Biobjective Online Bipartite Matching

Gagan Aggarwal
Google Research
Mountain View, CA
gagana@google.com

Yang Cai
MIT
Cambridge, MA
ycai@mit.edu

Aranyak Mehta
Google Research
Mountain View, CA
aranyak@google.com

George Pierrakos
U.C. Berkeley
Berkeley, CA
georgios@cs.berkeley.edu

ABSTRACT

Motivated by Online Ad allocation when there are multiple conflicting objectives, we introduce and study the problem of Biobjective Online Bipartite Matching, a strict generalization of the standard setting of Karp, Vazirani and Vazirani [9], where we are allowed to have edges of two colors, and the goal is to find a matching that is both large and balanced at the same time. We study both deterministic and randomized algorithms for this problem; after showing that the single color upper bounds of $1/2$ and $1-1/e$ carry over to our biobjective setting as well, we show that a very natural, albeit hard to analyze, deterministic algorithm achieves a competitive ratio of 0.343. We next show how a natural randomized algorithm matches this ratio, through a simpler analysis, and how a clever – and perhaps not immediately obvious – generalization of RANKING can beat the $1/2$ bound and get a competitive ratio of 0.573, coming close to matching the upper bound of 0.63.

Categories and Subject Descriptors

G.2.1 [**Discrete Mathematics**]: Combinatorics—*Permutations and Combinations*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph Algorithms*

General Terms

Algorithms, Theory

Keywords

Online Algorithms, Bipartite Matching

1. INTRODUCTION

Online Matching and Allocation is a topic that has received considerable interest in the recent past, both due to its practical importance in Internet Ad allocation, and also due to purely theoretical interest and advances that it has generated. The classic problem was introduced in [9]: There is a graph $G(U,V,E)$, with only U known in advance, and vertices from V arriving online in an adversarial order. As a vertex $v \in V$ arrives, its incident edges are revealed. The online algorithm can match v to some neighbor $u \in U$ which has not yet been matched (if any such exist), or choose to leave v unmatched. Each match made is irrevocable, and the goal is to maximize the size of the matching obtained.

Several simple ideas, such as an arbitrary choice of available neighbor (Greedy) or a random choice (Random) provide a competitive ratio of $1/2$, but no greater. The algorithm provided in [9], called RANKING, first permutes the vertices of U in a random permutation, and then matches each arriving vertex to that available neighbor which is highest according to the permutation. This algorithm is optimal, providing a ratio of $1-1/e$.

Motivated by applications in Internet advertising – in which U corresponds to advertisers, and V to arriving ad slots – many different variations of this problem have been studied recently, which includes different matching constraints, as well as different input models (we describe some of this related work in 1.4).

In this paper, we introduce a new direction in this literature. In all the variants studied so far, the constraints of the problem vary, but there is one objective function, namely the weight of the matching. However, in practice, there are typically multiple objective functions which an algorithm designer needs to consider, e.g. quality, ROI, revenue, efficiency, etc. Sometimes, these objective functions are not well-aligned and the designer needs to achieve a good balance among them as prescribed by business needs. In this paper, we introduce a new problem which captures this motivation.

1.1 Model and Problem Defintion

We consider the basic setting of online bipartite matching, to which we introduce the most natural notion of multiple objectives, as follows: There is a bipartite graph $G(U,V,E)$ with $|U| = |V| = n$, where each edge is colored either Red or Blue. As before, U is known

STOC 2013 Palo Alto, California USA

in advance, and the vertices of V arrive online together with their incident edges. The goal is to find a matching that "balances" the two colors, as described next.

Given a matching in G, let $MatchRed$ be the number of Red edges in the matching and let $MatchBlue$ be the number of Blue edges in the matching. Also let $MaxRed$ be the size of the maximum matching in the graph G restricted to its Red edges and let $MaxBlue$ be the size of the maximum matching on the Blue edges of G. Then, we define the objective function as

$$\min\left\{\frac{MatchBlue}{MaxBlue}, \frac{MatchRed}{MaxRed}\right\}$$

The competitive ratio of an online algorithm is defined as the minimum over all inputs of the ratio of its objective value to the optimal offline objective value.

We first observe that if $MaxBlue$ and $MaxRed$ are of very different magnitudes, then no deterministic algorithm A can achieve a non-trivial competitive ratio. Indeed, consider the following graph G_A, in which the adversary presents the algorithm with a sequence of blue edges $\{(u_1, v_1), \ldots, (u_i, v_i)\}$, each one connecting an arriving node v_i to a new node u_i (i.e. u_i's only neighbor so far is v_i), until A matches v_i to u_i through a blue edge (if this does not happen then $MatchBlue = 0$ and we are done). Once A matches v_i to u_i, the next node v_{i+1} is connected through a red edge to u_i, which is already taken. Continuing in this manner it is obvious that we can guarantee $MatchRed = 0$ (while $MaxRed > 0$), so $\min\{MatchBlue/MaxBlue, MatchRed/MaxRed\} = 0$, while the optimal objective function is strictly positive, resulting in a competitive ratio of 0.

In order to bypass this issue and get to the essence of the difficulty in handling two objective functions, we assume that $MaxBlue$ and $MaxRed$ are comparable. In fact, we will assume that there exists a perfect matching on Red edges as well as a perfect matching on Blue edges, i.e. $MaxBlue = MaxRed = n$. With this assumption, optimizing the objective function reduces to optimizing $\min\{MatchBlue, MatchRed\}$, and we will work with this objective function henceforth. Also, with this assumption, it is not hard to show that there exists a matching with $n/2 - o(n)$ Red as well Blue edges, giving us the following Proposition (proof in Appendix A).

PROPOSITION 1. *When $MaxRed = MaxBlue = n$, the offline optimal value of $\min\{MatchBlue, MatchRed\}$ is between $n/2 - O(\sqrt{n})$ and $n/2$.*

Notation: For the rest of the paper, for the given instance, we pick a perfect matching on Red edges, and a perfect matching on Blue edges, which we will refer to as the (canonical) Red and Blue perfect matchings in the graph. For $w \in U \cup V$ we define $N_b(w)$ as the match of w in the Blue perfect matching, and $N_r(w)$ as the match of w in the Red perfect matching.

1.2 Results

We start by looking at deterministic algorithms. As for any matching problem, we start by trying a greedy algorithm: We quickly note that if the algorithm always takes an edge whenever one is available (i.e., its other end point in U is not yet matched), then its competitive ratio can be no better than 0. Wlog assume that such an algorithm picks a Blue edge for the first arriving vertex if that vertex has exactly one incident edge of each color. Then for the instance in Figure 1 since such an algorithm never skips, it is forced to pick all the Blue edges, and no Red edges.

