Packing Returning Secretaries

Martin Hoefer
Goethe University Frankfurt/Main, Germany
mhoefer@cs.uni-frankfurt.de

Lisa Wilhelmi
Goethe University Frankfurt/Main, Germany
wilhelmi@cs.uni-frankfurt.de

Abstract

We study online secretary problems with returns in combinatorial packing domains with n candidates that arrive sequentially over time in random order. The goal is to accept a feasible packing of candidates of maximum total value. In the first variant, each candidate arrives exactly twice. All $2n$ arrivals occur in random order. We propose a simple 0.5-competitive algorithm that can be combined with arbitrary approximation algorithms for the packing domain, even when the total value of candidates is a subadditive function. For bipartite matching, we obtain an algorithm with competitive ratio at least $0.5721 - o(1)$ for growing n, and an algorithm with ratio at least 0.5459 for all $n \geq 1$. We extend all algorithms and ratios to $k \geq 2$ arrivals per candidate.

In the second variant, there is a pool of undecided candidates. In each round, a random candidate from the pool arrives. Upon arrival a candidate can be either decided (accept/reject) or postponed (returned into the pool). We mainly focus on minimizing the expected number of postponements when computing an optimal solution. An expected number of $\Theta(n \log n)$ is always sufficient. For matroids, we show that the expected number can be reduced to $O(r \log(n/r))$, where $r \leq n/2$ is the minimum of the ranks of matroid and dual matroid. For bipartite matching, we show a bound of $O(r \log n)$, where r is the size of the optimum matching. For general packing, we show a lower bound of $\Omega(n \log \log n)$, even when the size of the optimum is $r = \Theta(\log n)$.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Secretary Problem, Coupon Collector Problem, Matroids

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.65

Related Version A full version of the paper is available at https://arxiv.org/abs/1810.11216.

Funding Supported by a grant from the German-Israeli Foundation for Scientific Research and Development (GIF).

1 Introduction

The secretary problem is a classic approach to study optimal stopping problems: A sequence of n candidates are arriving in uniform random order. Each candidate reveals its value only upon arrival and must be decided (accept/reject) before seeing any further candidate(s). Every decision is final – once a candidate gets accepted, the game is over. Moreover, no rejected candidate can be accepted later on. The goal is to find the best candidate. An optimal solution is to discard the first (roughly) n/e candidates. From the subsequent ones we accept the first that is the best one among the ones seen so far. The probability to hire the best candidate approaches $1/e \approx 0.37$ when n tends to infinity.

29th International Symposium on Algorithms and Computation (ISAAC 2018).
Editors: Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao; Article No. 65; pp. 65:1–65:12
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The secretary problem and its variants have been popular since the 1960s. Significant interest in computer science emerged about a decade ago due to new applications in e-commerce and online advertising markets [2, 14]. For example, the classic secretary problem can be used to model a seller that wants to give away a single item, buyers arrive sequentially over time, and the goal is to assign the item to the buyer with highest value. More generally, online budgeted matching problems arise when search queries arrive over time, and the goal is to show the most profitable ads on the search result pages. The goal here is to design algorithms with good competitive ratio.

More recently, progress has been made towards a general understanding of online packing problems with random-order arrival, including matching [3, 20, 16], integer packing programs [22, 17], or independent set problems [13]. Most prominently, the matroid secretary problem has attracted a large amount of interest [2, 6]. Here the elements of a matroid arrive in uniform random order, and the goal is to construct an independent set with as high a value as possible. A central open problem in the area is the matroid secretary conjecture – is there a constant-competitive algorithm for every matroid in the random order model? The conjecture has been proved for a variety of subclasses of matroids [6]. Currently, the best-known algorithms for the general problem are $1/O(\log \log \text{rank})$-competitive [21, 8].

A strong assumption in the secretary problem is that every decision about a candidate must be made immediately without seeing any of the future candidates. Instead, in many natural admission scenarios candidates appear more than once, or they arrive and stay in the system for some time, during which a decision can be made. An interesting variant that captures this idea is the returning secretary problem [25]. Here each candidate is assigned two random time points from a bounded time interval. The earlier becomes the arrival time, the later the departure time. Hence, we can assume that each candidate arrives exactly twice, and all $2n$ arrivals occur in random order. The decision about acceptance of a candidate can be made between the first and the second arrival. More generally, for $k \geq 2$ each candidate chooses k random points, arrives at the earliest and leaves at the latest point. In this case, there are kn arrivals in random order. Vardi [25] showed an optimal algorithm for the returning secretary problem with $k = 2$, for which the probability of accepting the best candidate is about 0.768. For matroid secretary with $k = 2$ arrivals, a competitive ratio of 0.5, and for matching secretary a ratio $0.5625 - o(1)$ (with asymptotics in n) were shown.

In this paper, we significantly broaden and extend the results on the returning secretary problem towards general packing domains. We provide a simple algorithm that can be combined with arbitrary α-approximation algorithms and yields competitive ratios of $0.5 \cdot \alpha$ for all subadditive packing problems, including matroids, matching, knapsack, independent set, etc. Moreover, we improve the guarantees for matching secretary and provide bounds that hold in expectation for all n. We extend all our bounds to arbitrary $k \geq 2$. In addition, we study a complementary variant in which the decision maker is allowed to postpone the decision about a candidate. In this case, the goal is to minimize the number of postponements to guarantee an optimal or near-optimal solution in the end. These problems can be cast as a set of novel coupon collector problems, and we provide guarantees and trade-offs for matroid, matching and knapsack postponement.

