General Graphs are Easier than Bipartite Graphs: Tight Bounds for Secretary Matching

TOMER EZRA, Sapienza University of Rome
MICHAL FELDMAN, Tel Aviv University and Microsoft Research
NICK GRAVIN, ITCS, Shanghai University of Finance and Economics
ZHIHAO GAVIN TANG, ITCS, Shanghai University of Finance and Economics

Online algorithms for *secretary matching* in bipartite weighted graphs have been studied extensively in recent years. We generalize this study to secretary matching in *general* weighted graphs, for both vertex and edge arrival models.

Under vertex arrival, vertices arrive online in a uniformly random order; upon the arrival of a vertex v, the weights of edges from v to all previously arriving vertices are revealed, and the algorithm decides which of these edges, if any, to include in the matching. We provide a *tight* 5/12-competitive algorithm for this setting. Interestingly, it outperforms the best possible algorithm for secretary matching in bipartite graphs with 1-sided arrival, which cannot be better than $1/e$-competitive.

Under edge arrival, edges arrive online in a uniformly random order; upon the arrival of an edge e, its weight is revealed, and the algorithm decides whether to include it in the matching or not. For this setting we provide a 1/4-competitive algorithm, which improves upon the state of the art bound.

CCS Concepts: • **Theory of computation** → **Algorithmic game theory**; **Computational pricing and auctions**.

Additional Key Words and Phrases: Secretary problem; Online matching

ACM Reference Format:
Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. 2022. General Graphs are Easier than Bipartite Graphs: Tight Bounds for Secretary Matching. In *Proceedings of the 23rd ACM Conference on Economics and Computation (EC '22), July 11–15, 2022, Boulder, CO, USA.* ACM, New York, NY, USA, 30 pages. https://doi.org/10.1145/3490486.3538290

1 INTRODUCTION

A common tension in market scenarios, faced repeatedly by individuals and firms, is choosing the right timing to commit to a decision. This tension arises when one chooses their life-long partner, makes a reservation in Airbnb, accepts a job offer, or any other scenario where a decision should be made in the present, without knowing whether or to what extent a better option would arrive in the future.

The most basic mathematical model of such scenarios has been studied in the mathematical literature of optimal stopping theory. In a stopping problem, there are n rounds, and a sequence of n values $w_1, \ldots, w_n$ that are unknown from the outset. In every round t, the value w_t is revealed, and the decision maker makes an irrevocable decision whether to select w_t, in which case the process ends with value w_t, or continue to the next round (unless it's round n), in which case the value

EC '22, July 11–15, 2022, Boulder, CO, USA.

ACM ISBN 978-1-4503-9150-4/22/07…$15.00
https://doi.org/10.1145/3490486.3538290

w_t is lost forever and the process continues to round $t+1$. The goal is to maximize the obtained value. The *competitive ratio* of an algorithm ALG is the minimum ratio between the expected value obtained by ALG and the globally maximal value, over all value sequences $\mathbf{w} = (w_1, \ldots, w_n)$.

One can verify that not much can be done if the process is entirely adversarial. Two alternative models of stochastic variants have been studied extensively in the literature, and become to be known as the *secretary problem* [14, 16] and the *prophet inequality* [35, 36]. In the secretary problem, the value sequence $\mathbf{w}$ is arbitrary, but the values are assumed to arrive in a uniformly random order. The secretary problem is known to admit a competitive ratio of $1/e$, and this is the best possible ratio [8]. In the prophet setting, every value w_t is drawn (independently) from a probability distribution that is known from the outset. The prophet problem is known to admit a competitive ratio of $1/2$, and this is the best possible ratio [35, 36, 45].

A natural question arises: do these results extend to more complex stochastic optimization problems? This problem received a lot of attention in recent years in combinatorial structures such as uniform matroids [21, 33], graphical matroids [34], general matroids [4, 32], intersection of matroids [32], matching in graphs [9, 18, 31, 34], and general downward-closed feasibility constraints [44].

Of particular interest to this paper is the extension to *matching* problems in weighted graphs, where the goal is to select a matching of maximum weight. Matching problems have been of great interest in the last decade, partly due to their high applicability to Internet markets [39], such as online ad auctions, ridesharing platforms, online labor markets, and exchange markets for pairwise kidney exchange.

Two most commonly studied mathematical models of online matching are *edge arrival* and *vertex arrival* models. The edge arrival model, in which elements for selection arrive one by one, is perhaps the most natural model in the context of classic secretary and prophet settings (see, e.g., [34] on graphical matroids). On the other hand, the vertex arrival model (vertices arrive one by one, each vertex along with its incident edges to all previous vertices) is extremely natural in matching applications, as the arriving entities often correspond to the vertices. In fact, vast majority of the models in online matching literature are variants of the vertex arrival: including the seminal online matching model of the 1-sided vertex arrival in bipartite graphs [30], and vertex arrival in general graphs [15, 47].

Online matching with 1-sided vertex arrival. [34] introduced the following 1-sided bipartite matching setting, modeled as an online matching problem in a weighted bipartite graph $G = (L, R; E)$. A pool of jobs are available in the market (associated with vertices in R). Potential employees (associated with vertices in L) arrive one by one, in a *random* order. Upon the arrival of a potential employee her value for every job is revealed (the weights on the corresponding edges), and the algorithm should either match the employee to one of the available jobs or leave her unmatched. The goal is to maximize the total value of the matching in the market. The special case where there is a single job coincides with the classic secretary problem, thus the $1/e$ is the limit of what can be achieved. But the decision making process under the matching scenario is much more complex, as the algorithm should decide not only whether or not to match it, but also to whom, among all available jobs. [34] showed a $1/8$-competitive algorithm for this setting, and [31] improved it to a tight $1/e$-competitive algorithm, which settled the problem of 1-sided bipartite matching.

The analogous 1-sided bipartite matching setting in the prophet model[1] has been studied by [12]. Here too, the $1/2$ guarantee from the classic prophet setting extends to the more complex 1-sided bipartite matching.

[1]In prophet setting, the algorithm has stochastic information about the weights of the edges, but the arrival order is the worst-case, i.e., it is chosen by an adversary.

Online matching with general vertex arrival. The underlying structure of 1-sided bipartite matching is quite restricted, and does not capture more dynamic scenarios, where vertices from both sides of the market arrive dynamically, e.g., passengers and drivers, items and buyers, jobs and employees. Moreover, some scenarios cannot be captured by a bipartite graph at all, e.g., exchange markets for pairwise kidney exchange, or a pool of students who should be paired into roommates. Such scenarios are best captured by matching in a general graph, where upon the arrival of a vertex v, the weights on edges uv are revealed, for all previously arriving vertices u. The prophet version of this scenario has been studied by [9], who showed that the guarantee of $1/2$ extends even to this general matching setting.

That is, in the prophet model, the guarantee of $1/2$ obtained for the simplest setting extends all the way to matching in general graphs. It is only natural to ask whether the same extension holds in the secretary setting as well. This question leads to our main theorem, which reveals a new constant in the realm of secretary problems.

Main Theorem: Secretary matching in general graphs with vertex arrival admits a $5/12$-competitive algorithm. Moreover, $5/12$ is tight.

This result might seem confusing at a first glance, or even contradictory. Indeed, $5/12$ is strictly greater than $1/e$. Doesn't the $1/e$ upper bound, which holds in the classic secretary problem and the 1-sided bipartite matching, hold in this supposedly more general setting of matching in *general* graphs? The answer is that the upper bound of $1/e$ does not apply in our setting. In contrast to 1-sided bipartite matching, which encodes the classic secretary setting as a special case (with a single vertex on the static side), this instance does not impose a $1/e$ barrier in our setting. In particular, by the time the lonely vertex arrives in a random arrival order, half of its neighbors have already arrived (in expectation), leading to a competitive ratio of at least $1/2$, breaking the $1/e$ barrier[2].

For this reason, the proof of our tight upper bound requires a few new ideas and a non trivial construction which uses different techniques than the prior literature on secretary problem.

Remark: For the ordinal setting (i.e., if the algorithm is based only on pairwise comparisons of edges without observing associated numerical values) our result implies a competitive ratio of $\frac{5}{24}$.

Online matching with edge arrival. We now turn to online matching with edge arrival. In this model, the edges arrive one by one. Upon the arrival of an edge, its weight is revealed, and if both its endpoints are available, the algorithm makes an irrevocable decision whether to include it in the matching. This model has been studied in both the prophet and secretaries model, but unlike the vertex arrival model, no tight bounds are known. For prophet setting, the competitive ratio is known to lie between 0.337 and $3/7$ [9, 19, 41]. For secretary, [31] have established a competitive ratio of $1/(2e)$, by a reduction from edge arrival to vertex arrival in hypergraphs. The upper bound of $1/e$ from the classic secretary setting applies here. Our second result further improves this bound.

Theorem: Secretary matching in general graphs with edge arrival admits a $1/4$-competitive algorithm.

The design and analysis of our algorithm for edge arrival carry over to the more general online bipartite hypergraph secretary matching problem (with suitable adjustments). Algorithm 3 in Appendix B gives a competitive ratio of $1/d^{\frac{d}{d-1}}$, where $d+1$ equals the maximum size of the hyperedges. This improves (by a constant factor) upon the previous lower bound of $\frac{1}{ed}$ [31].

[2]In fact, for this particular example, one can easily show a competitive ratio strictly better than $1/2$.

			Secretary	Prophet
Vertex arrival	1-sided bipartite	LB	$\geq 1/e$ [31]	$\geq 1/2$ [12]
		UB	$\leq 1/e$ [8]	$\leq 1/2$ [35, 36]
	General graphs	LB	$\geq 5/12$ [Theorem 3.1]	$\geq 1/2$ [9]
		UB	$\leq 5/12$ [Theorem 4.1]	$\leq 1/2$ [35, 36]
Edge arrival		LB	$\geq 1/(2e)$ [31] $\geq 1/4$ [Theorem 5.2]	≥ 0.337 [9]
		UB	$\leq 1/e$ [8]	$\leq 3/7$ [41]

Table 1. Our results and previous results. UB and LB refer to upper and lower bounds, respectively.

1.1 Our Techniques

Vertex arrival: lower bound (positive result). At a very high level, our algorithm follows the standard explore & exploit approach for secretary problems, which is used essentially by all studies on secretary settings. More specifically, our algorithm resembles the one from [31] for matching with 1-sided vertex arrival. The algorithm begins with an exploration phase, where no matches are made. Then, in the exploitation phase, it finds at each step the optimum matching over the set of vertices that already arrived, and matches the latest vertex to its partner in the optimal matching whenever possible. Our algorithm differs from the one used in [31] in two ways: (i) it uses a longer exploration phase ($1/2$ instead of $1/e$ of the vertices), (ii), it ensures that the latest vertex *always* has a partner in the optimal matching computed on the graph induced by the vertices that have arrived (in particular, we treat differently the cases of odd and even number of vertices).

The key feature of our analysis is a precise accounting of the probability that a given vertex is matched at every step of the algorithm. This accounting turns out to be more challenging in our general vertex arrival model than under 1-sided vertex arrival [31]. Specifically, while in 1-sided arrival it is sufficient to establish an upper bound on the matching probability of vertices in the *offline side*, in general graphs every vertex can be either actively matched (i.e., matched upon its arrival) or passively matched (i.e., chosen as a partner of a vertex that arrives later), and there are non-monotone dependencies between matching probabilities of the vertices. That is, the event that a given vertex is matched may be either positively or negatively correlated with the event that another vertex is matched, depending on the set of vertices that have already arrived. For this reason, an upper bound on the probability of matching is insufficient, and we have to calculate the precise probability that a vertex is matched in every step of the algorithm. To this end, our algorithm ensures that the optimum matching is a perfect matching in every step of the algorithm by omitting a random vertex in odd rounds.

The same high level analysis approach is also pertinent in the cousin prophet inequality setting. Specifically, the Online Contention Resolution Schemes (OCRS) for matching in general graphs in a prophet setting [9] ensures that each vertex is matched to its realized partner with a constant probability (that does not depend on the vertex identity or its arrival time).

However, the accounting in the secretary setting is more complex: the probability of matching the t-th vertex depends on its arrival time t (but *not* on the identity of the vertex). This still results in a constant matching probability of every vertex due to the random arrival order, but leaves us with a much richer space of possible policies. Indeed, one can potentially condition the probability of matching a vertex to its (current) partner on the arrival time t. Algorithm 1 is derived as the solution to the respective optimization problem. Interestingly, it does not condition the matching probability on t (unlike our edge-arrival algorithm, see below) and achieves a tight competitive ratio of $5/12$.

We remark that the optimization problem is quite subtle compared to the simple constant probability policy used in the prophet setting from [9].

Vertex arrival: tight upper bound (negative result). It is known (see, e.g., [17]), but already quite non-trivial to establish, that there is a $1/e$ upper bound for the *game of googol* — the *cardinal* version of the classical secretary problem — where the algorithm observes a random sequence of arbitrarily large numbers and wishes to maximize the probability of stopping at the maximum number among them. In general, proving upper bounds for cardinal variants of the secretary problem is notoriously difficult. For example, only recently, Correa, Dütting, Fischer, and Schewior [5] provided a rather complex proof, based on the infinite version of Ramsey's theorem, that the best competitive ratio for prophet secretary with unknown i.i.d. priors is $1/e$.

Establishing a tight upper bound in our problem is particularly challenging. First, it is cardinal. Second, it has a complex combinatorial structure (rather than choosing a single element, it should choose a set of edges that forms a matching). Finally, the objective is to maximize the expected value of the selected matching rather than the probability of picking the maximum element.

Despite these difficulties, we managed to construct an instance of the secretary matching problem such that: (i) any α-competitive online algorithm for the cardinal setting can be converted into an α-competitive algorithm in the ordinal setting (in a cardinal setting, the online algorithm observes only pairwise comparisons between edges and the objective is to maximize the probability of selecting the maximum valued edge), and (ii) no online algorithm in the ordinal setting can give a better competitive ratio than $5/12$.

Note that (i) can only be achieved for a special family of instances, and it is not a priori clear that such a reduction can yield a tight upper bound. After identifying the right family of instances, our proof follows the high level structure of [5] (e.g., we also use the infinite version of Ramsey's theorem), but it requires several adjustments due to the more complex combinatorial nature of our matching problem. The derivation of (ii) requires substantial and subtle analysis. We first prove that a certain family of online algorithms in the ordinal setting is optimal for the given instance, and then find the algorithm that obtains the best possible competitive ratio within this family.

Edge arrival. [9] considered the prophet matching problem with edge arrival, and proposed an OCRS algorithm for the problem. The idea in the OCRS approach is to ensure that the probability of matching a realized edge uv is a constant fraction α of the probability that uv is in the optimum matching. On the one hand, it is desired to have α as large as possible. On the other hand, the event that both vertices u and v are not yet matched upon the arrival of the edge uv should have a sufficiently high probability. Following a simple union bound argument, setting $\alpha = 1/3$ was sufficient for the matching prophet setting.