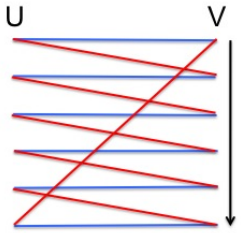

Figure 1: Deterministic Greedy does no better than 0.

Thus our first important observation is that any deterministic online algorithm has to sometimes *skip*, i.e., keep the arriving vertex unmatched even though it has an available neighbor. Note that this is very different from all previous settings in online matching and allocation problems, where greedy algorithms achieve a competitive ratio of 1/2. In Theorem 5 we prove an analogous upper bound of 1/2 for randomized algorithms.

Since the objective is to balance the two colors, a second natural approach is the following algorithm, called BALANCE[1]. Before describing BALANCE we introduce some notation which will we use repeatedly in the exposition and analysis of our deterministic algorithms.

First, we define *CurrentBlue* (resp. *CurrentRed*) as the number of Blue (resp. Red) edges in the current matching produced during the algorithm. We also define the following operations:

`NoPreference`: Match the arriving vertex to any available neighbor (irrespective of color).

`OnlyBlue`: Match the arriving vertex via an available Blue edge. If only Red edges are available then leave the vertex unmatched.

`OnlyRed`: Match the arriving vertex via an available Red edge. If only Blue edges are available then leave the vertex unmatched.

Algorithm BALANCE

For each arriving vertex v, switch case:

1. *CurrentBlue* = *CurrentRed* : `NoPreference`
2. *CurrentBlue* > *CurrentRed* : `OnlyRed`
3. *CurrentRed* > *CurrentBlue*: `OnlyBlue`

We will show in Theorem 1 that this algorithm has a competitive ratio of 1/3. In fact, we take BALANCE as the baseline algorithm that we will try to improve upon.

Next we try to improve upon this ratio. We observe that BALANCE attempts to keep the two colors *perfectly*

[1]Note that in the existing literature, *balance* typically refers to balancing the spend or the load on different vertices – here it refers to balancing the two colors in the current matching.

balanced, and *skips* an arriving vertex whenever the colors are unbalanced and no edge of the lagging color is available. So one can try giving the algorithm some leeway in how balanced the two colors need to be during its run. To do that we introduce two new operations:
`PreferBlue`: If a Blue edge is available, match the arriving vertex via one of them. Else match it via a Red edge, if one is available.
`PreferRed`: If a Red edge is available, match the arriving vertex via one of them. Else match it via a Blue edge, if one is available.
We are now ready to describe our next algorithm called c-BALANCE.

Algorithm c-BALANCE

For each arriving vertex v, switch case:

1. $CurrentBlue = CurrentRed$: `NoPreference`
2. $CurrentBlue > c \cdot CurrentRed$: `OnlyRed`
3. $CurrentRed > c \cdot CurrentBlue$: `OnlyBlue`
4. $CurrentRed < CurrentBlue \leq c \cdot CurrentRed$: `PreferRed`
5. $CurrentBlue < CurrentRed \leq c \cdot CurrentBlue$: `PreferBlue`

This algorithm tries to pick up some extra edges of the leading color (in case the arrival of edges of the leading colors slows down in the future), but not too many (as picking edges now reduces available vertices on the left, reducing the opportunity to pick lagging-color edges in the future).

In Theorem 2, we show that this algorithm achieves a competitive ratio of $\frac{2c}{(1+c)(2+c)}$, for $1 \leq c \leq 2$, and that this analysis is tight, i.e., there is a family of examples on which it performs no better. This function is maximized at $c = \sqrt{2}$, giving the algorithm a competitive ratio of 0.343. We note that while c-BALANCE seems more flexible than BALANCE (and its analysis is *much* more difficult), it does not perform significantly better. Nevertheless, it is our first example which beats the baseline of BALANCE.

From a hardness standpoint, in Theorem 3, we show that no deterministic algorithm can achieve a ratio better than 1/2. This follows from a generalization of the standard hard example for the classic single color problem. The gap between the best deterministic algorithm and the upper bound remains open.

Next we turn our attention to randomized algorithms. The performance of a randomized algorithm is defined as

$$Exp\ \min\{MatchRed, MatchBlue\}$$

Note that we have to be careful in making the definition. We do not define it as the minimum of the two expectations: This is an easier objective, and does not match our motivation. Indeed, to optimize the Min-of-Exps objective, one could simply flip a coin in the beginning to choose one of the two colors to optimize, and run RANKING only for edges of that color. This would give an expected $(1 - 1/e)/2$ matches for each color, and a ratio of $1 - 1/e$. Clearly, the Exp-of-Mins objective is 0 for this algorithm.

We first observe, in Theorem 4, that there is no randomized online algorithm that achieves a competitive ratio better than $1 - 1/e$. This follows from a slight modification of the standard hard instance (in the single color problem) of the upper-triangular adjacency matrix.

In terms of designing an algorithm, a natural first attempt to balance the two colors is the following:

Algorithm PROBGREEDY

For each arriving vertex v:
w.p. 1/2, match v via an available Red edge, if any.
w.p. 1/2, match v via an available Blue edge, if any.

Note that PROBGREEDY "skips" a vertex when it has available edges of a single color, but the vertex chooses the other color. In Theorem 6, we show that this algorithm achieves a competitive ratio of 1/3, with probability almost 1. Next, we analyze a generalization of the above algorithm:

Algorithm p-PROBGREEDY

For each arriving vertex v:
w.p. p, match v via an available Red edge, if any.
w.p. p, match v via an available Blue edge, if any.
w.p. $1 - 2p$, leave v unmatched.

This algorithm sometimes skips vertices even when edges of both colors are available. Surprisingly, it does better than PROBGREEDY for some values of p. In Theorem 6, we show that this algorithm has a competitive ratio of $\frac{2p(1-p)}{1+p}$ with probability almost 1. Amazingly, this is the same bound as that achieved by c-BALANCE for $p = \frac{1}{1+c}$, which also explains the identical competitive ratio of 1/3 achieved by BALANCE and PROBGREEDY. We do not know if there is a deeper connection between the two algorithms, and we leave this possibility as an open question. As in c-BALANCE, this function is maximized for $p = \sqrt{2} - 1$, giving a competitive ratio of 0.343.

