Secretary Matching with Vertex Arrivals and No Rejections

Mohak Goyal
Department of Management Science & Engineering, Stanford University
mohakg@stanford.edu

Abstract

Most prior work on online matching problems has been with the flexibility of keeping some vertices unmatched. We study three related online matching problems with the constraint of matching every vertex, i.e., *with no rejections.* We adopt a model in which vertices arrive in a uniformly random order and the non-negative edge-weights are arbitrary. For the capacitated online bipartite matching problem, in which the vertices of one side of the graph are offline and those of the other side arrive online, we give a 4.62-competitive algorithm when the capacity of each offline vertex is 2. For the online general (non-bipartite) matching problem, where all vertices arrive online, we give a 3.34-competitive algorithm. We also study the online roommate matching problem, in which each room (offline vertex) holds 2 persons (online vertices). Persons derive non-negative additive utilities from their room as well as roommate. In this model, with the goal of maximizing the social welfare, we give a 7.96-competitive algorithm. This is an improvement over the 24.72 approximation factor in prior work.

1 Introduction

Online allocation problems study scenarios in which the input information is presented in steps, and the algorithm must make irreversible decisions at each step. These problems have applications in various areas such as ride-hailing platforms (Dickerson et al. 2018) and ad auctions (Mehta 2012). In each of these applications, the future inputs are unknown and the goal is typically to maximize the revenue over the entire time horizon. Mathematically, several online allocation problems can be modeled as the problem of finding maximum-weight matchings on graphs.

One of the most commonly studied models of online matching is one with *vertex arrivals*. In this model, one vertex arrives in each time step and it reveals weights of edges from itself to the previously arrived and offline vertices. The algorithm is required to decide whether the current vertex should be matched and if yes, to which other vertex. This model is adopted in several seminal papers in online matching theory, for example (Karp, Vazirani, and Vazirani 1990; Kesselheim et al. 2013; Gamlath et al. 2019). Another well-studied model is one in which all the vertices are known at the outset and edges are revealed in an online fashion. The decision to include an edge in the matching has to be made when it is seen, before the next edge is observed. This is referred to as the edge arrival model (Korula and Pál 2009). In this paper, we study problems in the vertex arrival model. Further, we assume that all edge-weights are non-negative.

The models of online matching problems also vary in the form of stochasticity in the edge weights and in the arrival order of vertices. It is easy to see that if both the edge weights and the arrival order of vertices are arbitrary then any online algorithm cannot be competitive against its offline counterpart. Therefore, the commonly studied models are of one of two forms: *secretary matching* (Dynkin 1963; Gnedin 1994) and prophet inequalities (Krengel and Sucheston 1977). In the secretary matching model, vertices arrive in a uniformly random order but the edge-weights are arbitrary. Whereas, in the prophet inequalities setting, the edge weights are drawn independently from a known distribution but the vertex arrival order is arbitrary. In this paper, we consider the secretary matching model.

The secretary matching model of online matching is inspired by the classical secretary problem, in which a known number of job applicants arrive in a uniformly random order, are interviewed upon arrival, and at most one of them is hired. The irreversible decision to hire or reject an applicant is taken before the next applicant arrives. The goal is to maximize the probability of hiring the best applicant. The optimal algorithm is e-competitive. The problem was a folklore before being solved formally by (Dynkin 1963). See (Ferguson 1989) for historical details. The lower bound of factor e was given by (Gnedin 1994).

Most known algorithms for problems in the secretary matching model use an explore-and-exploit approach. No matches are made in the exploration phase. In the exploitation phase, the current vertex is matched only if it has an edge with weight above a threshold (Korula and Pál 2009). Alternatively, as in (Kesselheim et al. 2013), it is matched only if it is in a locally optimal matching computed over the vertices observed so far. An edge can be added to a matching only if it satisfies matching constraints, i.e., none of its end points have been matched previously by the algorithm.

All such algorithms make good use of the option of rejecting (i.e., not matching) several or all of the arriving vertices. This option may not be available in many resource allocation settings. Examples include the roommate market studied in (Chan et al. 2016; Huzhang et al. 2017) and mentor-mentee matching in education programs. In ride-hailing too, rejecting customers is a costly option for platforms and may be chosen only in rare situations. Therefore, in this paper we study three problems in online matching with the constraint of not rejecting any vertex.

The first problem is *capacitated bipartite matching*, in which vertices on one side of the graph are offline and have a fixed capacity and those on the other side arrive online and have capacity 1. We adopt an explore-and-exploit strategy which is given in detail in Section 3. The second problem is *general (non-bipartite) matching* in which all vertices arrive online. A vertex can either be matched on arrival, or can be kept *waiting* to be matched with a vertex that arrives later. It is not allowed to match two waiting vertices. For this problem, our algorithm runs in three phases: *exploration, selective matching*, and *forced matching*. It is inspired by the two-phase algorithm of (Ezra et al. 2020) for the general matching problem without the no-rejection condition. Details of our algorithm are given in Section 4. The third problem is *roommate matching* in Section 5, in which there are $n/2$ offline rooms, each with capacity 2, and n persons that arrive online must be assigned a room each. Persons derive utility from their room as well as roommate.

1.1 Related Work

Some extensions of the secretary problem include allowing multiple choices (Freeman 1983; Preater 1994; Gilbert and Mosteller 2006; Kleinberg 2005), hiring a senior and a junior secretary (Preater 1993), and hiring a team of k persons with submodular valuations over teams (Bateni, Hajiaghayi, and Zadimoghaddam 2013). The secretary problem was generalized to matroids by (Babaioff, Immorlica, and Kleinberg 2007). It is an open problem to find a constant factor approximation algorithm on general matroids. The current best is a $O(\log\log rank)$-competitive algorithm by (Lachish 2014) and (Feldman, Svensson, and Zenklusen 2014). Other works on the matroid secretary problem include (Soto 2013; Im and Wang 2011; Gharan and Vondrák 2013; Ma, Tang, and Wang 2016; Soto, Turkieltaub, and Verdugo 2021).

Secretary matching on bipartite graphs was introduced by (Korula and Pál 2009). They gave an 8-competitive algorithm. The problem was resolved by (Kesselheim et al. 2013) who gave an optimal e-competitive algorithm. More recently, (Reiffenhauser 2019) gave a truthful mechanism for secretary matching on bipartite graphs that attains the same competitive factor of e. Secretary matching on general graphs was solved recently by (Ezra et al. 2020). They gave an optimal 2.40-competitive algorithm, which is, surprisingly, even better than the best-possible on bipartite graphs.

