https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–24
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

The Secretary Problem with Independent Sampling

José Correa,[a] Andrés Cristi,[b] Laurent Feuilloley,[c] Tim Oosterwijk,[d,*] Alexandros Tsigonias-Dimitriadis[e]

[a] Department of Industrial Engineering, Universidad de Chile, Chile; [b] Center for Mathematical Modeling, Universidad de Chile, Chile; [c] CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69622 Villeurbanne, France; [d] School of Business and Economics, Vrije Universiteit, 1081 HV Amsterdam, Netherlands; [e] European Central Bank, 60314 Frankfurt am Main, Germany

*Corresponding author

Contact: correa@uchile.cl, https://orcid.org/0000-0002-3012-7622 (JC); andres.cristi.e@gmail.com, https://orcid.org/0000-0002-1227-2092 (AC); laurent.feuilloley@cnrs.fr, https://orcid.org/0000-0002-3994-0898 (LF); t.oosterwijk@vu.nl, https://orcid.org/0000-0002-8509-7002 (TO); alexandrostsigdim@gmail.com, https://orcid.org/0000-0002-7558-2215 (AT-D)

Received: May 12, 2021
Revised: October 2, 2022; November 28, 2023
Accepted: April 7, 2024
Published Online in Articles in Advance: June 17, 2024

https://doi.org/10.1287/mnsc.2021.01580

Abstract. The secretary problem is probably the most well-studied optimal stopping problem with many applications in economics and management. In the secretary problem, a decision maker faces an unknown sequence of values, revealed successively, and has to make irrevocable take-it-or-leave-it decisions. Her goal is to select the maximum value in the sequence. Although in the classic secretary problem the values of upcoming elements are entirely unknown, in many realistic situations, the decision maker has access to some information, for example, from past data. In this paper, we take a sampling approach and assume that before starting the sequence, each element is sampled independently with probability p. We study both the adversarial and the random arrival models. Our main result is to obtain the best possible algorithms for both settings and all values of p. As p grows to one, the obtained guarantees converge to the optimal guarantees in the full information case. Notably, we establish that the best possible algorithm in the adversarial order setting is a fixed threshold algorithm. In the random order setting, we characterize the best possible algorithm by a sequence of thresholds, dictating at which point in time we should accept a value. Surprisingly, this sequence is independent of p. We complement our theoretical results with numerical experiments on data of people playing the secretary problem repeatedly. Our results help explain some behavioral issues they raised and indicate that people play a strategy similar to our optimal algorithms from the start onwards, albeit slightly suboptimally.

History: Accepted by Chung Piaw Teo, optimization.
Funding: This work was partially funded by the Agencia Nacional de Investigación y Desarrollo Chile [Grants FB210005 (Centro de Modelamiento Matamático), AIM23_0004 (Millenium Institute for Research in Market Imperfections and Public Policy), AFB230002 (Instituto Sistemas Complejos de Ingeniería), ACT210005, and Fondo Nacional de Desarrollo Científico y Tecnológico 1220054], the Alexander von Humboldt Foundation with funds from the German Federal Ministry of Education and Research, and the German Research Foundation within the Research Training Group Advanced Optimization in a Networked Economy [Grant Graduiertenkolleg 2201]. The views expressed in this paper do not necessarily reflect those of the European Central Bank or the Eurosystem.
Supplemental Material: The online appendix and data files are available at https://doi.org/10.1287/mnsc.2021.01580.

Keywords: online decision • secretary problem • sampling • adversarial order • random order

1. Introduction

The secretary problem, in which we search for the best secretary of an online sequence of candidates, is probably the most well-studied optimal stopping problem. These problems, motivated as *decision making under uncertainty*, are characterized by a decision maker who needs to decide when to stop an input sequence of information and take an action upon stopping. Optimal stopping problems, and in particular the secretary problem, originally arose in connection to labor markets, which is also insinuated by the name of the secretary problem. However, they have applications in many subfields of economics and management, such as monetary theory, industrial organization, e-commerce, and finance.

In finance, a well-known application of high-dimensional optimal stopping is in option pricing, such as swing and American options (Chen and Goldberg 2018, 2019; Ciocan and Mišić 2020). Ideas from optimal stopping, often closely related to prophet inequalities, have been used to design posted price mechanisms in various scenarios (Chawla et al. 2010, Chen et al. 2019, Beyhaghi et al. 2021, Ma et al. 2021). Recently, Derakhshan et al. (2021) considered the problem of computing personalized reserve prices in online advertising using a data set of past bids. Moreover, Ma et al. (2021) use techniques from optimal stopping in assortment optimization. Babaioff et al. (2007) and Kleinberg (2005) show that generalizations of the classic secretary problem serve as a framework for online auctions. In industrial organization, extensions of the secretary problem have modeled situations where a group decision within a firm has to be made (Alpern and Baston 2017), or firms are competing to hire employers from a pool of candidates (Immorlica et al. 2006, Cownden and Steinsaltz 2014). Finally, because of its simplicity and broad applicability, variations of the secretary problem have been studied experimentally. Such papers usually describe the optimal policy for the scenario they are studying, and then, through field experiments, try to explain the cognitive strategies that the agents develop. Some examples include the classical secretary problem (Seale and Rapoport 1997), a cardinal independent and identically distributed (i.i.d.) setting (Angelovski and Güth 2020), choosing which apartment to rent (Zwick et al. 2003), trying to buy a plane ticket online (Baumann et al. 2020), and learning when to stop by playing a repeated secretary problem (Goldstein et al. 2020).

Mathematically, in the secretary problem, we are faced with a randomly permuted sequence of n elements with arbitrary values. The elements' values are revealed one at a time. Upon receiving an element, we need to make an irrevocable decision of whether we keep the value and stop the sequence or drop the value forever and continue observing the next. The goal is to maximize the probability of stopping with the largest value. For this problem, the best possible success guarantee has long been known to be $1/e$. The optimal algorithm is remarkably simple: Look at the first n/e values without taking any of them, and then stop with the first value larger than all values seen thus far (Lindley 1961, Dynkin 1963, Ferguson 1989). In the last decades, the secretary problem, its variants, and related basic optimal stopping problems such as the prophet inequality and the Pandora's box problem have been considered fundamental building blocks of online selection problems (Krengel and Sucheston 1977, 1978; Weitzman 1979; Doval 2018; Beyhaghi and Kleinberg 2019).

An essential limitation of the secretary problem for modeling real-world situations is the assumption that the values of the elements that have not yet been revealed are completely unknown. This is a very pessimistic assumption, as in realistic situations, one would expect to have some available information, coming, for instance, from the context or past data. As a consequence, the best possible $1/e$ success probability for the secretary problem can be substantially improved in many settings. This gives rise to the following natural question: What is a reasonable model to take into account this additional available information? A first approach is to assume that the numbers originate from a distribution that is known to the algorithm. This assumption is relevant when the process at hand has been repeated many times, and past data can be aggregated into a distribution. Along these lines, already in the 1960s, Gilbert and Mosteller (1966) considered the so-called full information secretary problem in which we additionally know that the elements' values are i.i.d. random variables from a known distribution. For this variant, they showed how to compute the optimal stopping rule by dynamic programming and were able to conclude, numerically, that the best possible success probability is $\gamma \approx 0.5801$. In subsequent work, Samuels (1991) finds an explicit expression for this quantity. Esfandiari et al. (2020) relaxed the i.i.d.-ness assumption, considering the problem when the elements' values are arbitrary independent random variables. They show that one can guarantee a success probability of 0.517, which, quite surprisingly, was very recently improved to γ by Nuti (2022). Interestingly, in this full information model with independent but not necessarily identical values, Allart and Islas (2015) showed that if the order is not random but adversarial, the optimal stopping rule guarantees a success probability of $1/e$.[1]

Although assuming no knowledge about the values seems too pessimistic, assuming that the full distribution is known might be too optimistic for most scenarios. Indeed, a typical situation would be that we have access to past data but not enough to safely reconstruct a distribution. These informational issues in optimal stopping have given rise to a stream of research aiming at understanding the relationship between the amount of information available and the success probabilities that can be derived. In this context, Azar et al. (2014) pioneered the study of data-driven versions of optimal stopping problems. Recently, Rubinstein et al. (2020) established a notable result in this direction for the classic prophet inequality.[2] They prove that a single sample from each distribution, rather than its full knowledge, is enough to achieve the optimal guarantee. Also, for the prophet secretary problem, the variant of the prophet inequality when the elements come in random order, one sample has been proven to be quite effective (Correa et al. 2020, Nuti and Vondrák 2023).

However, this *sampling* approach still assumes that there is an underlying distribution from which we can effectively sample. In many situations, this assumption may be strong, and ideally, we would like to combine the idea of having samples representing past data with having arbitrary values chosen adversarially, to ensure maximum robustness while requiring no distributional assumption. Recently, Kaplan et al. (2020) study such a model.[3] In their model, there are n arbitrary values, and they sample a fraction p of them at random. Then the nonsampled values are presented to the decision maker in either random order or adversarial order. Kaplan et al. (2020) design algorithms for maximizing the expectation rather than the probability of picking the maximum that translate into algorithms for data-driven versions of prophet inequalities.

In this paper, we consider an alternative sampling model, inspired by that of Campbell and Samuels (1981) and Kaplan et al. (2020). The main difference is that in our model, the sampling of each element is performed independently with the same fixed probability p. In other words, roughly a p-fraction of the elements are considered to be samples and revealed to the player upfront, before the nonsampled elements are revealed one by one, whose maximum value the player aims to obtain. Such data-driven versions are well motivated from several perspectives. First, in many applications, the decision maker has access to historical data that gives some insight into the distribution of future values. In our model, this information is captured in the form of samples that the decision maker knows a priori. Second, the model is robust in the sense that only minimal knowledge of the involved data are needed. Third, the general idea is closely related to machine learning methods that use predictors to learn the distribution (Goodfellow et al. 2016). The insight here is that for problems that can be modeled as data-driven versions of the secretary problem, these learning procedures are overly complicated: The simple combinatorial model presented in this paper already makes it possible to increase the solution quality even with modest sampling.

Of course, for large n our model is essentially equivalent to the model of Campbell and Samuels (1981) and Kaplan et al. (2020). However, our independent sampling has two crucial advantages. On the one hand, independence makes many mathematical calculations a lot simpler and thus allows to obtain simpler expressions. It allows dealing with instances of unknown size, which is often the case in practical applications. In particular, several of our results hold if we do not know n. A slight disadvantage of the independent sampling model is that we may end up sampling all n elements. For consistency in this case, we assume, by vacuity, that we *win* (i.e., pick the maximum). However, this is not very restrictive because, as we will see, the difficult instances involve large values of n for a fixed value of p.

Our main result is to obtain the best possible algorithms, that is, those maximizing the probability of selecting the largest element, for any prescribed sampling probability p. We present results for both the setting in which the order in which the elements are presented is random and for the adversarial order setting. These results uncover interesting relationships between the quality of the solution and the amount of past data available to a decision maker.

1.1. Problem

We are given n elements with values $\alpha_1, \dots, \alpha_n$, which are unknown to us, and an order $\sigma : [n] \to [n]$, where $[n] = \{1, \dots, n\}$. Each element is sampled independently with probability p. Let S be the (random) set of sampled elements and V be the remaining elements, also referred to as the online set or the set of online elements. First, the set S of sampled elements are revealed to us. Then the elements in V are presented to us in the order dictated by σ. Once an element is revealed, we either pick it and stop the sequence, or drop it forever and continue. The goal is to maximize the probability of picking the maximum valued element in V. In particular, it is not allowed to pick an element of S, which is justified by the fact that we consider S to represent past data. In the *adversarial order secretary problem with p-sampling* (AOSp), the order σ is chosen by an adversary that knows all values $\alpha_1, \dots, \alpha_n$ and the random sets S and V.[4] In the *random order secretary problem with p-sampling* (ROSp), the order σ is just a uniform random permutation.

Given n and an algorithm, we define its *success probability* as the infimum over all values $\alpha_1, \dots, \alpha_n$ of the probability that the algorithm stops with the maximum $\alpha_i \in V$. Moreover, the *success guarantee* of an algorithm is the infimum over all values of n of its success probability.

All algorithms considered in this paper are *ordinal*, that is, algorithms whose decision to stop at a given point depend only on the relative rankings of the values seen thus far and not on the actual values that have been observed, plus, possibly, on some external randomness. We observe that this is without loss of generality as for AOSp and ROSp general algorithms cannot perform better than ordinal algorithms. Indeed, as noted by Kaplan et al. (2020, theorem 2.3), a result of Moran et al. (1985) implies the existence of an infinite subset of the natural numbers where general algorithms behave like ordinal algorithms (for single selection ordinal objective functions such as ours). Therefore, and because the worst-case performance of our algorithms is attained as $n \to \infty$, our bounds apply to general algorithms; see also Theorem 1.

1.2. Our Results

For AOSp, we consider a very simple algorithm, that we call the k-max algorithm. Upon observing the sample set S it sets a threshold equal to the value of its kth largest element for $k = \lfloor 1/(1-p) \rfloor$. Then it accepts the first element in V whose value surpasses the threshold. If there are less than k samples, the algorithm accepts the first online value (we define the kth largest element from a set of less than k elements as $-\infty$).

We show that this algorithm achieves a success guarantee of $\lfloor 1/(1-p) \rfloor p^{\lfloor 1/(1-p) \rfloor}(1-p)$, so for instance, for $p = 1/2$, the guarantee evaluates to $1/4$. Although the proof of this fact is relatively easy, what is more surprising is that this guarantee is best possible. To prove the latter, we analyze a related optimal stopping problem, which we call the *last zero problem*. Suppose an adversary picks a number of identical blank cards n. Then, independently with probability p, each card is marked and you are informed about the total number of marked cards, but you ignore their position in the deck. Finally, one by one, you get to see the cards and whether they are marked or not. When you stop the sequence, you win if the card was the last blank card; otherwise, you lose. By using a related conflict graph over possible sequences, we show that for this problem, no ordinal algorithm can guess the last blank card with probability better than $\lfloor 1/(1-p)\rfloor p^{\lfloor 1/(1-p)\rfloor}(1-p)$. Then, we relate this problem to a different one, in which the objective is to guess the last number of an increasing sequence of unknown length. Finally, we go back to the original AOSp by considering an adversary that picks a growing sequence which at some point in time decreases to a low value, and this time is difficult to guess.