So far, randomization has not helped us get an improved competitive ratio, although the randomized algorithms are simpler in that they do not need to keep any state. Also, the analysis for p-PROBGREEDY is significantly simpler than that of c-BALANCE. The next question to ask is whether there is another randomized algorithm that actually improves upon the ratios we have so far. Stepping back, we recall the classic randomized algorithm for online bipartite matching (without colors), namely RANKING [9], defined below.

Algorithm RANKING

1. Pick a permutation σ of the vertices in U uniformly at random.
2. For each arriving vertex:
   Match it to the highest available neighbor (according to σ).

This algorithm by itself has a competitive ratio of 0, which can be seen again from the instance in Figure 1.

With probability almost 1, one of the first few arriving vertices will pick its Blue neighbor, and after that, every vertex is forced to pick its Blue neighbor. Is there a way to extend RANKING for our bi-objective problem?

In our first attempt, called DISJOINTRANKING, we try to remove edges from the graph in order to break up the bi-objective instance into two disjoint instances, one of each color, and then use RANKING on each instance independently.

Algorithm DISJOINTRANKING

1. Each vertex in $U \cup V$ picks a color uar, and throws away edges of that color. Note that only those edges which were not thrown away by either endpoint survive.
2. Run Algorithm RANKING on the resulting subgraph.

It is easy to see that this algorithm can be run online. Next we note that the first step creates a subgraph with two vertex partitions, such that no edges go from one part to the other, the edges within one part are all Blue, while the edges within the other part are all Red. Since each edge survives with probability $1/4$, there is a Blue (resp. Red) matching of size $n/4 - O(\sqrt{n})$ in the Blue (resp. Red) part, whp. Also, running RANKING on this graph is equivalent to running it on each part separately. Hence, we will obtain a matching with $\approx (1 - 1/e)\frac{n}{4}$ Red edges as well as $\approx (1 - 1/e)\frac{n}{4}$ Blue edges. This shows that the algorithm achieves a competitive ratio of $\frac{1-1/e}{2} \simeq 0.316$.

DISJOINTRANKING throws away $3/4$ of the edges and loses a factor of $1/2$ immediately (since the size of the guaranteed matching drops to $n/4$ and OPT is $n/2$). So one can try and throw away fewer edges. One way to do this is to let vertices of only one side (either U or V) pick a color at random and throw away incident edges of that color. This gives us two algorithms, namely LEFTSUBGRAPHRANKING and RIGHTSUBGRAPHRANKING, defined below:

Algorithm LEFTSUBGRAPHRANKING

1. Each vertex in U picks a color uar, and throws away edges of that color.
2. Run Algorithm RANKING on the resulting subgraph.

Algorithm RIGHTSUBGRAPHRANKING

1. Each vertex in V picks a color uar, and throws away edges of that color.
2. Run Algorithm RANKING on the resulting subgraph.

These algorithms throw away fewer edges than DISJOINTRANKING, thereby ensuring a matching having $\approx n/2$ edges of each color in the resulting subgraph. So we can hope for an online algorithm that gets a matching with $\approx (1-1/e)n/2$ edges of each color, achieving a ratio of $1-1/e$. However, we have lost the partition into a Red and a Blue instance and the analysis needs to take into account the interaction between edges of different colors in the resulting subgraph. This interaction results in some loss. We show in Theorem 8 that the competitive ratio of LEFTSUBGRAPHRANKING is $3 - 4/\sqrt{e} \simeq 0.573$, and this analysis is tight.

Although RIGHTSUBGRAPHRANKING is very similar, our technique does not carry over immediately; the reason is that the analysis of RANKING looks at misses on the permuted (offline) side of the graph. In LEFTSUBGRAPHRANKING the offline side is also the side that randomly picks colors (i.e., all randomness is decided by the offline nodes), which enables our analysis. This is not the case with RIGHTSUBGRAPHRANKING, which makes it hard to point which vertex to charge to; we leave the analysis of RIGHTSUBGRAPHRANKING as an interesting open question.

Another obvious open question is to close the gap for randomized algorithms. If there is a better algorithm, is it a different generalization of RANKING?

	Lower Bound (Algorithm)	Upper Bound (Impossibility)
Det. w/o Skips	0	0
Deterministic	0.343	0.5
Rand. w/o Skips	?	0.5
Randomized	$3 - 4/\sqrt{e} \simeq 0.573$	$1 - 1/e \simeq 0.63$

Figure 2: Table of Results

1.3 Techniques

We analyze c-BALANCE using a charging argument that starts off as usual – for every miss, we can point to a hit, namely its perfect matching neighbor (of the appropriate color) that must have been matched. This, by itself, gives a very weak bound; the new idea is that, instead of allowing every hit to be charged by a miss, we identify a set of hits that cannot be charged by misses – These are precisely the hits that cause the imbalance to grow during the `PreferRed` and `PreferBlue` operations, since every time this happens it is because the perfect matching neighbor of that hit (of the lagging color) is itself a hit (and not a miss).

PROBGREEDY balances the two colors probabilistically, and we show that a set of natural constraints holds whp, which puts a bound on the expected size of the minimum color. For LEFTSUBGRAPHRANKING too, we use a charging argument, except that each miss is charged to several hits just like in the analysis of RANKING. However, unlike RANKING, we charge each miss not just to hits in events obtained by changing the location of the hit, but also to hits in events obtained by changing the color choice of the miss vertex. This lets us charge each miss to $2n$ hits, which is crucial, since the original map of RANKING gives a much worse factor. This gives a factor of $3 - 4/\sqrt{e}$ for the minimum of the expectations of the two colors.

However, when we try to bound the expectation of the minimum color, even with this augmented charging map, we get some hits potentially getting charged twice, which could lead to a factor of $1 - 2/e \simeq 0.26$. The key insight that helps us overcome this problem is that our charging map has the property of "locality", i.e. misses in a given event map to hits in "nearby" events (according to an appropriate metric). This in turn lets

us claim that the number of hits that get double-charged are small, giving us a nearly identical competitive ratio of $3 - 4/\sqrt{e} - o(1)$ for the expectation of the minimum.

1.4 Related Work

Motivated by applications in Internet advertising, several variants and generalizations of Online Matching have been studied recently, starting with the "Adwords" problem ([11], [3]), which generalized the setting to that with bids and budgets. Further generalizations included the Weighted-Vertex Matching problem [1], weighted edges with free-disposal ("Display Ads") [6], among many others. There have also been variants which study stochastic inputs for all these settings, as well as stochastic rewards (we do not mention all the references, since this literature is quite extensive by now). Ours is an orthogonal direction, in introducing multiple objectives to this set of problems. We have started with the basic matching problem, but the question extends to the entire set of variants.