Another related line of work is on resource sharing. The offline roommate market was studied in (Chan et al. 2016). For a 2-persons-per-room model, they showed that maximizing social welfare is NP-hard and gave constant factor approximation algorithms for it. The online roommate market in the secretary setting was studied by (Huzhang et al. 2017) and they too gave constant factor approximation algorithms for social welfare maximization. (Bei and Zhang 2018) gave algorithms for assignment in ride-sharing. (Li and Li 2020) consider fairness considerations in resource sharing in general and for dorm assignment in particular.

1.2 Our Contributions

We have the following main results for secretary matching problems with no rejection:

1. For online capacitated bipartite matching, we give a 4.62-competitive algorithm when offline vertices have capacity 2 and a 5.46-competitive analysis of the algorithm of (Huzhang et al. 2017) when the offline vertices have capacity 1. They gave a 6.18-competitive factor analysis.
2. We give a 3.34-competitive algorithm for online general (non-bipartite) matching.
3. For online roommate matching, we give a 7.96-competitive algorithm. This result is an improvement over the 24.72 factor given by (Huzhang et al. 2017).

All our algorithms run in polynomial time. The competitive analysis results hold in expectation, which is taken over the randomness in the arrival order and in the algorithm.

2 Model and Preliminaries

An online algorithm is said to be c-competitive if its output has expected weight (or utility) at least $\frac{\text{OPT}}{c}$, where OPT is the weight (or utility) of the optimal offline solution. Denote the weights of matching M and edge e by $w(M)$ and $w(e)$ respectively. The three problems we study in this paper are described separately in the following subsections.

2.1 No-Rejection Capacitated Bipartite Matching

This problem is defined over bipartite graph $G = (L, R, W)$, where L is the set of offline vertices, R is the set of online vertices, and W is the set of edge weights. The offline vertices are known from the outset and the online vertices arrive in a uniformly random order. Each offline vertex has a fixed known capacity, which is the number of online vertices it can match with. Denote the capacity of vertex $u \in L$ by $c(u)$. We consider the case where $\sum_{u\in L} c(u) = n$, where n is the number of online vertices. We assume that G is a complete bipartite graph and that all edge weights are non-negative.

The objective is to maximize the weight of the matching. In this paper we study two important special cases of the capacities of offline vertices. The first case is $c(u) = 1$ for all $u \in L$ and the second case is $c(u) = 2$ for all $u \in L$. We refer to the former as BIPARTITEMATCHING1 and to the latter as BIPARTITEMATCHING2.

2.2 No-Rejection General Matching

In this problem, all n vertices $v \in V$ of graph $G = (V, W)$ arrive online in a uniformly random order and n is even[1]. We consider a complete graph with non-negative edge-weights given by W. The algorithm can match the current vertex to

[1]This is required due to the no-rejection condition. However, our algorithm and its competitive ratio analysis work also if n is odd and any one vertex is allowed to be kept unmatched.

a waiting vertex, or keep it waiting. However, it is required to match all vertices by the end of the process. The objective is to maximize the weight of the constructed matching. We refer to this problem as GENERALMATCHING.

2.3 No-Rejection Roommate Matching

This problem was defined in (Huzhang et al. 2017) who studied it as an online version of the offline roommate market model proposed by (Chan et al. 2016). In this problem, there are m rooms modelled as offline vertices and each room has 2 beds. $n = 2m$ people arrive online in a uniformly random order and each person must be assigned a room upon arrival. Each room is assigned to 2 persons.

Upon arrival, each person reveals her *room valuation* for each of the m rooms and *mutual utility* of being roommates with each of the persons who arrived before her. The mutual utility captures the sum of utilities enjoyed by both persons sharing a room. All room valuations and the mutual utility for every pair of persons are non-negative. Define a *room allocation* as a set of m tuples (r^i, v_1^i, v_2^i) where r^i denotes a room and v_1^i, v_2^i denote the persons assigned to room r^i. The utility of tuple (r^i, v_1^i, v_2^i) is the sum of 2 room valuations for room r^i made by v_1^i and v_2^i and the mutual utility of persons v_1^i and v_2^i. The total utility of a room allocation is the sum of utilities of all its m constituent tuples. The objective is to maximize the utility of the room allocation. We refer to this problem as ROOMMATEMATCHING.

In the following sections, we give algorithms and technical results for the three problems described in this section.

3 Online Capacitated Bipartite Matching

In this section we give online algorithms, ALG1 and ALG2, for problems BIPARTITEMATCHING1 and BIPARTITEMATCHING2 respectively. Both the algorithms run in two-phases and employ a explore-exploit strategy.

For ease of notation, we number the online vertices from 1 to n in their order of arrival. We use the integer variable v both as the number of a step and the name of the current vertex. Define k to be the stopping point of the exploration phase in both ALG1 and ALG2. For ALG1, we set $k = \lfloor 0.21n \rfloor$ and for ALG2 we set $k = \lfloor 0.25n \rfloor$. We give the technical results in the following subsections.

3.1 BIPARTITEMATCHING1

ALG1 for BIPARTITEMATCHING1 is the same as in (Huzhang et al. 2017) [Section 3.1], but with a different stopping point for the exploration phase, in which arriving online vertices are matched to uniformly randomly chosen unmatched offline vertices. When vertex v arrives in the exploitation phase, the algorithm computes an optimal matching M^v over all the offline vertices and online vertices from 1 to v. Denote the edge incident on v in M^v by e^v and the neighbor of v in M^v by $l(e^v)$. If $l(e^v)$ is available, then v is matched to it, otherwise v is matched to a uniformly randomly chosen available offline vertex.

