Algorithmica (2012) 62:333–348
DOI 10.1007/s00453-010-9457-2

Competitive Weighted Matching in Transversal Matroids

Nedialko B. Dimitrov · C. Greg Plaxton

Received: 1 March 2010 / Accepted: 27 September 2010 / Published online: 21 October 2010

Abstract Consider a bipartite graph with a set of left-vertices and a set of right-vertices. All the edges adjacent to the same left-vertex have the same weight. We present an algorithm that, given the set of right-vertices and the number of left-vertices, processes a uniformly random permutation of the left-vertices, one left-vertex at a time. In processing a particular left-vertex, the algorithm either permanently matches the left-vertex to a thus-far unmatched right-vertex, or decides never to match the left-vertex. The weight of the matching returned by our algorithm is within a constant factor of that of a maximum weight matching, generalizing the recent results of Babaioff et al.

Keywords Matching · Online algorithm · Secretary problem · Transversal matroid

1 Introduction

Motivated by applications related to auctions, mechanism design, and revenue management, Babaioff et al. recently introduced a generalization of the secretary problem called the matroid secretary problem [1]. In the matroid secretary problem, the goal is to build a maximum weight independent set, but we are constrained from knowing the full input to the problem. Instead, the matroid elements are revealed one at a time,

N.B. Dimitrov is supported by an MCD Fellowship from the University of Texas at Austin.
C.G. Plaxton is supported by NSF Grants CCF–0635203 and ANI–0326001.

N.B. Dimitrov (✉)
Operations Research Department, Naval Postgraduate School, 1411 Cunningham Rd., Monterey, CA 93943, USA
e-mail: ned@alumni.cs.utexas.edu

C.G. Plaxton
Computer Science Department, The University of Texas at Austin, 1616 Guadalupe St., Austin, TX 78701, USA
e-mail: plaxton@cs.utexas.edu

and we must immediately decide whether to include the revealed element in the independent set. In such a setting, an online algorithm is said to be *c-competitive* if it is able to produce an independent set with weight within a factor of c of the weight of a maximum weight independent set [3]. We say that an online algorithm is *competitive* if it is c-competitive for some constant c.

Babaioff et al. present competitive algorithms for the matroid secretary problem on bounded left-degree transversal matroids and graphic matroids. They also present a reduction showing that if we have a competitive algorithm for a matroid M, then we can construct a competitive algorithm for a truncated version of M. Babaioff et al. leave open the general matroid secretary problem and the central case of transversal matroids. As discussed later in this section, the case of transversal matroids unifies the existing results on the matroid secretary problem. In this paper we present a competitive online algorithm for weighted matching in transversal matroids, generalizing the results of Babaioff et al. Along with the reduction in Babaioff et al., our results also lead to competitive algorithms on truncated transversal matroids.

Informally, the online weighted transversal matroid matching problem can be described as follows. Consider a bipartite graph, with a set of m left-vertices and a set of n right-vertices. All edges adjacent to the same left-vertex have the same weight—we associate this weight with the left-vertex. The weighted transversal matroid matching problem (WTMM) asks us to find a maximum weight matching in this bipartite graph, and is solvable with the standard matroid greedy algorithm. In the *online* weighted transversal matroid matching problem (OWTMM), we are initially given only the total number of left-vertices, and then a uniformly random permutation of the left-vertices is revealed, one left-vertex at a time. When a vertex is revealed, we learn of both its weight and its incident edges. Upon seeing a particular left-vertex, without knowing the details of the remaining unrevealed left-vertices, we must immediately decide which right-vertex to match it to, if any. An unweighted version of this problem, where we simply try to maximize the cardinality of the final matching, was introduced by Karp et al. [7]. Karp et al. provide an algorithm that is $(1 - \frac{1}{e})$-competitive for the unweighted problem; the proof was recently simplified by Barinbaum and Mathieu [2, 7]. An open problem left by Babaioff et al. is to find an algorithm for OWTMM returning a matching with expected weight within a constant of the optimal matching in the corresponding WTMM problem. Theorem 7.3 presents such an algorithm.

In the literature, a transversal matroid is often specified by a set of elements E, and a set of subsets $A_1, \dots, A_n$ of E [10]. A subset $I = \{a_1, \dots, a_k\}$ of E is considered independent if there is an injective function f mapping I to $\{A_1, \dots, A_n\}$ such that $x \in f(x)$ for all inputs x. In our presentation, the set of elements E corresponds to the left-vertices, the sets $A_1, \dots, A_n$ correspond to the right-vertices, and there is an edge between an element of E and a set A_j if the element belongs to the set. An independent set then corresponds to a set of left-vertices for which there exists a matching to the right-vertices.

Perhaps the most well studied matroid secretary problem is the secretary problem, which first appeared as a folklore problem in the 1950s and has a long history [4, 5]. The problem was first solved by Lindley, who also presents a competitive algorithm for the secretary problem [11]. Competitive algorithms also exist for uniform matroids [8], bounded left-degree transversal matroids, graphic matroids, and truncated matroids [1]. For general matroids, the best known competitive ratio is $O(\log r)$ where r is the rank of the matroid [1].

With the exception of truncated matroids, where the result depends on Karger's matroid sampling theorem [6], all of the matroids for which a competitive algorithm is known are a special case of the transversal matroid. For example, the secretary problem is a transversal matroid with a single right-vertex. The uniform matroid of rank r is a transversal matroid on a complete bipartite graph with r right-vertices. Of course, bounded left-degree transversal matroids are a special case of the transversal matroids. And, finally, the competitive results for graphic matroids follow from a mapping to bounded left-degree transversal matroids, along with an algorithmic modification [1]. Thus, indeed, transversal matroids play a central role to the theory. For some remarks on the strong connection between general matroids and transversal matroids, see Sect. 8.