In another thread of work, there is a long research tradition of studying combinatorial optimization problems with multiple objectives, initiated by the paper of Papadimitriou and Yannakakis [14], which identifies a relaxed notion of *Pareto set* (the set of all solutions that are not dominated by any other in all objectives). Our problem fits only loosely in this literature: being an online problem, we are interested in implementing one balanced solution as the vertices arrive, not the entire Pareto set of trade-offs.

A similar motivation to ours can be found in a series of papers [4, 5, 13], which study the problem of designing an auction to balance revenue and efficiency. From a practical view, the importance of multiple objectives like quality, ROI and revenue in budgeted ad allocation, were described in [8].

An unrelated set of online problems with two objectives were studied in [7]. A related offline problem of exact matchings, i.e. finding a matching with exactly k Red edges was studied in [12, 10, 15], and the problem of approximately sampling and counting exact matchings was studied in [2].

2. DETERMINISTIC ALGORITHMS

2.1 Analysis of Balance

THEOREM 1. BALANCE *achieves a competitive ratio of* 1/3*, and this is tight.*

PROOF. Consider a miss of the algorithm, i.e., a vertex $v \in V$ which is left unmatched at the end of the algorithm. When v arrives, BALANCE might have performed one of the three operations, depending on the values of *CurrentRed* and *CurrentBlue*. We charge the miss v to a hit $u \in U$ as follows:

If the algorithm performed `OnlyRed`, then it means that it was looking for an available Red edge, but could not find one. This means that $N_r(v)$ had already been matched earlier. We charge the miss v to the hit $N_r(v)$ via the Red edge between v and $N_r(v)$. Similarly, if the algorithm performed `OnlyBlue`, we charge the miss v to the hit $N_b(v)$ via the Blue edge. If the algorithm performed `NoPreference`, this means that there was no available neighbor (of either color) and both $N_b(v)$ and $N_r(v)$ were already matched. We charge the miss v to the hit $N_b(v)$ via the Blue edge (choosing one of the two arbitrarily).

Note that each hit (of $u \in U$) gets charged at most two times: once by a miss $N_b(u)$ via a Blue edge and once by a miss $N_r(u)$ via a Red edge. Therefore the number of misses is no more than twice the number of hits. Let h denote all the hits of $u \in U$, and m denote all the misses of $v \in V$. Then, we have shown that $m \leq 2h$. Also, $h + m = n$. This gives that the total number of hits $h \geq n/3$. Now, let h_b and h_r denote the number of Blue and Red edges in the final matching obtained by the algorithm. Note that the algorithm has the invariant that $|CurrentBlue - CurrentRed| \leq 1$, so $|h_b - h_r| \leq 1$, making both h_b and h_r at least $n/6 - 1$. Since $OPT \leq n/2$, by Proposition 1, we get that the competitive ratio is 1/3.

The tightness of the analysis follows from the tightness of the analysis of c-BALANCE in the next subsection. □

2.2 Analysis of c-Balance

Next, we analyze c-BALANCE for any $c \geq 1$, which is a generalization of BALANCE (in particular 1-BALANCE is identical to BALANCE).

THEOREM 2. c-BALANCE *achieves a competitive ratio of* $\frac{2c}{(1+c)(2+c)}$*, and this is tight.*

PROOF. We first define our **Charging Scheme:** A miss of an arriving vertex v will be charged to:

- hit $N_b(v)$ via a Blue edge, if v performed an `OnlyBlue` operation,
- hit $N_r(v)$ via a Red edge, if v performed an `OnlyRed` operation,
- hit $N_b(v)$ via a Blue edge *or* hit $N_r(v)$ via a Red edge (we will make a specific choice later), if v performed any of the remaining operations, since in these three cases, we know that neither of the two neighbors was available.

Suppose that the algorithm never performs an `OnlyRed` operation. In this case, we can charge each miss $v \in V$ to the hit $N_b(v)$ via the Blue edge $(v, N_b(v))$. Since we charge a hit through a Blue edge alone, a hit $u \in U$ is charged at most once (by a miss $N_b(u)$). Thus, the number of misses is no more than the number of hits, giving us at least $n/2$ hits in total. Since the imbalance between colors is no more than $1 : c$, the number of hits of each color is at least $\frac{n}{2(1+c)}$. Also, $OPT \leq n/2$, giving a competitive ratio of $\frac{1}{1+c}$, which is greater than the claimed ratio. The same analysis holds if the algorithm never performs an `OnlyBlue` operation.

When the algorithm performs both an `OnlyRed` and an `OnlyBlue` operation, it has to charge some misses through a Red edge and some misses through a Blue edge. Therefore, a given hit u might get charged twice, once by $N_b(u)$ and once by $N_r(u)$, irrespective of how we choose to charge misses incurred while performing the remaining three types of operations. A naive analysis

that simply argues that the number of misses is no more than twice the number of hits will give a competitive ratio of $\frac{2}{3(1+c)} \leq \frac{1}{3}$.

Hence, we need a new insight into the dynamics of the algorithm. For this, we examine the `PreferRed` and `PreferBlue` operations more closely – in fact, we examine not the misses but the hits obtained by these operations. We observe that each time an arriving vertex v picks a Red edge during a `PreferBlue` operation, it does so because there is no Blue edge available. Let $u = N_b(v)$. Then u is a hit. Moreover, $N_b(u)$ (which is v) is also a hit, so hit u will never get charged by a miss through a Blue edge, since the only such charge can come from a miss of $N_b(u)$. Simply put, whenever the algorithm picks an edge of the leading color, thereby making the matching less balanced, we can identify a hit which cannot be charged by two misses.

If we can lower bound the number of such operations, we can get a better upper bound on the number of misses. For this, we note that when the algorithm performs an `OnlyRed` or an `OnlyBlue` operation, its matching is in a fairly unbalanced state (the colors are in the ratio $c : 1$). Since the algorithm starts off in a balanced state, it must have performed several `PreferBlue` / `PreferRed` operations in which it made the matching less balanced. All that remains is to count such operations.