Lemma 1. *For* BIPARTITEMATCHING1*, let* OPT *be the offline optimum value. The expected weight of edge e^v computed in line* 10 *of* ALG1 *is at least $\frac{\text{OPT}}{n}$.*

Algorithm 1: ALG1 for BIPARTITEMATCHING1

```
1:  $R' \leftarrow \emptyset$                          ▷ Set of online vertices seen so far
2:  $M \leftarrow \emptyset$                           ▷ Matching
3:  for every arriving vertex $v$ do
4:      $R' \leftarrow R' \cup \{v\}$
5:      if $v < \lfloor 0.21n \rfloor$ then           ▷ Exploration phase
6:          Uniformly randomly pick an available $v' \in L$
7:          $M \leftarrow M \cup \{(v', v)\}$
8:      else                                          ▷ Exploitation phase
9:          $M^v \leftarrow$ Optimal bipartite matching on $G(L, R')$
10:         $e^v \leftarrow$ Edge incident on $v$ in $M^v$
11:         if $M \cup \{e^v\}$ is a matching then
12:             $M \leftarrow M \cup \{e^v\}$
13:         else
14:             Uniformly randomly pick an available $v' \in L$
15:             $M \leftarrow M \cup \{(v', v)\}$
16:         end if
17:     end if
18: end for
19: return $M$
```

Proof. This result follows directly from Lemma 1 of (Kesselheim et al. 2013). In any step v of the algorithm, the identity and order of the vertices arrived so far can be modelled via the following random process: first choose a set R' of size v from R. Then determine the arrival order of these v vertices by iteratively selecting a vertex at random from R' without replacement. In step v, the algorithm calculates a optimal matching M^v on $G[L, R']$. Since the current vertex v can be seen as being selected uniformly at random from the set R', the expected weight of the edge e^v in M^v is $w(M^v)/v$. Also, since R' can be seen as being uniformly selected from R with size v we know $\mathbb{E}[w(M^v)] \geq \frac{v}{n}\text{OPT}$. Together we have, $\mathbb{E}[w(e^v)] \geq \frac{\text{OPT}}{n}$. □

Lemma 2. *For* BIPARTITEMATCHING1*, probability that edge e^v computed in line* 10 *of* ALG1 *can be added to the matching M, i.e., the 'if' condition of line* 11 *in* ALG1 *is 'true' is at least* $\frac{k}{v-1}\frac{n-v}{n}\left(1 + \frac{k}{n}\log\left(\frac{v}{k}\right) + \frac{k^2}{2n^2}\log^2\left(\frac{v}{k}\right) - o(1)\right)$.

Proof. Denote the probability that e^v can be added to the matching by $\mathbb{P}(v_{\texttt{success}})$. We start with a weak lower bound on $\mathbb{P}(v_{\texttt{success}})$. For $l(e^v)$ to be available at step v, it must neither be same as $l(e^u)$ for any $u < v$ nor should it have been picked in a previous random matching step. There are at most $v-1$ random matching steps before step v, therefore the probability of $l(e^v)$ being picked in a random matching is at most $\frac{v-1}{n}$. We use the randomness of the arrival order to bound the probability of $l(e^v)$ being same as $l(e^u)$ for any $u < v$. Consider a step $u < v$. Out of the u participating online vertices in M^u, the probability that $l(e^v)$ is matched to vertex u in M^u is at most $\frac{1}{u}$. This is because of the uniformly random order of the u participating vertices. Further, this is independent of the order of the vertices $1, \ldots, u-1$. Therefore, the event that for some $u' < u$, $l(e^{u'}) = l(e^v)$ is independent of the event $l(e^u) = l(e^v)$. Following inductively from steps $v-1$ to $k+1$, the probability that $l(e^v)$

was *not* matched to u in matching M^u prior to step v is:

$$\prod_{u=k+1}^{v-1} \mathbb{P}[l(e^u) \neq l(e^v)] \geq \prod_{u=k+1}^{v-1} \left(1 - \frac{1}{u}\right) = \frac{k}{v-1}. \quad (1)$$

Together with the bound on the probability of being picked in a random matching, this implies,

$$\mathbb{P}(v_{\texttt{success}}) \geq \left(1 - \frac{v-1}{n}\right)\left(\frac{k}{v-1}\right) \geq \frac{k(n-v)}{n(v-1)}. \quad (2)$$

Now we improve the bound in Equation (2) to get the result in the Lemma. Equation (2) implies that in any step $u > k$, the probability that a random match is done is at most $1 - \frac{k(n-u)}{n(u-1)}$. This is because the algorithm resorts to a random matching only if $l(e^u)$ is unavailable. If there is a random matching in step u, then the probability that $l(e^v)$ is picked in it is $\frac{1}{n-(u-1)}$ because there are $n-(u-1)$ vertices available. Therefore, the probability that $l(e^v)$ is *not* picked in a random matching is at least:

$$\prod_{u=1}^{k}\left(1 - \frac{1}{n-(u-1)}\right) \prod_{u=k+1}^{v-1}\left(1 - \frac{1 - \frac{k(n-u)}{n(u-1)}}{n-(u-1)}\right),$$

$$\geq \prod_{u=1}^{k} \frac{n-u}{n-(u-1)} \prod_{u=k+1}^{v-1} \frac{n-(u-1)-1+\frac{k(n-u)}{n(u-1)}}{n-(u-1)},$$

$$= \prod_{u=1}^{k} \frac{n-u}{n-(u-1)} \prod_{u=k+1}^{v-1} \frac{(n-u)\left[1+\frac{k}{n(u-1)}\right]}{n-(u-1)},$$

$$= \prod_{u=1}^{v-1} \frac{n-u}{n-(u-1)} \prod_{u=k+1}^{v-1} \left[1+\frac{k}{n(u-1)}\right],$$

$$\geq \frac{n-v}{n}\left[1 + \sum_{u=k+1}^{v-1} \frac{k}{n(u-1)} + \frac{1}{2}\Big(\sum_{u=k+1}^{v-1} \frac{k}{n(u-1)}\Big)^2 - \frac{1}{2}\Big(\sum_{u=k+1}^{v-1} \frac{k^2}{n^2(u-1)^2}\Big)\right], \quad (3)$$

$$\geq \frac{n-v}{n}\left(1 + \frac{k}{n}\log\left(\frac{v}{k}\right) + \frac{k^2}{2n^2}\log^2\left(\frac{v}{k}\right) - o(1)\right). \quad (4)$$

Equation (3) takes only the first 3 terms in the expansion of the product. Equation (4) follows by considering (3) as a Riemann sum with intervals of length 1 and choosing appropriate lower bound in each interval of the sum. The $-o(1)$ term captures $\sum_{u=k+1}^{v-1} \frac{-k^2}{2n^2(u-1)^2}$ and also the approximation of $\log(v-1)$ as $\log(v)$. This is possible because $v > k$ and k is a constant fraction of n. Combining Equations (4) and (1), we get the desired result. □

Theorem 1. ALG1 *is* 5.46*-competitive for* BIPARTITEMATCHING1.