We take a similar approach with respect to secretary matching; namely, we control the probability of selecting the last arriving edge at time t, given that it appears in the current optimal matching. However, unlike the simple solution of [9], we cannot simply set this probability to be a constant, since the edges arriving earlier generally have a higher chance to be in the current optimal matching. Instead, we set these probabilities $(\alpha_t)_{t=1}^{|E|}$ to be dependent on the time t, and obtain a recurrence relation on $(\alpha_t)_{t=1}^{|E|}$ that ensures that upon the arrival of an edge uv, both ends of the edge are available with a sufficiently high probability. Given a sequence $(\alpha_t)_{t=1}^{|E|}$, one can derive a good estimate on the competitive ratio of the corresponding online algorithm. We solve the resulting constrained optimization problem and obtain the sequence of $(\alpha_t)_{t=1}^{|E|}$ specified in Algorithm 2. Interestingly, despite the conceptual simplicity of Algorithm 2, it is unclear how to implement it in

polynomial time[3]. Thus, our result for edge arrival is information theoretic. Obtaining a poly-time algorithm with the same ratio remains an open problem.

In conclusion, our results for both vertex and edge arrival establish a close connection between secretary and prophet settings, and demonstrate that tools like OCRS that prove useful in prophet settings (e.g., [9]) can be used to improve the state of the art results for secretary settings. In particular, the tight result for prophet matching translates (with suitable adjustments) into a tight result for secretary matching in the vertex arrival model. This is in contrast to previous general-purpose results [7] connecting Contention Resolution Schemes and secretary problems, which suffered certain constant factor losses in the transition from one setting to another.

1.2 Related work

The 1-sided secretary matching problem studied by [34] can be considered as a generalization of the matroid secretary problem on transversal matroids[4], which was first introduced by [4].

They designed constant competitive algorithms for graphs with bounded degrees, and [6] generalized this result to arbitrary graphs. These results affirmatively answer the famous matroid secretary conjecture by [4] for transversal matroids. Whether $\Omega(1)$-competitive algorithms exist for the secretary problem on general matroids remains an intriguing question. Currently, the best known ratio is $\Omega(1/\log\log \text{rank})$ [37] and [13].

The $1/e$-competitive algorithm by [31] for 1-sided secretary matching is further extended to a truthful mechanism that attains the same competitive ratio by [43]. [34] also studied the secretary matching on hypergraphs and proposed $\Omega(1/d^2)$-competitive algorithms for d-hypergraphs[5]. This result is improved to $\Omega(1/d)$-competitive by [31].

Another line of work considers the secretary problem in the ordinal setting. That is, the algorithm is restricted to do pairwise comparisons between elements, without knowing the exact values of their weights. [22] designed constant competitive algorithms for several families of constraints, including matching, packing LPs and independent set with bounded local independence number. Specifically, they studied the same vertex-arrival model for general graphs as our model and designed an $\frac{e+1}{12e}$-competitive algorithm that only uses ordinal information. [46] studied the ordinal matroid secretary problem and achieved improved competitive ratios for transversal matroids, matching matroids, laminar matroids, etc.

The unweighted version and the vertex-weighted version of the 1-sided secretary matching problem (i.e., random arrival) have also been studied in the online algorithm literature. For the unweighted version when all edges have unit weight, [29] and [38] proved that the Ranking algorithm is 0.696-competitive. For the vertex-weighted version when each offline vertex is associated with a weight and all its incident edges have the same weight, [25] and [28] generalized the Ranking algorithm and achieved a competitive ratio of 0.663.

The general vertex arrival model adopted in this paper was first introduced by [47] in the online matching literature, where the focus is to select maximum size matching under adversarial vertex arrivals. This model is a natural generalization of the 1-sided online bipartite matching model by [30]. [47] designed a 0.526-competitive algorithm for the fractional version of the problem. [15] proposed a $1/2 + \Omega(1)$-competitive algorithm for the integral version of the problem. They also proved that no algorithm has a competitive ratio larger than $1/2$ in the edge arrival setting. Motivated by online ride-sharing, there has been a growing interest in online matching in the past

[3]The difficulty is that in order to estimate the probability that two given vertices u and v are both available upon the arrival of the edge uv, we need to understand the decisions of the online algorithm for different subsets of arrived edges.

[4]More specifically, any α-competitive algorithm for 1-sided secretary matching can be translated into an α-competitive algorithm for transversal matroids.

[5]A d-hypergraph is a hypergraph such that all its hyperedges have size at most d.

few years and other extensions of the 1-sided online bipartite matching model have been studied, including fully online matching [23, 24, 26] and edge-weighted online windowed matching [3]. The 1-sided online bipartite matching has also been extended to the vertex-weighted version [1], the edge-weighted version [10, 11], and the AdWords problem [27, 40].

The matching prophet problem with edge arrivals is first studied by [32] under the more general framework of matroid intersections. Their analysis gives a $\frac{1}{6}$-competitive algorithm for bipartite graphs. [19] explicitly studied the bipartite matching setting and designed a threshold-based $\frac{1}{3}$-competitive algorithm. They also showed an upper bound of $\frac{4}{9}$. The upper bound was improved to $\frac{3}{7}$, and the $\frac{1}{3}$ lower bound was extended to $1/(p+1)$-competitive ratio for p-dimensional matching in [2]. [9] designed an improved 0.337-competitive algorithm for general graphs.

For an intersection of two matroids with random arrival, [20] gives a randomized online algorithm that gives a better competitive ratio than $1/2$. Bipartite matching secretary with (unweighted) edge arrival is a special case of this setting.

2 MODEL AND PRELIMINARIES

The setting is presented by a graph $G = (V, E)$, where V is a set of n vertices. Each edge $e = uv \in E$ has weight $w_e \in \mathbb{R}$ and the vector $w \in \mathbb{R}^{|E|}$ contains the weights of all edges. Given a subset of the vertices $T \subseteq V$, we denote by $G(T)$ the subgraph induced by T. Similarly, given a subset of the edges $E' \subseteq E$, we denote by $G(E')$ the graph induced by E'. We consider two arrival models: (i) vertex arrival, and (ii) edge arrival, where the arriving elements are the vertices and edges, respectively. In both models, upon the arrival of an element, a matching decision should be made immediately and irrevocably, and the goal is to maximize the total weight of the resulting match.

A matching μ is a subset of E, where every vertex is matched to at most a single other vertex. We will also write $\mu(v)$ for the vertex matched with v in the matching μ. That is, if $uv \in \mu$ then $\mu(v) = u$ and $\mu(u) = v$. We denote by $w(\mu) \stackrel{\text{def}}{=} \sum_{e \in \mu} w_e$ the total weight of matching μ. In addition, given a subset of the vertices $T \subseteq V$ and a matching μ, write $\mu|_T = \{uv \in \mu | v, u \in T\}$ for the matching μ restricted to vertices in T. For a weight function w, we write $\mu^*(w)$ for the maximum weighted matching under w.

Vertex arrival. Under vertex arrival model, the vertices arrive in a *uniformly random* order. We rename the vertices $v_1, \ldots, v_n$ according to their arrival order, so that v_t is the vertex that arrives at time t. We denote by $V_t = \{v_1, \ldots, v_t\}$ the set of vertices that arrived up to time t, and by $G(V_t)$ the graph induced by V_t. Upon the arrival of vertex v_t, the weight $w_{v_t v_j}$ is revealed for all vertices $v_j \in V_{t-1}$. Consequently, v_t can either be matched to some available vertex $v_j \in V_{t-1}$ (in which case v_t and v_j are marked as unavailable) or left unmatched (in which case v_t remains available for future matches). We assume that the number of vertices n is known.

Without loss of generality, we may assume that G is a complete graph: we simply add 0-weight edges for the missing edges. Thus we may assume that for every $V' \subseteq V$ such that $|V'|$ is even, the maximum weighted matching of $G(V')$ matches all vertices of $G(V')$.

Edge arrival. Under edge arrival model, the edges arrive in a *uniformly random* order. We rename the edges $e_1, \ldots, e_m$ according to their arrival order, so that e_t is the edge that arrives at time t. We denote by $E_t = \{e_1, \ldots, e_t\}$ the set of edges that arrived up to time t, and by $G(E_t)$ the graph induced by E_t. Upon the arrival of edge $e_t = uv$, its weight w_{e_t} is revealed. If both u and v are available, then e_t can either be matched (in which case u and v are marked as unavailable) or left unmatched (in which case u and v remain available for future matches). We assume that the number of edges m is known.

We assume without loss of generality that the maximum weighted matching of $G(S)$ for any $S \subseteq E$ in both vertex and edge arrival model is unique. Indeed, we can perturb the weight of every edge by adding to it a random number in $[0, \epsilon]$, for a sufficiently small ϵ.[6]

3 SECRETARY MATCHING WITH VERTEX ARRIVAL: POSITIVE RESULT

In this section, we present an algorithm that gives a competitive ratio of 5/12. The algorithm ignores the first k vertices (exploration phase). Then, in every round t, it makes sure that the number of vertices is even. It does so by removing a random vertex r_t from V_{t-1} when t is odd. The algorithm finds maximum weighted matching μ_t in the graph $G(V_t)$ ($G(V_t \setminus \{v_{r_t}\})$ if t is odd). Since the matching is complete, v_t must have a partner $\mu_t(v_t)$ in this matching. If this partner is available, the algorithm matches v_t to it. See Algorithm 1.

ALGORITHM 1: 5/12-secretary matching for vertex arrival (for $k = \lfloor \frac{n}{2} \rfloor$)

Let $v_1, \ldots, v_n$ be the vertices in arrival order;
$A = V$, $\mu = \emptyset$; ▷ A is the set of available vertices, μ is the returned matching
for $t = k+1$ *to* n **do**
  Let $V_t = \{v_1, \ldots, v_t\}$; ▷ V_t is the set of vertices arrived up to time t
  if t *is odd* **then**
    Select $r_t \in \{1, \ldots t-1\}$ uniformly at random;
    Set $V'_t = V_t \setminus \{v_{r_t}\}$; ▷ delete a random vertex from $v_1, \ldots, v_{t-1}$
  else Set $V'_t = V_t$;
  Let μ_t be the maximum weighted matching in $G(V'_t)$;
  Let $e_t \stackrel{\text{def}}{=} v_t \mu_t(v_t)$;
  if $\mu_t(v_t) \in V_t \cap A$ **then**
    Add e_t to μ; ▷ add the chosen edge to the matching
    Remove v_t and $\mu_t(v_t)$ from A;
return *matching* μ

The following theorem asserts that Algorithm 1 achieves a competitive ratio of $\frac{5}{12}$.

Theorem 3.1. *Algorithm 1, with $k = \lfloor \frac{n}{2} \rfloor$, has a competitive ratio of $\frac{5}{12} - O(\frac{1}{n})$ for matching secretary with vertex arrival.*

Theorem 3.1 essentially shows that one can get a competitive ratio arbitrarily close to $\frac{5}{12}$. To see this, note that the algorithm can be modified by adding m auxiliary vertices (at random times) that are connected to all vertices with zero weight edges, and apply Algorithm 1 on the induced graph. Thus, the $O(\frac{1}{n})$ term in the theorem can be replaced by $O(\frac{1}{m})$, which can be made arbitrarily small. The running time of our algorithm is poly(m), i.e., we can achieve $\frac{5}{12} - \varepsilon$ approximation in time poly$(n, \frac{1}{\varepsilon})$.

[6] After the perturbation, for every two different matchings M_1, M_2, there exists an edge $e \in M_1 \Delta M_2$. Fix the perturbations of all edges but e. There is at most one value for the perturbation of e that makes the weights of M_1 and M_2 equal, which happens with probability 0. Since the number of different matchings is finite by the union bound the probability that any two matchings have the same weight is 0.

Proof. In our analysis, the event "matched$(u, \leq t)$" refers to the event that vertex u is matched either before the arrival of v_t or exactly in round t. The following lemma (whose proof is deferred to Appendix A.1) is used in the proof of Theorem 3.1.

Lemma 3.2. *For every $k \geq 3$, $t \geq k$, every possible realization $\widetilde{V}$ of V_t (i.e., $\widetilde{V} \subseteq V, |\widetilde{V}| = t$), and every vertex $u \in \widetilde{V}$, it holds that*

$$\mathbf{Pr}\left[\textit{matched}(u, \leq t) \mid V_t = \widetilde{V}\right] = \frac{2}{3}\left(1 - \frac{(t-3)! \cdot k!}{t! \cdot (k-3)!}\right). \tag{1}$$

Given some $t \geq k+1$, let μ_t denote the maximum weighted matching in $G(V'_t)$, and let μ^* be the maximum weighted matching in G. We give a lower bound on the expected weight of the edge e_t. Since e_t is a random edge chosen uniformly at random among the $\lfloor t/2 \rfloor$ edges in μ_t, for every possible realization $\widetilde{V}'$ of V'_t (note that $\widetilde{V}'$ is of size t if t is even, and of size $t-1$ if t is odd), it holds that $\mathbf{E}[w_{e_t}] = \frac{\mathbf{E}[w(\mu_t)]}{\lfloor t/2 \rfloor}$. Let $\mu^*|_{\widetilde{V}'} \stackrel{\text{def}}{=} \{ij \in \mu^* | i, j \in \widetilde{V}'\}$ be the matching μ^* restricted to vertices in $\widetilde{V}'$. Since μ_t is the maximum matching selected for the set of vertices V'_t, it holds that $\mathbf{E}[w(\mu_t)] \geq \mathbf{E}[w(\mu^*|_{V'_t})]$. Consider next $\mathbf{E}[w(\mu^*|_{V'_t})]$. The number of edges in V'_t is $\binom{2\lfloor \frac{t}{2} \rfloor}{2}$. Since all edges are symmetric (by random arrival) and the total number of edges is $\binom{n}{2}$, it holds that $\mathbf{E}[w(\mu^*|_{V'_t})] = \mathbf{E}[w(\mu^*)]\binom{2\lfloor \frac{t}{2} \rfloor}{2}/\binom{n}{2}$. We get:

$$\mathbf{E}\left[w_{e_t}\right] \geq \frac{1}{\lfloor t/2 \rfloor}\mathbf{E}\left[w(\mu^*|_{\widetilde{V}'})\right] = \frac{1}{\lfloor t/2 \rfloor}\mathbf{E}\left[w(\mu^*)\right]\binom{2\lfloor \frac{t}{2} \rfloor}{2}\Big/\binom{n}{2} = \frac{4 \cdot \lfloor t/2 \rfloor - 2}{n \cdot (n-1)}\mathbf{E}\left[w(\mu^*)\right]. \tag{2}$$