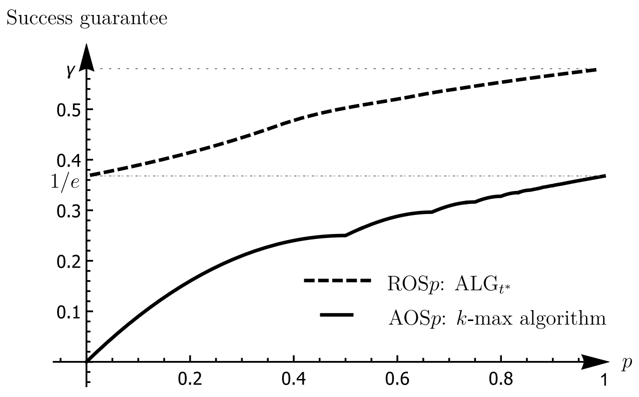
It is worth noting that this simple best possible algorithm does not use knowledge of n and, as opposed to most variants of the secretary problem, for AOSp knowledge of n is irrelevant in worst-case terms. Moreover, we discuss the case in which n is known but p is unknown. Here it is quite natural that the algorithm works again by simply estimating p using the size of the sample set. However, if neither n nor p are known, then no nontrivial success guarantee can be obtained.

For ROSp, we obtain a randomized algorithm with best possible success guarantee that works as follows. First, we assign to each of the n elements a uniformly random arrival time in the interval $[0,1]$, which implies that the elements arrive in uniform random order. All elements whose arrival time is less than p are placed in the sample set S. Then we find a sequence of time thresholds $0 < t_1 < t_2 < \cdots < 1$, dictating that if an element's arrival time is between t_i and t_{i+1}, the algorithm stops if its value is the maximum among elements arriving after p and it is among the i largest values of all elements seen so far. To obtain the success guarantee of this algorithm we first prove that for a fixed sequence $0 < t_1 < t_2 < \cdots < 1$, the success guarantee of the algorithm decreases with n. Then we write the optimization problem over the time thresholds, and interestingly, this turns out to be a separable concave optimization problem with a fairly simple solution. Moreover, the solution is universal in the sense that it does not depend on p. The resulting guarantee is thus easily computed and grows from $1/e$ when $p = 0$ to $\gamma \approx 0.58$ as $p \to 1$.[5] We also prove that this is a best possible algorithm. To this end, we first argue that ordinal algorithms in our model are essentially equivalent to a ranking function that determines what *global* ranking an element, which is a *local* maximum, should have to accept it. Here, by global ranking, we mean the ranking an element has among all samples and values revealed so far, and local ranking refers only to the values revealed and not to the samples. Finally, as n grows, this ranking function converges to a sequence of time thresholds as we defined them.

Figure 1 illustrates the success guarantee for our problems. For AOSp, it can be observed that the success guarantee can be bounded below by the function $p^{1/(1-p)}$ and bounded above by $(p-1)/(\log p)\cdot p^{-1/\log p}$ (see also the Online Appendix for details).

Finally, we demonstrate some of the strengths of our algorithms in practice, by evaluating them on the real-world data set of Goldstein et al. (2020). In particular, we first show that our algorithm for ROSp can help explain some behavioral issues raised by Goldstein et al. (2020); we then perform a sensitivity analysis on the k-max algorithm to test its robustness. Goldstein et al. (2020) set up an experiment in which people play repeated secretary problems. The values come from any of three possible distributions unknown to the players (the distribution is fixed for all games played by a person). They analyze a total of 48,336 games played by 6,537 players. Among other issues, Goldstein et al. study how close to optimal people play. However, they find difficulty in establishing what optimal means in their context, because for the first game that players played, optimal means simply the secretary algorithm, but after playing many games, optimal should mean something close to the dynamic program of Gilbert and Mosteller (1966). They thus consider several candidate models for the players' behavior and conclude that the closest to actual play is a multithreshold algorithm that is very much in the spirit of that of Gilbert and Mosteller. Interestingly, they find that by the fifth game, players have essentially learned the optimal thresholds (Goldstein et al. 2020, figure 9). However,

Figure 1. Best Possible Success Guarantee for ROSp and AOSp as a Function of p



they also find an apparent dichotomy between the strategy players use in the first few games and that used later on. Indeed, they state that "One possible explanation for the apparent change in strategy is that players spent the first few games primarily collecting information about the distribution and then switched to trying to actually win the game only in later games: that is, they spent the first few games exploring and then switched to exploiting only later."

Our results for ROSp, being optimal in a closely related model, shed more light on the players' strategies and strengthens existing insights in the work of Goldstein et al. (2020). To see this, observe that the first game the players face is just the normal secretary problem, or ROS0, the second closely corresponds to ROS$\frac{1}{2}$, the third to ROS$\frac{2}{3}$, and so on. With this observation, we are able to directly compare the performance of the players' strategies with that of our algorithms. Our experimental evaluation leads to two main conclusions: (a) the players' strategies are strongly correlated with our algorithm (i.e., players use similar, suboptimal series of thresholds), and (b) a good fraction plays near-optimally from the start; the players, in any case, improve their performance in the first few games by learning how to optimally use the information they gain (as Goldstein et al. (2020) also observe) and at some point their success rate stabilizes. For the sensitivity analysis of the k-max algorithm we observe what theory suggests: a wrong estimation of the parameter p in the first games does not affect the success rate much due to the robustness of the algorithm, but mistakes later on can be very costly, especially when p is overestimated.

A final remark we would like to make is the fact that our models and methods are not limited to the secretary problem. In the random order case, Azar et al. (2014) observed that many algorithms only use the random order to realize a random sample from the input and are thus order oblivious. Such a random sample is simply given in our model, and as such, existing order-oblivious algorithms can be adapted to the adversarial order secretary problem using our approach. On the other hand, by combining our approach for the secretary problem with algorithms for online problems in the random order case, we can find algorithms for diverse online problems. As an example, our results for the random order secretary problem can be extended easily to the weighted bipartite matching problem with the same performance guarantees by using the results of Kesselheim et al. (2013).

1.3. Further Related Literature

An interesting connection arises between our model and results when p is close to one and the so-called full information case. First, recall that Gilbert and Mosteller (1966) obtained the optimal algorithm with worst-case performance γ (Samuels 1982, 1991), in the secretary problem where the elements' values are taken as i.i.d. random variables from a known distribution. It may thus seem natural that our guarantee matches this quantity as $p \to 1$. However, this is far from obvious. Indeed, for the prophet inequality with i.i.d. values from an unknown distribution (a model that arguably gives more information than ours), Correa et al. (2019) proved that with $O(n^2)$ samples, one can achieve the best possible performance guarantee of the case with known distribution, and only very recently Rubinstein et al. (2020) improved this to $O(n)$ samples. This is in line with our result here because for p close to, but strictly less than one, the size of the sample set is linear in the size of V.

Still in the random order case, Correa et al. (2023) study the same sampling model we discuss in this paper but with the objective of maximizing the expected value of the chosen element. They obtain best possible ordinal algorithms for all values of p and the implied guarantees grow from $1/e$ when $p = 0$ to 0.745 when p tends to one, which is the optimal guarantee for the i.i.d. prophet inequality (Hill and Kertz 1982, Correa et al. 2021b), because in both cases the optimal guarantees converge to those of the full information case.

A more intriguing connection to the full information case pops up in the adversarial order case. In this context, Allart and Islas (2015), and independently Esfandiari et al. (2020), considered the adversarial order secretary problem in which an adversary chooses n distributions $F_1, \dots, F_n$. Then, independent values are drawn from these distributions and sequentially uncovered. A decision maker who knows $F_1, \dots, F_n$ needs to stop at the maximum realization. They prove that the optimal stopping rule is a simple single threshold algorithm, and the best possible success guarantee equals $1/e$. Although this problem has a similar flavor as our AOSp, and the optimal guarantee is the same, we are unaware of a precise connection.

On the other hand, our last zero problem, used as a tool for AOSp, is related to an old optimal stopping problem first studied by Bruss (2000). We face a sequence of n independent Bernoulli random variables where we know n and the distributions, and we want to stop with the last zero. Bruss obtains the optimal stopping rule for this problem, which also turns out to be a simple threshold rule. Our last zero problem is simpler in that the Bernoulli random variables are homogeneous. However, rather than knowing n, we only know the total number of ones. This subtle difference makes the problem substantially different.

The line of research exploring the use of additional information to improve solutions to problems in online decision making has gained momentum over the past years (Mahdian et al. 2012, Lykouris and Vassilvitskii 2021, Golrezaei et al. 2022). Our sampling approach can be considered in this setting as a case of the secretary problem with advice based on past data. Dütting et al.

(2021) study this problem from a more general perspective and use a factor revealing linear program to gain structural insight into the optimal policy, depending on the type of advice the algorithm is given.

Another recent line of work studies robust or semi-random versions of the classical secretary problem (Bradac et al. 2020, Kesselheim and Molinaro 2020). The main idea is that the problem input should be a mix of stochastic and adversarial parts. More specifically, in their (similar) models, some of the elements arrive at adversarially chosen times, and the rest at times uniformly randomly drawn from $[0,1]$. Their objective functions (and in some cases also the benchmarks) are quite different from ours. Kesselheim and Molinaro (2020) consider the knapsack secretary problem in this mixed model, whereas Bradac et al. (2020) design algorithms for selecting k items or maximizing the expectation under various matroid or knapsack constraints. It would be interesting to incorporate their ideas in our setting and study a problem that interpolates between ROSp and AOSp. The idea of mixing stochastic and adversarial inputs in online resource allocation has appeared earlier (Esfandiari et al. 2018, Hwang et al. 2021).

Outline of the Paper. Section 2 presents the model and some basic definitions formally. Then, Section 3 presents an overview of the techniques and results for the adversarial order case, and Section 4 does the same for the random order case. Section 5 follows with some insights into the results that can be obtained if we assume different knowledge of the parameters. Our numerical experiments can be found in Section 6. We close with a discussion about possible extensions of our results in Section 7. Most of the full proofs are deferred to the Online Appendix.

2. Model and Definitions

Let $p \in [0,1]$. We consider the following game between a player and an adversary. The game takes place in several phases.

- Phase 1: The adversary chooses an integer n and a set U of n integers.
- Phase 2: This phase is the only place where the game is different in AOS_p and ROS_p. In AOS_p the adversary chooses a permutation σ for U. In ROS_p σ is a random permutation, where each permutation is equally likely. U is then ordered according to σ.
- Phase 3: Every element of U is added to the sample set S independently with probability p, and otherwise it is added to the set $U \setminus S$ of online elements.
- Phase 4: The sample set is revealed to the player.
- Phase 5: The elements of the online set are presented to the player one after the other, according to the order σ. The player chooses at each step to continue or to stop the game. If the player continues, the next element is presented to her.

An instance of the game can be denoted as (U, σ, S), where U is the set of n values, σ is the permutation and S is the result of the sampling process. In particular, it is a sequence of values from U, presented in order σ, where the first $|S|$ values are considered to be sampled elements. We will identify an instance of the problem with such a sequence of values. A prefix r of the sequence then corresponds to the game until a certain decision point: $|r| < n$ values have been revealed at this time, and the player currently has the choice to stop the game at this last element of the prefix or to continue with the next, currently unrevealed, element.

We say that a player *wins* or an algorithm *succeeds* in an instance of the problem if it stops on the largest element of the online set $U \setminus S$. This allows us to formally define the success guarantee of an algorithm.

Definition 1. The *success guarantee* of a deterministic algorithm A for AOS_p is

$$\inf_n \min_{U,\sigma} \mathbb{P}_S(A \text{ succeeds on } (U,\sigma,S)),$$

where $\mathbb{P}_S$ takes the probability over the sampling phase. For ROS_p it is defined as

$$\inf_n \min_{U} \mathbb{P}_{S,\sigma}(A \text{ succeeds on } (U,\sigma,S)),$$

where $\mathbb{P}_{S,\sigma}$ takes the probability over the sampling phase and the permutation.

Observe that the success probability is unconditional on the sample set S.

We use a minimum for U, σ even if there is potentially an infinite number of possible sets U. This is because of the following result. This theorem is the analogue of theorem 2.3 in Kaplan et al. (2020), and the proof is essentially the same. For completeness, and because this statement is central to the paper, we sketch the following approach.

For the proof, we need the following definitions similar to Kaplan et al. (2020). We call two sequences $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ *order-equivalent* if $x_i < x_j \Longleftrightarrow y_i < y_j$ for all $1 \leq i, j \leq k$. The equivalence class of $x_1, \ldots, x_k$ is called its *order-type*. We say that a function is *order-invariant* on a set C if the value of the function on a sequence of elements from C depends only on the order-type of the sequence.

Theorem 1. *Suppose there exists an $\alpha \in [0,1]$ such that no ordinal algorithm can achieve a success guarantee of at least α for AOSp or ROSp. Then no cardinal algorithm can achieve a success guarantee of at least α for AOSp or ROSp.*

Proof. Any algorithm ALG for AOSp or ROSp can be described as a series of n functions $f_t : \mathbb{R}^{2n} \longmapsto \{0,1\}$.

Indeed, we can encode both the sample set and revealed values of the online set as sets of at most n reals (padding with zeros for example), and the decision (stop or continue) as a bit. Then f_t represents the function used for the decision at the tth step.

The functions f_t can be considered predicates defined on $\mathbb{R}^{2n}$. Because $\mathbb{R}$ is totally ordered, theorem 3.3 in Moran et al. (1985) ensures that there exists a subset $C \subset \mathbb{R}$ such that each function f_t is order-invariant on C^{2n}. Therefore, in any instance on this subset C, *ALG* only uses ordinal information and is, as such, an ordinal algorithm. Indeed, on this subset C, any cardinal algorithm behaves as an ordinal algorithm.

As a consequence, if on a fixed order of elements of C, the highest success guarantee any ordinal algorithm can achieve is α, then the success guarantee of any cardinal algorithm on the same order is also at most α. Thus, in this case, any cardinal algorithm has a success guarantee of at most α for AOSp or ROSp. The result follows. □

We will see that our positive results are actually also ordinal, and that they match the negative bounds. In particular, once we restrict to ordinal algorithms, we can assume that the input sequence is a permutation of $\{1, \dots, n\}$.

3. Adversarial Order

In this section, we study the adversarial order secretary problem with p-sampling (AOSp). We present the k-max algorithm and prove that it is optimal (in the worst-case sense) for this setting. All omitted proofs can be found in the Online Appendix.