Proof. We sum over the expected contributions of edges e^v to the weight of the matching M for all $v \in \{k+1, \dots, n\}$.

$$\frac{\mathbb{E}(w(M))}{\text{OPT}} \geq \sum_{v=k+1}^{n} \frac{\mathbb{E}(w(e^v))}{\text{OPT}} \mathbb{P}(v_{\texttt{success}}),$$

$$\geq \sum_{v=k+1}^{n} \frac{1}{n}\frac{k(n-v)}{n(v-1)}\Big(1 + \frac{k\log(\frac{v}{k})}{n} + \frac{k^2\log^2(\frac{v}{k})}{2n^2} - o(1)\Big),$$

$$\geq \frac{k}{n^2}\int_k^n (\frac{n-v}{v-1})(1 + \frac{k\log(\frac{v}{k})}{n} + \frac{k^2\log^2(\frac{v}{k})}{2n^2} - o(1))dv,$$

$$= \frac{k}{n^2}\Big[n\log\left(\frac{n}{k}\right) + \frac{k}{2}\log^2\left(\frac{n}{k}\right) - (n-k) - \frac{k}{n}\Big(n\log\left(\frac{n}{k}\right) - n + k\Big) + \frac{k^2}{6n^2}\log^3\left(\frac{n}{k}\right) - \frac{k^2}{2n^2}\Big(n\log^2\left(\frac{n}{k}\right) - 2n\log\left(\frac{n}{k}\right) + 2n - 2k\Big) - o(n)\Big]. \quad (5)$$

The second inequality results from Lemmas 1 and 2. To get the third inequality, we view the integral $\int_k^n$ as a Riemann sum with subdivisions into intervals of length 1 and use a simple upper bound on the function in each subdivision. The expression in Equation (5) is maximized at $k = \lfloor 0.21n \rfloor$ and attains the value 0.1833 as $n \to \infty$ which is $> \frac{1}{5.46}$. □

3.2 BIPARTITEMATCHING2

One natural algorithm for BIPARTITEMATCHING2 is to consider each offline vertex (with capacity 2) as two distinct vertices, each with capacity 1, and run ALG1. However, using the proposed ALG2, we utilize the fact that we have 2 chances to match each offline vertex, and we do not need to do random matchings unless both chances are used. This is also reflected in the competitive ratio of ALG2, which is at most 4.62 and is better than what we obtain for ALG1.

ALG2 ensures that no offline vertex is matched twice by random assignments until all offline vertices are matched at least once. ALG2 starts with an exploration phase in which only random matches are made. In the exploitation phase, it finds a match for arriving vertex v via an optimal matching M^v, and by a random selection if that match is not feasible.

The technical lemmas on the probability of availability of $l(e^v)$ follow and the competitive ratio is given in Theorem 2.

Lemma 3. *For* BIPARTITEMATCHING2, *let* OPT *be the offline optimum value. The expected weight of edge e^v computed in line* 12 *of* ALG2 *is at least* $\frac{\text{OPT}}{n}$.

Proof. Notice that the edge e^v is computed same as in ALG1. Make c identical copies of each offline vertex, each with capacity 1. The problem reduces to BIPARTITEMATCHING1 and this result follows from Lemma 1. □

Lemma 4. *In* ALG2 *for* BIPARTITEMATCHING2, *for vertices index* $k+1$ *to* $\lfloor n/2 \rfloor$, *the probability that edge e^v computed in line* 12 *can be added to matching M, i.e., the 'if' condition of line* 14 *is 'true' is at least* $\frac{n-v}{2v} - \frac{n^2}{32v^2} - o(1)$.

Lemma 5. *In* ALG2 *for* BIPARTITEMATCHING2, *for vertices index* $\lfloor n/2 \rfloor + 1$ *to* n, *the probability that edge e^v computed in line* 12 *can be added to matching M, i.e., the 'if' condition of line* 14 *is 'true' is at least* $\frac{3n(n-v)}{16v^2} - o(1)$.

The proofs of Lemmas 4 and 5 are given in Appendix A in the longer version of this paper (Goyal 2021).

Theorem 2. ALG2 *is* 4.62*-competitive for* BIPARTITEMATCHING2.

Algorithm 2: ALG2 for BIPARTITEMATCHING2

```
 1: $R' \leftarrow \emptyset$                          ▷ Set of online vertices seen so far
 2: $M \leftarrow \emptyset$                           ▷ Matching
 3: $L_a \leftarrow L$        ▷ Offline vertices available for random matchings
 4: for every arriving vertex $v$ do
 5:     $R' \leftarrow R' \cup \{v\}$
 6:     if $v < \lfloor n/4 \rfloor$ then                ▷ Exploration phase
 7:         Uniformly randomly pick a vertex $v' \in L_a$
 8:         $M \leftarrow M \cup \{(v', v)\}$
 9:         $L_a \leftarrow L_a \setminus \{v'\}$
10:     else                                     ▷ Exploitation phase
11:         $M^v \leftarrow$ Optimal capacitated matching on $G(L, R')$
12:         $e^v \leftarrow$ Edge incident on $v$ in $M^v$
13:         $l(e^v) \leftarrow$ Neighbor of $v$ via $e^v$
14:         if $M \cup \{e^v\}$ is a valid capacitated matching then
15:             $M \leftarrow M \cup \{e^v\}$
16:             $L_a \leftarrow L_a \setminus \{l(e^v)\}$
17:         else
18:             Uniformly randomly pick a vertex $v' \in L_a$
19:             $M \leftarrow M \cup \{(v', v)\}$
20:             $L_a \leftarrow L_a \setminus \{v'\}$
21:         end if
22:     end if
23:     if $L_a = \emptyset$ then                         ▷ Reset $L_a$
24:         $L_a \leftarrow \{v' \in L | v'$ is not matched to full capacity$\}$
25:     end if
26: end for
27: return $M$
```

Proof. We sum the expected contributions of the edges e^v to the weight of the matching M for all $v \in \{k+1, \ldots, n\}$. From Lemmas 3, 4, and 5, we get:

$$\frac{\mathbb{E}(w(M))}{\text{OPT}} \geq \frac{1}{\text{OPT}} \sum_{v=\lfloor \frac{n}{4} \rfloor + 1}^{n} \mathbb{E}(w(e^v))\mathbb{P}(v_{\text{success}})$$

$$\geq \frac{1}{n} \sum_{v=\lfloor \frac{n}{4} \rfloor + 1}^{n/2} \left(\frac{n-v}{2v} - \frac{n^2}{32v^2} \right) + \frac{1}{n} \sum_{v=n/2}^{n} \frac{3n(n-v)}{16v^2} - o(1)$$

$$\geq \int_{\frac{n}{4}}^{\frac{n}{2}} \left(\frac{n-v}{2vn} - \frac{n^2}{32v^2 n} \right) dv + \int_{\frac{n}{2}}^{n} \frac{3(n-v)}{16v^2} dv - o(1)$$