Writing μ for the matching obtained by Algorithm 1, and μ^* for the optimal matching, we get:

$$\begin{aligned}\frac{\mathbf{E}[w(\mu)]}{\mathbf{E}[w(\mu^*)]} &= \frac{1}{\mathbf{E}[w(\mu^*)]}\sum_{t=k+1}^{n}\mathbf{E}\left[w_{e_t} \cdot \mathbb{1}[\mu_t(v_t) \in V_t \cap A_t]\right] \\ &= \frac{1}{\mathbf{E}[w(\mu^*)]}\sum_{t=k+1}^{n}\mathbf{Pr}\left[\mu_t(v_t) \in V_t \cap A_t\right]\mathbf{E}\left[w_{e_t}\right], \end{aligned} \tag{3}$$

where A_t denotes the set A in the beginning of iteration t, and the second equality is derived below.

$$\begin{aligned} & \mathbf{E}\left[w_{e_t} \cdot \mathbb{1}[\mu_t(v_t) \in V_t \cap A_t]\right] \\ = & \sum_{\bar{v}_t, \bar{V}'_t}\mathbf{E}\left[w_{e_t} \cdot \mathbb{1}[\mu_t(v_t) \in V_t \cap A_t] \mid v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \cdot \mathbf{Pr}\left[v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \\ = & \sum_{\bar{v}_t, \bar{V}'_t} w_{e_t} \cdot \mathbf{E}\left[\mathbb{1}[\mu_t(v_t) \in V_t \cap A_t] \mid v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \cdot \mathbf{Pr}\left[v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \\ = & \sum_{\bar{v}_t, \bar{V}'_t} w_{e_t} \cdot \mathbf{Pr}\left[\mu_t(v_t) \in V_t \cap A_t \mid v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \cdot \mathbf{Pr}\left[v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \\ = & \mathbf{Pr}\left[\mu_t(v_t) \in V_t \cap A_t\right] \sum_{\bar{v}_t, \bar{V}'_t} w_{e_t} \cdot \mathbf{Pr}\left[v_t = \bar{v}_t, V'_t = \bar{V}'_t\right] \\ = & \mathbf{Pr}\left[\mu_t(v_t) \in V_t \cap A_t\right] \cdot \mathbf{E}\left[w_{e_t}\right]. \end{aligned} \tag{4}$$

The first and last equalities follow by the law of total probability (taking the randomness over v_t and V'_t, but not over the order of V'_t, and the random removed vertex). The second equality is by the fact

is by substituting the expectation of an indicator variable with the probability of the corresponding event. The fourth equality is by applying Lemma 3.2 with respect to time $t-1$.

Applying Lemma 3.2 again, we have that

$$\mathbf{Pr}\left[\mu_t(v_t) \in V_t \cap A\right] = 1 - \mathbf{Pr}\left[\text{matched}(u, \leq t-1)\right] \overset{(1)}{=} 1 - \frac{2}{3}\left(1 - \frac{(t-4)!\cdot k!}{(t-1)!\cdot(k-3)!}\right). \quad (5)$$

Applying Equations (2),(3), and (5), we get

$$\begin{aligned}\frac{\mathbf{E}[w(\mu)]}{\mathbf{E}[w(\mu^*)]} &\geq \frac{1}{\mathbf{E}[w(\mu^*)]}\sum_{t=k+1}^{n}\left(1-\frac{2}{3}\left(1-\frac{(t-4)!\cdot k!}{(t-1)!\cdot(k-3)!}\right)\right)\cdot\frac{4\cdot\lfloor t/2\rfloor-2}{n\cdot(n-1)}\cdot\mathbf{E}\left[w(\mu^*)\right]\\ &= \sum_{t=k+1}^{n}\left(1-\frac{2}{3}\left(1-\frac{(t-4)!\cdot k!}{(t-1)!\cdot(k-3)!}\right)\right)\cdot\frac{4\cdot\lfloor t/2\rfloor-2}{n\cdot(n-1)} \geq \frac{1}{3}+\frac{k^2}{n^2}-\frac{4k^3}{3n^3}-O\left(\frac{1}{n}\right),\end{aligned}$$

where the derivation of the last inequality is deferred to Appendix A.2. The last expression attains its maximum at $k=\frac{n}{2}$, achieving a competitive ratio of $\frac{5}{12}-O(\frac{1}{n})$. □

Remark: Algorithm 1 can be modified to a 5/24-competitive algorithm in the ordinal setting. Observe that the only step in which Algorithm 1 uses the edge weights is for constructing a maximum weighted matching μ_t in line 9. In the ordinal setting, instead of computing the max weight matching (which is not possible, since only pairwise comparisons are available), we greedily add the edges from largest to smallest. This procedure can be implemented using pairwise comparisons of edges, and gives at least half of the weight of the max weight matching in $G(V_t')$. Thus, we suffer an extra factor 2 loss. The rest of the analysis remains intact. The resulting $5/24 \approx 0.208$-competitive ratio improves upon the $\frac{e+1}{12e} \approx 0.114$ competitive ratio by [22]. In Section 4, we establish a tight upper bound of 5/12 for the cardinal setting which also applies to the ordinal setting. Closing the gap between the lower bound of 5/24 and the upper bound of 5/12 remains an open problem.

4 SECRETARY MATCHING WITH VERTEX ARRIVAL: A TIGHT UPPER BOUND

In this section we establish the following theorem showing that the competitive ratio of $\frac{5}{12}$ is tight.

Theorem 4.1. *No online algorithm has a competitive ratio better than $\frac{5}{12}$ for secretary matching with vertex arrival.*

We first give an overview of our proof approach for showing that the competitive ratio of $\frac{5}{12}$ is tight. The complete proof appears in Section 4.1.

We consider the following family of instances: G is a complete graph on n vertices, each vertex $v \in V(G)$ has a positive integer value $\lambda_v \in \mathbb{N}$, such that $\lambda_u \neq \lambda_v$ for any $u \neq v$. The weight of each edge uv is $w_{uv} = n^{3(\lambda_u+\lambda_v)}$. Thus, any instance can be specified by a set of values $\Lambda \subset \mathbb{N}$ of size $n = |\Lambda|$ on the graph vertices. Let $\binom{\mathbb{N}}{n}$ denote the set of all subsets of $\mathbb{N}$ of size n.

Our proof proceeds by reducing the following ordinal setting to the *cardinal* setting:

Ordinal setting. There are n vertices, ranked 1 to n according to an unknown ranking. The n vertices arrive in a random order $v_1, \ldots, v_n$. At time t, the algorithm observes the relative rank of v_t among $v_1, \ldots, v_t$, and decides whether to match it to an earlier unmatched vertex. The objective is to maximize the probability of matching together the top two vertices (ranked 1st and 2nd).

Our reduction to the cardinal setting from the ordinal setting goes through a third setting, termed the *hybrid* setting, described as follows:

Hybrid setting. Every vertex v has value λ_v, which is observed upon v's arrival (as in the cardinal settings), but the objective is to maximize the probability to match together the two highest value vertices (as in the ordinal setting).

Our reduction proceeds in three main steps (see Figure 1).

Step 1: Reducing the hybrid variant to the cardinal variant. We show (in Lemma 4.2) that for $\Lambda \in \binom{\mathbb{N}}{n}$, if $\mathsf{ALG}(\Lambda) \geq \alpha \cdot \mathsf{OPT}(\Lambda)$ in the cardinal setting, then ALG matches the top two vertices in Λ with probability $\alpha - O(\frac{1}{n})$.

Step 2: Simplifying competitive algorithms for the hybrid setting. We show (in Claims 4.1 and 4.2) that any α-competitive algorithm for the hybrid setting can be characterized by a collection of n set functions $f_i : \binom{\mathbb{N}}{i} \to [0,1]$ for each $i \in [n]$, where for every $\Lambda_i \in \binom{\mathbb{N}}{i}$, $f_i(\Lambda_i)$ denotes the probability that the algorithm matches the top two vertices in Λ_i given that it is *possible* (i.e., that one of these vertices arrives at time i, and both of them are unmatched).

Step 3: Reducing the ordinal variant to the hybrid variant. For this reduction, we apply the infinite Ramsey theorem (see Claim 4.3); this is similar to the approach taken by [5]. In particular, we find an infinite subset of values $T \subseteq \mathbb{N}$ for which the algorithm's decisions depend only on their ranking, but not on their numerical values; i.e., every function f_i is a constant on each $\Lambda_i \in \binom{T}{i}$ (up to a small error ε). Thus, for every set of values $\Lambda \subset T$ with $|\Lambda| \leq n$, ALG uses only information about the ranking of the values. Theorem 4.3 concludes the reduction, and also reveals useful properties of the optimal algorithm for the ordinal setting.

Finally, we analyze the ordinal algorithms ALG^o, which can be fully characterized by a vector $\vec{c} \in [0,1]^n$, where c_i is the probability ALG^o matches the top two vertices so far at step i, given that it is possible to match them, and establish an upper bound of $5/12 + O(\frac{1}{n})$ on its performance.

4.1 Full Analysis

4.1.1 The Ordinal Variant. The *ordinal* variant of the matching problem is the following:

- A set of n vertices are ranked, according to an unknown ranking, from 1st to nth. The 1st and 2nd vertices are referred to as the top two vertices.
- The vertices arrive sequentially, in a random order; let $v_1, \ldots, v_n$ denote the vertices in their arrival order.
- Upon the arrival of vertex v_t, the algorithm observes the relative rank of v_t among $v_1, \ldots, v_t$ (its rank is in $\{1, \ldots, t\}$), and must decide immediately and irrevocably whether to match it to an earlier unmatched vertex.
- The objective is to maximize the probability of matching together the top two vertices.

In the remainder of this section, we refer to our original secretary matching setting as the *cardinal* setting, and to this variant as the *ordinal* setting. Note that the two settings differ both in (i) the assumption about what is observable (a vertex's weight in the cardinal setting versus its relative rank in the ordinal setting), and in (ii) the objective function (maximize the expected total weight in the cardinal setting versus maximize the probability to match the top two vertices in the ordinal setting). The reduction goes through a third variant, which we refer to as the *hybrid* setting, which shares properties with both variants, as will be explained in Section 4.1.2.

An algorithm in the ordinal and hybrid settings is said to be α-competitive if it matches together the top two vertices with probability at least α.

Ordinal vs. Cardinal Classical Secretary. It is worthwhile to mention that the classical secretary problem has two variants as well: (i) the ordinal secretary problem, where the algorithm observes the relative rank of the arriving element, and aims to maximize the probability of selecting the

best element, and (ii) the cardinal secretary problem, where each element is associated with a value, which is observed upon arrival, and the algorithm aims to maximize the expectd value of the selected element.

It is straightforward to see that any algorithm in the ordinal setting preserves its competitive ratio when applied to the cardinal setting. On the other direction, a folklore result says that the cardinal setting is not easier than the ordinal one (recall that the best competitive ratios in both settings is $\frac{1}{e}$). The upper bound for the cardinal setting is nicely explained in the recent work of [5].

Our ordinal and cardinal variants for matching can be viewed as analogs of the cardinal and ordinal settings in the classic secretary problem. However, the matching setting is more involved in its combinatorial structure, and the multiple decisions that should be made. Indeed, it is not clear to us whether it is possible to adapt an algorithm for the ordinal setting to the cardinal matching setting (like in the classical secretary problem). Nevertheless, we establish a reduction from the ordinal setting to the cardinal setting.

4.1.2 Reduction: From Ordinal to Cardinal (through Hybrid). We shall focus on the following family of instances. An instance is described by a complete graph G on n vertices. Each vertex v of the graph is associated with a value $\lambda_v \in \mathbb{N}$, where $\mathbb{N}$ is the set of positive integers. The weight of each edge uv is determined by the values of the two endpoints: $w_{uv} = n^{3(i+j)}$ where $i = \lambda_u$, $j = \lambda_v$ are the values of its two endpoints. We assume that $\lambda_u \neq \lambda_v$ for every two distinct vertices u, v. Thus, every instance can be specified by a set of values $\Lambda \subset \mathbb{N}$ of size $n = |\Lambda|$ on the graph vertices. Let $\binom{\mathbb{N}}{n}$ denote the set of all such subsets Λ and in general $\binom{T}{i}$ denote the set of all subsets $\Lambda \subset T$ with $|\Lambda| = i$. Suppose the algorithm observes the vertex values directly, rather than the edge weights (this may only give more power to the algorithm). When clear in the context we refer to ij as the edge between vertices with values i and j, and denote the weight of this edge by w_{ij}.

The reduction from the ordinal setting to the cardinal setting proceeds in three main steps; an overview is given in Figure 1. In what follows, we give details for each step.

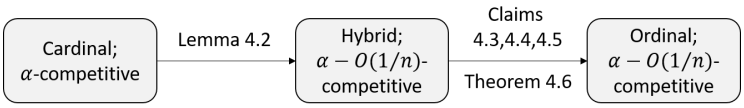

Fig. 1. An overview of the proof steps.

Step 1: Introducing the hybrid variant, and reducing the hybrid variant to the cardinal variant. The *hybrid* variant of the matching problem bridges between the cardinal and ordinal variants. In the hybrid variant, every vertex v is associated with value λ_v, which is observed upon v's arrival (as in the cardinal settings), but the objective is to maximize the probability to match together the two vertices with the highest values (similar to the ordinal setting). The following lemma reduces the hybrid variant of the problem to the cardinal variant.

Lemma 4.2. *Let $\Lambda \in \binom{\mathbb{N}}{n}$. If $ALG(\Lambda) \geq \alpha \cdot OPT(\Lambda)$ in the cardinal setting, then ALG matches the top two vertices in Λ with probability $\alpha - O(\frac{1}{n})$.*

Proof. Let i_1, i_2 be the two largest values of Λ. Consider the performance of the algorithm.

$$\begin{aligned}\mathrm{ALG}(\Lambda) &= \sum_{\{i,j\}\in\binom{\Lambda}{2}} \mathbf{Pr}\left[ij \text{ is selected}\right]\cdot w_{ij} \le \mathbf{Pr}\left[i_1 i_2 \text{ is selected}\right]\cdot w_{i_1 i_2} + \sum_{\{i,j\}\in\binom{\Lambda}{2}-\{i_1,i_2\}} w_{ij} \\ &\le \mathbf{Pr}\left[i_1 i_2 \text{ is selected}\right]\cdot w_{i_1 i_2} + \left(\binom{n}{2}-1\right)\cdot\frac{w_{i_1 i_2}}{n^3} = \left(\mathbf{Pr}\left[i_1 i_2 \text{ is selected}\right] + O\left(\frac{1}{n}\right)\right)\cdot w_{i_1 i_2}, \quad (6)\end{aligned}$$

where the second inequality follows from the fact that for every i, j, $w_{ij} = n^{3(i+j)} \le n^{3(i_1+i_2)-3} = \frac{w_{i_1 i_2}}{n^3}$.