Results and Contribution

In the secretary problem with k arrivals in Section 3, each candidiate arrives exactly k times. We propose a simple approach for general subadditive packing problems with returns, which can be combined with arbitrary α-approximation algorithms. It yields a competitive ratio of $0.5 \cdot \alpha$ for $k = 2$, and $\alpha \cdot (1 - 2^{-(k-1)})$ for $k \geq 2$.

For additive bipartite matching, we obtain a new algorithm that provides an improved competitive ratio of $0.5721 - o(1)$ for $k = 2$ with asymptotics in n. Moreover, we derive an algorithm with ratio 0.5459 for $k = 2$ for every n. Both algorithms rely on exact solution of partial matching problems. The algorithms can be combined with faster α-approximations for partial matchings, by spending at most an additional factor α in the competitive ratio. For the previous algorithm in [25], the algorithm description and proof of the ratio in the full version is slightly ambiguous.[1] Our algorithm clarifies and slightly improves upon this by including the twice-arrived and rejected candidates during a sample phase when computing partial matchings. Their removal yields free nodes in the offline partition for matching in later rounds.

In the postponing secretary problem in Section 4, there is a pool of n undecided candidates. In each round, a random candidate from the pool arrives. Upon arrival a candidate can be either decided (accept/reject) or postponed and returned into the pool. We strive to minimize the expected number of postponements to compute an optimal or near-optimal solution. Postponing everyone until all candidates are observed at least once is the coupon collector problem. Hence, with an expected number of $O(n \log n)$ postponements we reduce the problem to offline optimization. For general subadditive packing and an α-approximation algorithm, a simple trade-off shows an $(1 - \varepsilon) \cdot \alpha$-approximation using $O(n \ln {}^1\!/\!_\varepsilon)$ postponements.

Based on a property we term exclusion-monotonicity, we show significantly improved results when the desired solution has small cardinality. A bound of $O(r \log n)$ for the expected number of postponements holds when obtaining optimal solutions of size at most r in additive matroids and bipartite matching, and greedy 2-approximations for knapsack. For matroids, we can further improve the bound to $O(r' \ln {}^n\!/\!_{r'})$, where $r' = \min(r, n-r)$. This upper bound is at most n, and the worst-case is attained for uniform matroids. We fully characterize the expected number of postponements of every candidate in uniform matroids when the optimal solution is to be obtained. Finally, we conclude the paper with a lower bound that in general we might need $\Omega(n \log \log n)$ postponements even with an optimal solution of cardinality $O(\log n)$. Due to space constraints, all missing proofs are deferred to the full version of this paper.

Further Related Work

The literature on secretary online variants of packing problems and online stochastic optimization has grown significantly over the last decade. We restrict the review to the most directly related results. For a survey of classic variants of the secretary problem, see [10].

The bipartite secretary matching problem was first studied in the context of transversal matroids [3], where a decision about accepting an arriving vertex into the matching needs to be taken directly, but matching edges can be decided in the end. Later works required that the edges must also be decided upon arrival [20]. The best algorithm for both variants obtains a competitive ratio of $1/e$ [16]. Most work in computer science has been devoted to the matroid secretary problem. Currently, the best algorithms obtain a competitive ratio $1/O(\log \log \text{rank})$ [21, 8]. It is an open problem if a constant competitive ratio can be shown. For a survey of work on classes of matroids and further developments see [6].

[1] For example, the pseudo-code on page 12 does not become the algorithm for a single secretary when there is a single node in the offline partition. One would always accept the best secretary that arrived once in the sample phase. A better one arriving in later rounds is always rejected inside the for-loop. Also, the proof of Claim 5.6 seems to require both sides of the bipartite graph must have size n.

While above results are all for maximizing additive objective functions, recent work has started to consider submodular ones. For cardinality and matching constraints, constant-competitive algorithms exist for submodular secretary variants [18]. For matroids, there is a general technique to extend algorithms for additive objectives to submodular ones, which preserves constant competitive ratios [9].

Beyond matroids and matching, there are constant-competitive algorithms for knapsack secretary [1]. Prominent graph classes in networking applications allow good secretary algorithms for independent set [13]. The techniques for bipartite matching have been extended to secretary variants of combinatorial auctions and integer packing programs [22, 17]. Moreover, there are $1/O(\log n)$-competitive algorithms even in a general packing domain [23].

Additional model variants that have found interest are, for example, local secretary [5] (several decision makers try to simultaneously hire candidates based on local feedback), temp secretary [11] (candidates are hired only for a bounded period of time), or ordinal secretary [15, 24] (information available to the decision maker is only the total order of the candidates but not their numerical values).

Secretary postponement can be seen as a combinatorial extension of the coupon collector problem, a classic problem in applied probability. The elementary problem and its analysis are standard and discussed in many textbooks. The problem has many applications, and there is a plethora of variants that have been studied (see, e.g., [4, 12, 19]). To the best of our knowledge, however, the results for combinatorial packing problems derived in this paper have not been obtained in the literature before.