We do a change of variables by substituting v with xn,

$$= \int_{1/4}^{1/2} \frac{1-x}{2x} - \frac{1}{32x^2} dx + \int_{1/2}^{1} \frac{3(1-x)}{16x^2} dx - o(1),$$

$$> 0.2166 > \frac{1}{4.62}.$$

The $o(1)$ error term captures the difference between $k = \lfloor n/4 \rfloor$ and $n/4$ in the limit of the integration; this difference goes to 0 as $n \to \infty$. □

4 Online General Matching

For this problem, described in Subsection 2.2, we give ALG3 which runs in three phases and is inspired by (Ezra et al. 2020) who have a two-phase algorithm without the no-rejection condition. We number the vertices from 1 to n in the order of arrival. We use variable v both as the number of an iteration and as the name of the current vertex.

Algorithm 3: ALG3 for GENERALMATCHING

```
 1: $V' \leftarrow \{1, 2, \ldots \lfloor 6n/17 \rfloor\}$          ▷ Set of observed vertices
 2: $A \leftarrow \{1, 2, \ldots \lfloor 6n/17 \rfloor\}$            ▷ Set of waiting vertices
 3: $M \leftarrow \emptyset$                                  ▷ Matching
 4: for every arriving vertex $v > \lfloor 6n/17 \rfloor$ do
 5:     $V' \leftarrow V' \cup \{v\}$
 6:     if $v$ is even then
 7:         $\tilde{V} = V'$
 8:     else
 9:         $v_r \leftarrow$ randomly chosen from $V' \setminus \{v\}$
10:         $\tilde{V} = V' \setminus \{v_r\}$
11:     end if
12:     $M^v \leftarrow$ Optimal matching on $\tilde{V}$
13:     $e^v \leftarrow$ Edge incident on $v$ in $M^v$
14:     $l(e^v) \leftarrow$ Neighbor of $v$ via $e^v$
15:     if $M \cup \{e^v\}$ is a valid matching then
16:         $M \leftarrow M \cup \{e^v\}$
17:         $A \leftarrow A \setminus \{l(e^v)\}$            ▷ $l(e^v)$ is no longer waiting
18:     else
19:         if $|A| < (n + 1 - v)$ then                 ▷ Keep v waiting
20:             $A \leftarrow A \cup \{v\}$
21:         else                        ▷ Cannot keep any more vertices waiting
22:             Uniformly randomly pick a vertex $v' \in A$
23:             $M \leftarrow M \cup \{(v', v)\}$                 ▷ Forced matching
24:             $A \leftarrow A \setminus \{v'\}$
25:         end if
26:     end if
27: end for
28: return $M$
```

The first phase is of exploration in which all vertices are kept waiting and added to set A. Next is the selective matching phase. When vertex v arrives in this phase, if v is even, the algorithm computes an optimal matching M^v over the set of vertices V' seen so far. If v is odd, the algorithm uniformly randomly chooses a vertex v_r from $V' \setminus v$, and computes the optimal matching M^v over $V' \setminus v_r$. This ensures that v has a match in M^v. If the match of v in M^v (denoted by $l(e^v)$) is a waiting vertex (i.e., it is in A), then they are matched in M, otherwise v is kept waiting and added to A. The third phase is called forced matching. In this phase, the algorithm computes an optimal match M^v just as in the second phase. If $l(e^v)$ is in A, then v is matched to it, otherwise v is matched to a randomly chosen waiting vertex $v' \in A$.

Define k_e and k_s as the stopping points of the first and second phases respectively in ALG3. We set $k_e = \lfloor \frac{6n}{17} \rfloor$ and k_s is not fixed. The second phase ends when the number of waiting vertices, i.e., $|A|$ equals the number of vertices yet to arrive, i.e., $n + 1 - v$. In Lemma 9 we show that k_s is not much smaller than its expected value $\frac{12n}{17}$ w.h.p. for large n.

Lemma 6. *For* GENERALMATCHING, *let* OPT *be the offline optimum value. The expected weight of edge e^v computed in line* 13 *of* ALG3 *is at least* $\frac{4\lfloor v/2 \rfloor - 2}{n(n-1)}$OPT.

Proof. This result follows from Theorem 3.1 of (Ezra et al. 2020). See that V' is a uniformly randomly sampled subset of V of size v and therefore the expected weight of an edge with both end-points in V' is equal to the average weight of edges with end-points in V. Further, e^v can be seen as uniformly randomly sampled from the edges in M^v. $\square$

Lemma 7. *In* ALG3 *for* GENERALMATCHING, *for vertices index k_e+1 to k_s, the probability that edge e^v computed in line* 13 *can be added to the matching, i.e., the 'if' condition of line* 15 *is 'true,' is at least* $\frac{1}{3}(1+\frac{2(k_e-2)^3}{(v-1)^3})$.

Proof. This result follows directly from Lemma 3.2 of (Ezra et al. 2020). They do an elaborate accounting of the probability that a vertex is matched by the time that vertex v arrives, conditioned on the set of vertices arrived so far. $\square$

Lemma 8. *In* ALG3 *for* GENERALMATCHING, *for vertices k_s+1 to n, the probability that edge e^v computed in line* 13 *can be added to the matching, i.e., the 'if' condition of line* 15 *is 'true,' is at least* $\frac{1}{3}(1+\frac{2(k_e-2)^3}{(v-1)^3})(\frac{n-v+1}{n-k_s})$.