Finally, we have that $\mathrm{ALG}(\Lambda) \ge \alpha\cdot\mathrm{OPT}(\Lambda) \ge \alpha\cdot w_{i_1 i_2}$. Combining this with equation (6) concludes the proof. □

Lemma 4.2 reduces the hybrid setting to the cardinal setting. As such, it allows to change the objective in the cardinal setting to the objective in the ordinal setting. Note, however, that changing only the objective is not sufficient, as the online algorithm in the ordinal setting does not observe the values of the vertices, rather it observes only the relative ranking among them. In the following steps we reduce the ordinal variant to the hybrid variant.

Step 2: Simplifying the description of the algorithm for the hybrid variant. We now show that any α-competitive online algorithm for the hybrid variant admits a simplified description. Claim 4.1 formalizes the (trivial observation) that the algorithm's decision at each step t is simply a binary decision, and Claim 4.2 shows that this decision is "history independent".

Claim 4.1. *Let ALG be any α-competitive online algorithm in the hybrid setting. Then, there is an α-competitive algorithm ALG′ (in the hybrid setting) that in every step t, may (but not necessarily) only match the top two vertices up to step t.*

Proof. If at some step t, ALG matches two vertices u, v which are not the top two vertices up to t, then ALG′ that in such cases doesn't match any vertices, and proceeds as ALG thereafter as if u and v were matched, satisfies $\mathrm{ALG}'(\Lambda) = \mathrm{ALG}(\Lambda)$ for any set of values Λ. □

Thus, the algorithm's decision at time t is just a choice between (at most) two options: match the top two vertices so far, or not match them. However, such decision may still be quite complex: it may be randomized, it may depend on the values of the available vertices, on the arrival order, or on the previous choices of the algorithm. Our next claim simplifies the algorithm further. It shows that an optimal online algorithm is *history-independent*, i.e., the decision depends only on the vertex values and on whether matching the top two vertices is possible.

Claim 4.2. *Given a set of vertex values Λ_t at time t, we may assume without loss of generality that the algorithm's binary decision depends only on*

(1) *whether the top two vertices in Λ_t are not matched yet; and*
(2) *whether the last arriving vertex is one of the top two vertices in Λ_t.* [7]

Proof. To be formal, let us denote the arrival order of the vertices in Λ_t and prior decisions of the algorithm as a history $H(t, \Lambda_t)$. Consider any optimal online algorithm ALG. We are going to modify ALG so that it becomes *history-independent* at time t with the same performance guarantee.

Fix the set of values Λ_t. We consider the set $\mathcal{DH}(t, \Lambda_t)$ of all histories $H(t, \Lambda_t)$ for which the online algorithm has an option to match the two vertices with the highest values in Λ_t, i.e., situations that satisfy both conditions 1 and 2 from the statement of the Claim 4.2. Let us define

[7] Interestingly, it does not matter whether the last arriving vertex is the highest or second-highest, only whether it is one of the top two vertices.

an online algorithm ALG′ that for every observed history $H(t,\Lambda_t) \in \mathcal{DH}(t,\Lambda_t)$ forgets the real history and instead generates a contrafactual history $H'(t,\Lambda_t)$ drawn independently at random from $\mathcal{DH}(t,\Lambda_t)$, i.e., we generate a history $H' \in \mathcal{DH}(t,\Lambda_t)$ with the probability proportional to $\mathbf{Pr}[H(t,\Lambda_t) = H' \mid \Lambda_t]$. We keep ALG′ unchanged for any other set of values Λ_t or history $H \notin \mathcal{DH}(t,\Lambda_t)$. Starting from time t the algorithm ALG′ assumes that the real history was H' (not H) and makes all its future decisions according to H'. Note that by Claim 4.1 ALG′ would be a feasible algorithm. It holds that $\text{ALG}'(H \mid \Lambda_t) = \text{ALG}(H' \mid \Lambda_t)$, since all previous decisions of ALG before time t did not affect the two highest value vertices in Λ_t and, therefore, did not affect the two highest value vertices in Λ. Furthermore, the distribution of the contrafactual histories H' coincides by construction with the distribution of the actual histories $H \in \mathcal{DH}(t,\Lambda_t)$. Thus the expected performance of ALG′ is the same as ALG. Note that ALG′ is history independent for the given set of values Λ_t.

To conclude the proof, we need to apply the above transformation of an optimal algorithm ALG for every possible set of values Λ_t and every time $t \in [n]$. That can be easily done using, e.g., a backward induction on n. □

By Claims 4.1 and 4.2 we may restrict our attention to history-independent, binary-decision online algorithms. Any such algorithm can be characterized by a collection of n set functions $f_i : \binom{\mathbb{N}}{i} \to [0,1]$ for each $i \in [n]$, where for every $\Lambda_i \in \binom{\mathbb{N}}{i}$, $f_i(\Lambda_i)$ denotes the probability that the algorithm matches the top two vertices in Λ_i given that it is *possible*; i.e., that (i) one of these vertices arrives at time i, and (ii) both of them are unmatched.

Step 3: Reducing the ordinal variant to the hybrid variant. To complete the reduction from the ordinal setting to the hybrid setting, we use a similar approach to the upper bound proof from [5]. We fix an α-competitive online algorithm ALG for the hybrid variant, which is represented by a set of functions f_i for every $i \in [n]$.

Our goal is to find an infinite (or sufficiently large) subset $T \subseteq \mathbb{N}$ of values on which the algorithm's decisions do not depend on the actual values of the vertices, i.e., every function f_i is a constant on each $\Lambda_i \in \binom{T}{i}$. In this case ALG does not use any information about the actual values for every set of values $\Lambda \subset T$ with $|\Lambda| \le n$. That is, we can use ALG in the ordinal setting. We achieve this goal within an arbitrary small additive error ε, i.e., $f_i(\Lambda_i) = c_i \pm O(\varepsilon)$ for every $i \in [n]$, and every $\Lambda_i \in \binom{T}{i}$.

Claim 4.3. *For any collection of set functions $f_i : \binom{\mathbb{N}}{i} \to [0,1]$, $i \in [n]$ and any $\varepsilon > 0$ there is an infinite set $T \subset \mathbb{N}$ and constants $c_1, \dots, c_n \in [0,1]$, s.t. $f_i(\Lambda_i) = c_i + O(\varepsilon)$ for all $\Lambda_i \in \binom{T}{i}$, $i \in [n]$.*

Proof. The proof uses the infinite version of Ramsey's theorem. We find such a set T iteratively for $i \in [k]$, starting with $k = 1$ and up to $k = n$. We proceed by induction on k. The base of the induction is the case of $k = 0$, which holds trivially. Suppose, by the induction hypothesis, that we have an infinite set $T_k \subset \mathbb{N}$ and a set of constants $c_1, \dots, c_k$ such that $f_i(\Lambda_i) = c_i + O(\varepsilon)$ for all $\Lambda_i \in \binom{T_k}{i}$ and $i \in [k]$. Our goal is to find an infinite subset $T_{k+1} \subset T_k$ that satisfies the desired condition for f_{k+1}. Consider a complete hyper-graph on the set of vertices T_k with hyper-edges of size $k+1$. Each edge $\Lambda \subset T_k$, $|\Lambda| = k+1$, is colored in one of $1/\varepsilon$ colors: assign color $\lfloor f_{k+1}(\Lambda)/\varepsilon \rfloor$ to the edge Λ. By the infinite version of Ramsey's theorem [42], this hyper-graph admits an infinite monochromatic clique. Let the color of such a clique be C_{k+1}, and let the set of vertices in the clique be $T_{k+1} \subset T_k$. Set the constant $c_{k+1} = \varepsilon \cdot C_{k+1}$. Then $c_{k+1} \le f_{k+1}(\Lambda) < c_{k+1} + \varepsilon$ for any $\Lambda \in \binom{T_{k+1}}{k+1}$, i.e., $f_{k+1}(\Lambda) = c_{k+1} + O(\varepsilon)$ for any $\Lambda \subset T_{k+1}$, $|\Lambda| = k+1$. □

With this, we can conclude the reduction from the ordinal setting to the cardinal setting. As a bonus, we also reveal useful properties of an optimal algorithm in the ordinal setting.

Theorem 4.3. *Let ALG be an α-competitive online algorithm for the cardinal matching setting. Then, there is a $\alpha - O(\frac{1}{n})$-competitive online randomized algorithm ALG^o for the ordinal setting. Moreover, at every time t, ALG^o makes a binary decision on whether to match the two top vertices up to time t, and this decision may only depend on the time t and on whether this matching is possible.*

Proof. We use Lemma 4.2 to convert ALG into an $\alpha - O(\frac{1}{n})$-competitive algorithm $\mathsf{ALG}^{\mathrm{hyb}}$ for the hybrid setting. By Claims 4.1 and 4.2 we can convert $\mathsf{ALG}^{\mathrm{hyb}}$ to $\mathsf{ALG}^{\mathrm{hyb}}_O$ which makes decisions that depend only on the set of values Λ_t at time t and the possibility of matching the top two vertices in Λ_t. Finally, we can choose ε in Claim 4.3 to be $\varepsilon = O(\frac{1}{n^2})$ and find a set $T \subset \mathbb{N}$ such that $f_i(\Lambda_i) = c_i + O(\varepsilon)$ for every $\Lambda_i \in \binom{T}{i}$ and $i \in [n]$. Define the algorithm ALG^o for the ordinal setting as follows: Upon the arrival of the ith vertex, match the two top vertices up to time i with probability c_i, if this is possible. By the union bound, for every set $\Lambda \in \binom{T}{n}$,

$$\mathsf{ALG}^o(\Lambda) = \mathsf{ALG}^{\mathrm{hyb}}_O(\Lambda) + n \cdot O(\varepsilon) = \mathsf{ALG}^{\mathrm{hyb}}(\Lambda) + n \cdot O(\varepsilon) = \mathsf{ALG}^{\mathrm{hyb}}(\Lambda) + O(\frac{1}{n}).$$

Thus, ALG^o is an $\alpha - O(\frac{1}{n})$-competitive online algorithm for the ordinal setting, whose binary decisions depend only on the time t and the possibility of matching the two top vertices in Λ_t at time t. This concludes the proof. □

4.1.3 An Upper Bound for the Ordinal Setting. In this section we study the ordinal setting. Based on the analysis of Section 4.1.2, we restrict ourselves, without loss of generality, to algorithms that decide at each step whether to match the top two vertices so far, and the decision depends only on the time t and whether this matching is possible. Such an algorithm can be fully characterized by a vector $\vec{c} \in [0,1]^n$, where c_i is the probability that the algorithm matches the top two vertices so far at step i, given that it is possible to match them.

Our main theorem in this section is an upper bound of $5/12$ on the competitive ratio of any algorithm in the ordinal setting, whose proof is deferred to Appendix A.3.

Theorem 4.4. *In the ordinal setting, for any $\vec{c} \in [0,1]^n$, the corresponding algorithm matches the top two vertices with probability at most $\frac{5}{12} + O(\frac{1}{n})$.*

5 SECRETARY MATCHING WITH EDGE ARRIVAL

In this section we present an algorithm — Algorithm 2 — that gives a competitive ratio of $1/4$ for edge arrival. Let $e_1, \ldots, e_m$ be the edges in their arrival order. Let $E_t = \{e_1, \ldots, e_t\}$ denote the set of edges that arrived up to time t, and let μ_t^* denote the (unique) maximum weighted matching in $G(E_t)$.

Algorithm 2 ignores the first $\lfloor \frac{m}{2} \rfloor$ edges (exploration phase). Then, in every round t, upon the arrival of edge $e_t = uv$, it computes the probability that both u and v are available (the probability is taken with respect to the random arrival order of the edges E_{t-1} and the random choices of the algorithm in steps 1 to $t-1$); denote this probability by x_t. It then finds the maximum weighted matching μ_t^* in the graph induced by E_t. If $e_t \in \mu_t^*$, the algorithm matches e_t with probability $\frac{\alpha_t}{x_t}$, where α_t is given by the following formula (7):

$$\alpha_t = \begin{cases} 0 & \text{if } t \le \frac{m}{2} \\ 1 - 2\sum_{i=1}^{t-1} \frac{\alpha_i}{i} & \text{if } t > \frac{m}{2} \end{cases} \tag{7}$$

The algorithm ensures that every given edge $e_t = uv$ is matched with a certain probability α_t given by (7), whenever e_t is in the current maximum matching. This allows us to conveniently estimate the expected contribution of the maximum matching at time t and compare it to the maximum matching in the whole graph. Before doing that, we need to prove that the algorithm is well defined,

ALGORITHM 2: 1/4-competitive algorithm for secretary matching with edge arrival

Let $e_1, \ldots, e_m$ be the edges given in their arrival order;
$A = V, \quad \mu = \emptyset;$ ▷ A is the set of available vertices, μ is the returned matching
for $t = \lfloor \frac{m}{2} \rfloor + 1$ *to* m **do**
  Let $e_t = uv$ be the edge arriving at time t;
  $x_t \leftarrow \mathbf{Pr}[u, v \in A]$ (i.e., e_t is available); ▷ random: order of $e_1, ..., e_{t-1}$+ alg's choices
  $\mu_t^* \leftarrow$ the maximum weighted matching in $G(E_t)$;
  if $e_t \in \mu_t^*$ *and* $u, v \in A$ **then**
    begin with probability $\frac{\alpha_t}{x_t}$
      Add e_t to μ;
      Remove u and v from A;
return *matching* μ

i.e., that $x_t \geq \alpha_t$ for every t. Note that x_t is the probability that e_t is available, given a random order of edges in E_t (the edges arriving up to time t) that ends with e_t. As such, it depends on e_t, and the set E_t, but *not* on the order of edges within E_t (except for e_t being the t-th edge). We use $x_t(S, e)$ to denote the probability that e_t is available when $E_t = S$ and $e_t = e$. We use "matched$(u, < t)$" to denote the event that u becomes unavailable before time t, and "matched$(e, @t)$" to denote the event that edge e is matched exactly at time t. We use A_t to denote the random set of available vertices at time t.

Lemma 5.1. *For every time t, vertices u, v and set of edges Q of size $t-1$, given that $e_t = uv$ and $E_{t-1} = Q$, it holds that $x_t \geq \alpha_t$.*

Proof. We prove by induction on t. For the base case, where $t = \lfloor \frac{m}{2} \rfloor + 1$, the statement holds trivially since we have not matched anything yet, i.e., $x_t = 1$. Next, fix t and suppose the statement holds for all $t' \leq t-1$. Let $e_t = uv$ be the edge arriving at time t and the let $Q = E_{t-1}$ be the set of arrived edges before time t.