2 Packing Problems

We consider a *packing problem*, in which there is a set N of n *candidates*, and a set $\mathcal{S} \subseteq 2^N$ of *feasible solutions*. $\mathcal{S}$ is downward-closed, i.e. $S \in \mathcal{S}$ and $T \subseteq S$ implies $T \in \mathcal{S}$. For most parts, we assume that the *objective function* $w : 2^N \to \mathbb{R}_{\geq 0}$ is *additive*, i.e., there is a non-negative value $w : N \to \mathbb{R}_{\geq 0}$ for each candidate, and $w(S) = \sum_{e \in S} w(e)$ for all $S \subseteq N$. More generally, we will sometimes assume the objective function w is *monotone* and *subadditive*. If a packing problem has an α-*approximation algorithm*, then for any $N' \subseteq N$ the algorithm guarantees an approximation ratio $\alpha \leq 1$ for maximizing w over $\mathcal{S} \cap 2^{N'}$.

In a secretary variant, we know the number n upfront, and the candidates arrive in random order. Suppose a set N_i of candidates has arrived in rounds $1, \ldots, i$ and candidate $e \in N \setminus N_i$ arrives in round $i + 1$. Then e reveals all new feasible solutions with previously arrived candidates $(\mathcal{S} \cap 2^{N_i \cup \{e\}}) \setminus (\mathcal{S} \cap 2^{N_i})$ and their corresponding weight. In the additive case, this simply reduces to revealing the solutions and the weight $w(e)$.

We consider several specific variants. In *matroid secretary*, the set of candidates and the set of feasible solutions form a matroid. Upon arrival, a candidate reveals the new feasible solutions and their weights. In the additive variant with *known matroid*, all candidates and feasible solutions are known upfront. Candidates only reveal their weight upon arrival.

In *(bipartite) matching secretary*, there is a bipartite undirected graph $(N \cup V, E)$. The nodes in the offline partition V are present upfront. The candidates in the online partition arrive sequentially. The feasible solutions are the matchings in the arrived subgraph. Upon arrival, a candidate reveals its incident edges and weights of the new feasible solutions. In the additive version, the arriving candidate reveals a *weight per edge*, and the weight $w(M)$ of a matching M is the sum of edge weights. Upon accepting a candidate, the algorithm also has to decide which matching edge to include into M (since otherwise it is matroid secretary with transversal matroid).

For (additive) *knapsack secretary*, an arriving candidate e reveals its weight $w(e)$ and a size $b(e) \geq 0$. The size B of the knapsack is known upfront. The feasible solutions are all subsets of candidates such that their total size does not exceed B.

3 Secretaries with k Arrivals

Suppose that each candidate arrives exactly k times, and all these kn arrivals are presented in uniformly random order. Consider any subadditive secretary packing problem and the following simple algorithm. In the beginning, flip kn fair coins. The number of heads is the length of an initial sample phase. During the sample phase reject all candidates. Consider the set T of candidates that has appeared at least once and at most $k-1$ times in the sample phase. Apply the α-approximation algorithm to the instance based on $\mathcal{S} \cap 2^T$ to choose a feasible solution. Accept each candidate in the solution by the time of its k-th arrival.

▶ **Proposition 1.** *For any subadditive packing problem with an α-approximation algorithm, the secretary problem with k arrivals allows an algorithm with approximation ratio*

$$\beta = \alpha \cdot \left(1 - \frac{1}{2^{k-1}}\right) \quad .$$

Proof. Due to random order of arrival, we can simulate generation of T by attaching each of the kn coins to one arrival of one candidate. The arrival is in the sample phase if and only if the coin turns up heads. Then, the probability is $1/2^k$ for each of the following events: (1) a given candidate never appears in the sample phase, and (2) a given candidate appears k times in the sample phase. T is distributed as if we would include each candidate independently with probability $1 - \left(\frac{1}{2}\right)^{k-1}$.

Once T is created, we apply the α-approximation algorithm to the instance based on $\mathcal{S} \cap 2^T$ to choose a feasible solution. Note that every candidate in T will appear at least once after the sample phase and therefore is available for acceptance by our algorithm. Each element in T is sampled independently from N. Hence, as a simple consequence of subadditivity (see, e.g., [7, Proposition 2]), the value of the best feasible solution $S^*_T \subseteq T$ has value $w(S^*_T) \geq w(T \cap S^*) \geq \left(1 - \left(\frac{1}{2}\right)^{k-1}\right) \cdot w(S^*)$. By applying the α-approximation algorithm to T, we obtain a feasible solution S of value $w(S) \geq \alpha \cdot w(S^*_T) \geq \alpha \cdot \left(1 - \left(\frac{1}{2}\right)^{k-1}\right) \cdot w(S^*)$. ◀

For secretary matching, we improve upon this by using a slightly more elaborate approach. The algorithm again samples and rejects a number of candidates that is determined by kn independent coin flips with a suitable probability $p < 1$ (determined below). Hence, the length of the sample phase is distributed according to $\text{Binom}(kn, p)$. At the end of the sample phase it computes a matching M_s using an α-approximation algorithm for all known candidates and offline vertices V. It accepts into M the edges incident to candidates with at most $k-1$ arrivals in the sample. Each of them can be accepted upon their last arrival after the sample phase. The algorithm drops the edges from M_s incident to candidates that arrived k times in the sample. Let $V_s \subseteq V$ be the unmatched offline nodes.