Proof. In the forced matching phase, consider two processes running independently. The first process is that of finding a vertex $l(e^v)$ for v, as computed in lines 11 and 12 of ALG3 and starts at step k_e+1. The second process is selecting a vertex v' from the set of waiting vertices A uniformly at random and starts at step k_s+1. Vertex v' is matched to v if $l(e^v)$ is not available. We are interested in the probability of $l(e^v)$ being available when v arrives. For a lower bound of this probability, it is sufficient to consider the case that $l(e^v)$ was not picked earlier by either of the two processes. The probability of it not being picked in the first process (i.e., in optimal matchings M^u for $u<v$) is at least $\frac{1}{3}(1+\frac{2(k_e-2)^3}{(v-1)^3})$ by Lemma 7. The probability of $l(e^v)$ not being picked in the second process (random selection) is at least $(1-\frac{v-1-k_s}{n-k_s})$, since there are at most $(v-1-k_s)$ vertices picked in this process out of $n-k_s$ vertices. This expression simplifies to $\frac{n-v+1}{n-k_s}$. The result follows from multiplying these two probabilities. $\square$

Lemma 9. *For* ALG3*, the stopping point of the selective matching phase, k_s, is at least $\left(\frac{12-\delta}{17}\right)n$ for any small positive constant δ w.h.p. as $n\to\infty$.*

Proof. Denote the set of steps in the selective matching phase, i.e., $\{k_e+1,\ldots,k_s\}$ by V_{SM}. Further, denote the set of steps $\{k_e+1,\ldots,t\}$ where $t\le k_s$ by V_{SM}^t. Denote the indicator random variable of the event that ALG3 keeps vertex $v\in V_{\text{SM}}$ waiting when it arrives by X_v. By Lemma 7,

$$\mathbb{P}(X_v=1)\le 1-\frac{1}{3}(1+\frac{2(k_e-2)^3}{(v-1)^3})=\frac{2}{3}(1-\frac{(k_e-2)^3}{(v-1)^3}).$$

Clearly, X_v for $v\in V_{\text{SM}}^t$ are not independent random variables and therefore we cannot use a Chernoff bound on their sum. However, to obtain a useful concentration inequality, we prove the following property: for any subset $S\subseteq V_{\text{SM}}$,

$$\mathbb{P}\left[\Big(\prod_{u\in S}X_u\Big)=1\right]\le\prod_{u\in S}\mathbb{P}(X_u=1). \tag{6}$$

To prove Equation (6), we analyze how the random variables X_u and $X_{u'}$ depend on each other for any pair of steps (u,u'). Without loss of generality, let $u'<u$. See that $X_{u'}$ is independent of X_u because it is observed before step u. For dependence of X_u on $X_{u'}$, consider the case that $X_{u'}=1$. This event adds vertex u' to the set of waiting vertices A. Since this event does not remove any vertex from A, it does not increase the probability that $l(e^u)$ is not in A. That is, $\mathbb{P}(X_u=1|X_{u'}=1)\le\mathbb{P}(X_u=1)$. Extending this argument to all vertices in S that arrived before u, conditioned on the event $X_{u'}=1$ for all $\{u'\in S|u'<u\}$, the probability that u is kept waiting is at most $\mathbb{P}(X_u=1)$. This proves Equation (6). Intuitively, the process of keeping vertices waiting is *self-correcting* such that if too many vertices are waiting, then the probability of future vertices being kept waiting does not increase. (Panconesi and Srinivasan 1997) gave a methodology for obtaining a Chernoff-like bound using Equation (6). We derive it for our problem below.

Now we define t independent Bernoulli random variables y_u for $u\in V_{\text{SM}}^t$ such that $\mathbb{P}(y_u=1)=\mathbb{P}(X_u=1)$. Denote $X=\sum_{u\in V_{\text{SM}}^t}X_u$ and $Y=\sum_{u\in V_{\text{SM}}^t}y_u$. Notice that X and Y are functions of t; it is omitted from the notation for clarity. Clearly $\mathbb{E}(Y)=E(X)$. We will prove that for any $a>0$,

$$\mathbb{E}[e^{aX}]\le[e^{aY}]. \tag{7}$$

Since e^{az} can be expanded as $\sum_{s=0}^{+\infty}(a^s/s!)z^s$, for any $z\in\mathbb{R}$, by the linearity of expectation, we have $\mathbb{E}[e^{aX}]=\sum_{s=0}^{+\infty}(a^s/s!)\mathbb{E}[X^s]$ and $\mathbb{E}[e^{aY}]=\sum_{s=0}^{+\infty}(a^s/s!)\mathbb{E}[Y^s]$. To prove eq. (7), it suffices to show that for every $s=0,1,\ldots,\infty$, $\mathbb{E}[X^s]\le\mathbb{E}[Y^s]$. By the definition of X, we have $X^s=\sum_\sigma\prod_{j=1}^s X_{\sigma(j)}$ where the summation is over all permutations σ selecting s items from V_{SM}^t with replacement. By linearity of expectation, $\mathbb{E}[X^s]=\sum_\sigma\mathbb{E}[\prod_{j=1}^s X_{\sigma(j)}]$ and $\mathbb{E}[Y^s]=\sum_\sigma\mathbb{E}[\prod_{j=1}^s Y_{\sigma(j)}]$. To prove eq. (7), it now suffices to prove that for every permutation σ, $\sum_\sigma\mathbb{E}[\prod_{j=1}^s X_{\sigma(j)}]\le\mathbb{E}[\prod_{j=1}^s Y_{\sigma(j)}]$. Define Q to be the image of σ, that is Q is the set of distinct elements q such that $\sigma(j)=q$ for some j. We have:

$$\mathbb{E}\big[\prod_{j=1}^s X_{\sigma(j)}\big]=\mathbb{P}[X_u=1\;\forall\,u\in Q]\le\prod_{u\in Q}\mathbb{P}(X_u=1)$$

$$=\prod_{u\in Q}\mathbb{P}(Y_u=1)=\mathbb{P}[Y_u=1\;\forall\,u\in Q]=\mathbb{E}\big[\prod_{j=1}^s Y_{\sigma(j)}\big].$$

The inequality follows from (6). This proves (7). We apply Markov's inequality (Motwani and Raghavan 1995) to the non-negative random variable e^{aX} for $a>0$ and $\varepsilon>0$:

$$\begin{aligned}\mathbb{P}[X\ge(1+\varepsilon)\mathbb{E}[X]]&=\mathbb{P}[e^{aX}\ge e^{a(1+\varepsilon)\mathbb{E}[X]}]\\ &\le\mathbb{E}[e^{aX}]/e^{a(1+\varepsilon)\mathbb{E}[X]}\le\mathbb{E}[e^{aY}]/e^{a(1+\varepsilon)\mathbb{E}[Y]}.\end{aligned}\tag{8}$$