For simplicity, in the remainder of the proof, we omit the given $e_t = uv$ and $E_{t-1} = Q$ in all probabilities and indicator expressions. It holds that

$$\mathbf{Pr}\left[\text{matched}(u, < t)\right] = \sum_{\substack{e \ni u, \\ e \in Q}} \sum_{i=1}^{t-1} \mathbf{Pr}\left[e = e_i\right] \cdot \mathbf{Pr}\left[\text{matched}(e, @i) \mid e = e_i\right],$$

Clearly, $\mathbf{Pr}[e = e_i] = \frac{1}{t-1}$ due to the random arrival order. To calculate $\mathbf{Pr}[\text{matched}(e, @i) \mid e = e_i]$ for a given $e = (uv')$ and any given set $S = E_i \ni e$ of arrived edges at step $i < t$, note that by the induction hypothesis, $x_i \geq \alpha_i$, and whenever $e \in \mu^*(S)$, Algorithm 2 includes e in the matching with probability α_i precisely. We get

$$\begin{aligned}
\mathbf{Pr}\left[\text{matched}(e, @i) \mid e = e_i\right] &= \sum_{\substack{S \subseteq Q: \\ |S|=i,\ e \in S}} \mathbf{Pr}\left[E_i = S \mid e = e_i\right] \cdot \mathbb{1}\left[e \in \mu^*(S)\right] \cdot x_i(S, e) \cdot \frac{\alpha_i}{x_i(S, e)} \\
&= \alpha_i \cdot \sum_{\substack{S \subseteq Q: \\ |S|=i,\ e \in S}} \mathbf{Pr}\left[E_i = S \mid e = e_i\right] \cdot \mathbb{1}\left[e \in \mu^*(S)\right] = \alpha_i \cdot \mathbf{Pr}\left[e \in \mu_i^* \mid e = e_i\right].
\end{aligned}$$

It follows that

$$\begin{aligned}\mathbf{Pr}\left[\text{matched}(u,<t)\right] &= \sum_{\substack{e\ni u,\\ e\in Q}}\sum_{i=1}^{t-1}\frac{1}{t-1}\cdot\mathbf{Pr}\left[e\in\mu_i^*\mid e=e_i\right]\cdot\alpha_i \\ &= \frac{1}{t-1}\sum_{i=1}^{t-1}\alpha_i\sum_{\substack{e\ni u,\\ e\in Q}}\mathbf{Pr}\left[e\in\mu_i^*\mid e=e_i\right], \end{aligned} \tag{8}$$

where the last equality is obtained by changing the order of summation. We next prove that

$$\sum_{\substack{e\ni u,\\ e\in Q}}\mathbf{Pr}\left[e\in\mu_i^*\mid e=e_i\right]\le\frac{t-1}{i}. \tag{9}$$

It holds that

$$\sum_{\substack{e\ni u,\\ e\in Q}}\mathbf{Pr}\left[e\in\mu_i^*\mid e=e_i\right]=\sum_{\substack{e\ni u,\\ e\in Q}}\sum_{\substack{S\subseteq Q:\\ e\in S,|S|=i}}\mathbb{1}[e\in\mu_i^*\mid e=e_i,S=E_i]\cdot\mathbf{Pr}\left[S=E_i\mid e=e_i\right].$$

Since the arrival order is chosen uniformly at random, $\forall S\subset Q:|S|=i$ we have $\mathbf{Pr}[S=E_i\mid e=e_i]=\frac{1}{\binom{t-2}{i-1}}$, which can be written as $\frac{1}{\binom{t-1}{i}}\cdot\frac{t-1}{i}$. By changing the order of summation we get

$$\begin{aligned}\sum_{\substack{e\ni u,\\ e\in Q}}\mathbf{Pr}\left[e\in\mu_i^*\mid e=e_i\right] &= \sum_{\substack{S\subseteq Q:\\ |S|=i}}\sum_{\substack{e\ni u,\\ e\in S}}\mathbb{1}[e\in\mu_i^*\mid S=E_i]\cdot\frac{1}{\binom{t-1}{i}}\cdot\frac{t-1}{i} \\ &= \frac{t-1}{i}\sum_{\substack{S\subseteq Q:\\ |S|=i}}\sum_{\substack{e\ni u,\\ e\in S}}\mathbb{1}[e\in\mu_i^*\mid S=E_i]\cdot\mathbf{Pr}\left[S=E_i\right]\le\frac{t-1}{i}\sum_{\substack{S\subseteq Q:\\ |S|=i}}\mathbf{Pr}\left[S=E_i\right]=\frac{t-1}{i}.\end{aligned}$$

The second equality holds since $\mathbf{Pr}[S=E_i]=\frac{1}{\binom{t-1}{i}}$, and the inequality follows by observing that $\sum_{e\ni u,e\in S}\mathbb{1}[e\in\mu_i^*\mid S=E_i]\le 1$ since the events $e\in\mu_i^*$ are disjoint for different $e\ni u$. The last equality follows since $\sum_{S\subseteq Q:|S|=i}\mathbf{Pr}[S=E_i]=1$, as a partition into all possible realizations of E_i. This concludes the proof of Equation (9). We combine (8) and (9) and get the following probability bound that u (and similarly v) is matched before time t:

$$\mathbf{Pr}\left[\text{matched}(u,<t)\right]\le\sum_{i=1}^{t-1}\frac{\alpha_i}{i}\quad,\quad\mathbf{Pr}\left[\text{matched}(v,<t)\right]\le\sum_{i=1}^{t-1}\frac{\alpha_i}{i}$$

Applying the union bound to the events matched$(u,<t)$ and matched$(v,<t)$ the probability that both u and v are available upon the arrival of e_t is

$$x_t=\mathbf{Pr}\left[u,v\text{ available }@t\right]\ge 1-2\sum_{i=1}^{t-1}\frac{\alpha_i}{i}\overset{(7)}{=}\alpha_t.$$

This concludes the proof of Lemma 5.1. □

We are now ready to prove that Algorithm 2 is $\frac{1}{4}$-competitive.

Theorem 5.2. *Algorithm 2 has a competitive ratio of $\frac{1}{4}$.*

Proof. First, as the set of the first t edges E_t is chosen uniformly at random, the maximum matching μ_t^* in E_t has the expected total weight greater than or equal to the expected weight of the E_t edges in the optimal matching μ^*

$$\mathbf{E}\left[\sum_{e\in\mu_t^*} w_e\right] \geq \frac{t}{m}\cdot\sum_{e\in\mu^*} w_e. \tag{10}$$

Next, we prove by induction that for every $t > \frac{m}{2}$:

$$\alpha_t = \frac{\lfloor\frac{m}{2}\rfloor\cdot\lfloor\frac{m-2}{2}\rfloor}{(t-1)(t-2)}. \tag{11}$$

For $t = \lfloor\frac{m}{2}\rfloor + 1$, α_t is indeed 1. For $t > \lfloor\frac{m}{2}\rfloor + 1$,

$$\alpha_t = 1 - 2\sum_{i=1}^{t-1}\frac{\alpha_i}{i} = \alpha_{t-1} - 2\cdot\frac{\alpha_{t-1}}{t-1} = \frac{t-3}{t-1}\cdot\alpha_{t-1}.$$

The induction hypothesis now implies that $\frac{t-3}{t-1}\cdot\alpha_{t-1} = \frac{t-3}{t-1}\cdot\frac{\lfloor\frac{m}{2}\rfloor\cdot\lfloor\frac{m-2}{2}\rfloor}{(t-2)(t-3)} = \frac{\lfloor\frac{m}{2}\rfloor\cdot\lfloor\frac{m-2}{2}\rfloor}{(t-1)(t-2)}$, concluding (11).

We are now ready to establish the competitive ratio of Algorithm 2. Write μ for the matching returned by Algorithm 2. Edge $e_t = uv$ is matched in round t if: (i) it belongs to μ_t^*, and (ii) both u and v are available. Observing that condition (ii) happens with probability $x_t(S,e)$ for $S = E_t$, we get

$$\mathbf{E}\left[\sum_{e\in\mu} w_e\right] = \sum_{t=\lfloor\frac{m}{2}\rfloor+1}^{m}\sum_{\substack{S\subseteq E:\\|S|=t}}\sum_{e\in S}\mathbf{E}\left[w_e\cdot\mathbb{1}[e\in\mu_t^*] \mid e=e_t, E_t=S\right]\cdot\mathbf{Pr}\left[E_t=S, e=e_t\right]\cdot x_t(S,e)\cdot\frac{\alpha_t}{x_t(S,e)}.$$

Observe that $\sum_{e\in S}\mathbf{E}[w_e\cdot\mathbb{1}[e\in\mu_t^*] \mid e=e_t, E_t=S] = \mathbf{E}[\sum_{e\in\mu_t^*} w_e \mid E_t=S]$ and also that $\mathbf{Pr}[E_t=S, e=e_t] = \mathbf{Pr}[E_t=S]\cdot\frac{1}{t}$. We get

$$\begin{aligned}
\mathbf{E}\left[\sum_{e\in\mu} w_e\right] &= \sum_{t=\lfloor\frac{m}{2}\rfloor+1}^{m}\sum_{\substack{S\subseteq E:\\|S|=t}}\mathbf{E}\left[\sum_{e\in\mu_t^*} w_e \,\middle|\, E_t=S\right]\cdot\mathbf{Pr}\left[E_t=S\right]\cdot\frac{1}{t}\cdot\alpha_t \\
&= \sum_{t=\lfloor\frac{m}{2}\rfloor+1}^{m}\mathbf{E}\left[\sum_{e\in\mu_t^*} w_e\right]\cdot\frac{\alpha_t}{t} \overset{(11)}{=} \sum_{t=\lfloor\frac{m}{2}\rfloor+1}^{m}\mathbf{E}\left[\sum_{e\in\mu_t^*} w_e\right]\cdot\frac{1}{t}\cdot\frac{\lfloor\frac{m}{2}\rfloor\cdot\lfloor\frac{m-2}{2}\rfloor}{(t-1)(t-2)} \\
&\overset{(10)}{\geq} \sum_{t=\lfloor\frac{m}{2}\rfloor+1}^{m}\frac{t\cdot\sum_{e\in\mu^*} w_e}{m}\cdot\frac{1}{t}\cdot\frac{\lfloor\frac{m}{2}\rfloor\cdot\lfloor\frac{m-2}{2}\rfloor}{(t-1)(t-2)} \\
&= \left\lfloor\frac{m}{2}\right\rfloor\cdot\left\lfloor\frac{m-2}{2}\right\rfloor\cdot\frac{\sum_{e\in\mu^*} w_e}{m}\cdot\sum_{t=\lfloor\frac{m}{2}\rfloor+1}^{m}\frac{1}{(t-1)(t-2)} \geq \frac{1}{4}\sum_{e\in\mu^*} w_e
\end{aligned} \tag{12}$$

The last inequality follows by observing that $\frac{1}{(t-1)(t-2)} = \frac{1}{t-2} - \frac{1}{t-1}$, thus the sum telescopes to $\frac{1}{\lfloor\frac{m}{2}\rfloor-1} - \frac{1}{m-1}$; we have $\lfloor\frac{m}{2}\rfloor\cdot\lfloor\frac{m-2}{2}\rfloor\cdot\frac{1}{m}\left(\frac{1}{\lfloor\frac{m}{2}\rfloor-1} - \frac{1}{m-1}\right) = \frac{\lfloor\frac{m}{2}\rfloor}{m}\left(1 - \frac{\lfloor\frac{m-2}{2}\rfloor}{m-1}\right) > \frac{1}{2}\left(1-\frac{1}{2}\right) = 1/4$ (where the inequality holds for any $m \geq 2$). I.e., the coefficient of $\sum_{e\in\mu^*} w_e$ is at least $\frac{1}{4}$. This concludes the proof of the theorem. □

Remark: It is quite straightforward to implement Algorithm 2 in *exponential* time in the number of edges m using Monte Carlo simulations: one simply needs to compute all $x_t = x_t(S)$ depending on the set of visible edges $S \subset E$ (i.e., edges with known weights). We can easily compute all α_t using the simple explicit formula (11). Unfortunately, we do not know how to efficiently compute or estimate x_t in subexponential time (to estimate the probability that both ends of e_t are available, we need to predict at each step $i < t$ what Algorithm 2 would do for a random set of edges $S \subset E_t$). Thus, our result in Theorem 5.2 can be seen as an information-theoretic result. It remains an interesting open problem whether there is a poly-time online algorithm that matches this bound of $1/4$.

ACKNOWLEDGMENTS

This work is supported by Science and Technology Innovation 2030 –"New Generation of Artificial Intelligence" Major Project No.(2018AAA0100903), Innovation Program of Shanghai Municipal Education Commission, Program for Innovative Research Team of Shanghai University of Finance and Economics (IRTSHUFE) and the Fundamental Research Funds for the Central Universities. The first two authors are partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 866132), and by the Israel Science Foundation (grant number 317/17). Zhihao Gavin Tang is supported by NSFC grant 61902233. Nick Gravin is supported by NSFC grant 62150610500. Tomer Ezra is partially supported by the ERC Advanced Grant 788893 AMDROMA "Algorithmic and Mechanism Design Research in Online Markets" and the MIUR PRIN project ALGADIMAR "Algorithms, Games, and Digital Markets".

REFERENCES

[1] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. 2011. Online Vertex-Weighted Bipartite Matching and Single-bid Budgeted Allocations. In *SODA*. SIAM, 1253–1264.

[2] Noga Alon, Tristan Pollner, and S. Matthew Weinberg. 2020. Three Results on Prophet Inequalities on (Hyper-) Graphs. Personal communication.

[3] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. 2019. Edge Weighted Online Windowed Matching. In *EC*. ACM, 729–742.

[4] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. 2018. Matroid Secretary Problems. *J. ACM* 65, 6 (2018), 35:1–35:26.

[5] José Correa, Paul Dütting, Felix Fischer, and Kevin Schewior. 2019. Prophet inequalities for iid random variables from an unknown distribution. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC*. ACM, 3–17.

[6] Nedialko B. Dimitrov and C. Greg Plaxton. 2012. Competitive Weighted Matching in Transversal Matroids. *Algorithmica* 62, 1-2 (2012), 333–348.

[7] Shaddin Dughmi. 2020. The Outer Limits of Contention Resolution on Matroids and Connections to the Secretary Problem. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference) (LIPIcs)*, Artur Czumaj, Anuj Dawar, and Emanuela Merelli (Eds.), Vol. 168. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 42:1–42:18. https://doi.org/10.4230/LIPIcs.ICALP.2020.42

[8] E. Dynkin. 1963. The optimum choice of the instant for stopping a markov process.

[9] Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. 2020. Online Stochastic Max-Weight Matching: Prophet Inequality for Vertex and Edge Arrival Models. In *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, Péter Biró, Jason Hartline, Michael Ostrovsky, and Ariel D. Procaccia (Eds.). ACM, 769–787. https://doi.org/10.1145/3391403.3399513

[10] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. 2020. Edge-Weighted Online Bipartite Matching. In *FOCS*. IEEE, 412–423.

[11] Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. 2009. Online Ad Assignment with Free Disposal. In *WINE (Lecture Notes in Computer Science)*, Vol. 5929. Springer, 374–385.