Recall that we defined the *k-max algorithm* as in Algorithm 1: We set the kth largest value of the set S of sampled elements as a threshold, and the algorithm accepts the first element in the set V of online values whose value surpasses this threshold. If $|S| < k$, that is, if there are less than k sampled elements, then the algorithm accepts the first online element.[6] It turns out that this decision does not prevent us from achieving the success guarantee we aim for, as this case occurs with small probability only. From now on, we define $k = \lfloor 1/(1-p) \rfloor$ as the value of k for the k-max algorithm. This section is dedicated to proving the following theorem.

Algorithm 1 (k-Max Algorithm (for the Optimal k))

```
  $k = \lfloor \frac{1}{1-p} \rfloor$.
  if $|S| \geq k$ then
      $T \leftarrow$ the $k$-th largest value of $S$.
  else
      $T \leftarrow -\infty$
  end if
  while Current value $< T$ do
      Discard the value.
  end while
  Accept the current value (if there are no more values left, accept nothing).
```

Theorem 2. *Let $k = \lfloor 1/(1-p) \rfloor$. The k-max algorithm achieves a guarantee of $kp^k(1-p)$ for AOSp. Furthermore, no algorithm can achieve a better success guarantee.*

Naturally, when p tends to zero, the guarantee naturally tends to zero: If there are very few samples, the problem reduces to the secretary problem with adversarial order, which does not allow for any nontrivial success guarantee. What is more surprising is that when p is close to one, the success guarantee approaches $1/e$ (Figure 1), which is the performance obtained for the secretary problem with full knowledge of the distribution of the values of the elements (Allart and Islas 2015, Esfandiari et al. 2020).

The proof of the success guarantee of the algorithm is easy and appears in Section 3.1. The proof of its optimality is more advanced and requires new tools. Section 3.2 first introduces the new concepts we require before diving into the proof. A surprising fact of this proof is that it suffices to focus on the special case where the values of the elements are in nondecreasing order (thus where the algorithm aims to stop at the last element), with the twist that the algorithm is size-oblivious; that is, it does not know the size n of the instance.

3.1. Success Guarantee of the *k*-Max Algorithm

As the success guarantee of the k-max algorithm clearly depends on the value of k, we need to specify its value. Intuitively, the bigger the value of p, the higher the probability that the largest valued elements are sampled. Therefore, a larger value of p suggests to use a lower threshold. As is common for many threshold algorithms, there is a tradeoff between (1) setting the threshold too low and risking acceptance of an element that does not have the maximum online value and (2) setting it too high and risking finishing the game without selecting any element, which happens with probability p^k in our algorithm. The following lemma is the first part of Theorem 2 and establishes the success guarantee of the algorithm for the value $k = \lfloor 1/(1-p) \rfloor$.[7]

Lemma 1. *The k-max algorithm chooses the element of the online set with maximum value with probability $kp^k(1-p)$. In particular, for $k = \lfloor 1/(1-p) \rfloor$, its success guarantee is $\lfloor 1/(1-p) \rfloor p^{\lfloor 1/(1-p) \rfloor}(1-p)$.*

Proof. The k-max algorithm succeeds in an instance if *exactly one* of the k largest values of the adversarial input ends up in the online set and the $(k+1)$st largest ends up in the sample set. For the purpose of analysis, assume the elements are sequenced in nondecreasing order of their values. Thus, an instance in which the algorithm is successful is exactly a sequence ending in k sampled elements plus one online element that is somewhere in the last k entries of the sequence.

The probability that this happens equals $kp^k(1-p)$ because of the independent sampling. The second part of the lemma follows by substituting the value $k = \lfloor 1/(1-p) \rfloor$. □

3.2. Negative Result

We now focus on the proof that no algorithm can achieve a better success guarantee for AOSp than the k-max algorithm. We prove this bound for another setting, and then reduce this setting to AOSp in four steps. For the first step, we introduce a game called the last zero problem (Definition 4) in this section. We show that for $p = 1/2$ no deterministic ordinal size-oblivious (Definition 2) algorithm can have a success guarantee larger than what the k-max algorithm achieves for AOSp (Proposition 3). For the second step, in the Online Appendix companion we show how to generalize this to general values of p, randomized algorithms, and algorithms that are not size-oblivious. Therefore, this shows the bound for ordinal algorithms for the last zero problem. The third step is Proposition 1, which shows that this implies this negative result for ordinal algorithms for what we call the increasing case of AOSp, which is a special case of AOSp. This implies that the negative result also holds for ordinal algorithms for AOSp. Finally, Theorem 1 completes the fourth step by showing that the bound must then also be true for cardinal algorithms for AOSp.

3.2.1. Last Zero Problem.

We now focus on the proof for the negative result of Theorem 2. We will first prove it for the special case of $p = 1/2$ for deterministic size-oblivious algorithms. Size oblivious is a concept that intuitively entails that an algorithm "does not know n," that is, is unaware of the size of the instance. Informally, this implies that an algorithm that is presented with a certain sequence of elements and then needs to make a decision needs to make the same decision that it would have taken in other instances that revealed the same elements in the same order up to that point. Consider an algorithm and two instances I_1 and I_2 of different sizes n_1 and n_2, respectively, but with the same value of p. Suppose that until a certain point in the process of the revelation of the sequences, the algorithm happens to face the exact same set of samples and non-sampled elements in both instances, and is currently facing an online element of the same value in both instances. Thus, up to this point, the algorithm has access to exactly the same information (and possible beliefs over the size of the instance). Therefore, the algorithm needs to make the exact same (possibly randomized) decision in both situations, independent of n_1 or n_2. We formalize this notation as follows.

Definition 2. An algorithm *is size oblivious* if for a fixed value p and any pair of input sequences s_1 and s_2 of unequal length that have the same prefix of length $r \le \min\{|s_1|, |s_2|\}$, the algorithm will take the same decision after reading the prefix of length r in both sequences.

From now on, assume that any algorithm we consider is size oblivious.

For our main steps, the analysis starts by introducing the *last zero problem*. It turns out that a negative result for the last zero problem implies a negative result for AOSp under certain additional assumptions. The proof continues by removing these assumptions one at a time, until we retrieve the proof of Theorem 2 for size-oblivious algorithms. Finally, we generalize the proof to algorithms that are not size-oblivious, by showing that an algorithm cannot achieve a higher success guarantee with knowledge of the size of the instance (in worst case terms).

To introduce the last zero problem, we start by introducing two notions defined on bit strings.

Definition 3. The *norm* of a string s of bits is $\|s\|_1$, that is, the number of ones s contains. The total number of bits of such a string s is called its *length* and is denoted by $|s|$ or simply n.

The numbering of the entries of a string is counted starting from one. We will consistently use length for a bit string and size for an instance of the last zero problem, which we can now formally define.

Definition 4. For a fixed value of $p \in [0, 1]$, the *last zero problem* is defined as follows.

- Phase 1: An adversary picks a length n.
- Phase 2: A string of bits of length n is generated, where in each position independently the bit equals one with probability p and zero otherwise.
- Phase 3: The norm of the string is revealed to the player.
- Phase 4: The bits of the string are presented to the player one after the other. The player chooses at each step to continue or to stop the game. If the player continues, the next bit is presented to her.

The player wins if she stops on the last zero of the string.

Our restriction to only consider size-oblivious algorithms at the moment extends to the last zero problem, where the algorithm intuitively does not know the length of the string. Observe that this is crucial, as the game becomes trivial if this is not the case. Thus, we need not analyze the success guarantee of an algorithm for the last zero problem for a given size, but we need to prove that no algorithm can perform well on instances of any size. The success guarantee of an algorithm for the last zero problem is defined as follows.

Definition 5. The success guarantee of a deterministic algorithm A for the last zero problem is

$$\inf_{|s|} \mathbb{P}_s(A \text{ succeeds on } s),$$

where s denotes the random bit string.

Recall that the reason we introduced the last zero problem is because negative results for this problem imply negative results for AOSp under certain assumptions. The following proposition makes this formal. From now on, when we talk about the *increasing case* of AOSp, we refer to the special case of the AOSp problem in which the elements are revealed in nondecreasing order of their values.

Proposition 1. *Suppose there exists an $\alpha \in [0,1]$ such that no algorithm can achieve a success guarantee of at least α for the last zero problem. Then no size-oblivious ordinal algorithm can achieve a success guarantee of at least α for the increasing case of AOSp.*

Proof. We prove the statement by contraposition. Suppose that there exists a size-oblivious ordinal algorithm A that achieves a success guarantee of at least α for the increasing case of *AOSp*. We design an algorithm B for the last zero problem with success guarantee α.

At the start, B receives the number k of 1s in the sequence. It builds an increasing sequence *Seq* of $k+2$ values: $s_0 < s_1 < \cdots < s_k < s_{k+1}$. Then it feeds the subsequence $s_1 < \cdots < s_k$ (i.e., *Seq* with both extreme values removed) to A as a sample set of size k. It also initializes a counter c to 0, and creates an empty list L. In the online part, the algorithm B will receive bits, add new values in L, and call A when needed. More precisely, for every bit arriving, B does the following:

- If the bit is one, then it simply increments c, and continues.
- If the bit is zero, it adds a new element at the end of L, whose value is strictly between s_c and s_{c+1}, such that it is the largest value of L. Then it calls A with L as the list of online values seen so far (and sample set *Seq*), and B stops if and only if A stops.

Now consider an instance $I \in \{0,1\}^n$ of the last zero problem, and the following instance I' for the increasing case of *AOSp*: the full sequence is $1,2,\ldots,n$, among which the samples are $S = \{i | s_i = 1\}$. Because A is an ordinal size-oblivious algorithm, B succeeds on I if and only if A succeeds on I' with sample S. Hence,

$$\mathbb{P}_s(B \text{ succeeds on } s)$$
$$= \mathbb{P}_S(A \text{ succeeds on } I' \text{ with samples } S).$$

Because we assumed that the success probability of A is at least α, it must also be the case for B, which proves the proposition. □

For the remainder of this section, we consider the last zero problem. We identify an instance of the last zero problem with its bit string. We introduce the shorthand notation 0^ℓ and 1^ℓ for the string of length ℓ consisting of only zeros and ones, respectively.

3.2.2. Warm Up: Intuitive Proof for the Case $p = 1/2$.

As a warm up, we first present the negative result for deterministic size-oblivious algorithms for the special case of the last zero problem in which $p = 1/2$. For the sake of the proof, we also introduce the *no-zero rule* that specifies that if there are no online elements (i.e., all n elements are sampled), the algorithm fails. Indeed, at first sight this might contradict the assumption for AOSp that an algorithm succeeds in such an instance. However, as we will see, this rule actually becomes irrelevant for the generalization of the proof. Therefore, the use of this assumption in the analysis for the last zero problem does not pose a problem for the results for the AOSp problem, which uses the different assumption.

The following proposition is the first of a series of propositions, each one a generalization of the previous, until the proof of Theorem 2 is complete. The goal of the discussion and the proof sketch presented here is to informally introduce the tools required for the full proof.

Proposition 2. *For the last zero problem with $p = 1/2$ and the no-zero rule, no deterministic size-oblivious algorithm can achieve a better success guarantee than the k-max algorithm for AOSp.*

Proof. For $p = 1/2$, the k-max algorithm achieves a guarantee of $1/4$. Therefore, for the sake of contradiction, suppose that there exists an algorithm A for the last zero problem with $p = 1/2$ and the no-zero rule that achieves a guarantee strictly better than $1/4$. As a start, consider the decision of A when the adversary chooses $n = 1$. Then, there are two instances after sampling that both occur with probability $1/2$. The first case is that the instance is zero. Then, as the norm of the instance is revealed to A upfront, it knows that there is no one in the instance and is first presented a zero. The second case is that the instance is one. Then A knows there is a one in the instance, and it is announced from the start that the game is finished.

Recall that the success guarantee of A is defined as the infimum over all possible string lengths (instance sizes). Now observe that A fails in the second case because of the no-zero rule. Thus, to achieve at least $1/4$ for every length, A needs to succeed in the first case (recall that we restrict ourselves to deterministic algorithms for now). Because A is size oblivious, this implies that it needs to stop on the single zero in that instance. In other words, when it is revealed to A that the norm of the string is zero, it stops on the first bit.

Here comes the key observation. Suppose that the adversary chose $n = 2$ and the sampling resulted in the instance 00. Also in this instance A is presented with not a single one, and again observes a first zero. From earlier, we already deduced that A needs to stop at this first zero. Indeed, from the point of view of A, this is exactly the same situation as in the case where the instance was zero, because A is size oblivious. In other words, these two situations are *indistinguishable* for A, and it has to make the same decision. In the case of 00, this decision is wrong as the last zero is the second zero, hence A fails. We call such a situation a *conflict* between instances 0 and 00.

Conflict works in both directions. If an algorithm has a strategy that makes it succeed in 00, then after the first 0, it would wait, which would make it fail in the instance 0.

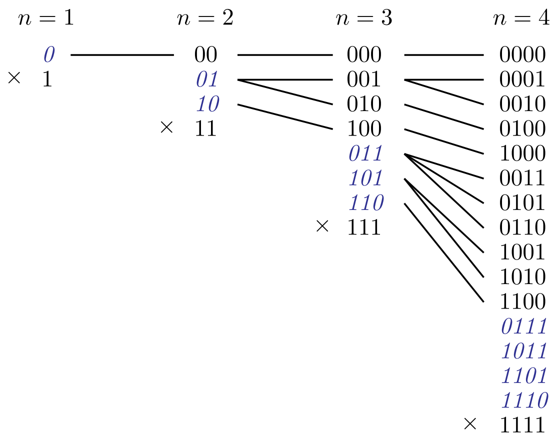
Another conflict occurs between the instances 01 and 001. On instance 001 one 1 is revealed to A and then it observes a first 0. This is exactly the same information as at the beginning of the instance 01. If A stops on this element then it succeeds in 01 but loses in 001. Conversely, if A waits and then stops on the next 0, it fails in 01 but wins in 001. Moreover, if A continues to wait it fails in both instances.

More generally, for every pair of instances there is a fairly simple criterion in each of the two directions to see if they are in conflict or not (Lemma 4). In particular, it turns out to be sufficient to decide the conflict between instances whose sizes differ only by one. Indeed, two instances s and s' of size n and $n + q$, respectively, are in conflict if and only if there is a series of conflicts (s, s_1), $(s_1, s_2), \ldots, (s_{q-1}, s')$, where s_i has size $n + i$ (Lemma 3). Then we can define the (infinite) *conflict graph* whose nodes represent all possible instances and the edges represent the conflict between nodes whose size differ by exactly one. From now on, we identify a node of the conflict graph with the corresponding instance and bit string. The conflict graph for size $n = 1$ to $n = 4$ is represented in Figure 2. In this graph, we can represent an algorithm as a subset of instances in which it succeeds. Such *selected instances* cannot be in conflict. In other words, they cannot be connected by a monotone path, where monotone means that the path traverses at most one node of each instance size.