The last inequality is due to (7) and the fact that $\mathbb{E}[X]=\mathbb{E}[Y]$. Now we use the standard steps for proving the Chernoff bound for Y. We use $a=\log(1+\varepsilon)$. Then,

$$\frac{\mathbb{E}[e^{aY}]}{e^{a(1+\varepsilon)\mathbb{E}[Y]}}\le\frac{e^{\mathbb{E}[Y](e^a-1)}}{e^{(1+\varepsilon)\mathbb{E}[Y]}}=\Big(\frac{e^\varepsilon}{(1+\varepsilon)^{1+\varepsilon}}\Big)^{\mathbb{E}[Y]}\le e^{\frac{-E[Y]\varepsilon^2}{3}}.$$

Together with (8), and using $\mathbb{E}[X]=\mathbb{E}[Y]$, this implies:

$$\mathbb{P}[X\ge(1+\varepsilon)\mathbb{E}[X]]\le e^{\frac{-E[X]\varepsilon^2}{3}}. \tag{9}$$

We now use this concentration bound on X to prove the result of the Lemma. Denote z as the number of steps in V_{SM} in which the current vertex is kept waiting. By definition of k_s, the number of vertices waiting after step k_s is equal to the number of vertices yet to arrive. That is: $k_e + z - (k_s - k_e - z) = n - k_s \implies z = \frac{n-2k_e}{2} = \frac{5n}{34}$. This implies that: for k_s to be $< \frac{12-\delta}{17}n$, we must have $X = 5n/34$ for $t < \frac{12-\delta}{17}n$. To be able to use Equation (9), we compute $\mathbb{E}[X]$ for $t = \frac{12-\delta}{17}n$. This is,

$$\mathbb{E}[X] = \sum_{u \in V_{\text{SM}}^t} \mathbb{P}(v_{\texttt{wait}}) = \sum_{u=\lfloor \frac{6n}{17} \rfloor + 1}^{\frac{12-\delta}{17}n} \left[\frac{2}{3}\left(1 - \frac{(k_e-2)^3}{(u-1)^3}\right) \right].$$

Upon solving, we get $\mathbb{E}[X] \le (\frac{5}{34} - \frac{7\delta}{204})n + o(n)$. Therefore, for $\varepsilon = \delta/10$, we have $(1+\varepsilon)\mathbb{E}[X] < 5n/34$. This, together with Equation (9), implies that for any constant $\delta > 0$, we get $k_s > \frac{12-\delta}{17}n$ w.h.p. as $n \to \infty$. □

Theorem 3. ALG3 *is* 3.34*-competitive for* GENERALMATCHING *w.h.p. as* $n \to \infty$.

Proof. We sum the expected contributions of the edges e^v to the weight of the matching M for all $v \in \{k_e+1, \ldots, n\}$. From Lemmas 6, 7, and 8, we get:

$$\frac{\mathbb{E}(w(M))}{\text{OPT}} \ge \sum_{v=k_e+1}^{n} \frac{\mathbb{E}(w(e^v))}{\text{OPT}} \mathbb{P}(e^v \text{is added to matching}),$$

$$\ge \frac{1}{n(n-1)}\Bigg[\sum_{v=k_e+1}^{k_s} \frac{4\lfloor v/2 \rfloor - 2}{3}\Big(1 + \frac{2(k_e-2)^3}{(v-1)^3}\Big)$$
$$+ \sum_{v=k_s+1}^{n} \frac{4\lfloor v/2 \rfloor - 2}{3}\Big(1 + \frac{2(k_e-2)^3}{(v-1)^3}\Big)\Big(\frac{n-v+1}{n-k_s}\Big)\Bigg],$$

$$\ge \frac{1}{n^2}\Bigg[\sum_{v=k_e+1}^{k_s} \frac{2v-4}{3}\Big(1 + \frac{2(k_e-2)^3}{(v-1)^3}\Big)$$
$$+ \sum_{v=k_s+1}^{n} \frac{2v-4}{3}\Big(1 + \frac{2(k_e-2)^3}{(v-1)^3}\Big)\Big(\frac{n-v+1}{n-k_s}\Big)\Bigg],$$

$$\ge \frac{1}{n^2}\Bigg[\int_{k_e}^{k_s} \frac{2v-4}{3}\Big(1 + \frac{2(k_e-2)^3}{(v-1)^3}\Big)dv$$
$$+ \int_{k_s}^{n} \frac{2v-4}{3}\Big(1 + \frac{2(k_e-2)^3}{(v-1)^3}\Big)\Big(\frac{n-v+1}{n-k_s}\Big)dv\Bigg].$$

Now we substitute the values of the parameters k_e and k_s. The value of k_e is $\lfloor \frac{6n}{17} \rfloor$. For any v, the value of the second integrand is smaller than that of the first integrand. Therefore, a lower bound of the expression is obtained by setting k_s to its smallest value. We had observed in Lemma 9 that k_s is at least $\left(\frac{12-\delta}{17}\right)n$ for any small positive constant δ with high probability for $n \to \infty$. We omit further calculations due to space constraints. On solving the integration, we get:

$$\frac{\mathbb{E}(w(M))}{\text{OPT}} \ge 0.30005 - \frac{10\delta}{289} - o(1).$$

By Lemma 9, the bound on k_s holds for any positive constant δ. Using $\delta = 10^{-5}$, we get the factor $> \frac{1}{3.34}$. □

Algorithm 4: ALG4 for ROOMMATEMATCHING

```
1: Draw a random variable $r$ from Uniform$[0,1]$
2: **if** $r \le 0.58$ **then**
3:     Run ALG2 on room valuations
4: **else**
5:     Run ALG3 on mutual utilities
6: **end if**
```

5 Online Roommate Matching

Recall the online roommate matching problem given in Subsection 2.3. If the mutual utilities of all pairs of persons are 0, then this problem is the same as BIPARTITEMATCHING2 with rooms as offline vertices, persons as online vertices, and room valuations as edge-weights. Whereas, if all room valuations are 0, then the problem is the same as GENERALMATCHING over persons as online vertices and mutual utilities as edge-weights. With these observations, we give ALG4 for ROOMMATEMATCHING which considers only the room valuations with probability p and only the mutual utilities with probability $1-p$. The competitive ratio in our analysis is minimized at $p = 0.58$.

Theorem 4. ALG4 *is* 7.96*-competitive for* ROOMMATEMATCHING *w.h.p. as* $n \to \infty$.