[12] Michal Feldman, Nick Gravin, and Brendan Lucier. 2015. Combinatorial Auctions via Posted Prices. In *SODA*. SIAM, 123–135.

[13] Moran Feldman, Ola Svensson, and Rico Zenklusen. 2018. A Simple $O(\log \log(\text{rank}))$-Competitive Algorithm for the Matroid Secretary Problem. *Math. Oper. Res.* 43, 2 (2018), 638–650.

[14] Thomas S. Ferguson. 1989. Who Solved the Secretary Problem? *Statist. Sci.* 4, 3 (08 1989), 282–289. https://doi.org/10.1214/ss/1177012493

[15] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. 2019. Online Matching with General Arrivals. In *FOCS*. IEEE Computer Society, 26–37.

[16] Martin Gardner. 1966. *New Mathematical Diversions from Scientific American.* Simon and Schuster, Chapter 3, problem 3. Reprint of the original column published in February 1960 with additional comments.

[17] Alexander V. Gnedin. 1994. A Solution to the Game of Googol. *Annals of Probability* 22, 3 (July 1994), 1588–1595.

[18] Nick Gravin, Zhihao Gavin Tang, and Kangning Wang. 2021. Online Stochastic Matching with Edge Arrivals. In *ICALP (LIPIcs)*, Vol. 198. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 74:1–74:20.

[19] Nikolai Gravin and Hongao Wang. 2019. Prophet Inequality for Bipartite Matching: Merits of Being Simple and Non Adaptive. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC*. 93–109.

[20] Guru Prashanth Guruganesh and Sahil Singla. 2017. Online Matroid Intersection: Beating Half for Random Arrival. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings (Lecture Notes in Computer Science)*, Friedrich Eisenbrand and Jochen Könemann (Eds.), Vol. 10328. Springer, 241–253.

[21] Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Tuomas Sandholm. 2007. Automated Online Mechanism Design and Prophet Inequalities. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. 58–65.

[22] Martin Hoefer and Bojana Kodric. 2017. Combinatorial Secretary Problems with Ordinal Information. In *ICALP (LIPIcs)*, Vol. 80. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 133:1–133:14.

[23] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. 2020. Fully Online Matching. *J. ACM* 67, 3 (2020), 17:1–17:25.

[24] Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. 2019. Tight Competitive Ratios of Classic Matching Algorithms in the Fully Online Model. In *SODA*. SIAM, 2875–2886.

[25] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. 2019. Online Vertex-Weighted Bipartite Matching: Beating 1-1/e with Random Arrivals. *ACM Trans. Algorithms* 15, 3 (2019), 38:1–38:15.

[26] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. 2020. Fully Online Matching II: Beating Ranking and Water-filling. *to appear in FOCS* (2020).

[27] Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. 2020. AdWords in a Panorama. In *FOCS*. IEEE, 1416–1426.

[28] Billy Jin and David P. Williamson. 2020. Improved Analysis of RANKING for Online Vertex-Weighted Bipartite Matching. *CoRR* abs/2007.12823 (2020).

[29] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. 2011. Online bipartite matching with unknown distributions. In *STOC*. ACM, 587–596.

[30] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. 1990. An Optimal Algorithm for On-line Bipartite Matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, STOC*. ACM, 352–358.

[31] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. 2013. An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions. In *ESA (Lecture Notes in Computer Science)*, Vol. 8125. Springer, 589–600.

[32] Robert Kleinberg and S. Matthew Weinberg. 2019. Matroid prophet inequalities and applications to multi-dimensional mechanism design. *Games and Economic Behavior* 113 (2019), 97–115.

[33] Robert D. Kleinberg. 2005. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*. SIAM, 630–631. http://dl.acm.org/citation.cfm?id=1070432.1070519

[34] Nitish Korula and Martin Pál. 2009. Algorithms for Secretary Problems on Graphs and Hypergraphs. In *ICALP (2) (Lecture Notes in Computer Science)*, Vol. 5556. Springer, 508–520.

[35] Ulrich Krengel and Louis Sucheston. 1977. Semiamarts and finite values. *Bull. Amer. Math. Soc.* 83, 4 (07 1977), 745–747.

[36] Ulrich Krengel and Louis Sucheston. 1978. On semiamarts, amarts, and processes with finite value. *Advances in Prob* 4, 197-266 (1978), 1–5.

[37] Oded Lachish. 2014. O(log log Rank) Competitive Ratio for the Matroid Secretary Problem. In *FOCS*. IEEE Computer Society, 326–335.

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

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

[40] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. 2007. AdWords and generalized online matching. *J. ACM* 54, 5 (2007), 22.

[41] Tristan Pollner. 2020. Two Problems In Combinatorial Optimization Under Uncertainty.
[42] Frank P Ramsey. 2009. On a problem of formal logic. In *Classic Papers in Combinatorics*. Springer, 1–24.
[43] Rebecca Reiffenhäuser. 2019. An Optimal Truthful Mechanism for the Online Weighted Bipartite Matching Problem. In *SODA*. SIAM, 1982–1993.
[44] Aviad Rubinstein. 2016. Beyond matroids: Secretary problem and prophet inequality with general constraints. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 324–332.
[45] Ester Samuel-Cahn et al. 1984. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *the Annals of Probability* 12, 4 (1984), 1213–1216.
[46] José A. Soto, Abner Turkieltaub, and Victor Verdugo. 2018. Strong Algorithms for the Ordinal Matroid Secretary Problem. In *SODA*. SIAM, 715–734.
[47] Yajun Wang and Sam Chiu-wai Wong. 2015. Two-sided Online Bipartite Matching and Vertex Cover: Beating the Greedy Algorithm. In *ICALP (1) (Lecture Notes in Computer Science)*, Vol. 9134. Springer, 1070–1081.

A MISSING PROOFS

A.1 Proof of Lemma 3.2

Lemma 3.2. *For every $k \geq 3$, $t \geq k$, every possible realization $\widetilde{V}$ of V_t (i.e., $\widetilde{V} \subseteq V, |\widetilde{V}| = t$), and every vertex $u \in \widetilde{V}$, it holds that*

$$\mathbf{Pr}\left[\mathit{matched}(u, \leq t) \mid V_t = \widetilde{V}\right] = \frac{2}{3}\left(1 - \frac{(t-3)! \cdot k!}{t! \cdot (k-3)!}\right). \tag{1}$$

Proof. To prove the lemma, we show that the probability that u is matched by time t, conditioned on $V_t = \widetilde{V}$ can be expressed by the following recursive formula:

$$\mathbf{Pr}\left[\text{matched}(u, \leq t) \mid V_t = \widetilde{V}\right] = p(k, t), \tag{13}$$

where

$$p(k,k) = 0 \quad \text{and} \quad p(k,t) = \frac{2}{t} + \frac{t-3}{t} \cdot p(k, t-1) \text{ for every } t \in \{k+1, \ldots, n\}. \tag{14}$$

We prove (13) by induction on t. For $t = k$, $p(k,k) = 0$ and (13) holds trivially. Consider next the case where $t > k$, and $V_t = \widetilde{V}$. For every set $T \subseteq V$ of even size, μ_T denotes the unique maximum matching of T (recall that the maximum matching is unique and matches all vertices) and for $u \in T$, $\mu_T(u)$ denotes the match of u in the μ_T. We distinguish between two cases, namely whether t is even or odd; in both cases V'_t is even.

Case 1: t is even. In this case $V'_t = V_t$. We partition the event that u is matched by time t, given that $V_t = \widetilde{V}$, into the following disjoint events: (i) $v_t = u$, (ii) $v_t = v$ for some $v \in \widetilde{V} \setminus \{u, \mu_{\widetilde{V}}(u)\}$, and (iii) $v_t = \mu_{\widetilde{V}}(u)$. Each one of these events occurs with probability $\frac{1}{t}$ (in (ii), $\frac{1}{t}$ is the probability of every given $v \in \widetilde{V} \setminus \{u, \mu_{\widetilde{V}}(u)\}$). We get:

$$\begin{aligned}
\mathbf{Pr}\left[\text{matched}(u, \leq t) \mid V_t = \widetilde{V}\right] &= \mathbf{Pr}\left[\text{matched}(u, \leq t) \mid V_t = \widetilde{V}, v_t = u\right] \frac{1}{t} \\
&+ \sum_{v \in \widetilde{V} \setminus \{u, \mu_{\widetilde{V}}(u)\}} \mathbf{Pr}\left[\text{matched}(u, \leq t) \mid V_t = \widetilde{V}, v_t = v\right] \frac{1}{t} \\
&+ \mathbf{Pr}\left[\text{matched}(u, \leq t) \mid V_t = \widetilde{V}, v_t = \mu_{\widetilde{V}}(u)\right] \frac{1}{t}.
\end{aligned}$$

If $v_t = u$, then u is matched by time t, iff $\mu_{\widetilde{V}}(u)$ is unmatched before u's arrival, which happens with probability $1 - p(k, t-1)$ by induction hypothesis for $V_{t-1} = \widetilde{V} \setminus \{u\}$. If v_t is neither u nor $\mu_{\widetilde{V}}(u)$, then u is matched by time t iff it is matched by time $t-1$. Finally, if $v_t = \mu_{\widetilde{V}}(u)$, then u is always

matched by t, since if it is unmatched before time t, it will be matched to $\mu_{\widetilde{V}}(u)$ upon arrival of $v_t = \mu_{\widetilde{V}}(u)$. Putting it all together we get:

$$\begin{aligned}\mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}\right] &= (1 - p(k, t-1))\frac{1}{t} + (t-2)p(k, t-1)\frac{1}{t} + \frac{1}{t} \\ &= \frac{2}{t} + \frac{t-3}{t} \cdot p(k, t-1) \\ &\overset{(14)}{=} p(k, t).\end{aligned}$$

Case 2: t is odd. Let r_t be the index of the random vertex that is dropped in line 7 of the algorithm. Then, $V'_t = \widetilde{V} \setminus \{v_{r_t}\}$. We partition the event that u is matched by time t, given that $V_t = \widetilde{V}$, into the following disjoint events: (i) $u = v_t$, (ii) $u = v_{r_t}$, and (iii) $u \neq v_t, v_{r_t}$. Each of the events (i) and (ii) occurs with probability $\frac{1}{t}$; event (iii) occurs with probability $\frac{t-2}{t}$. We get:

$$\begin{aligned}\mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}\right] &= \mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}, u = v_t\right]\frac{1}{t} \\ &+ \mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}, u = v_{r_t}\right]\frac{1}{t} \\ &+ \mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}, u \neq v_t, v_{r_t}\right]\frac{t-2}{t}.\end{aligned}$$

If $u = v_t$, then u is matched iff its match is available in round t, which happens with probability $1 - p(k, t-1)$, by induction. If $u = v_{r_t}$, then u is matched by time t iff it is matched by time $t-1$, which happens with probability $p(k, t-1)$ by the induction hypothesis for $V_{t-1} = \widetilde{V} \setminus \{v_t\}$. If $u \neq v_t, v_{r_t}$, then v_t, v_{r_t} are uniformly distributed among the pairs of vertices in $\widetilde{V} \setminus \{u\}$. To calculate the probability that u is matched by time t we separate the latter case into two disjoint events: (i) $\mu_t(v_t) = u$, in which case u is matched with probability 1; and (ii) $\mu_t(v_t) \neq u$, in which case u is matched only if it was matched before time t, which is $p(k, t-1)$ by induction. Thus,

$$\begin{aligned}\mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}, u \neq v_t, v_{r_t}\right] &= \mathbf{Pr}\left[\mu_t(v_t) = u \mid V_t = \widetilde{V}, u \neq v_t, v_{r_t}\right] \cdot 1 \\ &+ \mathbf{Pr}\left[\mu_t(v_t) \neq u \mid V_t = \widetilde{V}, u \neq v_t, v_{r_t}\right] \cdot p(k, t-1) \\ &= \frac{1}{t-2} \cdot 1 + \frac{t-3}{t-2}p(k, t-1).\end{aligned}$$

Putting it all together we get:

$$\begin{aligned}\mathbf{Pr}\left[\text{matched}(u, \le t) \mid V_t = \widetilde{V}\right] &= (1 - p(k, t-1))\frac{1}{t} + p(k, t-1)\frac{1}{t} \\ &+ \frac{t-2}{t} \cdot \left(\frac{1}{t-2} + \frac{t-3}{t-2} \cdot p(k, t-1)\right) = \frac{2}{t} + \frac{t-3}{t}p(k, t-1) = p(k, t).\end{aligned}$$

This concludes the proof of Equation (13).

It remains to solve the recursion. We prove by induction that

$$p(k, t) = \frac{2}{3}\left(1 - \frac{(t-3)! \cdot k!}{t! \cdot (k-3)!}\right). \tag{15}$$

For $t = k$ this holds trivially, since $p(k,k) = 0$. For $t > k$, suppose (15) holds for $t-1$; then,

$$\begin{aligned} p(k,t) &\overset{(14)}{=} \frac{2}{t} + \frac{t-3}{t} \cdot p(k,t-1) \\ &\overset{(15)}{=} \frac{2}{t} + \frac{t-3}{t} \cdot \frac{2}{3}\left(1 - \frac{(t-4)! \cdot k!}{(t-1)! \cdot (k-3)!}\right) \\ &= \frac{2}{3}\left(\frac{3}{t} + \frac{t-3}{t} \cdot \left(1 - \frac{(t-4)! \cdot k!}{(t-1)! \cdot (k-3)!}\right)\right) \\ &= \frac{2}{3}\left(1 - \frac{(t-3)! \cdot k!}{t! \cdot (k-3)!}\right), \end{aligned}$$

where the second equality holds by the induction assumption. This concludes the proof of Lemma 3.2. □

A.2 Missing Derivation from the Proof of Theorem 3.1

$$\begin{aligned} \frac{\mathbf{E}[w(\mu)]}{\mathbf{E}[w(\mu^*)]} &\geq \frac{1}{\mathbf{E}[w(\mu^*)]} \sum_{t=k+1}^{n} (1 - p(k,t-1)) \cdot \frac{4 \cdot \lfloor t/2 \rfloor - 2}{n \cdot (n-1)} \cdot \mathbf{E}\left[w(\mu^*)\right] \\ &\overset{(15)}{=} \sum_{t=k+1}^{n} \left(1 - \frac{2}{3}\left(1 - \frac{(t-4)! \cdot k!}{(t-1)! \cdot (k-3)!}\right)\right) \cdot \frac{4 \cdot \lfloor t/2 \rfloor - 2}{n \cdot (n-1)} \\ &\geq \sum_{t=k+1}^{n} \left(\frac{1}{3} + \frac{2 \cdot (t-4)! \cdot k!}{3 \cdot (t-1)! \cdot (k-3)!}\right) \cdot \frac{2t-4}{n \cdot (n-1)} \\ &\geq \sum_{t=k+1}^{n} \left(\frac{1}{3} + \frac{2(k-2)^3}{3 \cdot t^3}\right) \cdot \frac{2t-4}{n^2} \\ &\geq \frac{1}{n^2} \int_{k}^{n} \left(\frac{1}{3} + \frac{2(k-2)^3}{3 \cdot (t+1)^3}\right) \cdot (2t-4) \mathrm{d}\, t \\ &= \frac{1}{n^2} \cdot \frac{1}{3}\left(-\frac{4(k-2)^3}{t+1} + \frac{6(k-2)^3}{(t+1)^2} + (t+1)^2 - 6t\right)\Bigg|_{k}^{n} \\ &= -\frac{4k^3}{3n^3} + \frac{4k^2}{3n^2} + \frac{1}{3} - \frac{k^2}{3n^2} - O\left(\frac{1}{n}\right) \\ &= \frac{1}{3} + \frac{k^2}{n^2} - \frac{4k^3}{3n^3} - O\left(\frac{1}{n}\right), \end{aligned}$$

where to get the first inequality, we used the bound $\lfloor t/2 \rfloor \geq t/2 - 1/2$; to get the second inequality, we applied basic algebraic transformations to simplify each term under the summation; to get the third inequality, we estimated the integral $\int_{t=k}^{t=n}$ as a Riemann sum with the subdivision into equal intervals of length 1 and used a simple upper bound on the function's value in each subdivision interval; in the last two equalities all low-order terms are contained in the $O(\frac{1}{n})$ term (that vanishes for $n \to \infty$ and $k = \Theta(n)$).

