Improved Bounds for Online Stochastic Matching

Bahman Bahmani[⋆1] and Michael Kapralov[⋆⋆2]

[1] Stanford University, bahman@stanford.edu
[2] Stanford University, kapralov@stanford.edu

Abstract. We study the online stochastic matching problem in a form motivated by Internet display advertisement. Recently, Feldman et al. gave an algorithm that achieves 0.6702 competitive ratio, thus breaking through the $1-1/e$ barrier. One of the questions left open in their work is to obtain a better competitive ratio by generalizing their two suggested matchings (TSM) algorithm to d-suggested matchings (d-SM).

We show that the best competitive ratio that can be obtained with the *static* analysis used in the d-SM algorithm is upper bounded by 0.76, even for the special case of d-regular graphs, thus suggesting that a *dynamic* analysis may be needed to improve the competitive ratio significantly. We make the first step in this direction by showing that the RANDOM algorithm, which assigns an impression to a randomly chosen eligible advertiser, achieves $1 - e^{-d}d^d/d! = 1 - O(1/\sqrt{d})$ competitive ratio for d-regular graphs, which converges to 1 as d increases. On the hardness side, we improve the upper bound of 0.989 on the competitive ratio of any online algorithm obtained by Feldman et al. to $1 - 1/(e + e^2) \approx 0.902$. Finally, we show how to modify the TSM algorithm to obtain an improved 0.699 approximation for general bipartite graphs.

1 Introduction

Bipartite matching problems are among the central problems in combinatorial optimization with numerous applications. In this paper, we study a variant of the online bipartite matching problem motivated by applications in Internet advertising.

We are given a graph $G = (A, I, E)$, where A is a set of advertisers, I is the set of impression types and E is the set of edges between them. For $a \in A, i \in I$ the presence of an edge (a, i) indicates that advertiser a is bidding for impression type i. Advertisers are fixed and known in advance, while impressions of different types come online, and the task of the algorithm is to assign each impression to an advertiser as soon as the impression arrives or discard the impression. The objective is to maximize the number of matched impressions, subject to

⋆ Research supported by a Stanford Graduate Fellowship.
⋆⋆ Research supported by a Stanford Graduate Fellowship.

each advertiser being assigned at most one impression and each impression being assigned to at most one advertiser, i.e. subject to the allocation forming a matching.

If nothing is known about the graph G in advance and edges incident to impression types are revealed online, then we are in the online adversarial setting, for which an algorithm achieving the optimal approximation ratio of $1 - 1/e$ was given in [1]. A less restrictive model is the random order model, which has received a significant amount of attention recently. This model assumes that impression types and the graph G are unknown but appear in the random order. Even the greedy algorithm has a competitive ratio of $1 - 1/e$ in this model [2]. It is also known that no deterministic algorithm can achieve approximation ratio better than 0.75 and no randomized algorithm better than 0.83 [2]. However, the adversarial model or random order model may be too restrictive for applications, where statistics about frequencies of various impression types may be known. In the iid model, which we consider in this paper, we assume that impressions are drawn from a known distribution $\mathcal{D}$ on the set of *impression types* for which the edges to advertisers are known in advance. The $1 - 1/e$ bound of [1] was the best known approximation for the stochastic online model, until recently Feldman et al. gave an algorithm that achieves 0.6702 competitive ratio.They also showed that no algorithm can achieve a competitive ratio better than 0.989. Their main algorithmic technique is a novel application of the power of two choices consisting of carefully computing two edge-disjoint (near)-matchings offline to guide the online allocation. One of the questions left open by Feldman et al. is to obtain a better competitive ratio by generalizing the two suggested matchings (TSM) idea to d-suggested matchings (d-SM).

1.1 Our results and techniques

In this paper we give improved upper and lower bounds for three questions related to the stochastic matching problem.