1.1 Algorithm Motivations

Recall that the secretary problem is OWTMM with a single right-vertex and consider the following classic algorithm for the secretary problem. We sample the first $m/2$ left-vertices we see, rejecting all of them, but recording their edge weights. We set a price for the right-vertex equal to the maximum weight edge we see in the sample. We then match the right-vertex with the first non-sampled left-vertex whose edge weight exceeds the price, if we see such a left-vertex. The algorithm is competitive since with probability at least $1/4$, the second heaviest edge is sampled and the heaviest edge is not sampled.

This simple sample-and-price algorithm is the motivation for most of the competitive algorithms known for matroid secretary problems, again with the exception of truncated matroids. However, extending this algorithm to work for all, general transversal matroids is not straightforward. For example, Babaioff et al. show that a sample-and-price algorithm with an adaptive sampling time which sets the same price for all the right-vertices does not work. Babaioff et al. also show that a more complicated scheme, where the price required of a non-sampled left-vertex is determined by a circuit of sampled left-vertices also does not work.

One of the main issues that arises in trying to generalize the sample-and-price algorithm is a tension between the need to use sampled heavy left-vertices to price the right-vertices and the requirement that we not over-price too many right-vertices. Consider the example in Fig. 1a. If in the sample we see the left-vertex of weight 2, we should not over-price all the right-vertices at 2, since that prevents us from matching a large number of vertices of weight 1. The figure is only meant as an illustration, but can be extended to a counter-example for a heavy pricing method by adding $\log m$ clones of the left-vertex of weight 2. On the other hand, consider the example in Fig. 1b. If we do not set a price of 2 for the bottom-most right-vertex, we would prematurely match that right-vertex to a left-vertex of weight 1 instead of the infinite weight left-vertex. It is natural to consider more complex pricing schemes, such as dynamic prices that change throughout processing, or picking a random subset of the neighbors of a heavy left-vertex and pricing only those neighbors. However, it is both unclear if such schemes are effective and it is difficult to analyze them as they

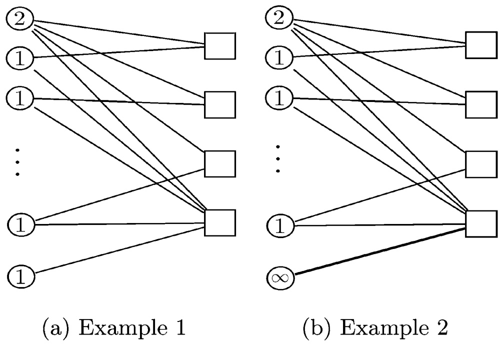

Fig. 1 Two example transversal matroids exhibiting the tension between using sampled heavy left-vertices for pricing and over-pricing the right-vertices. The figures are meant only to be illustrative, but can be extended to become counter-examples for certain pricing strategies. In **a**, we do not want to price all the right-vertices at 2, since we would miss many left-vertices of weight 1. In **b**, we want to price the bottom right-vertex at 2, since otherwise we would miss the infinite weight left-vertex

often introduce complicated probabilistic dependencies. It is this tension that leads Babaioff et al. to consider bounded left-degree transversal matroids.

For our results, we avoid the difficulties arising from more complex schemes with the concept of "candidate edges." The candidate edges we introduce have the following important properties. First, each left-vertex i has at most one candidate edge, uniquely determined by the sampled left-vertices heavier than i. In other words, given the sampled left-vertices heavier than i, the candidate edge is the same regardless of whether i is sampled, or where in the random order of non-sampled vertices it appears. Second, the candidate edges of the sampled left-vertices constitute a matching that is within a constant-factor of the max-weight matching on the sampled subgraph.

The analysis following from our definition of candidate edges is essentially the original sample-and-price analysis from the secretary problem, but applied to each right-vertex separately. The algorithm prices right-vertices using only the candidate edges. Furthermore, a non-sampled left-vertex can only be matched using its candidate edge. For a particular right-vertex, as in the secretary problem, we hope that the second-heaviest left-vertex with a candidate edge to the right-vertex is sampled, but the heaviest left-vertex with a candidate edge to the right vertex not sampled. Similarly to the secretary problem, this happens with at least $1/4$ probability.

The overall argument structure is as follows. In Sect. 2, we define some useful notation. In Sect. 3, we define candidate edges and show that they constitute a matching with weight within a constant factor of optimal on the sampled subgraph. In Sect. 4, to avoid any confusion from probabilistic dependencies, we analyze sampled and non-sampled matchings through counting arguments. Our counting arguments immediately imply that a matching resulting from candidate edges of non-sampled left-vertices has expected weight within a constant factor of the expected weight of the matching of candidate edges of sampled left-vertices. In Sect. 5, we show that the expected weight of the sampled candidate edge matching is within a constant factor of the max-weight matching on the entire graph. This completes the main technical arguments, since the non-sampled matching is within a constant factor of the sampled matching, which is within a constant of the optimal matching on the whole graph. In Sect. 6, we present a small but clarifying intermediate algorithm between the final online algorithm and the counting arguments presented earlier. Finally, in Sect. 7, we present the online algorithm and conclude the analysis.

2 Definitions

In this section, we formally define some quantities and notation we will use throughout the paper .

Fix a set of n right-vertices, numbered 0 to $n-1$.

Fix a set of m left-vertices, where each left-vertex i is described by a triple of (1) a real number weight, $w(i)$ (2) a unique integer ID and (3) a subset of the right vertices, $\mathrm{Right}(i)$. We define a total order on the left-vertices: we say a left-vertex i is less than a left-vertex i' if $w(i) > w(i')$ or $w(i) = w(i')$ and i has a smaller unique integer ID. We draw the reader's attention to the fact that smaller left-vertices have greater weight. From here on, we use the integers to denote the left-vertices, with 0 denoting the minimum left-vertex, 1 denoting the second minimum left-vertex and so forth. We draw the reader's attention to the fact that the ordering on the left-vertices is the same as the ordering on the corresponding integers.