A.3 Proof of Theorem 4.4

Theorem 4.4. *In the ordinal setting, for any $\vec{c} \in [0,1]^n$, the corresponding algorithm matches the top two vertices with probability at most $\frac{5}{12} + O(\frac{1}{n})$.*

Proof. Let v_i denote the vertex that arrives at time i, and let $\mathrm{I}_i, \mathrm{II}_i$ denote the respective top and second-top vertices among $v_1, \ldots, v_i$. Let O_i be the event that I_i remains unmatched by the end of step i, and let $p_i = \mathbf{Pr}[O_i]$, where the randomness is taken over the arrival order of the first i vertices. Clearly, $p_1 = 1$. In what follows, "matched$(e, @t)$" denotes the event that edge e is matched exactly at time t. For $i > 1$, we can express p_i by the following recursive formula:

$$\begin{aligned}
p_i =& \mathbf{Pr}\left[O_i \wedge (v_i = \mathrm{I}_i)\right] + \mathbf{Pr}\left[O_i \wedge (v_i = \mathrm{II}_i)\right] + \mathbf{Pr}\left[O_i \wedge (v_i \neq \mathrm{I}_i, \mathrm{II}_i)\right] \\
=& \frac{1}{i} \cdot \mathbf{Pr}\left[O_i \middle| v_i = \mathrm{I}_i\right] + \frac{1}{i} \cdot \mathbf{Pr}\left[O_i \middle| v_i = \mathrm{II}_i\right] + \frac{i-2}{i} \cdot \mathbf{Pr}\left[O_i \middle| v_i \neq \mathrm{I}_i, \mathrm{II}_i\right] \\
=& \frac{1}{i} \cdot \mathbf{Pr}\left[\bar{O_{i-1}} \vee \overline{\text{matched}(v_i \mathrm{I}_{i-1}, @i)} \;\middle|\; v_i = \mathrm{I}_i\right] + \frac{1}{i} \cdot \mathbf{Pr}\left[O_{i-1} \wedge \overline{\text{matched}(v_i \mathrm{I}_{i-1}, @i)} \;\middle|\; v_i = \mathrm{II}_i\right] \\
& + \frac{i-2}{i} \cdot \mathbf{Pr}\left[O_{i-1}\right] \\
=& \frac{1}{i} \cdot (1 - p_{i-1} c_i) + \frac{1}{i} \cdot p_{i-1}(1 - c_i) + \frac{i-2}{i} \cdot p_{i-1} \\
=& \frac{1}{i} \cdot (1 + (i-1)p_{i-1} - 2p_{i-1}c_i). \qquad (16)
\end{aligned}$$

The first equality follows by considering three disjoint cases for vertex v_i; the second equality holds since the distribution over the arrival order of $\{v_j\}_{j \in [i]}$ is uniform; to get the third equality we further consider cases where I_i will be matched: (a) if v_i is the top vertex so far, then it stays unmatched if either the vertex $\mathrm{II}_i = \mathrm{I}_{i-1}$ is already matched, *or* we don't match v_i to I_{i-1}, (b) if v_i is the second top vertex so far, then $\mathrm{I}_i = \mathrm{I}_{i-1}$ stays unmatched if it is unmatched at the step $i-1$ *and* it is not matched to v_i at step i, (c) if v_i is ranked lower than $\mathrm{I}_i, \mathrm{II}_i$, then $\mathrm{I}_i = \mathrm{I}_{i-1}$ stays unmatched if it is unmatched at step $i-1$; to obtain the fourth equality, we use the fact that the probability c_i of matching I_i and II_i (if it is possible) depends only on the time step i.

We next claim that the expected performance of the algorithm can be written as

$$\mathsf{ALG}(\vec{c}) = \frac{1}{\binom{n}{2}} \sum_{i=2}^{n} (i-1) \cdot p_{i-1} \cdot c_i. \qquad (17)$$

To see this, note that

$$\begin{aligned}
\mathsf{ALG}(\vec{c}) &= \sum_{i=2}^{n} \mathbf{Pr}\left[\{\mathrm{I}_i, \mathrm{II}_i\} = \{\mathrm{I}_n, \mathrm{II}_n\} \wedge v_i \in \{\mathrm{I}_i, \mathrm{II}_i\} \wedge O_{i-1} \wedge \text{matched}(\mathrm{I}_i \mathrm{II}_i, @i)\right] \\
&= \sum_{i=2}^{n} \mathbf{Pr}\left[\{\mathrm{I}_i, \mathrm{II}_i\} = \{\mathrm{I}_n, \mathrm{II}_n\}\right] \cdot \frac{2}{i} \cdot p_{i-1} \cdot c_i = \sum_{i=2}^{n} \frac{\binom{n-2}{i-2}}{\binom{n}{i}} \cdot \frac{2}{i} \cdot p_{i-1} \cdot c_i \\
&= \frac{1}{\binom{n}{2}} \sum_{i=2}^{n} (i-1) \cdot p_{i-1} \cdot c_i,
\end{aligned}$$

where the first equality follows from the law of total probability; the second equality follows from the following facts: 1) conditioned on $\{\mathrm{I}_i, \mathrm{II}_i\} = \{\mathrm{I}_n, \mathrm{II}_n\}$, the arrival order of $v_1, \ldots, v_i$ is chosen uniformly at random, 2) the events $v_i \in \{\mathrm{I}_i, \mathrm{II}_i\}$ and O_{i-1} are independent, 3) our algorithm matches $\mathrm{I}_i \mathrm{II}_i$ with probability c_i when possible, i.e., when $v_i \in \{\mathrm{I}_i, \mathrm{II}_i\}$ and I_{i-1} is unmatched.

For notation simplicity, let $q_i = ip_i$ for $i \in [n]$. By equations (16) and (17), in order to find the optimal $\vec{c} \in [0,1]^n$, it suffices to find the maximum of the following function[8] on $\vec{c} \in [0,1]^n$:

$$f(\vec{c}, \vec{q}) \stackrel{\text{def}}{=} \sum_{i=2}^{n} q_{i-1} c_i \qquad \text{where } q_i = 1 + \left(1 - \frac{2c_i}{i-1}\right) \cdot q_{i-1}, \text{ and } q_1 = 1 \tag{18}$$

We calculate the derivative of $f(\vec{c}, \vec{q}(\vec{c}))$ over c_i (note that $\vec{q}$ also depends on $\vec{c}$).

$$\begin{aligned}
\frac{df}{dc_i} &= q_{i-1} + \sum_{j=i+1}^{n} c_j \cdot \frac{dq_{j-1}}{dc_i} && (q_j \text{ does not depend on } c_i \text{ for } j < i) \\
&= q_{i-1} + \sum_{j=i+1}^{n} c_j \cdot \frac{dq_{j-1}}{dq_{j-2}} \cdot \frac{dq_{j-2}}{dq_{j-3}} \cdots \frac{dq_i}{dc_i} && \text{(by the chain rule)} \\
&= q_{i-1} + \sum_{j=i+1}^{n} c_j \cdot \prod_{k=i+1}^{j-1} \left(1 - \frac{2c_k}{k-1}\right) \cdot \left(-\frac{2q_{i-1}}{i-1}\right) && \text{(by the recursive formula of } q_j\text{, see Equation (18))} \\
&= q_{i-1} \cdot \left(1 - \frac{2}{i-1} \cdot \sum_{j=i+1}^{n} c_j \prod_{k=i+1}^{j-1} \left(1 - \frac{2c_k}{k-1}\right)\right).
\end{aligned}$$

Notice that $\frac{df}{dc_i}$ does not depend on c_i, i.e., f is a linear function of c_i. In particular, it means that the maximum of $f(\vec{c})$ on $\vec{c} \in [0,1]^n$ is achieved at either $c_i = 0$, or $c_i = 1$ depending on the sign of $\frac{df}{dc_i}$. Note that the maximum of f must be attained by some $\vec{c}$, since $[0,1]^n$ is a compact space and f is a continuous function. Moreover, we observe the following "monotonicity" property of the derivatives $\frac{df}{dc_i}$.

Claim A.1. *Let $i \geq 3$. If $\frac{df}{dc_i} \leq 0$, then $\frac{df}{dc_{i-1}} < 0$.*

Proof. Let $x_{i-1} = \sum_{j=i}^{n} c_j \prod_{k=i}^{j-1} \left(1 - \frac{2c_k}{k-1}\right)$ and $x_i = \sum_{j=i+1}^{n} c_j \prod_{k=i+1}^{j-1} \left(1 - \frac{2c_k}{k-1}\right)$. Thus $x_{i-1} = x_i \cdot \left(1 - \frac{2c_i}{i-1}\right) + c_i$. Then, $\frac{df}{dc_i} \leq 0$ iff $1 - \frac{2x_i}{i-1} \leq 0$ iff $x_i \geq \frac{i-1}{2}$.

We get:

$$\begin{aligned}
x_{i-1} = x_i \cdot \left(1 - \frac{2c_i}{i-1}\right) + c_i &= x_i + c_i \cdot \left(1 - \frac{2x_i}{i-1}\right) \geq x_i + 1 \cdot \left(1 - \frac{2x_i}{i-1}\right) \\
&= x_i \cdot \frac{i-3}{i-1} + 1 \geq \frac{i-1}{2} \cdot \frac{i-3}{i-1} + 1 = \frac{i-1}{2} > \frac{i-2}{2}.
\end{aligned}$$

Consequently $\frac{df}{dc_{i-1}} < 0$. □

Let $\vec{c} \in [0,1]^n$ be the vector at which $f(\vec{c})$ attains its maximum. Let ℓ be the largest index $i \in [n]$ such that $\frac{df}{dc_i} \leq 0$. Then, $\frac{df}{dc_i} > 0$ for all $i > \ell$, and by Claim A.1, $\frac{df}{dc_i} < 0$ for all $i < \ell$. Since f attains its maximum at $\vec{c}$, it must be that $c_i = 1$ for all $i > \ell$ and $c_i = 0$ for all $i < \ell$. We can also assume without loss of generality that $c_\ell = 0$, as $\frac{df}{dc_\ell} \leq 0$ (if $\frac{df}{dc_\ell} = 0$, then f does not depend on c_ℓ). We conclude that $c_i = 0$ for all $i \leq \ell$ and $c_i = 1$ for $i > \ell$. In other words, the algorithm is deterministic: it matches no vertices up to step ℓ, and thereafter matches the top two vertices so far whenever possible.

Next, we calculate the value of q_i for all i.

- For $i \leq \ell$, $c_i = 0$, and we get $q_i = 1 + q_{i-1} \cdot (1 - \frac{2 \cdot c_i}{i-1}) = 1 + q_{i-1}$. Thus, $q_i = i$.

[8] f does not depend on c_1.

- For $i > \ell$, $c_i = 1$, and we get $q_i = 1 + q_{i-1} \cdot (1 - \frac{2 \cdot c_i}{i-1}) = 1 + \frac{i-3}{i-1} \cdot q_{i-1}$. Multiplying the last equality by $(i-1)(i-2)$ gives

$$(i-1)(i-2)q_i - (i-2)(i-3)q_{i-1} = (i-1)(i-2). \tag{19}$$

  Summing the LHS of (19) over $j = \ell+1, \ldots, i$ (with j in the role of i) gives $(i-1)(i-2)q_i - \ell(\ell-1)(\ell-2)$ due to telescopic sum. Summing the RHS of (19) over $j = \ell+1, \ldots, i$ (with j in the role of i) gives $\sum_{j=\ell+1}^{i}(j-1)(j-2) = \frac{i(i-1)(i-2)}{3} - \frac{\ell(\ell-1)(\ell-2)}{3}$. We get that $q_i = \frac{i}{3} + \frac{2\ell(\ell-1)(\ell-2)}{3(i-1)(i-2)}$.

We are now ready to calculate the performance of the algorithm. Using Equation (17), the fact that $c_i = 0$ for all $i \le \ell$ and $c_i = 1$ for all $i > \ell$, and the values of q_i as calculated above gives

$$\begin{aligned}
\binom{n}{2} \cdot \mathrm{ALG} &= \sum_{i=2}^{n} q_{i-1}c_i = \sum_{i=\ell}^{n-1} q_i = \sum_{i=\ell}^{n-1}\left(\frac{i}{3} + \frac{2\ell(\ell-1)(\ell-2)}{3(i-1)(i-2)}\right) \\
&= \frac{\sum_{i=1}^{n-1} i - \sum_{i=1}^{\ell-1} i}{3} + \frac{2\ell(\ell-1)(\ell-2)}{3}\sum_{i=\ell}^{n-1}\left[\frac{1}{i-2} - \frac{1}{i-1}\right] \\
&= \frac{n^2 - n - \ell^2 + \ell}{6} + \frac{2}{3}\ell(\ell-1)(\ell-2)\left(\frac{1}{\ell-2} - \frac{1}{n-2}\right) \\
&= \frac{n^2}{6} + \frac{\ell^2}{2} - \frac{2\ell^3}{3n} + \left[\frac{\ell-n}{6} - \frac{2\ell}{3} + \frac{2(3\ell^2-2\ell)}{3n} - \frac{4\ell(\ell-1)(\ell-2)}{3n(n-2)}\right] \\
&= n^2 \cdot \left(\frac{1}{6} + \frac{\ell^2}{2n^2} - \frac{2\ell^3}{3n^3} + O\left(\frac{1}{n}\right)\right) \le \frac{5}{24}n^2 + O(n),
\end{aligned}$$

where the last equality holds since $0 < \ell \le n$; and the last inequality holds since the cubic function $g(x) \stackrel{\text{def}}{=} \frac{1}{6} + \frac{x^2}{2} - \frac{2x^3}{3}$ with $x = \frac{\ell}{n}$ attains its maximum on the interval $x \in [0,1]$ at $x = 0.5$, where the maximum value is $\frac{5}{24}$. Therefore, $\mathrm{ALG} \le \frac{5}{12} + O(\frac{1}{n})$, concluding the proof of Theorem 4.4. □

B GENERALIZATION TO HYPERGRAPHS

In this section we generalize Algorithm 2 to the online bipartite hypergraph secretary matching problem studied by [34] and [31].