First, we show that the *static* analysis inherent in the d-suggested matchings algorithm of [3] cannot achieve an competitive ratio better than 0.76, suggesting that a *dynamic* analysis of the allocation process would be needed to achieve a higher competitive ratio, even for the special case of d-regular graphs. We make a first step in that direction by proving

Theorem 1. *For any $\epsilon > 0$ as the size of the d-regular expected graph $G = (A, I, E)$ goes to ∞, with high probability the algorithm RANDOM has a competitive ratio at least as large as $1 - e^{-d}d^d/d! - \epsilon$.*

The analysis is based on a linear program that lower bounds the evolution of the allocation process as new impressions arrive. Interestingly, this bound coincides with the static bounds of $1 - 1/e$ and $1 - 2/e^2$ for $d = 1, 2$, but is strictly better than the static bound for any $d > 2$.

The proof of Theorem 1 proceeds by showing that RANDOM obtains a matching of size at least $(1 - e^{-d}d^d/d! - \epsilon)n$ whp. It is interesting to note that

this bound is tight, i.e. there exists a family of d-regular graphs for which the expected size of the matching constructed by RANDOM is $(1-e^{-d}d^d/d!)n+o(n)$.

On the upper bound side, we prove

Theorem 2. *The expected competitive ratio of any algorithm for online stochastic matching is bounded above by* $\frac{1-1/e+1/e^2}{1+1/e} \approx 0.901$.

This improves upon the 0.9898 hardness result obtained in [3].

Finally, we show that the $\frac{1-2/e^2}{4/3-2/3e} \approx 0.67$ approximation for general bipartite graphs obtained in [3] can be slightly improved without going beyond subgraphs of degree 2:

Theorem 3. *There exists an algorithm that for any* $\epsilon > 0$ *one has competitive ratio at least* $0.699 - \epsilon$ *with high probability.*

It is interesting to point out that the worst case competitive ratio is achieved in Theorem 3 in the regime when a bound on the performance of any online algorithm similar to Theorem 2 holds. In particular, a slightly better bound can be proved if one compares the performance of the algorithm to be best possible performance of an *online* algorithm. However, we do not pursue this direction here.

1.2 Other related work

The related online decision problem, the *ad allocation problem*, which is motivated by sponsored search auctions, has been studied extensively in the literature. In this problem, every edge $(a, i) \in E$ has a weight that corresponds to the bid of advertiser a for impression i, and each advertiser has a budget. Now in addition to forming a matching, the allocation of impressions to advertisers now has to satisfy budget constraints, and the objective is to maximize the total value of bids for the assigned impression. The offline version of the problem in NP-hard ([4]) and several approximation algorithms have been designed ([2], [5]). Approximation algorithms have also been designed for the online setting ([6], [2], [7], [8]), typically achieving $1 - 1/e$ approximation under the assumption that bids are small compared to budgets. An $1 - \epsilon$ approximation for any $\epsilon > 0$ has recently been obtained by [9] in the *random permutation model* under the assumption that the optimum is significantly larger than the maximum bid.

1.3 Preliminaries

We start with a formal definition of the problem. We are given a bipartite graph $G = (A, I, E)$ of advertisers A and impression types I, together with an integer number of impressions of type i that we expect to see. We denote this number by e_i and define $n := \sum_{i \in I} e_i$. We assume that at each step an impression of type i arrives with probability e_i/n and denote this distribution by $\mathcal{D}$. Then, n i.i.d. draws $i \sim \mathcal{D}$, of impression types arrive, and as soon as an impression i arrives, we should either assign it to an advertiser a such that $(i, a) \in E$, or not

assign i at all. Each advertiser $a \in A$ may be assigned at most once. Our goal is to maximize the number of assigned impressions.