For a nonempty subset A of left-vertices or right-vertices let $\mathrm{Min}(A)$ return the minimum vertex, as defined by the corresponding total order.

An edge is a pair (i, j), where i is a left-vertex and j belongs to $\mathrm{Right}(i)$. A matching is a set of edges M such that each vertex appears in at most one edge. For a matching M, let $\mathrm{Left}(M)$, $\mathrm{Right}(M)$, denote the left-vertices, right-vertices, in the matching, respectively.

For a set of left-vertices, A, we say $w(A) = \sum_{i\in A} w(i)$. For a matching M, we say $w(M) = w(\mathrm{Left}(M))$.

To facilitate our proofs, we define the following notation. For a subset of left-vertices L, let $\mathrm{Prefix}(L, i) = \{i' \in L \mid i' < i\}$. Similarly, for a matching M, let $\mathrm{Prefix}(M, i) = \{(i', j) \in M \mid i' < i\}$.

3 Candidate Edges and Their Properties

In this section we define candidate edges and show the two main properties discussed in Sect. 1.1. The first property, "each left-vertex i has exactly zero or one candidate edges, uniquely determined by the sampled left-vertices heavier than i" corresponds to Lemmas 3.1 and 3.2. The second property, "the candidate edges of the sampled left-vertices constitute a matching that is within a constant-factor of the max-weight matching on the sampled subgraph" corresponds to Lemma 3.5.

First, we define a function $\mathrm{Cands}(i, M)$ that receives a left-vertex i and a matching M, and returns an edge set. The $\mathrm{Cands}(i, M)$ function is as follows:

```
$M' := \mathrm{Prefix}(M, i)$
$A := \mathrm{Right}(i) - \mathrm{Right}(M')$
**if** $A = \emptyset$
    **return** $\emptyset$
**else**
    **return** $\{(i, \mathrm{Min}(A))\}$
```

Lemma 3.1 *For any left-vertex i and matching M*, $\mathrm{Cands}(i, M)$ *either returns the empty set*, *or* $\{(i, j)\}$, *where j is a right-vertex unmatched in* $\mathrm{Prefix}(M, i)$.

Proof Follows from the definition of Cands. □

Lemma 3.2 *For any left-vertex i and matchings M and M' with* $\mathrm{Prefix}(M, i) = \mathrm{Prefix}(M', i)$, *we have* $\mathrm{Cands}(i, M) = \mathrm{Cands}(i, M')$.

Proof Follows from the definition of Cands. □

We now define an algorithm for WTMM. Algorithm $\mathrm{AlgA}(L)$ takes a subset of left-vertices L and returns a matching and the algorithm is performed as follows:

```
$M := \emptyset$
**for** $i$ in increasing order in $L$
    $M := M \cup \mathrm{Cands}(i, M)$
**return** $M$
```

Recall that the total order on left-vertices is defined such that i is less than i' if $w(i) > w(i')$ or $w(i) = w(i')$ and i has a smaller unique integer ID.

Lemma 3.3 *For a subset of left-vertices L, let* $\tilde{M} = \mathrm{AlgA}(L)$, *then $\tilde{M}$ is a matching on L and* $\tilde{M} = \bigcup_{k\in L} \mathrm{Cands}(k, \tilde{M})$.

Proof We prove the lemma by first proving the following loop invariant in $\mathrm{AlgA}(L)$: M is a matching on $\mathrm{Prefix}(L, i)$ and $M = \bigcup_{k\in \mathrm{Prefix}(L,i)} \mathrm{Cands}(k, M)$.

The claimed invariant hold initially since $M := \emptyset$ and $i = \mathrm{Min}(L)$. Suppose the claim is true for M and i on entering the loop on which we process i. Let $M' = M \cup \mathrm{Cands}(i, M)$ and i' be the next left-vertex in order from L. We must show the claim holds for M' and i'.

Let $A = \mathrm{Cands}(i, M)$. We split the analysis in two cases. First, suppose $A = \emptyset$. Then, $M' = M$ and the claim holds for M' and i' simply because it holds for M and i.

Second, suppose $A = \{(i, j)\}$, for a right-vertex j unmatched in $\mathrm{Prefix}(M, i)$ (Lemma 3.1). Since $\mathrm{Prefix}(L, i') = \mathrm{Prefix}(L, i) \cup \{i\}$, the first part of the invariant holds.

For the second part of the invariant, we have

$$\begin{aligned} M' &= M \cup \mathrm{Cands}(i, M) \\ &= \bigcup_{k\in \mathrm{Prefix}(L,i)} \mathrm{Cands}(k, M) \cup \mathrm{Cands}(i, M) \\ &= \bigcup_{k\in \mathrm{Prefix}(L,i')} \mathrm{Cands}(k, M) \\ &= \bigcup_{k\in \mathrm{Prefix}(L,i')} \mathrm{Cands}(k, M') \end{aligned}$$

The second equality holds because the loop invariant holds for M and i, the third equality holds by the definition of i', and the final equality holds by Lemma 3.2. This proves the invariant.