For $n = 1$, the cross denotes that A will never succeed in the instance that consists of one 1, because of the no-zero rule. We write zero in cursive to denote that A succeeds in this instance, as it decides to select the last (and only) zero.

Now consider all instances of size two. Given that the size is two, each of them is equally likely to be the outcome of the sampling process, hence, each occurs with probability 1/4. Both 00 and 11 cannot be selected (because of the conflict to the left and the no-zero rule,

Figure 2. (Color online) Proof of Proposition 2, with the First Four Layers of the Conflict Graph



Notes. The instances that have a × are the ones where the player cannot win. The cursive instances are the ones used by the strategy explained in the proof.

respectively). Thus, to achieve a success guarantee strictly more than 1/4, A needs to succeed in both 01 and 10. Consequently, these instances need to be selected in the conflict graph.

Now, for $n = 3$, A fails in 000, 001, 010, and 100 because of conflicts and on 111 because of the no-zero rule. Therefore, A *must* succeed in 011, 101, and 110, because each instance has a probability of occurrence equal to 1/8. Finally, for size 4, the same argument as before shows that A fails in all instances except 0111, 1011, 1101, and 1110. However, these are only 4 of 16 cases, and thus, A cannot achieve a success guarantee strictly larger than 1/4 if the adversary chose $n = 4$. This is a contradiction, and therefore Proposition 2 is true. □

Of course, there are several limitations to this first proof of the claim:

- Our results should not rely on the arbitrary no-zero rule.
- The fact that the proof is only considering small sizes is a weakness, in the sense that it does not exclude the existence of algorithms that could possibly have a better success guarantee than the k-max algorithm, but only on instances of size at least some N_0.
- The sampling probability is fixed to 1/2, we require a proof for general values of p.
- The bound only applies to deterministic algorithms, not to randomized algorithms.

By addressing the second problem we resolve the first one automatically because for large enough instances, the probability of the case with no zero becomes negligible (for size n, this happens only with probability $1/2^n$).

The following sections show how we progressively bypass these issues, thereby generalizing Proposition 2 step by step. To do so, the next section first introduces the involved concepts formally.

3.2.3. Preliminaries: Conflict Graph. This section formalizes the intuition about the conflict graph used in the proof of Proposition 2. We first describe its generic structure and then show how to measure the performance of a deterministic algorithm in this framework. We do so by a series of rather simple lemmata. For the sake of conciseness, we defer their proofs to the electronic companion. Generally speaking, they are the formalization and generalization of the ideas of the proof of Proposition 2.

Conflict Graph Structure. We first define what it means for two instances to be in conflict. To this end, we define the following notation for an instance I, which we identify by its bit string. We denote by $I[a,b]$ the instance I restricted to the positions a to b (both included).

Definition 6. Consider two instances I_1 and I_2 of size n_1 and n_2, respectively, with $n_1 < n_2$, both containing at least one zero. Let r be the position of the last zero in I_1. The instances I_1 and I_2 are *in conflict*, denoted $I_1 \nsim I_2$, if $\|I_1\|_1 = \|I_2\|_1$ and $I_1[1,r] = I_2[1,r]$.

The following lemma explains why this notion is useful.

Lemma 2. *Let $I_1 \nsim I_2$ be two instances in conflict. No deterministic algorithm can succeed in both I_1 and I_2.*

We now define the conflict graph, which is the formal object of which Figure 2 depicts the start.

Definition 7. The *conflict graph* $G = (V,E)$ is an infinite graph in which the set of nodes $V = \cup_{i=1}^{\infty} V_i$, where $V_i = \{0,1\}^i$, corresponds to all finite strings of bits. In particular, every node $v \in V$ corresponds to a unique instance of the last zero problem that we identify with v. Then E is defined as the set $\{\{v_1,v_2\} | v_1 \nsim v_2 \text{ and } |v_2| = |v_1| + 1\}$. We call v_1, v_2 such that $\{v_1,v_2\} \in E$ *neighbors*.

Because every node of the conflict graph corresponds to a unique instance of the last zero problem, which corresponds to a unique bit string, we will use these terms interchangeably and use the same notation for the corresponding bit string, node, and instance. The length of a bit string corresponds with the size of the instance and node, so in particular, for a node v, $|v|$ denotes the length of the corresponding bit string.

When we draw the conflict graph, we order the nodes by increasing size as in Figure 2. By doing so, a path in the conflict graph is monotone if it goes from left to right without zigzagging.

Definition 8. A path $P = (v_1,\dots,v_k)$ in the conflict graph is called *monotone* if and only if $|v_{i+1}| = |v_i| + 1$ for all $i \in \{1,\dots,k-1\}$.

The next lemma formalizes why monotone paths are interesting to study.

Lemma 3. *$I_1 \nsim I_2$ if and only if there exists a monotone path in the conflict graph connecting I_1 and I_2.*

This lemma and its proof in the Online Appendix have several consequences for the structure of the conflict graph. The following lemma is immediate, formalizing when an instance I of size n is in conflict with an instance I' of size $n+1$ or an instance I'' of size $n-1$. Intuitively, $I \nsim I'$ if I' can be obtained by inserting a new zero anywhere after the last zero of the bit string of I. In the other direction, $I \nsim I''$ if I'' can be obtained from I by removing its last zero.

Lemma 4. *Consider an instance I of size n whose corresponding bit string contains at least one zero, and denote the position of the last zero by r. Then I is in conflict with exactly those instances I' of size $n+1$ for which there exists a position $s > r$ such that $I'[1,s-1] = I[1,s-1]$, $I'[s] = 0$ and $I'[s+1,n+1] = I[s,n]$. In the other direction, I is in conflict with the single instance I'' of size $n-1$ with $I''[1,r-1] = I[1,r-1]$ and $I''[r,n-1] = I[r+1,n]$.*

This lemma implies that every node of size $n > 1$ has at most one neighbor to its left, leading to the following definitions.

Definition 9. Let v be a node in the conflict graph. If it exists, its unique neighbor u with $|u| = |v| - 1$ is called its *parent*. All neighbors w of v such that $|w| = |v| + 1$ are called its *children*. The *degree* $\deg(v)$ of v is defined as its number of children. All nodes x with $|x| > |v|$ are called *descendants* of v if there exists a monotone path from v to x.

For a given size n there exist 2^n nodes, each corresponding to a different bit string of length n. In particular, we can now talk about the degree of instances. These adhere to the following structure.

Lemma 5. *Consider all 2^n nodes corresponding to instances of size n. For each $i \in \{1,\dots,n-1\}$, there are 2^{n-i} nodes of degree i. Moreover, a node with degree k has exactly one child of degree i for every $i \in \{1,\dots,k\}$.*

The reader can check that Lemma 5 is satisfied in Figure 2.

Weights in the Conflict Graph. Up to this point, we considered only the case $p = 1/2$ in which every instance of size n has a probability of $1/2^n$ of occurring as the result of the sampling process. However, whenever $p \neq 1/2$, instances of the same size have different associated probabilities. To incorporate this, for a fixed value of $p \in [0,1]$, we introduce weights of nodes and consider a weighted version of the conflict graph. We still only consider deterministic algorithms, and we study probabilities on the sampling process only at this point.

Definition 10. Let $p \in [0,1]$. We define the *weight* $w(v)$ of a bit string v of norm $\|v\|_1 = m$ and length $|v| = n$ as the probability that this bit string was the result of the sampling process in the last zero problem for an instance of size n, that is, $w(v) = p^m(1-p)^{n-m}$. We define the weight $w(v)$ of a node v in the conflict graph as the weight of the corresponding bit string.

For a fixed size n, the weights of the instances of size n sum to 1. These weights adhere to a very nice structure, which we will exploit to prove our results for $p \neq 1/2$. We will need to know the combined weights of nodes of a given size and degree, as well as the combined weight of their children of given degrees. To this end, we introduce some new notation.

Definition 11. Consider the conflict graph and let n be fixed. Define $V_{n,i} = \{v \in V : |v| = n, \deg(v) = i\}$ as the set of nodes of size n with degree i. Define $w_i = \sum_{v \in V_{n,i}} w(v)$ as their combined weight (which, as the notation suggests, turns out to be independent of n). Let $W_{ijn} = \{v \in V : |v| = n+1, \deg(v) = j, \exists u \in V_{n,i} : u \sim v\}$ be the set of nodes of size $n+1$ and degree j that are in conflict with any node in $V_{n,i}$. Then define $w_{ij} = \sum_{v \in W_{ijn}} w(v)$ (again, the next lemma shows its independence of n).

Note that w_{ij} is only positive for $j \leq i$ because of Lemma 5. We can now prove the following weighted version of this lemma.

Lemma 6. *For any size n, $w_i = p^{i-1}(1-p)$ and $w_{ij} = (1-p)w_i = p^{i-1}(1-p)^2$ for all $1 \leq j \leq i \leq n$.*

Algorithms in the Conflict Graph. We now turn to the connection between algorithms and the conflict graph. We start by linking the structure of the conflict graph to deterministic algorithms. We introduced the notions of instances in conflict to show that a deterministic algorithm can only succeed in one of these instances (Lemma 2). By Lemma 3, instances are in conflict if and only if they are connected by a monotone path. Hence, we can observe the following lemma.

Lemma 7. *Let P be a monotone path in the conflict graph. Any deterministic algorithm can succeed in at most one of the instances corresponding to the nodes of P.*

Because every node of the conflict graph corresponds to an instance, we can identify an algorithm with a partition of the nodes of the conflict graph: the nodes on which it succeeds and the nodes on which it fails. Lemma 7 constraints the structure of such a partition: Choosing a certain instance to succeed in prevents the algorithm from choosing other instances to succeed in. Let us introduce new terminology to simplify talking about this process.

Definition 12. We say that an algorithm *selects a node v* if it succeeds on the corresponding instance. A node that is in conflict with an instance of smaller size that the algorithm selected is said to be *removed*. For an algorithm A we denote its subset of selected nodes and removed nodes by A_S and A_R, respectively. For a given instance size n we denote by A_S^n and A_R^n the set of selected and removed nodes of size n, respectively, such that $A_S = \cup_{n=1}^{\infty} A_S^n$ and $A_R = \cup_{n=1}^{\infty} A_R^n$.

Not all disjoint sets of selected and removed nodes correspond to a real algorithm (in the sense that there might be no finite description of a corresponding method), but this is not an issue as we look for results regarding the nonexistence of certain algorithms. We will abuse terminology and use the word "algorithm" nevertheless. As selecting a node implies the removal of nodes of larger size, we will consider the execution of an algorithm to be in increasing order of the size of instances.

Let us introduce concepts for the quality of an algorithm for the last zero problem.

Definition 13. Let A be a deterministic algorithm for the last zero problem and let an instance size n be fixed. The *performance* of A for size n is the sum of the weights of the instances in which it succeeds, that is, $\text{perf}(A,n) = \sum_{v \in A_S^n} w(v)$. The *success guarantee* of A is the infimum of the performance of A over all instance sizes n, that is, $\inf_n \text{perf}(A,n)$.

Indeed, using Proposition 1, this corresponds to the definition of success guarantee for AOSp as in Definition 1. Recall that we aim for a bound on the success guarantee of any algorithm for AOSp, a goal we can achieve by proving a bound on the success guarantee of any algorithm for the last zero problem.

After introducing weights to generalize the conflict graph to generic values of p, let us make the final generalization by considering randomized algorithms. Where a deterministic algorithm either selects a node or not, a randomized algorithm labels each instance with a *selection probability* q. Concretely, this means the following. Suppose the algorithm faces the last 0 in this instance (but it is not aware of this). Then the algorithm stops with probability q (and succeeds in this instance). It continues the sequence with probability $1-q$, meaning it fails in this instance, but it might still succeed in any of its descendants.

The following lemma generalizes Lemma 2 to a nonbinary version for randomized algorithms.

Lemma 8. *Let I be an instance of the last zero problem. If any algorithm picks a selection probability q for I, its probability to succeed in any of the descendants of I is at most $1-q$.*

Similar to Definition 12, when a randomized algorithm selects an instance I with probability q, we say that it *removes a fraction* q of the descendants of I. The proof of Lemma 8 also shows that these *removed fractions* adhere to the following additive property: If an algorithm selects an instance I with probability q, and one of its descendants I' with probability r, then for any descendant I'' of I' its removed fraction is $q+r$ and the algorithm can select it with probability at most $1-q-r$.

Similarly, we define the performance of a randomized algorithm for a given size n as the product of the weight of a node multiplied by its selection probability, summed over all instances of size n. The worst case performance is then the infimum over n of these performances, corresponding to Definition 1.

Finally, the success guarantee of the k-max algorithm, proved in Lemma 1, can also be proved now using the alternative perspective of the conflict graph. The k-max algorithm roughly selects low degree nodes in every size n of the conflict graph in order to remove as little weight as possible from instances of larger size. A careful analysis indeed gives the same success guarantee $kp^k(1-p)$. For details, see the Online Appendix.

3.2.4. Proof for the Case of $p = 1/2$.

To introduce the main techniques behind the general proof, this section proves the special case of the negative results of Theorem 2 for deterministic size-oblivious algorithms for the case where n is larger than some constant N_0, and $p=1/2$. Indeed, Proposition 3 generalizes Proposition 2 by getting rid of the first two of its aforementioned limitations: It does not rely on the no-zero rule or small instance sizes.

For $p=1/2$, all nodes of size n have the same weight, namely $1/2^n$. Thus at any given size, the total fraction of selected nodes equals the total weight of the selected nodes.

Proposition 3. *For the last zero problem with $p=1/2$, no deterministic size-oblivious algorithm can have a better success guarantee than the k-max algorithm, even if we consider only instances of size larger than N_0, for any N_0.*

To prove Proposition 3, we will bound the success guarantee of any deterministic algorithm by considering a special class of algorithms.

Canonical Algorithms. More precisely, we consider a deterministic algorithm A that starts by selecting some nodes in the conflict graph for a certain size N_0. Consequently, all descendants of the selected nodes will be removed. A will then continue to the nodes of size N_0+1 and select a subset of the nodes of this size that have not been removed. Then it will continue to the next size and iterate this procedure. We will show that if the algorithm consistently selects at least a $1/4+\varepsilon$ fraction of the nodes for each size, this process cannot run forever, reaching a contradiction similar to the proof of Proposition 2.

