will match with whatever node is available. Components with an adversarial node *gain* an edge in the matching when the edge (v_{2i+1}, u_{2i+1}) is removed since the algorithm is forced into the right choice; if both edges (v_{2i+1}, u_{2i+1}) and (v_{2i+1}, u_{2i+2}) are removed, we neither gain nor lose. The expected gain is $\epsilon - \epsilon^2$. Components without an adversarial node lose an edge in the matching whenever either edge (v_{2i+1}, u_{2i+1}) or edge (v_{2i+2}, u_{2i+2}) is removed, and they lose an additional edge if all three edges of the component are removed. So the expected loss is $2\epsilon - \epsilon^2 + \epsilon^3$ Since there are d components with adversarial nodes and $n/2 - d$ without, this is a total of loss of

$$-d(\epsilon - \epsilon^2) + (n/2 - d)(2\epsilon - \epsilon^2 + \epsilon^3) = \epsilon(n - 3d) - O(\epsilon^2 n)$$

Hence, the total matching is size $n - \epsilon(n - 3d) + O(\epsilon^2 n)$, as claimed. ◀

▶ **Theorem 16.** *Given a predicted graph H with a perfect matching, suppose there are d adversarial nodes and $\epsilon = 0$ as described above in the (d, ϵ) semi-online model. Then there is an agnostic algorithm that does not know d that finds a matching of expected size $n - d$.*

Proof. Before any online nodes arrive, find a perfect matching M in H. In the online stage, as each node v arrives, we attempt to identify v with an online node in the predicted graph with the same neighborhood, and match v according to M. If no node in the predicted graph has neighborhood identical to v, we know that v is adversarial and we can simply leave it unmatched. (Note that adversarial nodes can mimic non-adversarial nodes, but it doesn't actually hurt us since they are isomorphic.) The predicted matching had size n, and we lose one edge for each adversarial node, so the obtained matching has size $n - d$. ◀

7.1 Fractional matchings for predictions with errors

In this section, we show that we can find an almost optimal *fractional* matching for the (d, ϵ) semi-online matching problem.

We use a result from [22], which gives a method of reconstructing a fractional matching using only the local structure of the graph and a single stored value for each offline node. They provide the notion of a *reconstruction function.* Their results extend to a variety of linear constraints and convex objectives, but here we need only a simple reconstruction function. For any positive integer k, define $g^k : (\mathtt{R}_0^+)^k \to (\mathtt{R}_0^+)^k$ by

$$g^k(\alpha_1, \alpha_2, \ldots, \alpha_k) = (\alpha_1 - \max(0, z),\ \alpha_2 - \max(0, z),\ \ldots,\ \alpha_k - \max(0, z))$$

where z is a solution to $\sum_j \min(\max(0, \alpha_j - z), 1) = 1$.

The reconstruction function g is this family of functions. Note that this is well-defined: there is always a solution z between -1 and the largest α_j, and the solution is unique unless $z \leq 0$.

The result of [22] assigns a value α_u to each u in the set of offline nodes, and reconstructs a matching on the fly as each online node arrives, using only the neighborhood of the online node and the stored α values. Crucially, the reconstruction assigns reasonable values even when the neighborhood is different than predicted. In this way, it is robust to small changes in the graph structure.

▶ **Lemma 17** (Restated from [22]). *Let g_i^k be defined as above, and let $H = (U, V, E_H)$ be a bipartite graph with a perfect matching of size $n = |U| = |V|$. Then there exist values α_u for each $u \in U$ (which can be found in polynomial time) such that the following holds: For all $v \in V$, define $x_{u_i,v} = g_i(\alpha_{u_1}, \alpha_{u_2}, \ldots \alpha_{u_k})$, where $u_1, u_2, \ldots, u_k$ is the neighborhood of v. Then x defines a fractional matching on H with weight n.*

Interested readers can find the proof in the full version. Given this reconstruction technique, we can now describe the algorithm:

- In the preprocessing phase, find the α_u values for all $u \in U$ using Lemma 17.
- In the online phase, for each online node v, compute $\tilde{x}_{u_i,v} = g_i(\alpha_{u_1}, \ldots, \alpha_{u_k})$, where $u_1, \ldots, u_k \in \Gamma_G(v)$, as described above. Assign weight $\tilde{x}_{u_i,v}$ to the edge from u_i to v; if that would cause node u_i to have more than a total weight 1 assigned to it, just assign as much as possible.