More formally, If $D(i)$ is the set of arrivals of the impression type i, the *realization graph* $\hat{G}$ is defined as $\hat{G} = (A, \hat{I}, \hat{E})$, where $\hat{I} = \cup_{i \in I} D(i)$ and $\hat{E} = \{(a, i') |\ i' \in D(i) \text{ and } (a, i) \in E\}$. Then, we would like to design an algorithm ALG, which having access to $G, \mathcal{D}$ and n finds a matching, in an online fashion, in $\hat{G}$, such that with high probability $ALG(\hat{G})/OPT(\hat{G}) \geq \alpha$ for some α, where $OPT(\hat{G})$ is the size of the maximum matching in $\hat{G}$.

It is shown in [3] that one can assume without loss of generality that $\mathcal{D}$ is the uniform distribution (indeed, it is sufficient to duplicate impression types in I an appropriate number of times to achieve that). Hence, in the rest of this paper we will make this assumption, and thus also have that $n = |I|$.

1.4 d-Suggested Matchings Algorithm

A family of algorithms for the introduced problem, namely d-Suggested Matching (d-SM), was presented in [3]. The d-SM algorithm works by finding d edge-disjoint (near) matchings in the expected graph, G, and then use these matchings to guide the online assignments of impressions.

More precisely, the algorithm finds (e.g. by using max flows in a boosted graph) d edge-disjoint near-matchings $M_1, M_2, \ldots, M_d$, and then for the j^{th} arrival of the impression type i, the algorithm tries to match the impression to the advertiser a connected to i in M_j. If a is already matched, then the impression is not assigned at all.

It is proved in [3] that the 1-SM algorithm achieves $1-1/e = 0.63$ competitive ratio, and that two near-matchings can be carefully chosen so that 2-SM (aka TSM) achieves $\frac{1-2/e^2}{4/3-2/3e} \approx 0.67$ competitive ratio for general bipartite graphs. But, the problem of finding a decomposition of general bipartite graphs into d edge-disjoint near-matchings to achieve a better competitive ratio is left open. This problem is trivial for d-regular graphs, which can be decomposed into a union of d edges-disjoint perfect matchings. However, we show that the d-SM algorithm cannot achieve a competitive ratio better than 0.76 for *any* d, thus suggesting that a *dynamic* analysis is needed to improve the current competitive ratio significantly. We make a first step towards such dynamic analysis by proving that the RANDOM algorithm achieves a competitive ratio of $1 - e^{-d}d^d/d! = 1 - O(1/\sqrt{d})$ on d-regular graphs.

1.5 Organization

In section 2 we prove an upper bound on the performance of the d-SM algorithm and analyze the performance of RANDOM algorithm on d-regular graphs. Section 3 contains the improved hardness result, and section 4 presents an improved version of the TSM algorithm.

2 Dynamic analysis for regular graphs

In this section we first upper bound the performance of the d-SM algorithm for any d and then analyze the RANDOM algorithm for allocation on regular graphs, showing that it achieves a competitive ratio of $1 - e^{-d}d^d/d!$ on d-regular graphs.

2.1 Upper-bounding the performance of d-SM

In this section, we give an upper-bound on the performance of any d-SM algorithm (for any value of d). In particular, we show that (with $|I| = n$) no d-SM algorithm can produce a matching of a larger size than $0.76n$, even on regular graphs:

Theorem 4. *No d-SM algorithm (for any value of d) can achieve a larger matching size than $0.76n$, even in expectation on regular graphs.*

The proof of this theorem is deferred to the full version of the paper. The bottleneck in the performance of the d-SM algorithm is that it is inherently a static algorithm. That is after it calculates the offline matchings, it doesn't consider the dynamics of the arrival sequence. In particular, near the end of the sequence, many of the advertisers are already matched, so when an impression arrives and checks its corresponding advertiser, there is a good chance that advertiser is already matched and the impression is thrown away. However, there may be other advertisers connected to this impression that are not yet matched and could be used for this impression but don't get utilized. In the next section, we present a simple dynamic algorithm that can get a much better competitive ratio for regular graphs.

2.2 The Random Algorithm

As in the previous section, $G = (A, I, E)$ is assumed to be a d-regular bipartite graph with $|I| = n$. The algorithm simply assigns each arriving impression to an unmatched connected advertiser chosen uniformly at random.