In the second phase, the algorithm follows ideas from [16, 25]. Upon arrival of a new candidate e, the algorithm computes an α-approximate matching M_e among V_s and all candidates with first arrival after the sample phase. If M_e contains an edge (e, v) incident to e, this edge is added into M if v is still unmatched. Otherwise the edge is discarded.

Since the algorithm can be combined with arbitrary α-approximation algorithms for matching, it also applies to, e.g., the k-arrival variant of ordinal secretary matching [15].

▶ **Theorem 2.** *For secretary matching with 2 arrivals and any α-approximation algorithm for offline matching with $\alpha \leq 1$, there is an algorithm with approximation ratio of* $0.5721 \cdot \alpha - o(1)$*. For k arrivals, the ratio becomes at least* $\alpha \cdot \left(1 - \frac{1}{2^{k-1}} + \frac{1}{2^{2k}} - \frac{1}{2^{2k} \cdot (2^k-1)^2}\right) - o(1)$*.*

Proof. By similar arguments as above, for each arrival of a secretary we can assume to flip a coin independently with probability $p < 1$ that determines if the arrival happens in the sample phase. Hence, each candidate has probability p^k to arrive exactly k times in the sample phase and $(1-p)^k$ to never arrive in the sample phase. Let M be the matching computed by the algorithm, M_1 the matching obtained right after the sample phase and M_2 the matching composed in the second phase. It holds $\mathbb{E}[w(M)] = \mathbb{E}[w(M_1)] + \mathbb{E}[w(M_2)]$.

For M_1 we interpret the random coin flips as a two-step process. First, for each candidate in N we flip a coin independently with probability $(1-(1-p)^k)$ whether the candidate arrives at least once in the sample phase. Then, we flip another independent coin with probability $p^k/(1-(1-p)^k)$ whether the candidate arrives k times in the sample phase. The first set of coin flips determines the matching M_s that evolves when we apply the α-approximation algorithm right after the sample phase. Since every candidate is included independently we have $\mathbb{E}[w(M_s)] \geq (1-(1-p)^k) \cdot \alpha \cdot w(M^*)$. Afterwards, the second set of coin flips determines the candidates that are dropped from M_s. They are determined independently, so $\mathbb{E}[w(M_1)] = \left(1 - \frac{p^k}{1-(1-p)^k}\right) \cdot \alpha \cdot \mathbb{E}[w(M_s)]$. In total, $\mathbb{E}[w(M_1)] \geq (1-(1-p)^k - p^k) \cdot \alpha \cdot w(M^*)$.

We denote by X the random number of candidates that arrived at least once during the sample phase. In the acceptance phase of the algorithm, we consider all $n - X$ candidates that have not arrived during the sample phase. Standard arguments [16, 25, 18] show that each of these newly arriving candidates contributes in expectation a value of $(\alpha \cdot (w(S^*))/n$. For the ℓ-th first arrival of a new candidate, the probability that the edge (e, v) suggested by the algorithm survives is the probability that the offline node $v \in V$ was not matched earlier, which is lower bounded by

$$\frac{p^k}{1-(1-p)^k} \cdot \prod_{r=X}^{\ell-1} \frac{r-1}{r} \quad = \quad \frac{p^k}{1-(1-p)^k} \cdot \frac{X-1}{\ell-1} \quad .$$

Hence, the expected value for M_2 is at least

$$\begin{aligned}\mathbb{E}[w(M_2) \mid X] &\geq \alpha \cdot w(M^*) \cdot \sum_{\ell=X}^{n} \frac{p^k}{1-(1-p)^k} \cdot \frac{X-1}{\ell-1} \cdot \frac{1}{n} \\ &\geq \alpha \cdot w(M^*) \cdot \frac{p^k}{1-(1-p)^k} \cdot \frac{X-1}{n} \cdot \ln \frac{n}{X} \quad .\end{aligned}$$

For constants p and k, standard Hoeffding bounds imply that $X = n(1-(1-p)^k) \pm o(n)$ with probability at least $1 - 1/n^c$ for suitable constant c (see, e.g., [25]). Hence,

$$\mathbb{E}[w(M)]/w(M^*) \quad \geq \quad \alpha\left((1-(1-p)^k - p^k) + p^k \cdot \ln\left(\frac{1}{1-(1-p)^k}\right)\right) - o(1) \quad , \quad (1)$$

where the asymptotics are in n. Numerical optimization shows that for $k = 2$ and $p \approx 0.49085$, the ratio becomes at least $0.57212 \cdot \alpha - o(1)$. See Table 1 for more numerical results.

Intuitively, the algorithm benefits from the unseen candidates after the sample phase and has a tendency to reduce the sample size. On the other hand, the candidates that come k times within the sample phase create the set of free nodes in V available for matching to later candidates. Overall, this leads to a small reduction in the sample size. For larger k this effect vanishes since the number of candidates that appear never or k times during the sample

Table 1 Near-optimal parameters p for the sample phase and resulting bounds for the competitive ratio (assuming $\alpha = 1$) derived by numerical optimization of function (1).

k	2	3	4	5	6	7
p	0.49085	0.498901	0.499826	0.499968	0.499994	0.499999
ratio	0.57212	0.766694	0.879033	0.938491	0.968995	0.984435

phase both become exponentially small. The optimal sampling parameter quickly approaches $p \to 0.5$. This maximizes the profit from candidates that are available for optimization immediately after the end of the sample phase. Thereby, the improvement over the simple procedure in Proposition 1 becomes smaller.