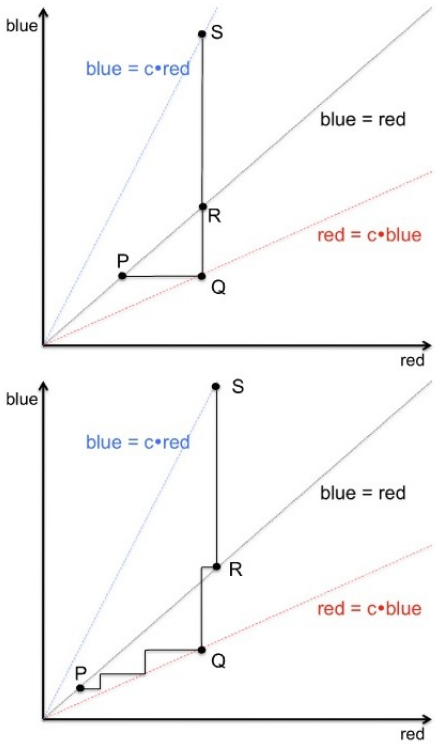

Figure 3: The states in the proof of Theorem 2; top graph is for the special (hardest) case, bottom graph is for the general case.

Special (hardest) case: In the following, we use the pair (*CurrentRed*, *CurrentBlue*) to summarize the current state of the matching. As a warm up to the full analysis, we will consider a particular form of run of the algorithm: in this the algorithm passes through the following states (for some x), and after leaving state S, it uses only `PreferRed`, `PreferBlue` and `NoPreference` operations. We will see later that this type of run is, in fact, the worst case for the algorithm.

P : (x, x)
Q : $(cx + 1, x)$
R : $(cx + 1, cx + 1)$
S : $(cx + 1, c(cx + 1) + 1)$

In this case, the Red edges picked in going from state P to state Q (except the very first one) are picked by the `PreferBlue` operation. Each of these, say for the arriving vertex v, identifies a hit $N_b(v)$ that will never be charged by a miss via a Blue edge. Thus, we have identified a total of $cx + 1 - x - 1 = (c - 1)x$ hits that will never be charged via a Blue edge. Let $Charge_B(\mathrm{Q})$ be the number of hits that occur up to state Q that can be charged via a Blue edge. The total number of hits up to state Q is $cx + 1 + x$. So

$$Charge_B(\mathrm{Q}) \leq (cx+1+x) - (c-1)x = 2x+1 \quad (1)$$

Similarly, the Blue edges picked in going from state R to state S (except the first one) are picked by the `PreferRed` operation and identify a set of hits that will never get charged via a Red edge. The number of such hits is $c(cx + 1) + 1 - (cx + 1) - 1 = (c - 1)(cx + 1)$. Let $Charge_R(\mathrm{S})$ be the number of hits that occur upto state S that can be charged via a Red edge. Since the total number of hits upto state S is $cx+1+c(cx+1)+1$,

$$\begin{aligned} &Charge_R(\mathrm{S}) \\ &\quad \leq (cx + 1 + c(cx + 1) + 1) - (c - 1)(cx + 1) \\ &\quad = 2cx + 3 \end{aligned} \quad (2)$$

Now we are ready to complete the specification of the **charging scheme**. Recall that for misses during operations `PreferBlue`, `PreferRed` and `NoPreference`, we had a choice to charge via a Red or a Blue edge. In the above run where the last "only" operation is an `OnlyRed` operation, we charge all of these misses via their Red perfect matching edge. The number of misses is equal to the number of charges made via Red edges plus the number of charges made via Blue edges. Since all the charges via Blue edges occur while performing `OnlyBlue` operations, all of them occur by the time the algorithm leaves state Q, and thus do not exceed $2x+1$, by Equation (1). Also, the total number of charges via Red edges to hits occurring up to state S do not exceed $2cx+3$, by Equation (2). Let m_1 be the total number of misses that are charged to hits occurring up to S. Then,

$$\begin{aligned} m_1 &\leq Charge_B(\mathrm{Q}) + Charge_R(\mathrm{S}) \\ &\leq 2x + 1 + 2cx + 3 \\ &= 2(c + 1)x + 4 \end{aligned} \quad (3)$$

Note that these include all the misses that occur before leaving state S, and may also include some misses that occur after leaving state S. The only misses that remain unaccounted for, call these m_2, are those that occur

after leaving state S and are charged via Red edges (as none of them occurs during an `OnlyBlue` operation) to some hit that occurs after leaving state S.

Let h_1 be the number of hits that occur before leaving state S and h_2 be the number of hits that occur after leaving state S. We just saw that

$$m_2 \leq h_2 \tag{4}$$

Also, from the description of state S,

$$h_1 = cx + 1 + c(cx + 1) + 1 \tag{5}$$

Combining equations (3), (4) and (5) and noting that $m_1 + m_2 + h_1 + h_2 = n$, we get

$$h_1 + h_2 \geq \frac{cn}{c+2} - O(1)$$

Since the imbalance between colors is no more than $1 : c$, the number of hits of each color is at least $\frac{cn}{(c+1)(c+2)}$. Also, $OPT \leq n/2$, which gives a competitive ratio of $\frac{2c}{(c+1)(c+2)}$.

Thus far, we have proved the claimed ratio for the special case when the run of the algorithm follows the path along states P, Q, R, S. It turns out that any deviation from this path only improves the competitive ratio, and in fact the tight example for the algorithm follows such a path.

The intuition is that when the algorithm picks up some Blue edges during `PreferBlue` operations on its way from doing `NoPreference` operations to doing `OnlyBlue` operations, the number of hits that can be charged by Blue misses does not increase in proportion to the increase in the number of hits. More concretely, an extra y Blue hits during `PreferBlue` operations creates the need for an extra cy Red hits during the `PreferBlue` operations in order to start doing `OnlyBlue` operations. So we have an extra $(1 + c)y$ hits of which cy cannot be charged by Blue misses, leaving only a $1/c$ fraction of the extra hits chargeable by Blue misses. This is better than what we need to get the final ratio – recall that for state Q, we could only claim that $(c - 1)x$ of the $(c+1)x$ hits upto that state cannot be charged by Blue misses, leaving a $2/(c+1)$ fraction of the hits chargeable by Blue hits.

Tight Example: Figure 4 provides a tight example for c-BALANCE. It follows the path in the special case outlined above.

We postpone the full analysis for the general case as well as a detailed explanation of the tight example to Appendix B. □

COROLLARY 1. *$\sqrt{2}$–Balance achieves a competitive ratio of* $0.3431\ldots$.

2.3 An Upper Bound of 1/2

THEOREM 3. *No deterministic algorithm can achieve a competitive ratio greater than* $1/2$.