Proof. Let OPT be the social welfare of the optimal offline room allocation. Let OPT$_{RV}$ and OPT$_{MU}$ be the social welfare of the offline room allocations which maximize only the sum of room valuations and mutual utilities respectively. Then we have OPT$_{RV}$ + OPT$_{MU}$ $\ge$ OPT. Let U denote the expected social welfare of the room allocation given by ALG4. By Theorems 2 and 3, we have $U \ge 0.58 \cdot \frac{1}{4.62} \cdot \text{OPT}_{RV} + 0.42 \cdot \frac{1}{3.34} \cdot \text{OPT}_{MU} \ge 0.1257 \cdot (\text{OPT}_{RV} + \text{OPT}_{MU}) \ge 0.1257 \cdot \text{OPT} > \frac{\text{OPT}}{7.96}$. □

6 Conclusion

In this paper we do the first detailed analysis of online matching problems with a no-rejection condition. We argue that this is a natural constraint in several resource allocation and resource sharing scenarios. We give constant factor approximation algorithms for capacitated bipartite matching, general matching, and roommate matching problems in the online no-rejection setting. The roommate matching problem captures scenarios where multiple persons may be assigned to use a single resource and there are positive externalities from the other persons using the same resource.

For future work, an important theoretical direction is to find lower bounds of competitive ratios for these problems. Simple lower bounds follow from the corresponding problems without the no-rejection condition. This is $e(\approx 2.73)$ for BIPARTITEMATCHING1 and 2.40 for GENERALMATCHING. Another interesting problem is to design truthful mechanisms for these online matching and resource allocation problems. In the proposed algorithms, an arriving vertex has the incentive to misreport its valuations (i.e., edge-weights) to potentially get a better match. For example, it is a dominant strategy for an arriving vertex to report zero valuations for the resources that are no longer available.

Acknowledgements

I am very grateful to Ashish Goel, Zhihao Jiang, and Anmol Kagrecha for their thoughtful comments on this work. This research was supported by the Office of Naval Research and the Future of Digital Currency Initiative (FDCI) at Stanford University.

References

Babaioff, M.; Immorlica, N.; and Kleinberg, R. 2007. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 434–443.

Bateni, M.; Hajiaghayi, M.; and Zadimoghaddam, M. 2013. Submodular secretary problem and extensions. *ACM Transactions on Algorithms (TALG)*, 9(4): 1–23.

Bei, X.; and Zhang, S. 2018. Algorithms for trip-vehicle assignment in ride-sharing. In *AAAI Conference on Artificial Intelligence*.

Chan, P.; Huang, X.; Liu, Z.; Zhang, C.; and Zhang, S. 2016. Assignment and pricing in roommate market. In *AAAI Conference on Artificial Intelligence*, volume 30.

Dickerson, J.; Sankararaman, K.; Srinivasan, A.; and Xu, P. 2018. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. In *AAAI Conference on Artificial Intelligence*, volume 32.

Dynkin, E. B. 1963. The optimum choice of the instant for stopping a Markov process. *Soviet Mathematics*, 4: 627–629.

Ezra, T.; Feldman, M.; Gravin, N.; and Tang, Z. G. 2020. Secretary Matching with General Arrivals. *arXiv preprint arXiv:2011.01559*.

Feldman, M.; Svensson, O.; and Zenklusen, R. 2014. A simple O (log log (rank))-competitive algorithm for the matroid secretary problem. In *Proceedings of the twenty-sixth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1189–1201.

Ferguson, T. S. 1989. Who solved the secretary problem? *Statistical science*, 4(3): 282–289.

Freeman, P. 1983. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, 189–206.

Gamlath, B.; Kapralov, M.; Maggiori, A.; Svensson, O.; and Wajc, D. 2019. Online matching with general arrivals. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 26–37.

Gharan, S. O.; and Vondrák, J. 2013. On variants of the matroid secretary problem. *Algorithmica*, 67(4): 472–497.

Gilbert, J. P.; and Mosteller, F. 2006. Recognizing the maximum of a sequence. In *Selected Papers of Frederick Mosteller*, 355–398. Springer.

Gnedin, A. V. 1994. A solution to the game of googol. *The Annals of Probability*, 1588–1595.

Goyal, M. 2021. Secretary Matching With Vertex Arrivals and No Rejections. arXiv preprint arXiv:2112.07140.

Huzhang, G.; Huang, X.; Zhang, S.; and Bei, X. 2017. Online Roommate Allocation Problem. In *IJCAI*, 235–241.

Im, S.; and Wang, Y. 2011. Secretary problems: Laminar matroid and interval scheduling. In *Proceedings of the twenty-second annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1265–1274.

Karp, R. M.; Vazirani, U. V.; and Vazirani, V. V. 1990. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM Symposium on Theory of Computing (STOC)*, 352–358.

Kesselheim, T.; Radke, K.; Tönnis, A.; and Vöcking, B. 2013. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European symposium on algorithms*, 589–600. Springer.

Kleinberg, R. 2005. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the sixteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 630–631. Citeseer.

Korula, N.; and Pál, M. 2009. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 508–520. Springer.

Krengel, U.; and Sucheston, L. 1977. Semiamarts and finite values. *Bulletin of the American Mathematical Society*, 83(4): 745–747.

Lachish, O. 2014. O (log log rank) competitive ratio for the matroid secretary problem. In *IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, 326–335.

Li, B.; and Li, Y. 2020. Fair resource sharing and dorm assignment. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 708–716.

Ma, T.; Tang, B.; and Wang, Y. 2016. The simulated greedy algorithm for several submodular matroid secretary problems. *Theory of Computing Systems*, 58(4): 681–706.

Mehta, A. 2012. Online Matching and Ad Allocation. *Theoretical Computer Science*, 8(4): 265–368.

Motwani, R.; and Raghavan, P. 1995. *Randomized algorithms*. Cambridge university press.

Panconesi, A.; and Srinivasan, A. 1997. Randomized distributed edge coloring via an extension of the Chernoff–Hoeffding bounds. *SIAM Journal on Computing*, 26(2): 350–368.

Preater, J. 1993. The senior and junior secretaries problem. *Operations research letters*, 14(4): 231–235.

Preater, J. 1994. On multiple choice secretary problems. *Mathematics of Operations Research*, 19(3): 597–602.

Reiffenhauser, R. 2019. An optimal truthful mechanism for the online weighted bipartite matching problem. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1982–1993.

Soto, J. A. 2013. Matroid secretary problem in the random-assignment model. *SIAM Journal on Computing*, 42(1): 178–211.

Soto, J. A.; Turkieltaub, A.; and Verdugo, V. 2021. Strong algorithms for the ordinal matroid secretary problem. *Mathematics of Operations Research*.