Before we proceed to the proof, we make a crucial observation based on Lemma 5.

Lemma 9. *Consider two nodes v, w in the conflict graph with $\deg(v)=\deg(w)$. Then the subtree $T(v)$ consisting of v and its descendants is isomorphic to the subtree $T(w)$ consisting of w and its descendants.*

Because of these isomorphisms, some choices an algorithm can make are essentially equivalent.

Lemma 10. *Consider an algorithm A for the last zero problem with $p=1/2$ such that there exists an instance $I_1 \in A_S$ and an instance $I_2 \notin \{A_S \cup A_R\}$ with $|I_1|=|I_2|$. Then there exists another algorithm A' with $I_2 \in A'_S$ and $I_1 \notin A'_S$ with the same success guarantee as A.*

With this important observation at hand, we can prove that it suffices to restrict our attention to algorithms of a canonical form, in order to reduce the large variety of possible algorithms. We say that an algorithm follows a *small degrees first strategy* if for any instance size the algorithm considers, among the nodes it did not previously remove, it selects the nodes with the smallest degrees. This strategy does not define a single algorithm as many nodes have the same degree. Indeed, the k-max algorithm for AOSp is closely related to these small degrees first strategies for the last zero problem: We will elaborate on this in the Online Appendix.

Lemma 11 (Small Degrees First Strategy). *Consider the last zero problem for $p=1/2$. For every size-oblivious algorithm A there exists an algorithm that adheres to the small degrees first strategy that achieves the same performance as A for every n.*

From now on, we restrict ourselves to considering algorithms that follow the small degrees first strategy.

Cover Ratio. Consider an algorithm A. It partitions the nodes of the conflict graph of any given size n in three partitions: the set A_S of selected nodes, the set A_R of removed nodes, and the set $V \setminus (A_S \cup A_R)$ of nodes that are neither selected nor removed (the algorithm succeeds in any node in the first set and fails in the latter two). It might be counter-intuitive at first that the third set is nonempty as selecting any such node can only improve $\mathrm{perf}(A,n)$. However, this would remove all of its descendants. Because the success guarantee is defined as $\inf_n \mathrm{perf}(A,n)$, lowering $\mathrm{perf}(A,n')$ for some size $n'>n$ could therefore decrease it. Based on this consideration we introduce the following definition.

Definition 14. Let A be an algorithm for the last zero problem that selects the set of nodes A_S and removes the set of nodes A_R. Fix a size n. The *cover ratio* for A and size n is $\rho_n = \sum_{v \in A_S^n \cup A_R^n} w(v)$, that is, the combined weight of the nodes A selects and removes.

Observe that this expression with a sum over only A_S instead of $A_S \cup A_R$ equals perf(A,n). In the special case that $p = 1/2$, all instances of size n have equal weight and therefore $\rho_n = (|A_S| + |A_R|)/2^n$ is just the fraction of the total number of instances of size n that are either selected or removed.

Armed with this definition, we can present the proof of Proposition 3. The proof sketch of Proposition 2 showed the intuition behind the proof, here we state the formal arguments. The idea behind the proof is to show that selecting strictly more than $1/4$ of the instances for many successive sizes implies that the cover ratio increases with the instance size in such a way that for some instance size it is impossible to select that many instances. This shows by contradiction that there is no deterministic algorithm that has a success guarantee of $1/4 + \varepsilon$ for any $\varepsilon > 0$.

Lemma 11 implies that we can restrict ourselves to an algorithm A that for every instance size n selects a $1/4 + \varepsilon$ fraction of the nonremoved instances in increasing order of degrees (breaking ties arbitrarily between instances of the same degree as they constitute isomorphic subtrees). Then A repeats this for the nonremoved instances of size $n+1$, which we refer to as the next *step*. Without loss of generality, we can assume that A starts at size N_0 with no removed nodes.

We now analyze the dynamics of A and in particular the dynamics of the cover ratio. First, observe that at size N_0, no nodes have been removed thus far. As A selects a $1/4 + \varepsilon$ fraction of the nodes and half of all these nodes have a degree of one, the algorithm selects only nodes of degree 1. For a certain number of sizes, starting from N_0, the algorithm can select only degree 1 nodes. We call this the *first phase* of the algorithm.

Claim 1. *Consider the last zero problem for $p = 1/2$ and an algorithm as described previously. After t steps in the first phase of the algorithm, the cover ratio ρ_{N_0+t} equals $(1/4 + \varepsilon) \cdot \sum_{i=1}^{t} 1/2^{i-1}$.*

Note that $(1/4 + \varepsilon) \cdot \sum_{i=1}^{t} 1/2^{i-1}$ tends asymptotically toward $(1 + \varepsilon)/2$ as k grows, for a given $\varepsilon > 0$. In particular, this means that there exists an instance size s such that $\rho_s > 1/2$, which is the total fraction of nodes with degree 1. This implies, in turn, that A cannot only select nodes of degree 1 from instance size s onwards and is forced to start selecting degree 2 nodes to achieve a performance of $1/4 + \varepsilon$ for these instance sizes. This is the start of a *second phase* in which A also selects degree 2 nodes.

Claim 2. *Consider the last zero problem for $p = 1/2$ and an algorithm as described previously. Let s be the instance size at which the second phase of the algorithm starts. Then $\rho_{i+1} \geq \rho_i + \varepsilon$ for all $i \geq s$.*

These two claims imply Proposition 3, and its proof can be found in the Online Appendix. Intuitively, if the cover ratio grows by the same additive factor in each step, at some point it will exceed the value 1, which completes the proof by contradiction.

3.2.5. Generalizing the Proof.

The previous section proved the negative results of Theorem 2 for size-oblivious deterministic algorithms for $p = 1/2$. In this section, we sketch the generalization for general values of p, for randomized algorithms, and for non–size-oblivious algorithms. All details are in the Online Appendix.

Generalization to Any Value of p. To generalize the previous results beyond the case of $p = 1/2$ is a bit more complicated, because when $p \neq 1/2$ the instances of the same size do not have the same probability of occurring. For example, for $p = 3/4$, succeeding in the instance 1^k0 contributes more to its performance for that instance size than to succeed in the instance 0^{k+1}, as the first has probability $(3/4)^k(1/4)$ to occur and the second has probability $(1/4)^{k+1}$. Recall that we associated these probabilities as weights to the nodes in the conflict graph (Definition 10).

As such, the swapping argument of Lemma 10 no longer works. To overcome this issue, we introduce two *local operators*: One for high-degree nodes and one for low-degree nodes. The high-degree local operator removes a node of high degree and selects its children instead, and the low-degree local operator selects a node of low degree and removes its descendants instead. We can show that both local operators improve the *average performance* of an algorithm, which is the average of perf(A,n), perf$(A,n+1)$, ..., perf$(A,n+t)$ in a window $[n, n+t]$.

To complete the proof, we introduce an algorithm that we call the *fill-in strategy*, which selects all nonremoved nodes of degree up to $\lfloor 1/(1-p) \rfloor$ and prove it has optimal average performance among all size-oblivious deterministic algorithms. Finally, we analyze the k-max algorithm in the conflict graph and note that it is very consistent in the sense that it selects the same total weight for every size. Therefore, its average performance and worst case performance (i.e., success guarantee) are arbitrarily close, and they match the average performance of the optimal fill-in strategy.

Generalization to Randomized Algorithms. Not much is needed to extend the results to size-oblivious randomized algorithms for general values of p. We define the randomized version of the two previously mentioned local operators and show that they improve the average performance as well. The rest of the proof follows immediately from all arguments for the previous generalization.

Generalization to Non–Size-Oblivious Algorithms. The generalization to algorithms that are not size oblivious is very similar in spirit to all the previous arguments. The idea is to mimic the proof for size-oblivious algorithms by drawing a parallel between a different set of instances of AOSp and the last zero problem. Recall that Proposition 1 used instances of AOSp where the elements are ordered in increasing value. To prove Theorem 2 for algorithms that are not size oblivious, consider a set of instances for AOSp generated by an adversary who chooses, next to the values and an instance size n, another integer m. Then, he first orders an increasing sequence of numbers until index m, followed by very low values. Intuitively, such an instance is similar to the instances considered previously, as knowing the value of n does not provide much useful information. An algorithm succeeds if it stops on the last online value before m, whose value is unknown. The proof requires some additional arguments, as the algorithm knows an upper bound on m and can estimate its value roughly based on the number of samples of very low value, but the proof technique is very similar to the proof shown in this section. The full details are in the Online Appendix.

4. Random Order

In this section, we study the second problem of this paper: the random order secretary problem with p-sampling ROSp. To analyze this case it is useful to have the following equivalent setting.

4.1. Continuous Time Arrival Model

We are given the values $\alpha_1, \ldots, \alpha_n$, and nature samples n uniformly random and independent arrival times $(\tau_i)_{i=1}^n$ in the interval $[0,1]$. Now S contains all elements α_i such that $\tau_i < p$ and V contains all other elements. We get to observe all elements in S beforehand. Then, we observe one by one the elements in V in the order given by the τ_is.

We show equivalence between ROSp and the continuous time arrival model by showing that any algorithm A for the original setting can be applied to the continuous time model, obtaining the same success guarantee, and vice versa. Consider an algorithm for ROSp. It is easy to see that in the continuous time model each element is in S independently with probability p, and that the elements in V are revealed in uniformly random order, as in ROSp. Therefore, we can use the algorithm A and simply ignore the arrival times. Consider now an algorithm A' for the continuous time model. There are no arrival times in ROSp, but we can simulate them: we can sample $|S|$ uniform arrival times in the interval $[0,p]$ and assign them to the elements of S in an arbitrary way, and sample $|V| = n - |S|$ uniform arrival times in $[p,1]$ and assign them to each observed element in V. Notice that since in ROSp each element is in S independently with probability p and the elements in V are revealed in uniformly random order, the simulated arrival times distribute exactly as n uniform arrival times in $[0,1]$. Thus, if the algorithm A' requires observing the arrival times, we can simply pass it the simulated arrival times and we obtain a randomized algorithm for ROSp with the same success probability as if we were applying it to the continuous time model.

From now on, we consider the continuous time arrival model. Consider the family of algorithms ALG_t, described in Algorithm 2. The algorithm is parameterized by a sequence $t = (t_i)_{i\in\mathbb{N}}$ such that $0 \le t_1 < t_2 < \cdots < 1$, which it takes as input. The algorithm starts by setting a threshold equal to the largest sampled element and it accepts an online element revealed between time t_1 and t_2 if it exceeds this threshold and is the largest online element revealed thus far. Between time t_2 and t_3, it accepts an element if and only if it is larger than the second largest sampled element, and it is the largest online element revealed thus far, and so on. After every t_k value, it decreases the threshold to the kth largest sampled element. In general, the algorithm ALG_t accepts an element α_i if it is the largest element of V seen thus far, and it is larger than the kth largest element in S, where k is such that $t_k \le \tau_i < t_{k+1}$. In other words, between times t_k and t_{k+1} the algorithm sets as threshold the kth largest element of S for each k. It only accepts elements larger than this threshold and larger than any online value seen thus far. We prove that the best possible success guarantee is attained in this family of decreasing threshold algorithms. Even though these algorithms are designed for the continuous time model, because of the aforementioned equivalence, they are also optimal in the original setting.

Algorithm 2 (Time-Threshold Algorithm ALG_t for ROS_p)
```
for $i = 1, \ldots, |S|$ do
    $s_i \leftarrow$ the $i$-th largest element in $S$.
end for
$s_i \leftarrow -\infty$ if $i > |S|$.
for $j = 1, \ldots, |V|$ do
    $\sigma(j) \leftarrow$ the index of the $j$th observed element of $V$.
    if $\tau_{\sigma(j)} < t_1$ then
        Discard the value
    else
        $\ell \leftarrow \max\{\ell' : t_{\ell'} \le \tau_{\sigma(j)}\}$.
        if $\alpha_{\sigma(j)} > s_\ell$ and $\alpha_{\sigma(j)}$ is the largest element of $V$ seen so far then
            Accept the value and stop the game
        else
            Discard the value
        end if
    end if
end for
```

Theorem 3. *There exists a universal sequence t, independent of p and n, such that ALG_t obtains the best possible* *success guarantee for ROSp. Furthermore, when $p = 0$, this guarantee is equal to $1/e$, and when p tends to one, the guarantee tends to $\gamma \approx 0.58$, the optimal success guarantee in the full-information secretary problem.*[8]

We prove this theorem in two main steps. First, we find the sequence t^* that maximizes the success guarantee of ALG_t. Then, we find an expression for the optimal success probability when p and n are given, and prove that for fixed p it converges to the success guarantee of ALG_{t^*} when n tends to infinity. In this section, we state the lemmas and sketch the proofs. The full proofs can be found in the Online Appendix.

To find the optimal sequence t^*, we start by studying the success probability of algorithm ALG_t, for any sequence t, sample rate p and instance size n. We prove that in fact the worst case for this class of algorithms is when n is very large. The approach of approximating the problem when n is large by a continuous time problem was pioneered by Bruss (1984) and has been used for different optimal stopping problems (Immorlica et al. 2006, Chan et al. 2015).

Lemma 12. *For any sequence t and sampling probability p, the success probability of ALG_t in ROSp decreases with n.*

To prove the lemma the idea is to inductively couple the realizations of the arrival times in instances of sizes n and $n+1$. We show that if ALG_t fails for a given realization of the arrival times of the largest n values in the instance of size n, then ALG_t also fails for any possible realization of the arrival time of the smallest (the $n+1$th largest) value, in the instance of size $n+1$. This implies that the probability of failure increases with n.

By Lemma 12, the success guarantee of ALG_t is simply the limit of its success probability when n grows to infinity. We calculate these probabilities and obtain an explicit formula for the limit in the following lemma. Interestingly, the formula turns out to be fairly simple.

Lemma 13. *Fix a sequence t and a sampling probability p. The success guarantee of ALG_t in ROSp is given by*

$$\sum_{i=1}^{\infty} p^{i-1} \cdot \left(1 - (p \vee t_i) - \int_{(p\vee t_i)}^{1} \sum_{j=1}^{i} \frac{t-(p\vee t_i)}{t^j}\,\mathrm{d}t\right). \quad (1)$$

where $p \vee t_i = \max\{p, t_i\}$.

We then focus our attention on optimizing this success guarantee. Surprisingly, it turns out the problem of maximizing Equation (1) is separable and concave, so we can simply impose the first-order conditions to obtain the optimum. Perhaps even more surprising is that these first-order conditions are independent of p, and therefore, the optimal sequence t^* is also independent of p, as the following lemma shows.