Let $H = (L \cup R, E)$ be the underlying edge-weighted $(d+1)$-hypergraph. Each edge in E has the form (v, S) where $v \in L, S \subseteq R$ and $|S| \le d$. All vertices in R are given in advance and the vertices in L arrive online uniformly at random. We assume $|L| = m$, and m is known in advance. Upon the arrival of vertex v, all its incident hyperedges are revealed with the corresponding weights. The edge arrival model studied in Section 5 can be viewed as a special case of the online bipartite 3-hypergraph matching. Specifically, for the underlying graph $G = (V, E_G)$ of the edge arrival problem we construct a hypergraph in which $R = V$ and each vertex in L corresponds to an edge in E_G. That is, each vertex $\ell \in L$ corresponding to the edge $e = (uv) \in E_G$ has only one incident hyperedge $\{\ell, u, v\}$ in the hypergraph.

Let $L_t = \{\ell_1, \ldots, \ell_t\}$ denote the set of vertices that arrived up to time t, and let μ_t^* denote the (unique) maximum weighted matching in $H(L_t \cup R)$. The algorithm (see Algorithm 3) ignores the first $\lfloor f_d \cdot m \rfloor$ vertices (exploration phase), where $f_d = 1/d^{\frac{1}{d-1}}$. Then, in every round t, upon the arrival of vertex ℓ_t, we first find the maximum weighted matching μ_t^* in the graph induced by $L_t \cup R$. Let e_t be the incident edge of ℓ_t in μ_t^*. If ℓ_t is not matched in μ_t^*, let e_t be a null edge for notation simplicity. We compute the probability that edge e_t is available (the probability is taken with respect to the random arrival order of the vertices L_{t-1} and random choices of the algorithm

in steps 1 to $t-1$); denote this probability by x_t. Then, the algorithm matches e_t with probability $\frac{\alpha_t}{x_t}$, where α_t is given by the following formula (20):

$$\alpha_t = \begin{cases} 0 & \text{if } t \le f_d \cdot m \\ 1 - d \cdot \sum_{i=1}^{t-1} \frac{\alpha_i}{i} & \text{if } t > f_d \cdot m \end{cases} \tag{20}$$

ALGORITHM 3: Algorithm for online bipartite hypergraph secretary matching

Let $\ell_1, \ldots, \ell_m$ be the vertices of the left side L given in their arrival order;
$A = R$, $\mu = \emptyset$; ▷ A is the set of available vertices, μ is the returned matching
for $t \in \{\lfloor f_d \cdot m \rfloor + 1, ..., m\}$ **do**
  $\mu_t^* \leftarrow$ the maximum weighted matching in $H(L_t \cup R)$;
  Let $e_t = (\ell_t, S_t)$ be the incident edge of ℓ_t in μ_t^*; ▷ If ℓ_t is not matched in μ_t^*, let $S_t = \emptyset$
  $x_t \leftarrow \mathbf{Pr}[S_T \subseteq A]$ (e_t is available); ▷ random: order of $\{e_1, ..., e_{t-1}\}$ + algorithm's choices
  if $S_t \subseteq A$ **then**
    begin with probability $\frac{\alpha_t}{x_t}$
      Add e_t to μ;
      Remove S_t from A;
return *matching* μ

Observe that the edge e_t we consider to take at time t only depends on the set of arrived vertices L_t at time t and the last arriving vertex ℓ_t. The algorithm ensures that every edge e_t is matched with a certain probability α_t given by (20). This allows us to conveniently estimate the expected contribution of the maximum matching at time t and compare it to the maximum matching in the whole graph. Before doing that, we need to prove that the algorithm is well defined, i.e., that $x_t \ge \alpha_t$ for every t.

Note that x_t is the probability that e_t is available, given a random order of vertices in L_t (the vertices arriving up to time t) that ends with ℓ_t. As such, the probability depends on ℓ_t, and the set L_t, but *not* on the order of vertices within L_t (except for ℓ_t being the t-th vertex).

Recall that "matched$(u, < t)$" denotes the event that $u \in R$ becomes unavailable before time t, and "matched$(e, @t)$" denotes the event that edge e is matched exactly at time t.

Lemma B.1. *For every time t, vertex $v \in L$, and a set of vertices Q, if $\ell_t = v$ and $L_{t-1} = Q$, then $x_t \ge \alpha_t$.*

Proof. The proof proceeds by induction on t. For the base case, where $t = \lfloor f_d \cdot m \rfloor + 1$, the statement holds trivially since we have not matched anything yet, i.e., $x_t = 1$. Next, fix t and suppose the statement holds for all $t' \le t-1$. Let $v = \ell_t$ be the vertex arriving at time t and let $Q = L_{t-1}$ be the set of arrived vertices up to time t. Observe that given $v = \ell_t, Q = L_{t-1}$, the edge $e_t = (v, S)$ is fixed. For simplicity, we omit conditioning on $\ell_t = v, L_{t-1} = Q$ in all probabilities and

indicator expressions in the remainder of the lemma's proof. For each vertex $u \in R$, it holds that

$$\mathbf{Pr}\left[\text{matched}(u, < t)\right] = \sum_{i=1}^{t-1} \sum_{\substack{Q_i \subseteq Q \\ |Q_i|=i}} \sum_{z_i \in Q_i} \mathbf{Pr}\left[\ell_i = z_i, L_i = Q_i\right] \cdot \mathbb{1}[u \in e_i \mid \ell_i = z_i, L_i = Q_i] \cdot \mathbf{Pr}\left[\text{matched}(e_i, @i) \mid \ell_i = z_i, L_i = Q_i\right]. \quad (21)$$

Notice that $\mathbf{Pr}[\text{matched}(e_i, @i) \mid \ell_i = z_i, L_i = Q_i] = \alpha_i$, according to the design of our algorithm and the induction hypothesis, and that $\mathbf{Pr}[\ell_i = z_i, L_i = Q_i] = \frac{1}{i}\mathbf{Pr}[L_i = Q_i]$ due to the random arrival order of the vertices. We get

$$\mathbf{Pr}\left[\text{matched}(u, < t)\right] = \sum_{i=1}^{t-1} \frac{\alpha_i}{i} \cdot \sum_{\substack{Q_i \subseteq Q \\ |Q_i|=i}} \mathbf{Pr}\left[L_i = Q_i\right] \sum_{z_i \in Q_i} \mathbb{1}[u \in e_i \mid \ell_i = z_i, L_i = Q_i].$$

The maximum matching μ_i^* is fixed for a given Q_i and $u \in e_i$ if and only if (i) u is matched in μ_i^* and (ii) its corresponding online vertex arrives at time i. That is, $\sum_{z_i \in Q_i} \mathbb{1}[u \in e_i \mid \ell_i = z_i, L_i = Q_i] \leq 1$. Consequently, we have that

$$\mathbf{Pr}\left[\text{matched}(u, < t)\right] \leq \sum_{i=1}^{t-1} \frac{\alpha_i}{i} \cdot \sum_{\substack{Q_i \subseteq Q \\ |Q_i|=i}} \mathbf{Pr}\left[L_i = Q_i\right] \cdot 1 = \sum_{i=1}^{t-1} \frac{\alpha_i}{i}. \quad (22)$$

Finally, since S_t contains at most d vertices, applying the union bound to the events matched$(u, < t)$ for all $u \in S_t$, we have that the probability that e_t is available is

$$x_t = \mathbf{Pr}\left[e_t \text{ available } @t\right] \geq 1 - d \sum_{i=1}^{t-1} \frac{\alpha_i}{i} \overset{(20)}{=} \alpha_t.$$

This concludes the proof of Lemma B.1. □

We are now ready to conclude the competitive analysis of Algorithm 3. Our competitive ratio has the same asymptotic order $\Omega(\frac{1}{d})$ as the previous best bound of $\frac{1}{ed}$ by [31], but the constant factor of e improves when d goes to infinity. We assume without loss of generality that the number of vertices m in L is large. Indeed, we can slightly modify Algorithm 3 by adding a number of dummy vertices with no edges to R.

Theorem B.2. *Algorithm 3 has a competitive ratio of* $1/d^{\frac{d}{d-1}}$.

Proof. As the set of the first t vertices L_t is chosen uniformly at random, the expected total weight of the maximum matching μ_t^* is greater than or equal to the expected weight of those edges of L_t in the optimal matching μ^*, i.e.,

$$\mathbf{E}\left[\sum_{e \in \mu_t^*} w_e\right] \geq \frac{t}{m} \cdot \sum_{e \in \mu^*} w_e. \quad (23)$$

Next, we prove by induction on t that for every $t > \lfloor f_d \cdot m \rfloor$:

$$\alpha_t = \prod_{i=1}^{d} \frac{\lfloor f_d \cdot m \rfloor + 1 - i}{t - i}. \quad (24)$$

For $t = \lfloor f_d \cdot m \rfloor + 1$, α_t is indeed 1. For $t > \lfloor f_d \cdot m \rfloor + 1$,

$$\alpha_t = 1 - d \sum_{i=1}^{t-1} \frac{\alpha_i}{i} = \alpha_{t-1} - d \cdot \frac{\alpha_{t-1}}{t-1} = \frac{t-d-1}{t-1} \cdot \alpha_{t-1} = \frac{t-d-1}{t-1} \prod_{i=1}^{d} \frac{\lfloor f_d \cdot m \rfloor + 1 - i}{t-1-i}.$$

The induction hypothesis now implies (24).

We are now ready to establish the competitive ratio of Algorithm 3. Write μ for the matching returned by Algorithm 3. Edge $e_t = (v, S)$ is matched in round t if all vertices in S are available (this happens with probability x_t). Under this event, e_t is matched with probability α_t / x_t. Therefore,

$$\mathbf{E}\left[\sum_{e \in \mu} w_e\right] = \sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \sum_{\substack{V \subseteq L, \\ |V|=t}} \sum_{v \in V} \mathbf{E}\left[w_{e_t} \mid \ell_t = v, L_t = V\right] \cdot \mathbf{Pr}\left[\ell_t = v, L_t = V\right] \cdot \alpha_t.$$

Observe that $\sum_{v \in V} \mathbf{E}[w_{e_t} \mid \ell_t = v, L_t = V] = \mathbf{E}[\sum_{e \in \mu_t^*} w_e \mid L_t = V]$ and also that $\mathbf{Pr}[\ell_t = v, L_t = V] = \mathbf{Pr}[L_t = V] \cdot \frac{1}{t}$. We get

$$\begin{aligned}
\mathbf{E}\left[\sum_{e \in \mu} w_e\right] &= \sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \sum_{\substack{V \subseteq L, \\ |V|=t}} \mathbf{E}\left[\sum_{e \in \mu_t^*} w_e \,\middle|\, L_t = V\right] \cdot \mathbf{Pr}\left[L_t = V\right] \cdot \frac{1}{t} \cdot \alpha_t \\
&= \sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \mathbf{E}\left[\sum_{e \in \mu_t^*} w_e\right] \cdot \frac{\alpha_t}{t} \overset{(24)}{=} \sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \mathbf{E}\left[\sum_{e \in \mu_t^*} w_e\right] \cdot \frac{1}{t} \cdot \prod_{i=1}^{d} \frac{\lfloor f_d \cdot m \rfloor + 1 - i}{t-i} \\
&\overset{(23)}{\geq} \sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \frac{t \cdot \sum_{e \in \mu^*} w_e}{m} \cdot \frac{1}{t} \cdot \prod_{i=1}^{d} \frac{\lfloor f_d \cdot m \rfloor + 1 - i}{t-i} \\
&= \prod_{i=1}^{d} (\lfloor f_d \cdot m \rfloor + 1 - i) \cdot \frac{\sum_{e \in \mu^*} w_e}{m} \cdot \sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \prod_{i=1}^{d} \frac{1}{t-i}. \qquad (25)
\end{aligned}$$

Observe that

$$\prod_{i=1}^{d} \frac{1}{t-i} = \frac{1}{d-1} \left(\prod_{i=1}^{d-1} \frac{1}{t-i-1} - \prod_{i=1}^{d-1} \frac{1}{t-i} \right).$$

The summation over t in Equation (25) telescopes to

$$\sum_{t=\lfloor f_d \cdot m \rfloor + 1}^{m} \prod_{i=1}^{d} \frac{1}{t-i} = \frac{1}{d-1} \cdot \left(\prod_{i=1}^{d-1} \frac{1}{\lfloor f_d \cdot m \rfloor - i} - \prod_{i=1}^{d-1} \frac{1}{m-i} \right).$$

Thus, we have

$$\begin{aligned}
&\frac{1}{m}\cdot\prod_{i=1}^{d}\left(\lfloor f_d\cdot m\rfloor+1-i\right)\cdot\sum_{t=\lfloor f_d\cdot m\rfloor+1}^{m}\prod_{i=1}^{d}\frac{1}{t-i}\\
=&\frac{1}{m}\cdot\prod_{i=1}^{d}\left(\lfloor f_d\cdot m\rfloor+1-i\right)\cdot\frac{1}{d-1}\cdot\left(\prod_{i=1}^{d-1}\frac{1}{\lfloor f_d\cdot m\rfloor-i}-\prod_{i=1}^{d-1}\frac{1}{m-i}\right)\\
=&\frac{1}{d-1}\cdot\left(\frac{\lfloor f_d\cdot m\rfloor}{m}-\prod_{i=1}^{d}\frac{\lfloor f_d\cdot m\rfloor+1-i}{m+1-i}\right)\\
=&\frac{1+o(1)}{d-1}\left(f_d-f_d^d\right)=(1+o(1))/d^{\frac{d}{d-1}},
\end{aligned}$$

where to get the second to the last equality we used that $\frac{\lfloor f_d\cdot m\rfloor-i}{m-i}\approx f_d$ for fixed i and f_d as m goes to infinity; the last equation follows from the definition of $f_d=1/d^{\frac{1}{d-1}}$. To sum up, the coefficient of $\sum_{e\in\mu^*}w_e$ in Equation (25) is at least $1/d^{\frac{d}{d-1}}$; this concludes the proof of the theorem. □

Remark: In this section we study the extension of the edge arrival model to hypergraphs. The extension of the vertex arrival model to hypergraphs remains an open problem.