The lemma statement follows from following the same reasoning as in the inductive step above, but taken for the final iteration of the loop. □

Lemma 3.4 *Let M^* be a max-weight matching on L and M be the matching returned by* AlgA(L). *If $(i, j) \in M^*$ and i is unmatched in M, then there is a i' such that $(i', j) \in M$ and $w(i') \geq w(i)$.*

Proof By Lemma 3.3, $M = \bigcup_{k\in L} \mathrm{Cands}(k, M)$. Since i is not matched in M and by Lemma 3.1, we have $\emptyset = \mathrm{Cands}(i, M)$. By the definition of Cands, the empty set can only be returned if $\mathrm{Right}(i) \subseteq \mathrm{Right}(\mathrm{Prefix}(M, i))$. In other words, every right-vertex in $\mathrm{Right}(i)$ is matched to a left-vertex less than i in M, completing the proof. □

Lemma 3.5 *Let M^* be a max-weight matching on L, and M be the matching returned by* AlgA(L). *Then $w(M) \geq \frac{1}{2}w(M^*)$.*

Proof By summing the inequality in Lemma 3.4 over left-vertices matched in $M^* - M$, we have $w(M) \geq w(M^* - M)$. By definition of intersection, we have $w(M) \geq w(M^* \cap M)$. Combining the two inequalities, we have $2w(M) \geq w(M^*)$. □

4 Counting Arguments

In this section, to avoid confusion with probabilistic dependencies, we analyze sampled and non-sampled matchings through counting arguments. As stated in Sect. 1.1, our counting arguments, in particular Lemma 4.9, immediately imply that a matching resulting from candidate edges of non-sampled left-vertices has expected weight at least $1/4$ of the expected weight of the matching of candidate edges of sampled left-vertices. From this section we only export Lemma 4.1, which is used to connect the counting arguments with the final online algorithm, and Lemma 4.9.

Let α be a binary string and α_i be the ith character in the string. Intuitively, the reader should think of a 0 in the ith position of α as sampling the left-vertex i and of a 1 in the ith position as not sampling i. For two binary strings α and β, let $\alpha\beta$ denote concatenation. For a binary string α of length at most m, we define the sets of edges $M_0(\alpha)$, $M_2(\alpha)$, $E_0(\alpha)$ recursively as follows.

$$M_0(\epsilon) = E_0(\epsilon) = \emptyset$$

$$M_2(\alpha) = \mathrm{Cands}(|\alpha|, M_0(\alpha))$$

$$M_0(\alpha 0) = M_0(\alpha) \cup M_2(\alpha)$$

$$M_0(\alpha 1) = M_0(\alpha)$$

$$E_0(\alpha 0) = E_0(\alpha)$$

$$E_0(\alpha 1) = E_0(\alpha) \cup M_2(\alpha)$$

Finally, we also define $E_1(\alpha) = M_0(\alpha) \cup E_0(\alpha)$ and $M_1(\alpha)$ to be $\{(i,j) \in E_0(\alpha) \mid j \text{ appears at most once in } E_0(\alpha)\}$. It is not difficult to show that $M_0(\alpha)$, $M_1(\alpha)$ and $M_2(\alpha)$ are matchings while $E_0(\alpha)$ and $E_1(\alpha)$ are sets of edges.

We give the reader a loose intuitive interpretation of these definitions. Intuitively, one can think of processing the left-vertices in order of increasing weight as we increase the length of α. Then, $M_2(\alpha)$ represents the $|\alpha|$th candidate edge; $M_0(\alpha)$ represents a matching created from the sampled left-vertices; $E_0(\alpha)$ represents a set of edges created from the non-sampled left-vertices such that each non-sampled left-vertex appears at most once; $E_1(\alpha)$ represents a set of all candidate edges, regardless of whether the corresponding left-vertex is sampled; and $M_1(\alpha)$ represents a matching created from the non-sampled left-vertices.

Lemma 4.1 *For a binary string α of length at most m, let $A = \{i \mid \alpha_i = 0\}$ and $B = \{i \mid \alpha_i = 1\}$. We have, $M_0(\alpha) = \bigcup_{i\in A} \mathrm{Cands}(i, M_0(\alpha))$ and $E_0(\alpha) = \bigcup_{i\in B} \mathrm{Cands}(i, M_0(\alpha))$.*

Proof We prove the claim by induction on the length of α. For $\alpha = \epsilon$, the claim follows from the definition of $M_0(\epsilon)$ and $E_0(\epsilon)$. The inductive claim follows from Lemma 3.2 and the recursive definitions of $M_0(\alpha)$ and $E_0(\alpha)$. □

For a set of edges A, let $\deg(A, j)$ denote the degree of the right-vertex j in A. For a left-vertex i and a right-vertex j, we partition the set of binary strings to assist in our counting arguments as follows.

$$\begin{array}{ll}
\alpha \in S_0(i,j) & \text{if } |\alpha| < i, \deg(E_1(\alpha), j) = 0 \\
\alpha \in S_1(i,j) & \text{if } |\alpha| = i, \deg(E_1(\alpha), j) = 0, M_2(\alpha) = \{(i,j)\} \\
\alpha \in S_2(i,j) & \text{if } \deg(E_0(\alpha), j) = \deg(E_1(\alpha), j) = 1, \alpha = \beta\gamma, \beta \in S_1(i,j) \\
\alpha \in S_3(i,j) & \text{if } \deg(E_0(\alpha), j) = 1, \deg(E_1(\alpha), j) > 1, \alpha = \beta\gamma, \beta \in S_2(i,j) \\
\alpha \in S_4(i,j) & \text{otherwise.}
\end{array}$$

We give the reader some intuitive interpretation of the above sets. For a particular pair (i,j): $S_0(i,j)$ represents strings where j has never been returned by Cands and we have not yet reached i; $S_1(i,j)$ represents strings where Cands has never before returned j, we have just now reached i and Cands returns $\{(i,j)\}$; $S_2(i,j)$ represents strings where j has been returned exactly once by Cands, when j was returned by Cands it was along with i and i was non-sampled; $S_3(i,j)$ represents strings where the first time Cands returned j it was along with i and i was non-sampled, then j was returned again with some other, sampled vertex i'; finally, $S_4(i,j)$ represents all other strings.