Lemma 14. *Fix a sampling probability p. The sequence t^* defined as the unique solution of the equations*

$$\ln\left(\frac{1}{t_i^*}\right) + \sum_{j=1}^{i-1} \frac{(1/t_i^*)^j - 1}{j} = 1, \text{ for all } i \in \mathbb{N}, \quad (2)$$

maximizes Equation (1). In particular, t^* does not depend on p.

Now that we have the best algorithm in the family, we prove that its success guarantee is actually the best possible. To do this, we first characterize the algorithm that achieves the highest success probability for fixed sampling probability p and instance size n.

For a nondecreasing function $\ell : [n] \to [n]$, we define the *sequential-ℓ-max algorithm* in the following way.

Definition 15. Let $\ell : [n] \to [n]$. The *sequential-ℓ-max algorithm* accepts the ith observed value (considering the values from S and the ones that have been revealed from V) if it is the largest seen thus far from V, and it is larger that the $\ell(i)$th largest value from S.

We prove that the optimal algorithm is in this class.

Lemma 15. *Fix a sampling probability p and an instance size n. There is a function ℓ such that the sequential-ℓ-max algorithm obtains the best possible success probability for instances of size n of ROSp.*

To conclude the optimality of ALG_{t^*} we show that the success probability of the best sequential-ℓ-max algorithm for each n converges to Equation (1) for some sequence t, when n grows to infinity. To this end, we first calculate the success probability of a sequential-ℓ-max algorithm.

Lemma 16. *Fix n, p, and a nondecreasing function ℓ. Consider an integer h such that $0 \le h < n$, and define $\hat{\ell}(i) = \min\{\ell(i), h+1\}$ for all $i \in [n]$. The success probability of the sequential-ℓ-max algorithm, conditional on $|S| = h$, is given by*

$$\frac{1}{n-h}\left(1 - \prod_{j=0}^{\hat{\ell}(h+1)-1} \frac{h-j}{n-j}\right) + \sum_{i=h+1}^{n-1}\left(\sum_{r=h+1}^{i} \frac{1}{n-i}\left(\frac{1}{i-h}\prod_{j=0}^{\hat{\ell}(r)-1}\frac{h-j}{i-j} - \frac{1}{n-h}\prod_{j=0}^{\hat{\ell}(r)-1}\frac{h-j}{n-j}\right) - \frac{1}{n-h}\prod_{j=0}^{\hat{\ell}(i+1)-1}\frac{h-j}{n-j}\right). \quad (3)$$

We then show that there is a limit for the optimal ℓ in a continuous space and use a Riemann sum analysis to obtain Equation (1) in the limit, proving Theorem 3.

Lemma 17. *Fix a sampling probability p. For each $n \in \mathbb{N}$, choose $\ell_{p,n}$ so that the sequential-$\ell_{p,n}$-max algorithm achieves the best possible success probability for fixed p and n. There* *exists a sequence t such that the success probability of the sequential-$\ell_{p,n}$-max algorithm converges to Equation (1) when n grows to infinity.*

Finally, we study the success guarantee of ALG_{t^*} in the border values of p and show that it actually becomes equal to the best possible among all algorithms. It is easy to see that the success guarantee is $1/e$ when $p = 0$. When $p = 0$, Equation (1) simplifies to $t_1 \ln(1/t_1)$, and Equation (2) yields $t_1^* = 1/e$. Substitution gives the success guarantee of $1/e$. The case when p tends to one is a bit more involved and requires some tedious calculations. We evaluate Equation (1) with the first-order approximation $t_i^* \approx t_i' := 1 - c/i$, where c is a constant. To fix c we impose that (t_i') satisfies Equation (2) in the limit when $i \to \infty$. More precisely, we take c such that

$$1 = \lim_{i\to\infty} \ln\left(\frac{1}{1-c/i}\right) + \sum_{j=1}^{i-1} \frac{(1-c/i)^{-j} - 1}{j}$$
$$= \int_0^1 \frac{e^{cx}-1}{x}\mathrm{d}x.$$

With this in hand, we use a Riemann sum analysis to show the next lemma, which states that when p tends to one, this approximation converges to the explicit expression of Samuels (1982, 1991) for γ.

Lemma 18. *Let $t_i' = 1 - c/i$, where c is the solution of $\int_0^1 (e^{cx}-1)/x\,\mathrm{d}x = 1$. When evaluated in t', Equation (1) tends to*

$$\gamma = e^{-c} + (e^{-c} - 1 - c)\int_1^\infty x^{-1}e^{-cx}\,\mathrm{d}x \approx 0.5801, \qquad (4)$$

when p tends to one.

4.1. Computation of the Time Thresholds

In this section, we discuss how to compute the optimal time thresholds t^*. By Lemma 14, t^* does not depend on p, nor in n, and therefore it is enough to compute them once. However, because t^* is an infinite sequence, a reasonable question is how well we can do if we compute only finitely many of these thresholds.

Denote by $\gamma(p)$ the optimal success guarantee for a given $p \in [0,1)$. To achieve a success guarantee of $\gamma(p) - \varepsilon$ for a given $\varepsilon > 0$, it is sufficient to compute $O([1/\varepsilon \cdot (1-p)])$ many thresholds within an $O(\varepsilon^2(1-p)^2)$ margin of error each. The reason for this is that our algorithm can fail (compared with ALG_{t^*}) if the best element of the interval $[p,1]$ falls too close to the thresholds (closer than the margin of error), or after the last threshold we computed, which by the first-order approximation is $1 - O(\varepsilon(1-p))$. However, the best element of the interval $[p,1]$ falls in a set of measure $\varepsilon(1-p)$ with probability ε, and therefore we can ignore this event while only losing ε in the success probability.

Table 1. Approximation of the First 10 Optimal Time Thresholds Within an Error of 10^{-7}

$t_1^* \approx 0.3678794$	$t_6^* \approx 0.8709762$
$t_2^* \approx 0.6422006$	$t_7^* \approx 0.8887973$
$t_3^* \approx 0.7518116$	$t_8^* \approx 0.9022956$
$t_4^* \approx 0.8101810$	$t_9^* \approx 0.9128731$
$t_5^* \approx 0.8463645$	$t_{10}^* \approx 0.9213851$

Observe that for each $i \in \mathbb{N}$, Equation (2) gives a separate equation on a single variable, whose unique solution is t_i^*. Also, the left-hand side is monotone and continuous. Therefore, we can approximate each threshold t_i^* independently by doing a binary search on each of these equations.

If we assume that the left-hand side of Equation (2) can be computed in $O(i)$ time for a given i (because it has i terms), we obtain the following.

Lemma 19. *For any given $p \in [0,1]$, instance size n, and $\varepsilon > 0$, we can compute thresholds $\tilde{t}$ that approximate t^* and such that $ALG_{\tilde{t}}$ has a success probability of at least $\gamma(p) - \varepsilon$ in time $O(1/[\varepsilon^2(1-p)^2] \cdot \log([1/\varepsilon(1-p)]))$.*

Table 1 provides the first 10 optimal time thresholds.

5. Robustness with Respect to the Knowledge of the Parameters

In this section, we briefly discuss the impact of the knowledge of the parameters on the guarantees that can be obtained. There are two parameters for both AOSp and ROSp: the number of elements n and the sampling probability p. The performance of an algorithm can vary a lot depending on its presumed knowledge about these parameters.

For AOSp we already discussed that knowledge of n is irrelevant in worst case terms, and the next section gives some insights into its performance for given values of n. It also presents numerical results for sensitivity analysis on the value of p. To complete the picture, we analyze the robustness with respect to estimation errors in the parameter p theoretically in this section. First, if p is unknown but n is known, we show that the ratio of the number of samples to the total number of elements gives a good estimate of p and that using the k-max algorithm with this estimate is basically optimal. More specifically, assume we are given a set S of h samples, drawn independently from an initial set consisting of n values in total, using some (unknown) value of p. The remaining $n - h$ samples form the online set V. In this setting we adapt the k-max algorithm simply by setting the threshold to the kth largest sample, where $k = \lfloor n/(n-h) \rfloor$, and accepting the first value of the online set that is above the threshold. This variation of the k-max algorithm boils down to simply estimating p as $\hat{p} = h/n$ and using $\hat{p}$ to determine the desired value of k. By standard concentration arguments, we can prove that the estimate $\hat{p}$ is accurate with high probability, and thus we obtain the following theorem, whose proof can be found in the Online Appendix.

Theorem 4. *Consider AOSp with unknown p, where h samples are drawn independently from a set of n elements. The k-max algorithm with $\hat{p} = h/n$ achieves the best possible success guarantee up to a factor $1 - \varepsilon$ with high probability.*

Second, for AOSp where both p and n are unknown, we show that no nontrivial guarantee can be obtained. The intuition behind this strong negative result results from the situation in which the algorithm is given very few samples. In this case, it does not know whether the instance is very short (in which case it should stop early), or the sampling probability is very low (in which case it should wait longer).

Theorem 5. *When both p and n are unknown, no algorithm can get a positive success guarantee for AOSp.*

For ROSp, we have shown that the optimal algorithm ALG_{t^*} does not depend on p, and knowledge of the uniform random arrivals suffices to obtain the optimal guarantee. Therefore, ALG_{t^*} achieves the best possible success guarantee, even when n is unknown. The next section provides more insights into its quality for finite n. Conversely, if p is unknown and n is known and large, then we can sample uniform random arrival times for each value and obtain with ALG_{t^*} the best success guarantee. Indeed, the sampled arrival times themselves will provide a sharp estimate of p.

On a more applied note, whenever it is reasonable to assume that the values come in random order, it is usually also safe to assume that this random order comes from random arrival times. In case the arrival times are random but not uniform, the time thresholds t^* can be transformed using the distribution function of the arrival times, and again, it is possible to obtain the optimal success guarantee.

6. Numerical Experiments

In this section, we implement our algorithms and evaluate them on the data set of Goldstein et al. (2020). In their paper, they design a large-scale online experiment in which people repeatedly play a secretary problem. The values each player faces are drawn i.i.d. from a distribution unknown to them and their total number is the same in every game. The main goal is to study experimentally the evolution of the players' stopping behavior. In particular, the main research question posed is whether the players progressively learn a near-optimal stopping strategy as they gain more experience. Goldstein et al. (2020) use a Bayesian comparison framework to model the players' behavior and conclude that the estimated thresholds are indeed very close to the ones of Gilbert and Mosteller (1966) (i.e., the optimal ones for i.i.d. values from a known distribution) after only a few games.

We start by observing that our independent sampling model can be applied to their repeated secretary problem in a straightforward way; the first game is precisely the classic secretary problem or ROS0. Now, the second game closely corresponds to ROS1/2; we can imagine the values of the first game as our samples and the values of the second game as the online values. All values are i.i.d., and the two sets have equal sizes. Our independent sampling model with $p = 1/2$ and the values of both games as the n input values $\alpha_1, \alpha_2, \ldots, \alpha_n$, would also result in splitting them into two sets of roughly equal size. Applying the same reasoning, the third game closely corresponds to ROS2/3, and so on; the ith game corresponds to ROS$(i-1)/i$. In general, our model and this repeated secretary problem would be equivalent in the limit as $n \to \infty$, but for small values of n, we essentially ignore the variance of the independent sampling process. Because our algorithms are guaranteed to be optimal in a very similar model, they can serve as a meaningful benchmark for studying the players' strategies across games. By doing so, we hope to provide new insights and strengthen the existing ones regarding the players' behavior. In particular, and as mentioned in Section 1.3, our new comparison can help explain up to which extent players optimally use the information they have at each game and, as a result, resolve some behavioral issues raised by Goldstein et al. (2020). Because the values are i.i.d. (thus, closer to random order than adversarial) and players could choose the order of inspecting the elements (of course without knowing their values or the distribution over them), ALG_t is the natural candidate to use as the main benchmark, especially because its guarantee converges to the algorithm of Gilbert and Mosteller (1966).

First, we describe in more detail the behavioral experiment, the methodology, and the results obtained in Goldstein et al. (2020). The participants of the online experiment were directed to a simple interface, in which they were presented with a number of boxes containing money (i.e., hidden values). They could open the boxes in any order they wanted and decide when to stop. The interface would not let them terminate the game if the value with which they stopped was not a local maximum. After the end of each game, they were told if they won, how far away they were from the maximum value, and they would observe all the values of that game. They were also incentivized to play at least six games. The values for each player were drawn independently from an identical unknown distribution across all games. Each player was randomly assigned to one of three candidate distributions with the same support but with very different density functions; the first had high negative skew, the second was the uniform distribution, and the third had high positive skew. Each player was also randomly assigned to play games of 7 boxes or 15 boxes (recall that the game becomes harder as the number of boxes increases). In the end, 6,637 people participated and were distributed across the aforementioned random choices. They played a total of 48,836 games, for an average of 7.39 games per player.

The data set of the experiments contains the following information for each player: in which distribution and to which number of boxes they were assigned, how many games they played, the number of each game (if it was the first game, the second, and so on), the value and the rank of the box with which they stopped at each game, and the values of all *opened* boxes. What is not included in the data set of Goldstein et al. (2020), although the players observed them at the end of each game, are the values of the remaining closed boxes. Nevertheless, this is not a major issue; because we know the distribution from which the values were drawn, we can complement each instance with random values that would be very similar to the realized ones.

Their data show that the players rapidly improve their performance in early games, thus exhibiting substantial learning. The effects of learning are not as strong in later games, and the players eventually converge to a probability of success that is 5–10 percentage points below the theoretical optima (that is, the guarantee given by the optimal algorithm of Gilbert and Mosteller (1966) with perfect knowledge of the distribution). The authors focus on modeling the players' strategies and attempt to fit several models they define to the data. In the first game, they observe that the thresholds belong to the class of what they call "value oblivious" strategies, which also contains the solution to the classical secretary problem. After the first game, the authors conclude that the players play according to a multithreshold strategy. Therefore, there is a switch in strategies from the first to the second game. The optimal algorithm of Gilbert and Mosteller (1966) and our optimal algorithm for ROS*p* also compute a series of (decreasing) thresholds. To investigate at which rate players learn to play optimally, Goldstein et al. (2020) compute the players' estimated thresholds for the seven box games; they show that these converge quickly to the optimal ones of Gilbert and Mosteller (1966) and are very close to them already in the fourth game. Finally, they give some potential explanations for the strategy switch from the first game to the second (see also Section 1.3). One important takeaway from that work is that for such fundamental optimal stopping problems, their results indicate that "the optimal procedure is likely to give a close approximation of human behavior. This is in contrast to many other areas of economics." Our experimental results, which we present next, will provide strong further support to this insight.