Note that we make the online computation based on the neighborhood in G, the realized graph, although the α_u values were computed based on H, the predicted graph. We have the following.

▶ **Theorem 18.** *In the (d, ϵ) semi-online matching problem in which the predicted graph has a perfect matching, there is a deterministic agnostic algorithm that gives a fractional matching of size $n(1 - 2\epsilon - \delta)$ in expectation, taken over the randomized realization of the graph. The algorithm does not know the value of d or the value of ϵ in advance.*

Proof. If the realized graph were exactly as predicted, we would give the fractional assignment x guaranteed in Lemma 17, which has weight n. However, the fractional matching that is actually realized is somewhat different. For each online node that arrives, we treat it the same whether it is adversarial or not. But we have a few cases to consider for analysis:

- Case 1: The online node v is adversarial. In this case, we forfeit the entire weight of 1 in the matching. We may assign some fractional matching to incident edges. However, we count this as 'excess' and do not credit it towards our total. In this way, we lose at most δn total weight.
- Case 2: The online node v is not adversarial, but it has extra edges added through a random process. There are at most ϵn such nodes in expectation. In this case, we treat them the same as adversarial. We forfeit the entire weight of 1, and ignore the 'excess' assignment. This loses at most ϵn total weight in expectation.
- Case 3: The online node v is as exactly as predicted. In this case, we correctly calculate x_{uv} for each $u \in \Gamma(v)$. Further, we assign x_{uv} to each edge, unless there was already 'excess' there. Since we never took credit for this excess, we will take x_{uv} credit now. So we do not lose anything in this case.
- Case 4: The online node v is as predicted, except each edge is removed with probability ϵ (and no edges are added). In this case, when we solve for z, we find a value that is bounded above by the true z. The reason is that in the predicted graph, we solved $\sum_{u\in\Gamma(v)} \min(\max(0, \alpha_u - z), 1) = 1$ for z when computing g. In the realized graph, this same sum has had some of its summands removed, meaning the solution in z is at most what it was before. So the value of $\tilde{x}_{uv}$ that we calculate is at least x_{uv} for all u in the realized neighborhood. We take a credit of x_{uv} for each of these, leaving the rest as excess. Note that we have assigned 0 to each edge that was in the predicted graph but missing in the realized graph. Since each edge goes missing with probability ϵ, this is a total of at most ϵn in expectation.

So, the total amount we lose in expectation is $2\epsilon n + \delta n$. Since the matching in the predicted graph has weight n, the claim follows. ◀

7.2 Semi-Online Algorithms For Ski Rental

In this section, we consider the semi-online ski rental problem. In the classical ski rental problem, a skier needs to ski for an unknown number of days and on each day needs to decide whether to rent skis for the day at a cost of 1 unit, or whether to buy skis for a higher cost of b units and ski for free thereafter. We consider a model where the skier has perfect predictions about whether or not she will ski on a given day for a few days in the time horizon. In addition, she may or may not ski on the other days. For instance, say the skier knows whether or not she's skiing for all weekends in the season, but is uncertain of the other days. The goal is to design an algorithm for buying skis so that the total cost of skiing is competitive with respect to the optimal solution for adversarial choices for all the days for which we have no predictions.

Let x denote the number of days that the predictions guarantee the skier would ski. Further, it is more convenient to work with the fractional version of the problem so that it costs 1 unit to buy skis and renting for z (fractional) days costs z units. In this setting, we know in advance that the skier will ski for at least x days. There is a randomized algorithm that guarantees a competitive ratio of $1/(1-(1-x)e^{-(1-x)})$. Our analysis is a minor extension of an elegant result of [11].

▶ **Theorem 19.** *There is a* $\dfrac{e}{e-(1-x)e^x}$ *competitive randomized algorithm for the semi-online ski-rental problem where x is a lower bound of the number of days the skier will ski.*

Proof. Without loss of generality, we can assume that all the days with a prediction occur before any of the adversarial days arrive. Otherwise, the algorithm can always pretend as if the predictions have already occurred, since only the number of skiing days is important and not their order. Recall that x denotes the number of days that the predictions guarantee the skier would ski. Let $u \geq x$ be the actual number of days (chosen by the adversary) that she will ski. Since buying skis costs 1, the optimal solution has a cost of $\min(u, 1)$. Clearly, if $x \geq 1$, we must always buy the skis immediately and hence we assume that $0 \leq x < 1$ in the rest of the section. Further, even the optimal deterministic algorithm buys skis once $z = 1$, so we may assume that $u \leq 1$.