Lemma 4.2

$$\begin{array}{rcl}
\alpha \in S_1(i,j) & \Rightarrow & \alpha 0 \in S_4(i,j), \alpha 1 \in S_2(i,j) \\
\alpha \in S_2(i,j), \quad \exists i' M_2(\alpha) = \{(i',j)\} & \Rightarrow & \alpha 0 \in S_3(i,j), \alpha 1 \in S_4(i,j)
\end{array}$$

$$\begin{aligned} \alpha \in S_2(i,j), \quad \forall i' M_2(\alpha) \neq \{(i',j)\} \quad &\Rightarrow \quad \alpha 0, \alpha 1 \in S_2(i,j) \\ \alpha \in S_3(i,j) \quad &\Rightarrow \quad \alpha 0, \alpha 1 \in S_3(i,j) \\ \alpha \in S_4(i,j) \quad &\Rightarrow \quad \alpha 0, \alpha 1 \in S_4(i,j) \end{aligned}$$

Proof The statement follows straightforwardly by case analysis from the recursive definitions of $M_0(\alpha)$, $E_0(\alpha)$, $E_1(\alpha)$ and the partition $S_0(i,j)$, $S_1(i,j)$, $S_2(i,j)$, $S_3(i,j)$, $S_4(i,j)$. We provide some intuitive discussion.

For the first implication, appending 0 to α places $\{(i,j)\}$ in M_0, which places $\alpha 0$ in S_4. On the other hand, appending 1 to α places $\{(i,j)\}$ in E_0, which places $\alpha 1$ in S_2.

For the second implication, appending 0 to α places $\{(i',j)\}$ in M_0, which places $\alpha 0$ in S_3. On the other hand, appending 1 to α places $\{(i',j)\}$ in E_0, which places $\alpha 1$ in S_4.

For the third implication, neither appending 0 nor 1 to α increases the degree of j in E_1, so both extensions of α are in S_2.

For the fourth implication, since α is in S_3, we know that j is in M_0. This means that j can no longer be returned by Cands (Lemma 3.1). Thus regardless of the appended character, the extension of α is in S_3.

For the fifth implication, since appending a character to α does not decrease the length of the string or decrease the degree of j in E_0 or E_1, both extensions of α are in S_4. □

Lemma 4.3 *For any right-vertex j, left-vertex i and integer k such that $i < k \leq m$, we have*

$$|S_2(i,j) \cap \{0,1\}^k| + 2|S_3(i,j) \cap \{0,1\}^k| = 2^{k-i-1}|S_1(i,j)|.$$

Proof We prove the lemma by induction on k with a base case $k = i + 1$.

For the base case, we have $|S_2(i,j) \cap \{0,1\}^{i+1}| \geq |S_1(i,j)|$ since for every $\alpha \in S_1(i,j)$ we have that $|\alpha| = i$ by the definition of $S_1(i,j)$ and by Lemma 4.2 we have $\alpha 1 \in S_2(i,j)$. By the definition of $S_2(i,j)$, all strings in $S_2(i,j)$ have length at least $i+1$. Also by the definition of $S_2(i,j)$, all strings of length $i+1$ in $S_2(i,j)$ are equal to $\alpha 1$ for some string in $S_1(i,j)$. Thus, we have $|S_2(i,j) \cap \{0,1\}^{i+1}| \leq |S_1(i,j)|$. By the definition of $S_3(i,j)$, all strings in $S_3(i,j)$ have length at least $i+2$. Thus, $|S_3(i,j) \cap \{0,1\}^{i+1}| = 0$, completing the base case.

To show the inductive step, notice that the right-hand side of the claimed equality exactly doubles as we increase k by one. We must show that the left-hand side of the equality also exactly doubles. By Lemma 4.2, for every $\alpha \in S_2(i,j)$, either $\alpha 0 \in S_3(i,j)$ or both $\alpha 0, \alpha 1 \in S_2(i,j)$. In either case, the count from the first summand of the left-hand side of the equality doubles when we increase k by one. Again, by Lemma 4.2, for every $\alpha \in S_3(i,j)$, we have $\alpha 0, \alpha 1 \in S_3(i,j)$. So, the count from the second summand of the left-hand side of the equality also doubles. So the left-hand side at least doubles when we increase k by one.

By Lemma 4.2, the only ways for a string extended by one character to be in $S_2(i,j)$ or $S_3(i,j)$ is by extending a string in $S_1(i,j)$, $S_2(i,j)$ or $S_3(i,j)$. In the previous paragraph, we accounted for extensions from strings in $S_2(i, j)$ or $S_3(i, j)$. All strings in $S_1(i, j)$ have length i, but in the inductive case $k > i + 1$. Thus, the left-hand side of the claimed equality exactly doubles. □

Lemma 4.4 *For any right-vertex j, left-vertex i and integer k such that $i < k \leq m$, we have*

$$|(S_2(i, j) \cup S_3(i, j)) \cap \{0, 1\}^k| \geq 2^{k-i-2}|S_1(i, j)|$$

Proof By Lemma 4.3, we have that $|S_2(i, j) \cap \{0, 1\}^k| + 2|S_3(i, j) \cap \{0, 1\}^k| = 2^{k-i-1}|S_1(i, j)|$. We can increase the left-hand side to get $2|S_2(i, j) \cap \{0, 1\}^k| + 2|S_3(i, j) \cap \{0, 1\}^k| \geq 2^{k-i-1}|S_1(i, j)|$. Since $S_2(i, j)$ and $S_3(i, j)$ are disjoint, we have $|(S_2(i, j) \cup S_3(i, j)) \cap \{0, 1\}^k| \geq 2^{k-i-2}|S_1(i, j)|$. □

Lemma 4.5 *For any right-vertex j and integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_1(\alpha), j) \geq \sum_{0 \leq i < k} w(i) 2^{k-i-2}|S_1(i, j)|$$

where $w(M_1(\alpha), j)$ denotes the weight of the left-vertex matched to j in $M_1(\alpha)$ or zero if j is unmatched.