6.1. Experimental Setup

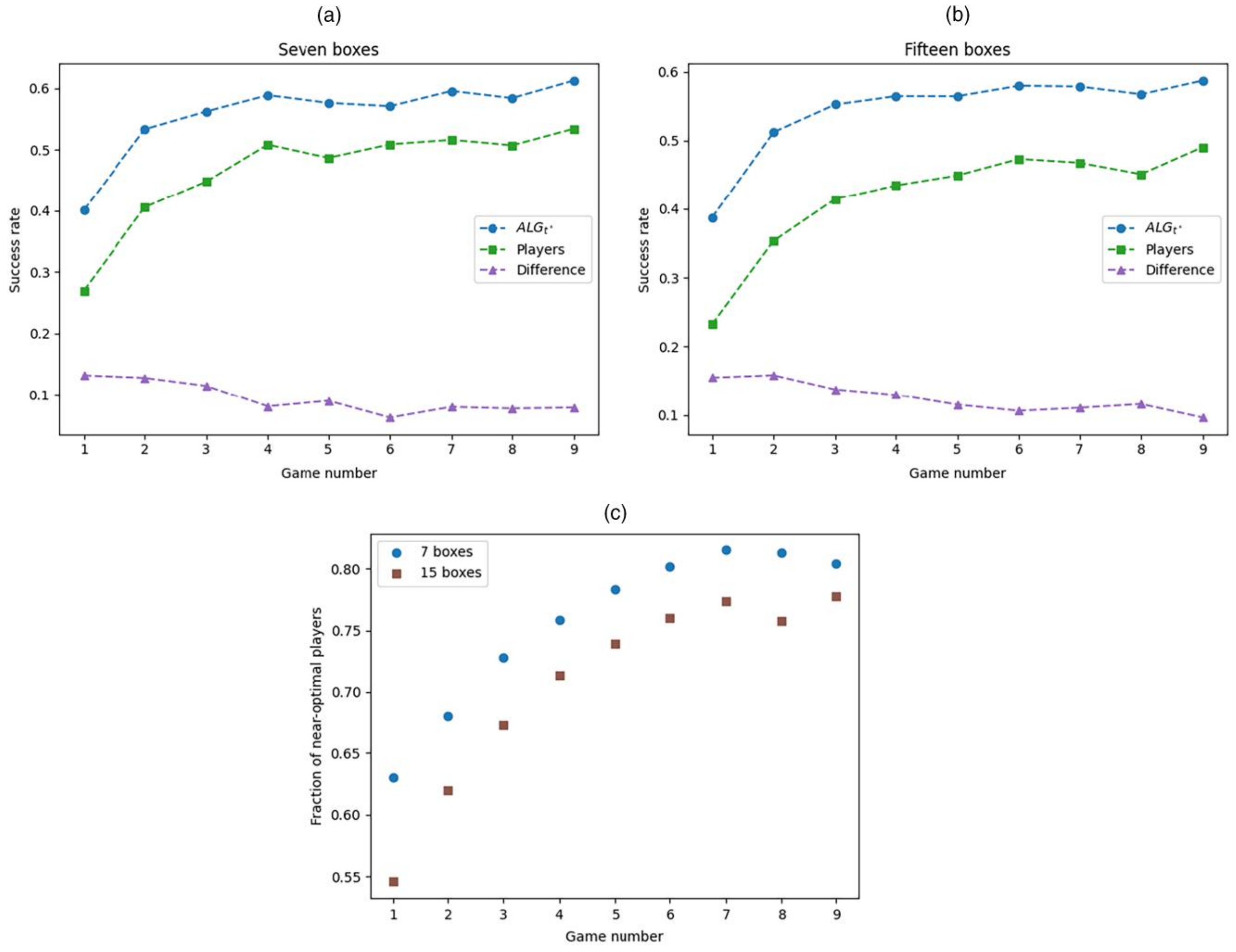
For our experiments, the first thing to do is to complete the missing data of the real-world data set. Because for each game, the rank and the value of the box the player stopped with, the value distribution, and the values of the opened boxes are known, we generate the values of the unopened boxes using rejection sampling. This is a necessary step for ALG_t because it uses the whole sample set to fix the thresholds. We generate 100 such complete instances by independently generating the missing data 100 times and take the average success rate of ALG_t (i.e., the fraction of instances in which the algorithm stopped with the maximum value) for each game as our final result. For a fair comparison with the work of Goldstein et al. (2020), we also perform no cleaning on the data by excluding, for example, players who did not put any effort into winning. After the end of each game, we add the values of the boxes to our sample set S. Thus, we know that, for example, in the third game, the samples are twice as many as the online values. In this case, as we explained before, we run our algorithm with $p = 2/3$. All our results are plotted for the first nine games. We made this choice so that, on the one hand, we are consistent with some of the important figures of Goldstein et al. (2020) (e.g., figures 8 and 9) and, on the other hand, because less than a quarter of the players played more than nine games.

6.2. Experimental Results for the Optimal ROS*p* Algorithm

We present in Figure 3 the results of evaluating ALG_t on the data of Goldstein et al. (2020) complemented with the generated values of the unopened boxes. To calculate the series of decreasing thresholds of ALG_t, we have to solve Equation (2); it is faster to search for the solution for each threshold using binary search and stop when we are ε-close to the solution of the equation (here, we set $\varepsilon = 10^{-7}$).

As we observe from Figure 3, (a) and (b), the evolution of the success rates of the players and ALG_t exhibit a very strong correlation, both for 7 and 15 boxes. This suggests that the players play according to some strategy that is similar to ALG_t all along (i.e., a series of decreasing thresholds), but (slightly) suboptimally. Moreover, the difference in the success rates is similar across games, and in particular, it slightly decreases in the first few games, and from game 4 onward it remains stable. These results strengthen the belief that for this type of simple online selection problems the optimal algorithm provides good insights into how players behave. Figure 3(c) provides additional supporting evidence to the claim that the players quickly learn how to play strategies that are close to the optimal. The figure shows the percentage of players that played close to optimal in the sense that the ranking of

Figure 3. (Color online) Comparison of the Players' Behavior with the Optimal Algorithm for ROSp for 7 and 15 Boxes



Notes. (a) The success rate of the players and ALG_t for the first nine games and their difference for seven boxes. (b) The success rate of the players and ALG_t for the first nine games and their difference for 15 boxes. (c) Percentage of players who picked a value with ranking at most one away from the value ALG_t picks.

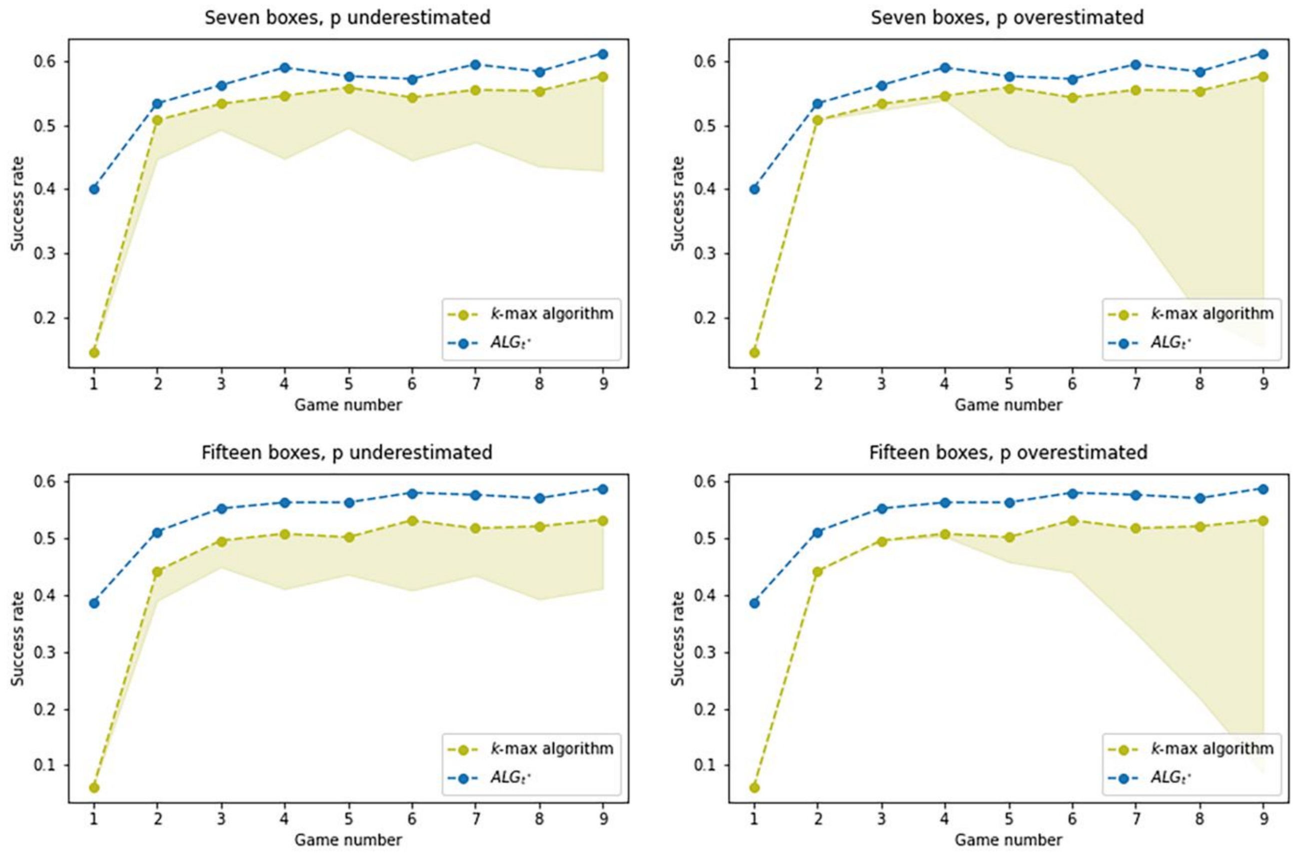
the chosen value of a near-optimal player and that of ALG_t differ by at most one. We observe that there is a strong learning effect in the first approximately four games. Starting at game 6, it remains relatively stable, and it even decreases slightly in the subsequent games. Nonetheless, around 80% of the players (and a bit less for the harder case of 15 boxes) learn how to play close to optimal after only a few games.

6.3. Experimental Results for the Optimal AOS*p* Algorithm

In this section, we evaluate the k-max algorithm on the same data set. We perform a sensitivity analysis for the k-max algorithm by allowing an estimation error in the value of p up to 0.1. Figure 4 shows the performance of the algorithm again in the first nine games and the deviation in its guarantees by allowing p to vary. For reference, we plot again the success rate of ALG_t. There are separate plots for the error resulting from an underestimation and an overestimation because they exhibit different types of deviation. The performance of the k-max algorithm has a huge improvement between games 1 and 2, and after game 3, its performance remains stable and only a few percentage points below that of ALG_t.

As for the robustness of the k-max algorithm with respect to misspecification of the parameter p, Figure 4 shows what one might expect: Running the algorithm with slightly wrong p does not play a big role for the first few games because the same or a close (in ranking) sample will be selected as the threshold (recall, e.g., that AOSp takes the same threshold for all $p \in [0, 1/2)$). Later on, however, we see that the deviation from the guarantee of k-max starts increasing. Because the value of p is only slightly different for two consecutive games, using some p' instead of p will likely result in very different threshold choices. For this reason, the deviation when p gets overestimated explodes (resulting in practically no

Figure 4. (Color online) Evaluation of AOSp on the Data Set of Goldstein et al. (2020)



Notes. The errors correspond to wrong estimation of p up to 0.1. For each of the cases of 7 and 15 boxes, the left plot corresponds to an underestimation of the right value of p and the right plot to an overestimation.

guarantee by game 9), whereas in the case of underestimation, the deviation increases smoothly with the number of games. Take as an example game 8: For this game, the correct p is $7/8$ and the resulting k is 8; that is, the threshold is set to the eighth largest value from a previous game. If we were to underestimate p by 0.1, we would set $k' = 4$. This misspecified k' is different from the correct one, but not too far away, so we can still hope to catch the maximum in some instances, despite the threshold being higher than before. On the other hand, if we were to overestimate p by 0.1, we would set $k'' = 40$. This is way off the right value of k, resulting in setting a very low threshold (we would expect to accept an online value among the very first ones) and thus achieving a much lower guarantee.

Further Discussion and Insights. From Figures 3 and 4, we get a good view of how our algorithms perform for finite, and in particular, small values of n. ALG_t achieves guarantees that are slightly above its worst-case theoretical ones and, as p increases, it follows a similar trajectory to the worst case (which occurs when $n \to \infty$). The case for the k-max algorithm is slightly different: on the one hand, it also follows a similar trajectory to the theoretical worst-case guarantee as p increases, but, on the other hand, it performs much better than the worst-case bounds. Finally, recall that Section 5 includes an analysis of the robustness of the k-max algorithm for each game separately, where we show how the deviation from the success rate changes as the estimation error of p varies.

7. Extensions

We expect that the ideas developed in this paper will prove useful in other contexts related to online decision making. To motivate further work, we discuss two relatively straightforward extensions of our model and results. These extensions are concerned with the well-studied matroid secretary problem. Here, the decision maker faces a sequence of the elements of a given ground set and needs to select a subset, subject to the constraint that the selected set has to be an independent set of an underlying matroid.

7.1. Graphical Matroid Secretary Problem with Independent Sampling

First, consider the graphical matroid secretary problem, in which the underlying matroid is graphical (i.e., the independent sets are forests of an undirected graph). For this problem, Korula and Pál (2009) gave a $1/(2e)$-competitive algorithm for the case in which the elements are presented to the decision maker in random order. In a nutshell, Korula and Pal fix an ordering of the vertices and with probability $1/2$ they orient all edges from the lower numbered vertex to the higher numbered vertex, and with probability $1/2$ they orient all edges in the other direction. Then they run for each vertex independently a standard secretary algorithm to find the maximum-weight edge leaving this vertex. It is not difficult to see that this gives a $1/(2e)$-approximation.

Now consider the random order graphical matroid secretary problem with independent sampling, in which every edge is sampled independently a priori with a fixed probability p and the goal is to select the maximum-weight independent set of the nonsampled elements. Let $\alpha^*_R(p)$ be the success guarantee of ROSp obtained in this paper. The analysis of Korula and Pál (2009) immediately yields a success guarantee of $\alpha^*_R(p)/2$ for this extension to graphical matroids.