It is well known that a simple 2×2 graph yields a hardness result of $1/2$ for the single color case; the idea is to reduce from the single color case, constructing a graph G in the biobjective setting (i.e. a graph with both a Blue and a Red perfect matching) such that if an algorithm can achieve a competitive ratio better than $1/2$ on G then the same algorithm could beat the $1/2$ upper bound for the single color case, thus reaching a contradiction. Unfortunately the standard 2×2 graph cannot be easily turned into a graph with two perfect matchings, so instead we resort to the use of an adaptive version of the standard hard instance for the randomized case (in the single color problem) of the upper-triangular adjacency matrix, for which we argue that a deterministic algorithm cannot get more than $n/2$ edges (details can be found in the full paper).

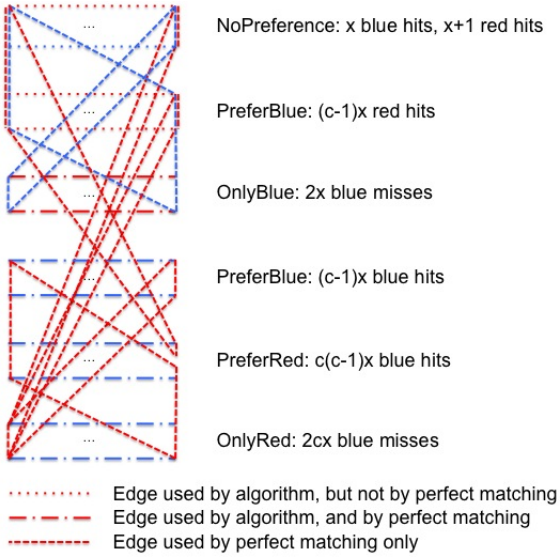

Figure 4: The tight example of Theorem 2.

3. UPPER BOUNDS FOR RANDOMIZED ALGORITHMS

THEOREM 4. *No randomized algorithm can achieve a competitive ratio greater than* $1 - 1/e$.

The main idea is to find a (family of) examples, such that the graph has a Blue as well as a Red perfect matching, with the following property. If we ignored the colors, then (as a single color problem) no randomized algorithm can beat $1 - 1/e$ on this family. This would imply the claim. We modify the standard upper triangular matrix example to achieve this; the proof is provided in Appendix C.

As in the case of deterministic algorithms, we show a limit on the performance of algorithms that always pick an edge if one is available.

THEOREM 5. *No randomized algorithm that never skips can achieve a competitive ratio greater than* $1/2$.

The idea is to modify the example in Figure 1, to get a distribution over two examples, which would give the claim via Yao's lemma (proof in Appendix C).

4. ANALYSIS OF P-PROBGREEDY

THEOREM 6. *p-*PROBGREEDY *achieves a competitive ratio of* $\frac{2p(1-p)}{1+p}$ *with high probability.*[2]

PROOF. Let m_b, m_r, h_b, h_r be the random variables that denote the number of Blue and Red misses and Blue and Red hits respectively (a Blue (resp. Red) hit (resp. miss) is a hit that happens at a node that has selected Blue as its color). We start by noting that

$$m_b + h_b = |\{v \in V \text{ that picked Blue}\}| = pn \pm o(n) \text{ whp} \quad (6)$$

$$m_r + h_r = |\{v \in V \text{ that picked Red}\}| = pn \pm o(n) \text{ whp} \quad (7)$$

where both statements follow from Chernoff bounds and "with high probability" stands for always, except with probability exponentially small in n. From now on we will ignore the lower order term $o(n)$, as it will not affect our analysis.

Next we will relate the number of Red and Blue misses to the total number of hits, through the following charging argument: every time a node $v \in V$ that picked Blue (resp. Red) as its color has a miss, we charge it through a Blue (resp. Red) edge to its neighbor $N_b(v)$ (resp. $N_r(v)$).

We say a hit (u, v) is Blue chargeable if $v \neq N_b(u)$ and v arrives earlier than $N_b(u)$. Now define a collection of random variables $\{X_i\}_{i\in[n]}$ as follows:

$$X_i = \begin{cases} 1-p & \text{there exists the i-th Blue chargeable hit (u,v), where } N_b(u) \text{ chooses Blue;} \\ -p & \text{there exists the i-th blue chargeable hit (u,v), where } N_b(u) \text{ chooses Red or skip;} \\ 0 & \text{otherwise} \end{cases}$$

It is clear that a hit (u, v) will be charged through a Blue edge, only if it is Blue chargeable and $N_b(u)$ chooses Blue. So $m_b \leq B$, where B is a random variable that equals to the size of $\{X_i | X_i > 0\}$.

Now we want to bound B. To do that, we make the following important observation: $\{X_i\}_{i\in[n]}$ is a martingale. The proof is simple

$$\begin{aligned} & \mathbb{E}[X_i | X_1, X_2, \ldots, X_{i-1}] \\ = \; & q(p \cdot (1-p) - (1-p)p) + (1-q) \cdot 0 \\ = \; & 0 \end{aligned}$$

where q is the probability that there exists the i-th Blue chargeable hit conditioned on $X_1, X_2, \ldots, X_{i-1}$. As $|X_i| \leq 1$, by Azuma's inequality, we have

$$\Pr\left[\sum_{i=1}^{n} X_i \geq n^{3/4}\right] \leq \exp(-\frac{n^{1/2}}{2}).$$

If we let the random variable S to be $|\{X_i | X_i \neq 0\}|$, the above inequality means that whp $(1-p)B - p(S-B) \leq n^{3/4}$. So whp $B \leq pS + n^{3/4}$.

Notice that S is always smaller than $h_r + h_b$, and $h_r + h_b \geq pn$ whp. Therefore,

$$m_b \leq B \leq (p + n^{-1/5})(h_r + h_b) \text{ whp.}$$

From now on, we will ignore the $n^{-1/5}$ as it will not affect our competitive ratio. Via an identical argument we also get $m_r \leq p(h_b + h_r)$ whp; combining these two inequalities with (6) and (7) we get that, with high probability, the number of Blue and Red hits must be feasible solutions of the following linear system:

$$\begin{aligned} (p+1)h_b + ph_r &\geq pn \\ ph_b + (p+1)h_r &\geq pn \\ 0 \leq h_b, h_r &\leq pn \end{aligned}$$

From the system above it follows easily that the two feasible solutions which minimize the objective $\min\{h_b, h_r\}$ are $h_b = \frac{p(1-p)}{1+p}n, h_r = pn$ and $h_b = pn, h_r = \frac{p(1-p)}{1+p}n$. Since the optimal offline solution achieves an objective of at most $\frac{n}{2} + o(n)$ (see Proposition 1) it follows that the competitive ratio of *p-*PROBGREEDYis at most $\frac{2p(1-p)}{1+p}$.