Proof By the definitions of M_1, S_2 and S_3, we have that $(i, j) \in M_1(\alpha)$ if and only if $\alpha \in (S_2(i, j) \cup S_3(i, j))$. Furthermore, by the definition of M_1, if α has length k, then only left-vertices less than k can be matched in M_1. The left-hand side of the claimed inequality is thus equal to $\sum_{0 \leq i < k} w(i)|(S_2(i, j) \cup S_3(i, j)) \cap \{0, 1\}^k|$. Applying Lemma 4.4 gives the desired result. □

Lemma 4.6 *For any integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_1(\alpha)) \geq \sum_{0 \leq i < k} \sum_{0 \leq j < n} w(i) 2^{k-i-2}|S_1(i, j)|$$

Proof Follows from summing the result of Lemma 4.5 over all $0 \leq j < n$. □

Lemma 4.7 *For any right-vertex j and integer k such that $k \leq m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha), j) \leq \sum_{0 \leq i < k} w(i) 2^{k-i}|S_1(i, j)|$$

where $w(M_0(\alpha), j)$ is equal to the weight of the left-vertex matched to j in $M_0(\alpha)$ or zero if j is unmatched.

Proof To prove this lemma, we first introduce some helpful claims and definitions. For any binary string α of length at most k define $f(\alpha)$ as the set of proper prefixes β of α such that $M_2(\beta) = \{(|\beta|, j)\}$. The following two claims follow directly from the definitions.

Claim 1: For any binary string α of length at most k, we have $\deg(E_1(\alpha), j) = |f(\alpha)|$.

Claim 2: For any binary string α of length at most k, we have $|f(\alpha)| = 0$ implies $w(M_0(\alpha), j) = 0$.

Let A denote all $\alpha \in \{0,1\}^k$ such that $f(\alpha) \neq \emptyset$. For all $\alpha \in A$ let $g(\alpha)$ denote the shortest string in $f(\alpha)$.

Claim 3: For any α in A, we have $f(g(\alpha)) = \emptyset$. Since any proper prefix of $g(\alpha)$ is also a proper prefix of α.

Claim 4: For any α in A, we have $\deg(E_1(g(\alpha)), j) = 0$ and $M_2(g(\alpha)) = \{(|g(\alpha)|, j)\}$. Follows from Claims 1 and 3.

Claim 5: For any α in A, we have $0 \le |g(\alpha)| < k$ and $g(\alpha) \in S_1(|g(\alpha)|, j)$. Follows from Claim 4, the definition of S_1 and since $g(\alpha) \in f(\alpha)$.

Claim 6: For any α in A, we have $w(M_0(\alpha), j) \le w(|g(\alpha)|)$. Since $M_0(\alpha)$ is a matching, $\deg(M_0(\alpha), j)$ is either zero or one. If it is zero, the claim is trivial. If it is one, then $M_0(\alpha)$ contains a unique (i, j) for some left-vertex i. Thus, $M_2(\beta) = \{(i, j)\}$ for some proper prefix β of α of length i. By the definition of g, $|g(\alpha)| \le |\beta| = i$. So, the claim follows.

Claim 7: For all $0 \le i < k$ and β in $S_1(i, j)$ we have $|g^{-1}(\beta)| \le 2^{k-i}$. Since $\beta \in S_1(i, j)$, we have $|\beta| = i$. Since $g(\alpha)$ is a prefix of α, $|g^{-1}(\beta)|$ is at most the number of k bit extensions of β, which is 2^{k-i}.

We are now ready to prove the lemma

$$\begin{aligned}
\sum_{\alpha\in\{0,1\}^k} w(M_0(\alpha), j) &= \sum_{\alpha\in A} w(M_0(\alpha), j) \\
&= \sum_{0\le i<k} \sum_{\beta\in S_1(i,j)} \sum_{\alpha\in g^{-1}(\beta)} w(M_0(\alpha), j) \\
&\le \sum_{0\le i<k} \sum_{\beta\in S_1(i,j)} \sum_{\alpha\in g^{-1}(\beta)} w(|g(\alpha)|) \\
&= \sum_{0\le i<k} \sum_{\beta\in S_1(i,j)} \sum_{\alpha\in g^{-1}(\beta)} w(i) \\
&\le \sum_{0\le i<k} \sum_{\beta\in S_1(i,j)} w(i)2^{k-i} \\
&= \sum_{0\le i<k} w(i)2^{k-i}|S_1(i, j)|
\end{aligned}$$

where the first step follows from Claim 2 and the definition of A; the second step follows from Claim 5; step three follows from Claim 6; step four follows since $\alpha \in g^{-1}(\beta)$ and $\beta \in S_1(i, j)$ implies $i = |\beta| = |g(\alpha)|$; step five follows from Claim 7; and step 6 is immediate. □

Lemma 4.8 *For any integer k such that $k \le m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha)) \le \sum_{0 \le i < k} \sum_{0 \le j < n} w(i) 2^{k-i} |S_1(i,j)|.$$

Proof Follows from summing the result of Lemma 4.7 over all $0 \le j < n$. □

Lemma 4.9 *For any integer k such that $k \le m$, we have*

$$\sum_{\alpha \in \{0,1\}^k} w(M_1(\alpha)) \ge \frac{1}{4} \sum_{\alpha \in \{0,1\}^k} w(M_0(\alpha))$$

Proof Follows from Lemmas 4.8 and 4.6. □

5 Analysis Under a Probability Distribution

In this section we begin working with probability distributions and show that the expected weight of the sampled candidate edge matching is within a constant factor of the max-weight matching on the entire graph (Lemma 5.2). We tie these results with Sect. 4, to show that the expected weight of the non-sampled matching is within a constant the weight of a max-weight matching on the entire graph (Lemma 5.4), completing the main technical portion of the argument. The only result exported from this section are Lemmas 5.3 and 5.4.