In this section, we analyze the performance of this algorithm and show that it achieves a significantly better competitive ratio than d-SM on regular graphs.

We give a few definitions first.

Definition 1.
- *We denote the t^{th} arriving impression by i_t. Whenever we refer to time t, we mean the time just before the arrival of i_t.*
- *$A_u(t)$ is the set of unmatched advertisers at time t.*
- *S_t is all the state information at time t, i.e. the sequence of all impressions that have arrived by time t as well as the sequence of all assignments made by the algorithm up to time t.*

– *The set of impressions of remaining degree k at time t is denoted by $I_k(t)$. In other words,*

$$I_k(t) = \{i| \ |N(i) \cap A_u(t)| = k\} \quad \forall\, 0 \leq k \leq d$$

where $N(i)$ is the neighborhood of impression i.
– *The fraction of impressions of remaining degree k at time t:*

$$R_k(t) = \frac{|I_k(t)|}{n}$$

– *For any $i \in I$, $a \in A$, and $1 \leq t \leq n$:*

$$P(i,a,t) = Pr[i \text{ gets matched to } a \text{ at time } t \mid i_t = i,\, S_t]$$

– *for any $i \in I$,*

$$P(i,t) = Pr[\text{the remaining degree of } i \text{ changes at time } t \mid S_t]$$

– *For $1 \leq k \leq d$,*

$$P_k(t) = E[fraction\ of\ impression\ types\ whose\ remaining\ degree\ changes\ from\ k\ to\ k-1\ at\ time\ t|\ S_t]$$

– *We denote the expectations of the random variables defined above by corresponding lower-case letters. That is,*

$$r_k(t) = E[R_k(t)],\ p(i,a,t) = E[P(i,a,t)],$$
$$p(i,t) = E[P(i,t)],\ p_k(t) = E[P_k(t)]$$

where all expectations are with respect to the randomness in S_t.
– *We denote the matching constructed by the algorithm up to time t by M_t and the final matching (i.e. after the arrival of i_n) by M.*

Fact 5 *The following are clear:*

$$P(i,t) = \sum_{a \in N(i) \cap A_u(t)} \sum_{j \in N(a)} P(j,a,t)/n$$

$$P_k(t) = \frac{1}{n} \sum_{i \in I_k(t)} P(i,t)$$

We would like to analyze $|M|$, the size of the final matching constructed by the algorithm. We start with a lemma, which is proved in the full version of the paper.

Lemma 1. *$|M|$ is sharply concentrated around $E[|M|]$.*

So, we only need to analyze the expectations (or in other words, only $E[|M|]$). We have the following lemmas, proved in the full version of the paper:

Lemma 2.

$$E[|M|] = \sum_{t=1}^{n}(1 - r_0(t))$$

Lemma 3.

$$r_k(t+1) - r_k(t) = p_{k+1}(t) - p_k(t) \quad \forall\, 1 \le k < d$$

$$r_0(t+1) - r_0(t) = p_1(t)$$

$$r_d(0) = 1,\ \sum_{k=0}^{d} r_k(t) = 1,\ r_k(t) \ge 0$$

The next lemma is essential in our analysis. But, before presenting it, we notice that since the algorithm picks a connected unmatched advertiser uniformly at random at each time, we have:

$$P(i,a,t) = \frac{1\{a \in N(i) \cap A_u(t)\}}{|N(i) \cap A_u(t)|}$$

This fact is used in the proof of the lemma, which is given in the full version of the paper:

Lemma 4.