□

THEOREM 7. PROBGREEDY *achieves a competitive ratio of* 1/3 *and this is tight.*

PROOF. The fact that the competitive ratio is at least 1/3 follows from the previous theorem. Figure 5 provides an example where the ratio is no more than 1/3. It is explained in more detail in Appendix D. □

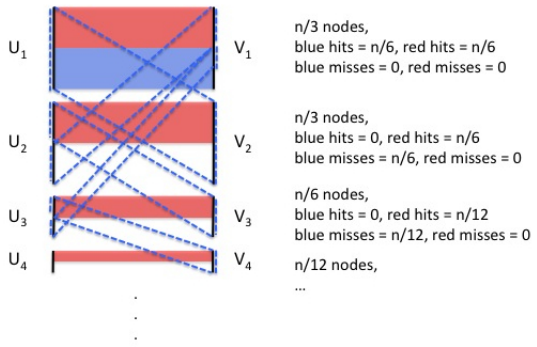

Figure 5: The tight example of Theorem 7.

5. ANALYSIS OF LEFTSUBGRAPHRANKING

In this section we prove the main result of this paper on the performance of LEFTSUBGRAPHRANKING. Recall the definition of the algorithm:

Algorithm LEFTSUBGRAPHRANKING

1. Each vertex in U picks a color uar, and throws away edges of that color.
2. Run Algorithm RANKING on the resulting subgraph.

THEOREM 8. *For any constant* $\epsilon > 0$*, and sufficiently large* n*, the competitive ratio of* LEFTSUBGRAPHRANKING *is at least* $3 - \frac{4}{\sqrt{e}} - 4\epsilon - \frac{10}{\sqrt{n}}$.

[2]Note that, amazingly, this ratio is equal to the ratio attained by c-BALANCE for $p = \frac{1}{1+c}$.

We will begin by showing that for each color, the expected number of edges of that color in the matching produced by the algorithm is at least $(3 - \frac{4}{\sqrt{e}})\frac{n}{2}$. Note that this, by itself, does not imply a competitive ratio of $3 - \frac{4}{\sqrt{e}}$, as the objective fuction is the expectation of the minimum color's edges in the matching, rather that the minimum of the two expectations. However, this is clearly necessary, and we will show how to extend the argument to prove an almost identical ratio for the expectation of the minimum, giving Theorem 8.

5.1 Bounding the Minimum of the two Expectations

To prove a bound on the expected number of edges of a given color, we use a charging argument, which is an augmentation of the charging argument for RANKING (see [1] for one such proof). The probability space of the algorithm consists of all tuples (σ, λ), where σ is a permutation of U and $\lambda \in \{r, b\}^n$ is the vector of colors [3] chosen by vertices in U. In order to describe the charging argument, we make the following definitions.

DEFINITION 1. *For any σ, $u \in U$ and $i \in [n]$, let σ_u^i be the permutation obtained by removing u from σ and inserting it back into σ at position i. Likewise, for any λ, $u \in U$ and $d \in \{r, b\}$, let λ_u^d be such that $\lambda_u^d(u) = d$ and $\lambda_u^d(v) = \lambda(v)$ for all $v \neq u$.*

DEFINITION 2 (**Set of Misses and Hits**). *Define Q_t^d (resp. R_t^d) as the set of all event–position pairs $(\sigma, \lambda), t)$, s.t. the vertex in position t in σ (say u) has $\lambda(u) = d$, and was matched (resp. unmatched) under (σ, λ). Formally:*

$$\begin{aligned} Q_t^d = &\{(\sigma, \lambda, t) : \sigma(u) = t,\ \lambda(u) = d \text{ and} \\ &\quad u \text{ is matched in the event } (\sigma, \lambda)\} \\ R_t^d = &\{(\sigma, \lambda, t) : \sigma(u) = t,\ \lambda(u) = d \text{ and} \\ &\quad u \text{ is unmatched in the event } (\sigma, \lambda)\} \end{aligned}$$

Recall that $N_d(u)$ denotes the partner of u in the perfect matching of color d. We are now ready to define the main charging map ψ from a miss to a set of hits.

Consider a triple (σ, λ, t) s.t. that vertex u at position t in σ is unmatched and let $d = \lambda(u)$. Then, we define the d-ChargingMap of the triple (σ, λ, t) as the set of all triples (σ', λ', s), such that $\sigma' = \sigma_u^i$ for some $i \in [n]$ and $\lambda' = \lambda_u^d$ for some $d \in \{r, b\}$, and s is the rank of the vertex to which $N_d(u)$ is matched in the event (σ', λ').

DEFINITION 3 (**Charging Map**). *For every $(\sigma, \lambda, t) \in R_t^d$, define the map*

$$\begin{aligned} \psi_d(\sigma, \lambda, t) = \{&(\sigma_u^i, \lambda_u^{d'}, s) : 1 \le i \le n,\ d' \in \{r, b\}, \\ &\sigma(u) = t \text{ and } N_d(u) \text{ is matched to } u' \\ &\text{with } \sigma_u^i(u') = s \text{ in the event } (\sigma_u^i, \lambda_u^{d'})\} \end{aligned}$$

The following lemma proves that the above charging maps are well-defined (i.e. s always exists) and that $s \le t$ for all such mappings. This essentially follows the standard alternating path argument for RANKING (see e.g. [1]), except that we also need to argue that the same holds even when u flips its color and throws away a different set of edges.

LEMMA 1. *For any ranking σ and coloring λ, if vertex u has rank t in σ and is left unmatched in the event (σ, λ), then $N_{\lambda(u)}(u)$ is matched to some node u' in the event $(\sigma_u^i, \lambda_u^b)$ and some node u'' in the event $(\sigma_u^i, \lambda_u^r)$, for any $1 \le i \le n$. Moreover, $\sigma_u^i(u') \le t$ and $\sigma_u^i(u'') \le t$.*

The proof can be found in Appendix E. We next show that for a fixed t, the set-values $\psi_d(\sigma, \lambda, t)$ are disjoint for different σ or λ.