More formally, we use $\ln(1+x) \geq x - x^2$ in (1) and obtain

$$\mathbb{E}\left[w(M)\right]/w(M^*) \geq \alpha\left((1-(1-p)^k - p^k) + \frac{p^k \cdot (1-p)^k}{1-(1-p)^k} - \frac{p^k(1-p)^{2k}}{(1-(1-p)^k)^2}\right) - o(1)\ .$$

Note that $\ln(1+x) \leq x$, so we deteriorate the expression only by the last negative term. For growing k, the optimal value of p approaches 0.5 very quickly, and we bound

$$\begin{aligned}\mathbb{E}\left[w(M)\right]/w(M^*) &\geq \alpha\left(\left(1-\frac{1}{2^k}-\frac{1}{2^k}\right)+\frac{\frac{1}{2^k}\cdot\frac{1}{2^k}}{1-\frac{1}{2^k}}-\frac{\frac{1}{2^k}\cdot\frac{1}{2^{2k}}}{(1-\frac{1}{2^k})^2}\right)-o(1)\\ &= \alpha\left(1-\frac{1}{2^{k-1}}+\frac{1}{2^{2k}}-\frac{1}{2^{2k}\cdot(2^{2k}-2^{k+1}+1)}\right)-o(1)\ .\end{aligned}$$ ◀

In contrast to [25], our algorithm computes an optimal (or α-approximate) matching after the sampling phase for the set of *all* candidates that arrived during that phase (instead of the ones that arrived only once). All candidates that arrived k times are dropped. This creates free nodes of V to be matched to subsequently arriving candidates. The ratios depend asymptotically on n, since the guarantee in the second phase relies on concentration bounds for X, the number of candidates that arrive at least once in the sampling phase.

Alternatively, one can replace the second phase by recursively applying the sampling phase. More formally, after the sampling phase is done and matching M_1 is added to M, we apply the same sampling phase to V_s and the candidates that have not arrived so far. In this way, we can iterate the sampling step and re-apply it to the unseen candidates and left-over nodes of the offline partition. The resulting ratios do not require concentration bounds.

▶ **Corollary 3.** *For secretary matching with 2 arrivals and any α-approximation algorithm for offline matching with $\alpha \leq 1$, there is an algorithm with approximation ratio of* $0.5459 \cdot \alpha$ *for every $n \geq 1$. For k arrivals, the ratio becomes at least* $(1 - \frac{1}{2^{k-1}} + \frac{1}{2^{2k}} - \frac{2^k-1}{2^{2k}\cdot(2^{2k}-2^k-1)}) \cdot \alpha$ *for every $n \geq 1$.*

4 Postponing Secretaries

Now suppose that for each arriving candidate the algorithm can decide (accept/reject) or postpone it. The goal is to compute an optimal or near-optimal solution with a small expected number of postponements. Consider any algorithm for the postponement problem. We cluster the execution into rounds. Round i are the arrivals from and including the i-th *unique arrival* (i.e., the i-th time a candidate arrives for the first time) and before the $(i+1)$-th unique arrival. Clearly, there are always $n-1$ rounds in the execution of any algorithm.

If we simply postpone every candidate until we have seen all n candidates, we have full information to make accept/reject decisions for all candidates. Then the problem reduces to the classic coupon collector problem, and the expected number of returns is $\Theta(n \log n)$. Our goal is to examine how we can improve upon this baseline.

We first consider a general result for subadditive packing. To reduce the expected number of returns to $\Theta(n)$, it is sufficient to sacrifice a constant factor in the approximation ratio. We obtain the following FPTAS-style trade-off between postponements and solution quality.

▶ **Proposition 4.** *For any[2] $\varepsilon > 2/n$ and any subadditive packing problem with α-approximation algorithm, there is an $\alpha \cdot (1-\varepsilon)$-approximation algorithm with an expected number of postponements of $\mathbb{E}[R] < n \cdot \ln(2/\varepsilon)$.*

Proof. We postpone every candidate until round $\lceil n(1-\varepsilon) \rceil$. Then, we run the α-approximation algorithm on the subset of arrived candidates. By the same arguments as in Proposition 1, this yields an $\alpha(1-\varepsilon)$-approximation.

Let R^i be the number of postponements in round i. Clearly, by linearity of expectation, $\mathbb{E}[R] = \sum_{i=1}^{n-1} \mathbb{E}[R^i]$. In each round, the number of postponements is the number of rounds until we see the next unique arrival, and, hence, distributed according to a negative binomial distribution. Therefore, their expected number is

$$\mathbb{E}[R] \leq \sum_{i=1}^{\lceil n(1-\varepsilon) \rceil} \left(\frac{n}{n-i} - 1 \right) = n \cdot \sum_{i=1}^{\lceil n(1-\varepsilon) \rceil} \frac{1}{n-i} - \lceil n(1-\varepsilon) \rceil$$

$$\leq n \cdot (\ln(n) - \ln(n\varepsilon - 1) - 1 + \varepsilon) \leq n \cdot (-\ln(\varepsilon - 1/n)) < n \ln\left(\frac{2}{\varepsilon}\right) . \qquad ◀$$

4.1 Exclusion-Monotone Algorithms

We obtain significantly better guarantees for packing problems and algorithms with a monotonicity property. Consider a packing problem and any algorithm $\mathcal{A}$. We denote by $\mathcal{A}(T)$ the solution computed by $\mathcal{A}$ when applied to $T \subseteq N$.