Define a function Sample, which takes an m-bit binary string α such that $\text{Sample}(\alpha) = \{i \mid \alpha_i = 0\}$. We introduce a probability distribution $\mathcal{P}$ on m-bit binary strings α. In $\mathcal{P}$, each α_i independently has an equal chance of $\alpha_i = 0$ and $\alpha_i = 1$.

Lemma 5.1 *Let M^* be a max-weight matching and M_α denote a max-weight matching on the subgraph induced by* Sample(α) *for a binary string α. Then,* $\text{Exp}[w(M_\alpha)] \ge \frac{1}{2} w(M^*)$.

Proof We have $\text{Exp}[w(M_\alpha)] \ge \sum_{i \in \text{Left}(M^*)} \Pr[\alpha_i = 0] w(i) = \frac{1}{2} w(M^*)$, where the first step follows from the linearity of expectation and observing that the matching M_α has a weight at least as big as the weight of a matching $M'_\alpha = \{(i,j) \in M^* \mid \alpha_i = 0\}$. □

Lemma 5.2 *Let M^* be a max-weight matching. Then, the following inequality holds*:

$$\text{Exp}[w(\text{AlgA}(\text{Sample}(\alpha)))] \ge \frac{1}{4} w(M^*)$$

Proof Let M_α be a max-weight matching on Sample(α), for a string α. Then, we have

$$\mathrm{Exp}[w(\mathrm{AlgA}(\mathrm{Sample}(\alpha)))] \geq \mathrm{Exp}\left[\frac{1}{2}w(M_\alpha)\right] \geq \frac{1}{4}w(M^*)$$

where the first step follows from Lemma 3.5 and the second step follows from Lemma 5.1 and the linearity of expectation. □

Lemma 5.3 *Let α be any m-bit binary string and $A = \mathrm{Sample}(\alpha)$. We have $M_0(\alpha) = \mathrm{AlgA}(A)$.*

Proof Follows from the definition of AlgA and M_0. □

Lemma 5.4 *Let M^* be a max-weight matching. We have $\mathrm{Exp}[w(M_1(\alpha))] \geq \frac{1}{16}w(M^*)$.*

Proof We have

$$\begin{aligned}\mathrm{Exp}[w(M_1(\alpha))] &\geq \frac{1}{4}\mathrm{Exp}[w(M_0(\alpha))]\\ &= \frac{1}{4}\mathrm{Exp}[w(\mathrm{AlgA}(\mathrm{Sample}(\alpha)))]\\ &\geq \frac{1}{16}w(M^*)\end{aligned}$$

where the first inequality follows from Lemma 4.9, the equality follows from Lemma 5.3, and the final inequality follows from Lemma 5.2. □

6 Intermediate Algorithm

In this section we analyze a useful intermediate algorithm between the counting arguments and the final online algorithm. In specific, in the counting argument, we process the non-sampled left-vertices in decreasing order of weight. In this section we use Lemma 4.1 to argue that we can process the non-sampled left-vertices in an arbitrary order. This is similar to what happens in the original sample-and-price algorithm in the secretary problem. In the secretary problem, we depend on the fraction of time when the second highest bidder is sampled and the highest bidder is not. When this happens, we can process the non-sampled bidders in an arbitrary order, since only one of them meets the required price.

We define an algorithm $\mathrm{AlgB}(\alpha)$ that takes an m-bit binary string α, and returns a matching. The $\mathrm{AlgB}(\alpha)$ function is as follows:

```
$M := \mathrm{AlgA}(\mathrm{Sample}(\alpha))$
$A := \{0, \ldots, m\} - \mathrm{Sample}(\alpha)$
$E := \emptyset$
```

```
for $i$ in arbitrary order from $A$:
    $E := E \cup \text{Cands}(i, M)$
return the matching of pairs $(i, j)$ in $E$ where $j$ appears at
    most once in $E$
```

Lemma 6.1 *For any m-bit binary string α, we have* $\text{AlgB}(\alpha) = M_1(\alpha)$.

Proof The results of Lemma 5.3 give that for any m-bit binary string α we have $M_0(\alpha) = \text{AlgA}(\text{Sample}(\alpha))$. Applying Lemma 4.1, we also have that $E_0(\alpha) = \bigcup_{i \in A} \text{Cands}(i, M_0(\alpha))$, where A is as in AlgB. Thus, at the end of the loop in AlgB, E is equal to $E_0(\alpha)$. The lemma statement follows from the definition of $M_1(\alpha)$ and the last line of AlgB. □

7 Online Algorithm

In this section, we define and analyze the final online algorithm, which is closely related to AlgB. The main difference between the two algorithms is that the online algorithm must rely on the random permutation of left-vertices for sampling whereas up to now we have discussed a simpler direct sampling method, where each element has an equal chance of being sampled or not. With Lemma 7.2 we show that the direct sampling method and the method of sampling from the random permutation induce the same distribution. The main theorem follows immediately from our previous results.