7.2. Laminar Matroid Secretary Problem with Independent Sampling

Second, consider the laminar matroid secretary problem, in which underlying matroid is laminar.[9] For this class, a sequence of papers have obtained constant factor guarantees (Jaillet et al. 2013, Ma et al. 2016) until the currently best-known factor of 5.16 (Soto et al. 2021). All these papers rely on the idea of first (binomially) sampling a fraction of the elements of the matroid to guide the posterior decisions. However, the final goal is to compare against the optimal solution that includes even the sampled elements. An interesting direction will be to study the performance guarantees of these algorithms when the benchmark is the optimal solution of the online set as in this paper.

Moreover the technique of first using independent sampling is ubiquitous in secretary problems with combinatorial constraints. Therefore, we believe that understanding the performance guarantees when compared with the optimum of the online set is interesting not only from a theoretical perspective but also from a practical viewpoint since these samples can be interpreted as historical data.

Acknowledgments

The authors thank Daniel Goldstein, Preston McAfee, Siddharth Suri, and James Wright for kindly sharing their data with us; the review team whose remarks improved the contents and exhibition of this work significantly; Pranav Nuti for bringing up the work of Campbell and Samuels (1981); and Matthew Faw for pointing out the connection to the graphical and laminar matroid secretary problem. An earlier version of this article has been published in the proceedings of the ACM-SIAM Symposium on Discrete Algorithms 2021 conference (Correa et al. 2021a).

Endnotes

[1] Esfandiari et al. (2020) also obtained this result.

[2] The classic prophet inequality asserts that when faced with a sequence of n independent random variables, $X_1, \ldots, X_n$, a decision maker who knows their distributions and is allowed to stop the sequence at any time, can obtain, in expectation, at least half the reward of a prophet who knows the values of each realization.

[3] Interestingly, the model was proposed much earlier by Campbell and Samuels (1981) and recently rediscovered.

[4] Our results, and in particular the upper bounds on the success probability, remain true if the adversary knows all values $\alpha_1, \ldots, \alpha_n$ but not the result of the sampling process; that is, she does not know the random sets S and V.

[5] We recently became aware of the work of Campbell and Samuels (1981) who obtain very similar results. Indeed, they consider the dependent sampling version described earlier and obtain that the optimal success guarantee converges to γ as the fraction sampled grows to one. Their methods, however, are very different from ours and are significantly more complicated.

[6] Recall that we define the kth largest element from a set of less than k elements as $-\infty$.

[7] Observe that this lemma still holds in the setting where the order of the online elements is determined by the adversary after sampling because our algorithm is order oblivious.

[8] The optimal guarantee $\gamma \approx 0.58$ was first obtained numerically by Gilbert and Mosteller (1966). An explicit formula for γ was later found by Samuels (1982, 1991).

[9] A laminar family is a collection $\mathcal{A}$ of subsets of a ground set E such that, for any two intersecting sets, one is contained in the other. For a capacity function c on $\mathcal{A}$, a laminar matroid is given by the family of independent sets $\{I : |I \cap A| \leq c(A), \text{ for all } A \in \mathcal{A}\}$.

References

Allart P, Islas J (2015) A sharp lower bound for choosing the maximum of an independent sequence. *J. Appl. Probability* 53:1041–1051.

Alpern S, Baston V (2017) The secretary problem with a selection committee: Do conformist committees hire better secretaries? *Management Sci.* 63(4):1184–1197.

Angelovski A, Güth W (2020) When to stop: A cardinal secretary search experiment. *J. Math. Psych.* 98:102425.

Azar P, Kleinberg R, Weinberg SM (2014) Prophet inequalities with limited information. *Proc. Twenty-Fifth Annual ACM-SIAM Sympos. Discrete Algorithms (SODA '14)* (Society for Industrial and Applied Mathematics, Philadelphia), 1358–1377.

Babaioff M, Immorlica N, Kempe D, Kleinberg R (2007) A knapsack secretary problem with applications. Charikar M, Jansen K, Reingold O, Rolim JDP, eds. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. APPROX RANDOM 2007*, Lecture Notes in Computer Science, vol. 4627 (Springer, Berlin, Heidelberg).

Baumann C, Singmann H, Gershman SJ, von Helversen B (2020) A linear threshold model for optimal stopping behavior. *Proc. Natl. Acad. Sci. USA* 117(23):12750–12755.

Beyhaghi H, Kleinberg R (2019) Pandora's problem with nonobligatory inspection. *Proc. 2019 ACM Conf. Econom. Comput. (EC '19)* (Association for Computing Machinery, New York), 131–132.

Beyhaghi H, Golrezaei N, Paes Leme R, Pál M, Sivan B (2021) Improved revenue bounds for posted-price and second-price mechanisms. *Oper. Res.* 69(6):1805–1822.

Bradac D, Gupta A, Singla S, Zuzic G (2020) Robust algorithms for the secretary problem. *11th Innovations Theoret. Comput. Sci. Conf.*

(ITCS 2020), Leibniz International Proceedings in Informatics (LIPIcs), vol. 151 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 32:1–32:26.

Bruss FT (1984) A unified approach to a class of best choice problems with an unknown number of options. *Ann. Probability* 12(3):882–889.

Bruss FT (2000) Sum the odds to one and stop. *Ann. Probability* 28(3):1384–1391.

Campbell G, Samuels S (1981) Choosing the best from the current crop. *Adv. Appl. Probability* 13(3):510–532.

Chan HT-H, Chen F, Jiang SH-C (2015) Revealing optimal thresholds for generalized secretary problem via continuous LP: Impacts on online k-item auction and bipartite k-matching with random arrival order. *Proc. 2015 Annual ACM-SIAM Sympos. Discrete Algorithms (SODA)* (SIAM, Philadelphia), 1169–1188.

Chawla S, Hartline J, Malec D, Sivan B (2010) Multi-parameter mechanism design and sequential posted pricing. *Proc. Forty-Second ACM Sympos. Theory Comput. (STOC '10)* (Association for Computing Machinery, New York), 311–320.

Chen Y, Goldberg D (2018) Beating the curse of dimensionality in options pricing and optimal stopping. Preprint, submitted July 6, https://arxiv.org/abs/1807.02227.

Chen Y, Goldberg D (2019) A new approach to high-dimensional online decision-making.

Chen Y, Farias VF, Trichakis N (2019) On the efficacy of static prices for revenue management in the face of strategic customers. *Management Sci.* 65(12):5535–5555.

Ciocan DF, Mišić VV (2020) Interpretable optimal stopping. *Management Sci.* 68(3):1616–1638.

Correa J, Cristi A, Epstein B, Soto J (2020) The two-sided game of googol and sample-based prophet inequalities. *Proc. 2020 ACM-SIAM Sympos. Discrete Algorithms (SODA)* (SIAM, Philadelphia), 2066–2081.

Correa J, Cristi A, Epstein B, Soto J (2023) Sample-driven optimal stopping: From the secretary problem to the i.i.d. prophet inequality. *Math. Oper. Res.* 49(1):441–475.

Correa J, Dutting P, Fischer F, Schewior K (2019) Prophet inequalities for i.i.d. random variables from an unknown distribution. *Proc. 2019 ACM Conf. Econom. Comput. (EC '19)* (Association for Computing Machinery, New York), 3–17.

Correa J, Cristi A, Feuilloley L, Oosterwijk T, Tsigonias-Dimitriadis A (2021a) The secretary problem with independent sampling. *Proc. 2021 ACM-SIAM Sympos. Discrete Algorithms (SODA)* (SIAM, Philadelphia), 2047–2058.

Correa J, Foncea P, Hoeksma R, Oosterwijk T, Vredeveld T (2021b) Posted price mechanisms and optimal threshold strategies for random arrivals. *Math. Oper. Res.* 46(4):1452–1478.

Cownden D, Steinsaltz D (2014) Effects of competition in a secretary problem. *Oper. Res.* 62(1):104–113.

Derakhshan M, Golrezaei N, Paes Leme R (2021) Linear program-based approximation for personalized reserve prices. *Management Sci.* 68(3):1849–1864.

Doval L (2018) Whether to open Pandora's box. *J. Econom. Theory* 175:127–158.

Dütting P, Lattanzi S, Paes Leme R, Vassilvitskii S (2021) Secretaries with advice. *Proc. 22nd ACM Conf. Econom. Comput. (EC '21)* (Association for Computing Machinery, New York), 409–429.

Dynkin EB (1963) The optimum choice of the instant for stopping a Markov process. *Soviet Math. Dokl.* 4:627–629.

Esfandiari H, Korula N, Mirrokni V (2018) Allocation with traffic spikes: Mixing adversarial and stochastic models. *ACM Trans. Econom. Comput.* 6(3–4):14.

Esfandiari H, Hajiaghayi MT, Lucier B, Mitzenmacher M (2020) Prophets, secretaries, and maximizing the probability of choosing the best. *Internat. Conf. Artificial Intelligence Statist.* (PMLR, New York).

Ferguson TS (1989) Who solved the secretary problem? *Statist. Sci.* 4(3):282–296.

Gilbert J, Mosteller F (1966) Recognizing the maximum of a sequence. *J. Amer. Statist. Assoc.* 61:35–73.

Goldstein DG, McAfee PR, Suri S, Wright JR (2020) Learning when to stop searching. *Management Sci.* 66(3):1375–1394.

Golrezaei N, Jaillet P, Zhou Z (2022) Online resource allocation with samples. Available at SSRN.

Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning* (MIT Press, Cambridge, MA).

Hill T, Kertz R (1982) Comparisons of stop rule and supremum expectations of i.i.d. random variables. *Ann. Probability* 10(2): 336–345.

Hwang D, Jaillet P, Manshadi V (2021) Online resource allocation under partially predictable demand. *Oper. Res.* 69(3):895–915.

Immorlica N, Kleinberg RD, Mahdian M (2006) Secretary problems with competing employers. Spirakis P, Mavronicolas M, Kontogiannis S, eds. *Internet and Network Economics. WINE 2006*, Lecture Notes in Computer Science, vol. 4286 (Springer, Berlin, Heidelberg).

Jaillet P, Soto JA, Zenklusen R (2013) Advances on matroid secretary problems: Free order model and laminar case. Goemans M, Correa J, eds. *Integer Programming and Combinatorial Optimization. IPCO 2013*, Lecture Notes in Computer Science, vol. 7801 (Springer, Berlin, Heidelberg).

Kaplan H, Naori D, Raz D (2020) Competitive analysis with a sample and the secretary problem. *Proc. 2020 ACM-SIAM Sympos. Discrete Algorithms (SODA)* (SIAM, Philadelphia), 2082–2095.

Kesselheim T, Molinaro M (2020) Knapsack secretary with bursty adversary. *47th Internat. Colloquium Automata Languages Programming (ICALP 2020)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 168 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 72:1–72:15.

Kesselheim T, Radke K, Tönnis A, Vöcking B (2013) An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. Bodlaender HL, Italiano GF, eds. *Algorithms - ESA 2013. ESA 2013*, Lecture Notes in Computer Science, vol. 8125 (Springer, Berlin, Heidelberg).

Kleinberg R (2005) A multiple-choice secretary algorithm with applications to online auctions. *Proc. Sixteenth Annual ACM-SIAM Sympos. Discrete Algorithms (SODA '05)* (Society for Industrial and Applied Mathematics, Philadelphia), 630–631.

Korula N, Pál M (2009) Algorithms for secretary problems on graphs and hypergraphs. Albers S, Marchetti-Spaccamela A, Matias Y, Nikoletseas S, Thomas W, eds. *Automata, Languages and Programming. ICALP 2009*, Lecture Notes in Computer Science, vol. 5556 (Springer, Berlin, Heidelberg).

Krengel U, Sucheston L (1977) Semiamarts and finite values. *Bull. Amer. Math. Soc. (New Series)* 83:745–747.

Krengel U, Sucheston L (1978) On semiamarts, amarts, and processes with finite value. *Adv. Probability* 4:197–266.

Lindley DV (1961) Dynamic programming and decision theory. *J. Roy. Statist. Soc. Ser. C Appl. Statist.* 10:39–51.

Lykouris T, Vassilvitskii S (2021) Competitive caching with machine learned advice. *J. ACM* 68(4):1–25.

Ma W, Simchi-Levi D, Zhao J (2021) Dynamic pricing (and assortment) under a static calendar. *Management Sci.* 67(4):2292–2313.

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

Mahdian M, Nazerzadeh H, Saberi A (2012) Online optimization with uncertain information. *ACM Trans. Algorithms* 8(1):1–29.

Moran S, Snir M, Manber U (1985) Applications of Ramsey's theorem to decision tree complexity. *J. ACM* 32(4):938–949.

Nuti P (2022) The secretary problem with distributions. *Integer Programming and Combinatorial Optimization, 23rd International*

Conference, IPCO 2022, Eindhoven, The Netherlands, June 27–29, 2022, Proceedings (Springer-Verlag, Berlin, Heidelberg), 429–439.

Nuti P, Vondrák J (2023) Secretary problems: The power of a single sample. *Proc. 2023 Annual ACM-SIAM Sympos. Discrete Algorithms (SODA)* (SIAM, Philadelphia), 2015–2029.

Rubinstein A, Wang JZ, Weinberg SM (2020) Optimal single-choice prophet inequalities from samples. *11th Innovations Theoret. Comput. Sci. Conf. (ITCS 2020)*, Leibniz International Proceedings in Informatics (LIPIcs), vol. 151 (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany), 60:1–60:10.

Samuels S (1982) Exact solutions for the full information best choice problem. Technical report number 82-17, Department of Statistics, Purdue University, West Lafayette, IN.

Samuels S (1991) Secretary problems. Ghosh B, Sen P, eds. *Handbook of Sequential Analysis* (CRC Press, Boca Raton, FL), 381–405.

Seale DA, Rapoport A (1997) Sequential decision making with relative ranks: An experimental investigation of the "secretary problem". *Organ. Behav. Human Decision Processes* 69(3):221–236.

Soto J, Turkieltaub A, Verdugo V (2021) Strong algorithms for the ordinal matroid secretary problem. *Math. Oper. Res.* 46(2): 642–673.

Weitzman M (1979) Optimal search for the best alternative. *Econometrica* 47(3):641–654.

Zwick R, Rapoport A, King Chung Lo A, Muthukrishnan AV (2003) Consumer sequential search: Not enough or too much? *Marketing Sci.* 22(4):503–519.