$$\sum_{k'=1}^{k} p_{k'}(t) \le \frac{d}{n} \sum_{k'=1}^{k} r_{k'}(t)$$

From the previous two lemmas, we get that to lower-bound $E[|M|]$, we can solve the following LP:

$$\begin{array}{lll}
\text{maximize} & \sum_{t=1}^{n} r_0(t) & \\
\text{s.t.} & r_0(t+1) - r_0(t) = p_1(t) & 1 \le t \le n-1 \\
& r_k(t+1) - r_k(t) = p_{k+1}(t) - p_k(t) & \forall\, 1 \le k < d,\ 1 \le t \le n-1 \\
& r_d(t+1) - r_d(t) = -p_d(t) & 1 \le t \le n-1 \\
& r_d(1) = 1,\ r_k(1) = 0 & \forall\, 0 \le k < d \\
& \sum_{k'=1}^{k} p_{k'}(t) \le \frac{d}{n} \sum_{k'=1}^{k} r_{k'}(t) & \forall\, 1 \le k \le d,\ 1 \le t \le n
\end{array}$$

We present the solution to the above LP. To simplify the presentation of the solution, we introduce the following notation:

Definition 2. *For any non-negative integers k, t:*

$$\theta(k,t) = \left(\frac{d}{n}\right)^k \left(1 - \frac{d}{n}\right)^{t-k-1} \binom{t-1}{k}$$

By the properties of the binomial coefficients, we have $\theta(k,t) = 0$ for any $k > t-1$, and also $\sum_{k=0}^{t-1} \theta(k,t) = 1$.

Now, the following proposition, proven in the full version of the paper, gives the LP solution:

Proposition 1. *The solution to the above LP is as follows:*

$$r_{d-k}(t) = \theta(k,t) \;\; \forall\, 0 \le k < d,\, 1 \le t \le n$$
$$r_0(t) = 1 - \sum_{j=1}^{d} r_j(t) \;\; \forall\, 1 \le t \le n$$
$$p_j(t) = \frac{d}{n} r_j(t) \;\; \forall\, 1 \le j \le d,\; 1 \le t \le n$$

From the above proposition, we get that the performance of the algorithm is bounded as follows:

$$\frac{E[|M|]}{n} \ge \frac{1}{n} \sum_{k=0}^{d-1} \sum_{t=1}^{n} \left(\frac{d}{n}\right)^k \left(1 - \frac{d}{n}\right)^{t-1-k} \binom{t-1}{k}$$

It can be seen that as $n \to \infty$, the above summation converges to:

$$\sum_{k=0}^{d-1} \int_0^1 e^{-dx} (dx)^j / j! dx = 1 - e^{-d} d^d / d!$$

This and the sharp concentration of $|M|$ give the following result:

Theorem 6. *For any $\epsilon > 0$ as the size of the d-regular expected graph $G = (A, I, E)$ converges to ∞, with high probability the algorithm RANDOM achieves a competitive ratio at least as large as $1 - e^{-d} d^d / d! - \epsilon$.*

It remains to note that the bound on the size of the matching is tight: it is achieved for a family of graphs G_n that consist of a disjoint union of n/d copies of $K_{d,d}$.

Comparing this lower bound on the performance of RANDOM (which converges to 1 as $1 - 1/\sqrt{d}$ when $d \to \infty$) with the constant (0.76) upper bound on the performance of d-SM, we conclude that the RANDOM algorithm achieves a significantly better competitive ratio than d-SM on regular graphs.

3 Hardness

Feldman et al. [3] prove that no online algorithm for the stochastic matching problem can achieve a better competitive ratio than 0.99. In this section, we prove that in fact no online algorithm can achieve a better competitive ratio than 0.902:

Theorem 7. *The expected competitive ratio of any algorithm for online stochastic matching is bounded above by* $\frac{1-1/e+1/e^2}{1+1/e} \approx 0.901062$.

Proof. We exhibit an instance $G = (A, I, E)$ of the problem for which no online algorithm yields a matching of expected size larger than $\frac{1-1/e+1/e^2}{1+1/e}|I|$, even though a matching of size $(1-\epsilon)|I|$ exists with high probability. Let $G = (I_1 \cup I_2, A_1 \cup A_2, E)$ be defined as follows:

1. $|I_1| = |A_2| = n$, $|A_1| = |I_2| = n/e$.
2. There is a perfect matching M between I_1 and A_2.
3. There is a complete graph between I_2 and A_2, and between I_1 and A_1.