▶ **Definition 5.** A sequence of subsets $(N_i)_{i \in \mathbb{N}}$ with $N_i \subseteq N$ is called inclusion-monotone if $N_i \subseteq N_j$ for all $i \leq j$. An algorithm $\mathcal{A}$ is called *r-exclusion-monotone* if for every inclusion monotone sequence there is a sequence of subsets $(D_i)_{i \in \mathbb{N}}$ with $\mathcal{A}(N_i) \subseteq D_i \subseteq N_i$, $|D_i| \leq r$ and $N_i \setminus D_i \subseteq N_j \setminus D_j$ for all $i \leq j$.

Intuitively, to determine its solution for any subset of available elements N_i, an r-exclusion-monotone algorithm $\mathcal{A}$ can restrict attention to a set D_i of at most r elements. Moreover, $\mathcal{A}$ is such that any element $e \in N_i \setminus D_i$ that is discarded must never be reconsidered when more elements become available.

This property is exhibited in a variety of important packing domains. For these problems we can obtain more fine-grained, significantly improved guarantees based on solution size.

[2] For $\varepsilon \leq 2/n$, the bound remains $\Theta(n \log n)$ by simply observing all applicants and computing an α-approximation.

▶ **Proposition 6.** *The following algorithms are r-exclusion-monotone.*

- *Optimal algorithm* Opt *for matroids. r is the rank of the matroid.*
- *Optimal algorithm* Opt *for bipartite matching. r is the maximum cardinality of any matching*[3]*.*
- Greedy *0.5-approximation algorithm for knapsack. Here $r = |S| + 1$ with S a feasible packing of the knapsack with maximum cardinality.*

Now consider candidates arriving in random order with postponements. Obviously, the set of arrived candidates forms an inclusion-monotone sequence. In our algorithm Maintain-$\mathcal{A}$, we apply the r-exclusion-monotone algorithm $\mathcal{A}$ in the beginning of round i to the set N_i of arrived candidates. Maintain-$\mathcal{A}$ immediately rejects any candidate as soon as it is not contained in D_i. It keeps postponing the candidates in D_i. Finally, Maintain-$\mathcal{A}$ accepts the candidates in $\mathcal{A}(N)$ after the last round. Note that for the following result, Maintain-$\mathcal{A}$ does not have to know n, r or any properties of the unseen candidates. The following guarantee significantly improves over the simple bound given in Proposition 4 when the solution is drawn from a small subset of elements.

▶ **Theorem 7.** *Consider a packing problem with an r-exclusion-monotone α-approximation algorithm $\mathcal{A}$. The corresponding algorithm* Maintain-$\mathcal{A}$ *computes an α-approximation with an expected number of postponements $\mathbb{E}[R] = \Theta(r \ln n/r')$, where $r' = \min(r, n-r)$.*

Proof. Consider the execution of the algorithm in rounds as discussed above. In each round, let U_i denote the number of candidates that are still undecided (i.e., either have not arrived or have been left undecided in earlier rounds). In round i we have seen exactly i candidates. Thus, given U_i undecided candidates, the expected number of postponements R^i in round i is given by a negative binomial distribution and amounts to

$$\mathbb{E}\left[R^i \mid U_i\right] = \left(\frac{U_i}{n-i} - 1\right) .$$

To bound U_i we note that, trivially, $U_i \le n$. Moreover, the number of candidates that have arrived and are undecided is $U_i - (n-i)$. Since Maintain-$\mathcal{A}$ postpones only candidates in the set D_i, we have that $U_i - (n-i) \le r$. This implies $U_i \le \min(n, n-i+r)$ and yields

$$\begin{aligned}
\mathbb{E}[R] &\le \sum_{i=1}^{r-1}\left(\frac{n}{n-i} - 1\right) + \sum_{i=r}^{n-1}\left(\frac{n-i+r}{n-i} - 1\right)\\
&= n\sum_{i=1}^{r-1}\frac{1}{n-i} - r + r\sum_{i=r}^{n-1}\frac{1}{n-i}\\
&\le n\left(\frac{1}{n-r+1} + \ln\left(\frac{n-1}{n-r+1}\right)\right) + r\left(\frac{1}{r} - 1 + \ln\left(\frac{n-1}{r}\right)\right)\\
&= \left(2 + \frac{r-1}{n-r+1} - r\right) + n\ln\left(\frac{n-1}{n-r+1}\right) + r\ln\left(\frac{n-1}{r}\right) .
\end{aligned}$$

Clearly, the first term in the bracket is at most 1. For $r \ge n-r+1$, the second term is larger than the third term and amounts to $O(r \ln {}^{n}\!/\!_{r'})$. For $r \le n-r+1$, we upper bound

$$n\ln\left(\frac{n-1}{n-r+1}\right) = n\ln\left(1 + \frac{r-2}{n-r+1}\right) \le (r-2) + \frac{(r-2)(r-1)}{n-r+1} < 2r - 4 .$$

[3] Recall that *vertices in one partition* arrive and get postponed, along with their incident edges. If *single edges* arrive and must be postponed individually, the property might not hold (c.f. Example 10 below).