CLAIM 1. *If $(\sigma, \lambda, s) \in \psi_d(\sigma', \lambda', t)$ and $(\sigma, \lambda, s) \in \psi_d(\sigma'', \lambda'', t)$, then $\sigma' = \sigma''$ and $\lambda' = \lambda''$.*

PROOF. WLOG, we can assume $d = r$. It is clear that except for the node at rank t the two orderings and colorings are the same. Let $u' \in U$ be the vertex with $\sigma(u') = s$, and let $\bar{u} \in V$ be the vertex to which u' is matched in the event (σ, λ). Finally let $u \in U$ be the vertex which $\bar{u}$ is matched to in the perfect red matching. Since we are using the charging map $\psi_r(\cdot)$, it follows that $\lambda(u) = \lambda'(u) = r$ and $\sigma(u) = \sigma'(u) = t$. □

We can now prove the bound on the expectation of each color.

LEMMA 2. *Both the number of red matches and the number of blue matches are at least $(\frac{3}{2} - \frac{2}{\sqrt{e}})n$.*

PROOF. We will show the lemma for the number of red matches, as the proof for the blue matches is identical. Let x_t be the probability that the node in position t has a red match ($x_t = \frac{|Q_t^r|}{2^n n!}$).

By Claim 1, we have:

$$\begin{aligned} &\frac{1}{2} - x_t \le \frac{1}{2n} \sum_{s \le t} (x_s + y_s) \\ \implies\ &\frac{1}{2} - x_t \le \frac{1}{2n} \sum_{s \le t} (x_s + \frac{1}{2}). \end{aligned}$$

Letting $z_s = x_s + 1/2$, we can rewrite the above formula as

$$1 - z_t \le \frac{1}{2n} \sum_{s \le t} z_s$$

Therefore, we get

$$\sum_{t \le n} z_t \ge (2 - \frac{2}{\sqrt{e}})n.$$

So

$$\sum_{t \le n} x_t \ge (\frac{3}{2} - \frac{2}{\sqrt{e}})n \approx 0.2869n$$

□

[3] Through this section, we will use r and b to denote color Red and Blue respectively.

5.2 Bounding the Expectation of the Minimum Color

In order to bound the expectation of the number of matches of the minimum color, we need to consider two parts of the probability space – the events for which Red is the trailing color (where we need to count the number of Red matches) and the events for which Blue is the trailing color (where we need to count the number of Blue matches), breaking ties arbitrarily. However in doing so, a new difficulty arises – when we were bounding the expected number of matches of one color, we were guaranteed that the charging maps for all misses of that color are disjoint. However, this is no longer true when we look at the charging maps for Red misses for some of the events and the charging map for Blue misses for the remaining events – the same hit might appear in the charging map for some Blue miss and for some Red miss. Naively allowing double charging will lead to a large loss in competitive ratio, giving a ratio of $1 - 2/e \simeq 0.26$.

Here we make the second key observation of the analysis that lets us show essentially the same bound for expectation of the minimum color, with a loss of only lower order terms. Let $\Delta(\sigma, \lambda)$ be the difference between the number of Red matches and Blue matches in the matching produced by the algorithm for the event (σ, λ). We observe that our charging map is "local", i.e. misses in an event (σ, λ) map to hits in events whose Δ value is within 2 of $\Delta(\sigma, \lambda)$, as shown in Lemma 3. If we partition the events by their Δ value, we want to count the number of Blue misses for events whose Δ value is positive and the number of Red misses for the rest. By the lemma, we know that Blue misses of interest charge to events whose Δ value is at least -1, while the Red misses of interest charge to events whose Δ value is at most 2. So the only events whose hits get double-charged are those whose Δ value is in $\{-1, 0, 1, 2\}$. If we could show that the number of hits in such events is small, we will be done. Unfortunately, we do not know how to show this. What we can show is that there is some $k \in [1, 4\sqrt{n}]$ s.t. the probability mass of hits in events with Δ value in $\{k, k+1, k+2, k+3\}$ is no more than $\frac{1}{\sqrt{n}}$, as shown in Fact 1. So if we look at the charges from Blue misses in events with Δ value greater than $k+1$, and the charges from Red misses from the remaining events, we will double charge only a small number of the hits, thereby giving us the required bound.

The full proof can be found in Appendix F.

Tight example: Interestingly, our analysis is tight. Consider the following family of graphs. Let the upper triangular entries (without the diagonal) and the bottom left entry of the adjacency matrix filled with Red edges, and the rest filled with Blue edges. This guarantees two perfect matchings. But running LeftSubgraphRanking on this graph gives no more than $(\frac{3}{2} - \frac{2}{\sqrt{e}})n$ red matches (Figure 6 provides an illustration; details can be found in the full paper).

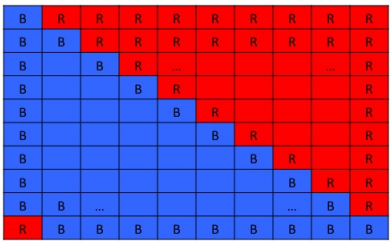

Figure 6: A tight example for LeftSubgraphRanking.

6. REFERENCES

[1] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. *SODA*, 2010.

[2] N. Bhatnagar, D. Randall, V.V. Vazirani, and E. Vigoda. Random bichromatic matchings. *Algorithmica*, 50(4):418–445, 2008.

[3] N. Buchbinder, K. Jain, and J.S. Naor. Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue. In *ESA*, 2007.

[4] Constantinos Daskalakis and George Pierrakos. Simple, optimal and efficient auctions. In *WINE*, 2011.

[5] Ilias Diakonikolas, Christos H. Papadimitriou, George Pierrakos, and Yaron Singer. Efficiency-revenue trade-offs in auctions. In *ICALP (2)*, pages 488–499, 2012.

[6] Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *WINE*, 2009.

[7] M. Flammini and G. Nicosia. On multicriteria online problems. *Algorithms-ESA 2000*, 2000.

[8] Chinmay Karande, Aranyak Mehta, and Ramakrishnan Srikant. Optimizing budget constrained spend in search advertising. In *WSDM, 2013 (to appear)*.

[9] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for online bipartite matching. In *STOC*, 1990.

[10] AV Karzanov. Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics and Systems Analysis*, 1987.

[11] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. In *FOCS*, 2005.

[12] K. Mulmuley, U.V. Vazirani, and V.V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1987.

[13] Silvio Micali Pablo Azar, Constantinos Daskalakis and Matt Weinberg. Optimal and efficient parametric auctions. In *SODA*, 2013.

[14] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, 2000.

[15] T. Yi, K.G. Murty, and C. Spera. Matchings in colored bipartite networks. *Discrete Applied Mathematics*, 121(1):261–277, 2002.