We first show that with high probability there exists a matching of size $(1+1/e-\epsilon)n$ in the realization graph. By the balls and bins analysis, there will be $(1-1/e)n$ distinct arrivals in I_1. Route the first arrivals in I_1 to A_2 and the rest to A_1 (this is possible since there is a complete graph between I_1 and A_1, and $|A_1| = |I_1|/e$). This leaves $|A_2|/e$ advertisers in A_2 unmatched. These are matched to impressions from I_2. Hence, a matching of size $(1+1/e-\epsilon)n$ exists with high probability.

Consider an advertiser $a \in A_2$ that is matched by ALG to an impression in I_2. Fix a time $t \in [1, (1+1/e)n]$ (recall that time t corresponds to the moment just before the arrival of the t^{th} impression). We call advertiser a *good* if $M(a)$ has not arrived and *bad* otherwise (where $M(a)$ is the impression connected to a by an edge in M). Denote the number of good advertisers at time t by X_t, and the number of good advertisers at the end of the sequence by $X = X_{(1+1/e)n+1}$. Note that the size of the final matching constructed by ALG is upper bounded by $n/e + (1-1/e)n + X$.

At time t an impression $i \in I_1$ arrives with probability $1/(1+1/e)$ and it is a unique arrival incident on a good advertiser with probability X_t/n. Hence, we have:

$$\mathbf{E}[X_{t+1} - X_t | X_t] \leq \frac{1/e}{1+1/e} - \frac{1}{1+1/e} X_t/n, X_1 = 0,$$

which implies that

$$E[X_t] \leq (n/e)\left(1 - \left(1 - \frac{1}{(1+1/e)n}\right)^{t-1}\right)$$

This yields that $E[X] \leq (n/e)(1-e^{-1})$, and hence, in expectation, at most $\frac{1}{e}(1-e^{-1})n$ of the advertisers that are matched to I_2 are good at the end of the sequence. This concludes that the expected size of the matching constructed by ALG can not be larger than $n/e + (1-1/e)n + 1/e(1-1/e)n = (1+1/e-1/e^2)n$, and hence ALG's competitive ratio can not be better than $\frac{1+1/e-1/e^2}{1+1/e} \approx 0.901062$, which finishes the proof. □

4 Improved competitive ratio for general graphs

In this section we present an algorithm for the online stochastic matching problem in general graphs that yields an 0.699-approximation to the offline optimum.

We start by giving an outline of the algorithm of [3]. In the offline phase, the TSM algorithm constructs a boosted flow graph G_f, where each $a \in A$ is connected to a source s by an edge with capacity 2, each $i \in I$ is connected to a sink t by an edge with capacity 2 and each edge of G is directed from A to I and assigned capacity 1. One then finds an integral maxflow in G_f. Denote the edges that carry flow by E_f. The flow edges form a union of paths and cycles, which are colored blue and red in an alternating fashion (with some extra care taken for various types of paths). The online algorithm proceeds as follows: (1) assigns the first arrival along the blue edge if it is still available, and discards otherwise, (2) assigns the second arrival to the red edge if it is still available, discards otherwise.

It was shown in [3] that this algorithm yields an $(1-2/e^2)/(4/3-2/(3e)) \approx 0.67029$ approximation, which is tight for their algorithm. In what follows we show how to modify the flow graph constructed by the TSM algorithm to obtain an approximation ratio of 0.699 against the optimal offline algorithm.

Our algorithm works on the flow graph constructed by the algorithm of [3]. As in [3], denote the reachability min-cut corresponding to the max-flow in the boosted flow graph G_f by $(A_S \cup I_S, A_T \cup I_T)$, where $A_S \cup I_S$ is the source side of the cut and $A_T \cup I_T$ is the sink side, and denote the set of edges crossing the cut by E_δ. It was shown in [3] that the min-cut can always be chosen so that the edges in E_δ form a matching, and we make that assumption on E_δ.