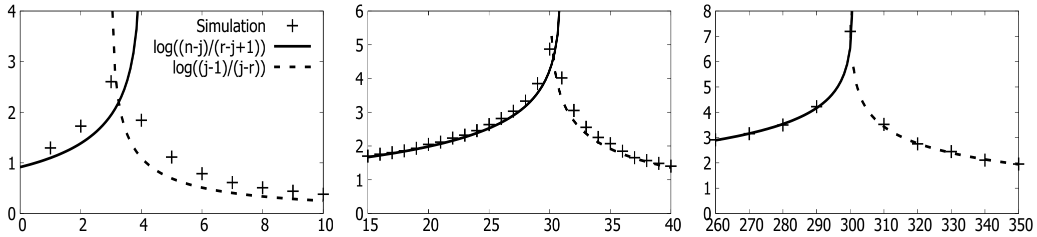

Figure 1 Number of postponements of MaintainOPT in a uniform matroid with $n = 10$ and $r = 3$ (left), $n = 100$ and $r = 30$ (middle), and $n = 1000$ and $r = 300$ (right). The x-axis is the index of the candidate in the sorted order. The y-axis shows the average number of postponements over 5000 runs. The $O(1)$-terms in Theorem 9 turn out to be small. They appear to be maximal for candidates r and $r + 1$, but seem to vanish for growing n.

Thus, the asymptotics are dominated by the third term, and $\mathbb{E}[R] = O(r \ln {}^{n}\!/_{r'})$. A similar calculation using elementary lower bounds shows that $\mathbb{E}[R] = \Omega(r \ln {}^{n}\!/_{r'})$. ◀

4.2 Matroids

We adjust MaintainOPT for *known matroids*, i.e. when the structure of the matroid is known upfront (only the weights of the elements are revealed). In this case, we can assume $r \leq n/2$, since for $r \geq n/2$ we can consider finding a minimum-weight basis in the dual matroid. We adjust algorithm MaintainOPT in the following way. Instead of postponing all elements in the current optimum until the end, we can accept some elements earlier. In particular, we can directly accept an element e as soon as there is no unseen element that can force e to leave the optimum solution. This allows to significantly improve the number of returns to below n for any rank of the matroid.

▶ **Theorem 8.** *For the class of all matroids with rank r, the expected number of postponements R in* MaintainOPT *with known matroid is maximized for the uniform matroid. It is bounded by $\mathbb{E}[R] = \Theta(r' \ln {}^{n}\!/_{r'})$, where $r' = \min(r, n - r)$. For every matroid it holds that $\mathbb{E}[R] < n$.*

Note that for any postponement problem, a simple calculation shows that the expected number of postponements of any single candidate can always be upper bounded by $\ln n + 1$. In contrast, the previous theorem shows that, on average, we need less than one postponement per candidate to compute even an optimal solution in matroids. However, they can be quite unbalanced over the candidates. We fully characterize the expected number of postponements in the uniform matroid with $r \leq n/2$. The worst candidate in the optimal solution (i.e., the r-th best candidate) asymptotically gets the largest expected number of postponements. The expected number is decreasing quickly for better and worse candidates.

▶ **Theorem 9.** *For* MaintainOPT *with known uniform matroid of rank $r \leq n/2$, the expected number of postponements R_j of the j-th best candidate is bounded by*

$$\mathbb{E}[R_j] \leq \begin{cases} \ln\left(\dfrac{n-j}{r-j+1}\right) + O(1) \ , & \text{for } j = 1, \ldots, r, \\[2ex] \ln\left(\dfrac{j-1}{j-r}\right) + O(1) \ , & \text{for } j = r+1, \ldots, n. \end{cases}$$

Based on our experiments in Figure 1 the $O(1)$ terms are small and even seem to vanish for large n. The logarithmic function captures the number of postponements rather precisely.

For matroids, the number of postponements of MaintainOPT with known matroid is always at most n. Instead, for bipartite matching the number of postponements of MaintainOPT must grow to $\Theta(n \log n)$ when r becomes large, even if the graph is known.

▶ **Example 10.** Consider a simple cycle of length $2n$ and number the vertices consecutively around the cycle. Suppose the $r = n$ even vertices form the offline partition V, and the n odd vertices arrive in random order. The edge weights can be arbitrary, but an adversary chooses them to be in $[1, 1 + \varepsilon]$. Then, unless we see all vertices, we cannot decide which of the two perfect matchings will be the optimal one. MaintainOPT needs to see all vertices to be able to decide the matching edges. We recover the coupon collector problem.

The example also applies when the *edges* of the bipartite graph are candidates that arrive in random order (rather than the vertices). In order to guarantee that an optimal solution is returned with probability 1 in the end, all $2n$ candidate edges need to remain undecided until the last unique arrival. This shows, in particular, that the bound of $O(r' \ln {}^{n}/_{r'})$ for MaintainOPT for known matroids cannot be extended to known *intersections of matroids*.