Let $p_x(z)$ denote the probability that we buy the skis on day z, and let $q(x)$ denote the probability that we buy skis immediately. Recall that p_x is implicitly a function of the prediction x. Given a fixed number of days to ski u, we can now compute the expected cost of the algorithm as

$$\mathrm{Cost}(x, u) = q(x) + \int_0^u (1+z) \cdot p_x(z) dz + \int_u^1 u \cdot p_x(z) dz$$

Our goal is to choose a probability distribution p so as to minimize $\mathrm{Cost}(x,u)/\min(u,1)$ while the adversary's goal is to choose u to maximize the same quantity. We will choose p_x and q so that $\mathrm{Cost}(x,u)/\min(u,1)$ is constant with respect to u. As we noted, $u \leq 1$, so $\min(u,1) = u$. Setting the $\mathrm{Cost}(x,u) = c \cdot u$ for constant c and taking the derivative with respect to u twice gives us

$$0 = \frac{\partial}{\partial u} p_x(u) - p_x(u)$$

Of course, p_x must also be a valid probability distribution. Thus, we set $p_x(z) = (1 - q(x)) \cdot \dfrac{e^z}{e - e^x}$ for $z \geq x$. For $z < x$, we set $p_x(z) = 0$ since there is no reason to buy skis while $z < x$ if we did not already buy it immediately.

Recalling that we set $\mathrm{Cost}(x,u) = c \cdot u$, we can substitute $p_x(z)$ and solve for $q(x)$, finding

$$q(x) = \frac{xe^x}{e - (1-x)e^x}$$

Hence, the competitive ratio is thus given by

$$\frac{\mathrm{Cost}(x,u)}{u} = \frac{1}{u}\left(q(x) + \frac{1-q(x)}{e-e^x} \int_x^u (1+z)e^z dz + \frac{1-q(x)}{e-e^x} \int_u^1 u \cdot e^z dz \right)$$

Substitute $q(x)$, and after some manipulation, this becomes

$$\frac{\mathrm{Cost}(x,u)}{u} = \frac{e}{e-(1-x)e^x}$$

Note that when $x = 0$, this becomes the classic ski rental problem, and the above bound is $e/(e-1)$, as expected. ◀

References

1 S. Albers and M Hellwig. Semi-online scheduling revisited. *Theor. Comput. Sci.*, 443:1–9, 2012.

2 J. Balogh and J Békési. Semi-on-line bin packing: a short overview and a new lower bound. *Cent. Eur. J. Oper. Res.*, 21(4):685–698, 2013.

3 Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královic, Richard Královic, and Tobias Mömke. On the Advice Complexity of Online Problems. In *ISAAC*, pages 331–340, 2009.

4 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 2005.

5 Sebastien Bubeck and Aleksandrs Slivkins. The best of both worlds: Stochastic and adversarial bandits. In *COLT*, pages 42.1–42.23, 2012.

6 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

7 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *SODA*, pages 468–485. Society for Industrial and Applied Mathematics, 2012.

8 Pascal Van Hentenryck and Russell Bent. *Online stochastic combinatorial optimization.* The MIT Press, 2009.

9 Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000. doi:10.1016/S0304-3975(99)00140-1.

10 Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

11 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP Acknowledgment and Other Stories about e/(e-1). *Algorithmica*, 36(3):209–224, 2003.

12 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

13 Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

14 Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving Online Algorithms Using ML Predictions. In *NIPS*, 2018.

15 Euiwoong Lee and Sahil Singla. Maximum Matching in the Online Batch-Arrival Model. In *IPCO*, pages 355–367, 2017.

16 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *ICML*, pages 3302–3311, 2018.

17 Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Trans. Algorithms*, 8(1):2:1–2:29, 2012.

18 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *STOC*, pages 597–606, 2011.

19 Andres Muñoz Medina and Sergei Vassilvitskii. Revenue Optimization with Approximate Bid Predictions. In *NIPS*, pages 1856–1864, 2017.

20 Aranyak Mehta. Online Matching and Ad Allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

21 Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *SODA*, pages 1690–1701, 2012.

22 Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *EC*, pages 109–118, 2010.