The intuition behind the algorithm is as follows. Recall that the analysis of the TSM algorithm uses the reachability cut $(A_S \cup I_S, A_T \cup I_T)$ in the boosted flow graph to bound the optimum in $\hat{G}$. The key insight is that this bound on OPT can sometimes be improved by using a different cut in $\hat{G}$. In order to exploit this fact, however, we first modify the flow graph obtained in the TSM algorithm as described in Algorithm 1 below. Intuitively, the modification is based on the fact that the value of the flow in the boosted graph G_f does not translate directly into the performance of TSM on the subgraph given by E_f. In particular, the performance of the algorithm improves if the flow obtained via the max-flow computation in G_f is rerouted so that it is more evenly spread among vertices in A_S and I_T. This can be done using two max-flow computations. The two min-cuts obtained from these computations are then used to define a subgraph H of G for which we can bound OPT more carefully. The cut that we use to bound $OPT(\hat{H})$ then depends on the structure of the new set of flow edges that was obtained.

Denote by G_S the subgraph induced by vertices of $A_S \cup I_S$ in G and by G_T the subgraph induced by vertices of $A_T \cup I_T$ in G. For $k = 0, 1, 2$ define

$$A_S^k = \{a \in A_S : a \text{ carries } k \text{ units of flow in } E_f\}$$
$$I_T^k = \{i \in I_T : i \text{ carries } k \text{ units of flow in } E_f\}.$$

Algorithm 1

Input: $G = (A, I, E)$, the set of edges E_f carrying max-flow in G_f. The min-cut $(A_S \cup I_S, A_T \cup I_T)$.

Output: E^* - a modified set of paths and cycles in G.

1. Orient edges of G_S from A_S to I_S, orient flow edges from I_S to A_S.

2. Connect vertices in A_S^0 to a source by edges of capacity 1, vertices in A_S^2 to a sink by edges of capacity 1, assign capacity 1 to edges of G.

3. Find max-flow in G_S. Denote the set of edges carrying flow by E_f^S.

4. Orient edges of G_T from I_T to A_T, orient flow edges from A_T to I_T.

5. Connect vertices in I_T^0 to a source by edges of capacity 1, vertices I_T^2 to a sink by edges of capacity 1, assign capacity 1 to edges of G.

6. Find max-flow in G_T. Denote the set of edges carrying flow by E_f^T.

7. Set $E^* := E_\delta \cup E_f^T \cup E_f^S$.

8. Decompose E^* into a disjoint union of paths and cycles. Color the edges of the paths and cycles as follows (the same as in [3], given here for completeness):

- Color cycles alternately blue and red;
- Color odd length paths alternately blue and red, with more blue than red;
- For even paths that start and end in I, color the first two edges blue, then alternate red and blue.
- For even paths that start and end in A, alternate blue and red.

9. *(TSM)* At runtime, assign the first arrival along the blue edge if it is still available, discard otherwise. Assign the second arrival to the red edge if it is still available, discard otherwise.

We now analyze the performance of Algorithm 1. We start with some definitions. Denote by the $\mathcal{P}^\delta$ and $\mathcal{C}^\delta$ the set of paths and cycles respectively in E^* that contain edges from E_δ. Note that since there are no flow edges between I_S and A_T, paths and cycles in the flow graph are either contained in one of G_S and G_T or contain an edge of E_δ. Here and in what follows we will sometimes view the set of paths and cycles $\mathcal{P}^\delta \cup \mathcal{C}^\delta$ as a set of vertices.