4.3 Exclusion-Monotonicity and Solution Size

For r-exclusion-monotone algorithms $\mathcal{A}$ the algorithm Maintain-$\mathcal{A}$ needs at most $O(r \ln n)$ postponements. One might hope that for any r-exclusion-monotone algorithm the parameter r is tied closely to the solution size of the algorithm. Then a large number of returns in Maintain-$\mathcal{A}$ would be caused by $\mathcal{A}$ returning a solution with many elements. This, however, is not the case – even if we are guaranteed that the size of the optimal solution is $\Theta(\log n)$, an expected number of $\Omega(n \log \log n)$ postponements for MaintainOPT can be required.

▶ **Theorem 11.** *There is a class of instances of the independent set problem with every optimal solution of size $|I^*| = 3 \ln n$, for which the expected number of postponements R in* MaintainOPT *is $\mathbb{E}[R] = \Omega(n \ln \ln n)$.*

References

1 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A Knapsack Secretary Problem with Applications. In *Proc. 10th Workshop Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 16–28, 2007.

2 Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online Auctions and Generalized Secretary Problems. *SIGecom Exchanges*, 7(2), 2008.

3 Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. 18th Symp. Discrete Algorithms (SODA)*, pages 434–443, 2007.

4 Arnon Boneh and Micha Hofri. The coupon-collector problem revisited – a survey of engineering problems and computational methods. *Comm. Stat. Stoch. Models*, 13(1):39–66, 1997.

5 Ning Chen, Martin Hoefer, Marvin Künnemann, Chengyu Lin, and Peihan Miao. Secretary Markets with Local Information. In *Proc. 42nd Intl. Coll. Automata, Languages & Programming (ICALP)*, volume 2, pages 552–563, 2015.

6 Michael Dinitz. Recent advances on the matroid secretary problem. *SIGACT News*, 44(2):126–142, 2013.

7 Uriel Feige. On Maximizing Welfare When Utility Functions Are Subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009.

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

9 Moran Feldman and Rico Zenklusen. The Submodular Secretary Problem Goes Linear. *SIAM J. Comput.*, 47(2):330–366, 2018.

10 Thomas Ferguson. Who Solved the Secretary Problem? *Statistical Science*, 4(3):282–289, 1989.

11 Amos Fiat, Ilia Gorelik, Haim Kaplan, and Slava Novgorodov. The Temp Secretary Problem. In *Proc. 23rd European Symp. Algorithms (ESA)*, pages 631–642, 2015.

12 Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier. Birthday Paradox, Coupon Collectors, Caching Algorithms and Self-Organizing Search. *Disc. Appl. Math.*, 39(3):207–229, 1992.

13 Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online Independent Set Beyond the Worst-Case: Secretaries, Prophets and Periods. In *Proc. 41st Intl. Coll. Automata, Languages & Programming (ICALP)*, volume 2, pages 508–519, 2014.

14 MohammadTaghi Hajiaghayi, Robert Kleinberg, and David Parkes. Adaptive limited-supply online auctions. In *Proc. 5th Conf. Electronic Commerce (EC)*, pages 71–80, 2004.

15 Martin Hoefer and Bojana Kodric. Combinatorial Secretary Problems with Ordinal Information. In *Proc. 44th Intl. Coll. Automata, Languages & Programming (ICALP)*, pages 133:1–133:14, 2017.

16 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions. In *Proc. 21st European Symp. Algorithms (ESA)*, pages 589–600, 2013.

17 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal Beats Dual on Online Packing LPs in the Random-Order Model. In *Proc. 46th Symp. Theory of Comput. (STOC)*, pages 303–312, 2014.

18 Thomas Kesselheim and Andreas Tönnis. Submodular Secretary Problems: Cardinality, Matching, and Linear Constraints. In *Proc. 20th Workshop Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 16:1–16:22, 2017.

19 John Kobza, Sheldon Jacobson, and Diane Vaughan. A Survey of the Coupon Collector's Problem with Random Sample Sizes. *Methodol. Comput. Appl. Probab.*, 9:573–584, 2007.

20 Nitish Korula and Martin Pál. Algorithms for Secretary Problems on Graphs and Hypergraphs. In *Proc. 36th Intl. Coll. Automata, Languages & Programming (ICALP)*, pages 508–520, 2009.

21 Oded Lachish. O(log log Rank)-Competitive Ratio for the Matroid Secretary Problem. In *Proc. 55th Symp. Foundations of Comput. Sci. (FOCS)*, pages 326–335, 2014.

22 Marco Molinaro and R. Ravi. Geometry of Online Packing Linear Programs. *Math. Oper. Res.*, 39(1):46–59, 2014.

23 Aviad Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proc. 48th Symp. Theory of Comput. (STOC)*, pages 324–332, 2016.

24 José Soto, Abner Turkieltaub, and Victor Verdugo. Strong Algorithms for the Ordinal Matroid Secretary Problem. In *Proc. 29th Symp. Discrete Algorithms (SODA)*, pages 715–734, 2018.

25 Shai Vardi. The Returning Secretary. In *Proc. 32nd Symp. Theoret. Aspects of Comput. Sci. (STACS)*, pages 716–729, 2015. Full version CoRR abs/1404.0614. `arXiv:1404.0614`.