Define the online algorithm as follows. Initially, we are given the set of right-vertices, and the total number of left-vertices we will see, m. The algorithm ONLINE proceeds in two phases.

```
First phase:
        $k := \text{Bin}(m, \frac{1}{2})$, where Bin is the binomial distribution.
        Reject the first $k$ vertices, not matching them to anything.
        Let $B$ be the set of all the rejected vertices.
        $M_0 := \text{AlgA}(B)$.
Second phase:
        We are given $M_0$ from the first phase.
        We build a matching $M_1$, initialized to $\emptyset$.
        On receiving a left-vertex $i$:
            $A := \text{Cands}(i, M_0)$
            if $A \neq \emptyset$ and the right-vertex in $A$ is unmatched in $M_1$
                $M_1 := M_1 \cup A$
        return the matching $M_1$
```

Lemma 7.1 *Let α be a m-bit binary string, $B = \text{Sample}(\alpha)$ and M_1^B be a matching returned by ONLINE when B is sampled in the first phase. Then, $w(M_1^B) \geq w(\text{AlgB}(\alpha))$.*

Proof ONLINE and AlgB perform the same operations on the vertices that are not sampled, with the small optimization that ONLINE matches a right vertex j to the first left-vertex i such that $\{(i, j)\} = \text{Cands}(i, M_0)$, while AlgB does not match any right-vertex j that is returned twice by Cands. $\square$

Lemma 7.2 *Consider a set A of m elements. Let $\mathcal{P}$ be the probability distribution where each $a \in A$ independently has an equal chance of being sampled or not sampled.*

Let $\mathcal{P}'$ be the probability distribution where we first pick a k from $\text{Bin}(m, \frac{1}{2})$. Then, from a uniformly random permutation of A, we only sample the first k elements.

The probability distributions $\mathcal{P}$ and $\mathcal{P}'$ are equal.

Proof We simply prove that each particular sample set $B \subseteq A$ appears with the same probability in both distributions. The probability of any particular sample B under $\mathcal{P}$ is $\frac{1}{2^m}$. We must show the same is true under $\mathcal{P}'$. Let $|B| = k'$. The probability of B under $\mathcal{P}'$ is $[\frac{k'!(m-k')!}{m!}] \cdot [\frac{m!}{k'!(m-k')!} \frac{1}{2^m}]$, where the first term comes from the probability that the elements in B come first in the random permutation, and the second term comes from the probability that k' is chosen as the cutoff for the sampling. $\square$

Theorem 7.3 *Let M^* be a max-weight matching. Given a uniformly random permutation of the left-vertices, ONLINE returns a matching whose weight is least $\frac{1}{16}w(M^*)$ in expectation.*

Proof Let $\mathcal{P}$ and $\mathcal{P}'$ be as defined in Lemma 7.2. We must show that the expected weight of the matching produced by ONLINE under $\mathcal{P}'$ is at least $\frac{1}{16}w(M^*)$.

By Lemma 7.1, for each possible sample selection ONLINE returns a matching with weight at least as large as the matching returned by AlgB on the same sample. Thus, the expectation of ONLINE under $\mathcal{P}'$ is at least as large as the expectation of AlgB under $\mathcal{P}'$. Lemma 7.2, gives us the result that the expectation of AlgB(α) when α is drawn from $\mathcal{P}'$ is the same as that when α is drawn from $\mathcal{P}$. Lemma 6.1 gives us the result that the expectation of AlgB(α) under $\mathcal{P}$ is the same as the expectation of $M_1(\alpha)$ under $\mathcal{P}$. Finally, Lemma 5.4 shows that the expectation of $M_1(\alpha)$ under $\mathcal{P}$ is at least $\frac{1}{16}w(M^*)$, completing the result. $\square$

8 Concluding Remarks

Recently, Korula and Pál have shown that the techniques introduced in this paper are applicable to some generalizations of transversal matroid matching [9]. In this section we make some remarks on the strong connection between transversal matroids and general matroids. The connection comes from the following characterization of a basis: B is a basis of a matroid M iff B is a minimal set having non-empty intersection with every co-circuit of M [12]. With this characterization, one can think of a general matroid as a bipartite graph in the following way. Let the matroid elements be the left-vertices and co-circuits be the right-vertices. Let there be an edge between an element and a co-circuit if the element belongs to the co-circuit. An independent set in the general matroid is then a combinatorial structure which is close to a matching, but not the same. Consider taking a particular element into an independent set we are constructing. On taking in the element, we cover all the co-circuits containing that element because they have non-empty intersection with the constructed independent set. After that, to increase the independent set, we can only take elements which cover some uncovered co-circuits. So, in a sense, independent sets match left-vertices to subsets of right-vertices. Perhaps it is possible to come up with a sample-and-price scheme for pricing co-circuits to extend the results of this paper to general matroids, solving the matroid secretary problem?

References

1. Babaioff, M., Immorlica, N., Kleinberg, R.: Matroids, secretary problems, and online mechanisms. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, January 2007, pp. 434–443 (2008)
2. Birnbaum, B., Mathieu, C.: On-line bipartite matching made simple. SIGACT News **39**, 80–87 (2008)
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
4. Ferguson, T.: Who solved the secretary problem? Stat. Sci. **4**, 282–289 (1989)
5. Freeman, P.R.: The secretary problem and its extensions: a review. Int. Stat. Rev. **51**, 189–206 (1983)
6. Karger, D.: Random sampling and greedy sparsification for matroid optimization problems. Math. Program. **82**, 41–81 (1998)
7. Karp, R.M., Vazirani, U.V., Vazirani, V.V.: An optimal algorithm for on-line bipartite matching. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, New York, NY, USA, May 1990, pp. 352–358 (1990)
8. Kleinberg, R.: A multiple-choice secretary algorithm with applications to online auctions. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, January 2005, pp. 630–631 (2005)
9. Korula, N., Pál, M.: Algorithms for secretary problems on graphs and hypergraphs. arXiv:0807.1139v1. July 2008
10. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Dover, Mineola (2001)
11. Lindley, D.V.: Dynamic programming and decision theory. Appl. Stat. **10**, 39–51 (1961)
12. Oxley, J.: What is a matroid? Cubo **5**, 179–218 (2003)