Orient edges of G_S and G_T as described above (note that edges from E_δ are not oriented since reachability is defined only within G_S and G_T). Denote by $\mathcal{P}_S^*$ the set of paths in $E^* \cap E(G_S)$ that are reachable from $A_S \cap (\mathcal{P}^\delta \cup \mathcal{C}^\delta)$ using edge orientation in G_S and by $\mathcal{P}_T^*$ the set of paths in $E^* \cap E(G_T)$ that are reachable from $I_T \cap (\mathcal{P}^\delta \cup \mathcal{C}^\delta)$ using edge orientation in G_T. Also, define $\mathcal{P}^* := \mathcal{P}_S^* \cup \mathcal{P}_T^*$.

Let H be the subgraph induced by $\mathcal{P}^\delta \cup \mathcal{C}^\delta \cup \mathcal{P}^*$. Define $\tilde{A}_T := A_T \setminus V(H)$, $\tilde{I}_T := I_T \setminus V(H)$, $\tilde{I}_S := I_S \setminus V(H)$, $\tilde{A}_S := A_S \setminus V(H)$.

All proofs from this section have been deferred to the full version of the paper.

The following lemmas are important for our analysis:

Lemma 5. *There are no edges between $\tilde{I}_T$ and $A \setminus \tilde{A}_T$, and no edges between $\tilde{A}_S$ and $I \setminus \tilde{I}_S$*

Lemma 6. *For any $\epsilon > 0$ one has*

$$ALG(\hat{G}) \geq (1 - 2/e^2)|\tilde{A}_T| + (1 - 2/e^2)|\tilde{I}_S| + ALG(\hat{H}) - \epsilon$$
$$OPT(\hat{G}) \leq |\tilde{A}_T| + |\tilde{I}_S| + OPT(\hat{H}) + \epsilon$$

with high probability.

We now proceed to prove that Algorithm 1 gives a 0.699-approximation to the best offline solution. In light of Lemma 6 and the fact that $0.699 < 1 - 2/e^2$ it suffices to prove this guarantee for the graph H.

Lemma 7. *For any $\epsilon > 0$ one has $ALG(\hat{H})/OPT(\hat{H}) \geq 0.699 - \epsilon$ whp.*

We can now prove Theorem 3:
Proof of Theorem 3: It follows from Lemma 7 and Lemma 6 that for any $\epsilon > 0$ Algorithm 1 achieves a competitive ratio of at least $0.699 - \epsilon$. □

Remark 1. It can be seen from the proof of Lemma 7 that the worst case performance of Algorithm 1 occurs when $(1 - 1/e)|E_\delta| = |A'_S| - |I'_S|$, i.e. in the regime in which a bound similar to Theorem 2 on the performance of any online algorithm holds. It is in fact possible to get a slightly better bound if one compares the performance of Algorithm 1 against the best possible performance of an online algorithm, but we do not present that analysis here for the sake of clarity. It can also be noted that the original analysis of the TSM algorithm is tight even when compared against the performance of the best possible online algorithm.

References

1. Karp, R., Vazirani, U., Vazirani, V.: An optimal algorithm for online bipartite matching. STOC (1990)
2. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to adwords. SODA (2008)
3. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: Beating $1 - 1/e$. FOCS (2009)
4. Azar, Y., Birnbaum, B., Karlin, A., Mathieu, C., Nguyen, C.: Improved approximation algorithms for budgeted allocations. ICALP (2008)
5. Srinivasan, A.: Budgeted allocations in the full-information setting. RANDOM/APPROX (2008)
6. Mehta, A., Saberi, A., Vazirani, U., Vazirani, V.: Adwords and generalized online matching. FOCS (2005)
7. Buchbinder, N., Jain, K., Naor, J.: Online primal-dual algorithms for maximizing ad-auctions. ESA (2007)
8. Kalyanasundaram, B., Pruhs, K.R.: An optimal deterministic algorithm for online b -matching. Theoretical Computer Science (2000)
9. Devanur, N., Hayes, T.: The adwords problem: online keyword matching with budgeted bidders under random permutations. EC (2009)
10. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)