Journal of Machine Learning Research 23 (2022) 1-37 Submitted 6/21; Published 4/22

The Two-Sided Game of Googol*

José Correa CORREA@UCHILE.CL
Department of Industrial Engineering
Universidad de Chile
Santiago, 8320000, Chile

Andrés Cristi ANDRES.CRISTI@ING.UCHILE.CL
Department of Industrial Engineering
Universidad de Chile
Santiago, 8320000, Chile

Boris Epstein BORIS.EPSTEIN@COLUMBIA.EDU
Decision, Risk and Operations Division
Columbia Business School
New York, NY 10027, USA

José Soto JSOTO@DIM.UCHILE.CL
Department of Mathematical Engineering and Center for Mathematical Modeling CNRS IRL 2807
Universidad de Chile
Santiago, 8320000, Chile

Editor: Sebastien Bubeck

Abstract

The secretary problem or game of Googol are classic models for online selection problems. In this paper we consider a variant of the problem and explore its connections to data-driven online selection. Specifically, we are given n cards with arbitrary non-negative numbers written on both sides. The cards are randomly placed on n consecutive positions on a table, and for each card, the visible side is also selected at random. The player sees the visible side of all cards and wants to select the card with the maximum hidden value. To this end, the player flips the first card, sees its hidden value and decides whether to pick it or drop it and continue with the next card. We study algorithms for two natural objectives: maximizing the probability of selecting the maximum hidden value, and maximizing the expectation of the selected hidden value. For the former objective we obtain a simple 0.45292-competitive algorithm. For the latter, we obtain a 0.63518-competitive algorithm. Our main contribution is to set up a model allowing to transform probabilistic optimal stopping problems into purely combinatorial ones. For instance, we can apply our results to obtain lower bounds for the single sample prophet secretary problem.

Keywords: optimal stopping, prophet inequalities, data-driven decision making, secretary problem, sampling

∗. A prelimirary version appeared in the Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020).

1. Introduction

In the classic game of Googol we are given n cards with n different arbitrary positive numbers written on them. The cards are shuffled and spread on a table with the numbers facing down. The cards are flipped one at a time, in a uniform random order, and we have to decide when to stop. The goal is to maximize the probability that the last flipped card has the overall greatest number.

In this work we study a variant of this problem that we call *The two-sided game of Googol*. Similar to the classic version, we are given n cards that we have to flip in random uniform order. However, here the cards have numbers on both sides, so we have $2n$ different arbitrary non-negative numbers instead of n, written on each side of each card. The cards are shuffled and spread on a table so that, independently for each card, either side faces up with probability $1/2$. We can see all the numbers that landed facing up (while the other side is hidden), and flip one card at a time, revealing the number that was facing down. Again we have to decide when to stop. We study the *secretary* and the *prophet* variants. In the secretary variant, the goal is to maximize the probability of stopping at the maximum number over the numbers that landed facing down. In the prophet variant, the goal is to maximize the ratio between the expectation of the last number revealed before stopping, and the expected maximum of the numbers that landed facing down.

1.1 Optimal Stopping and Prophet Inequalities

The two-sided game of Googol naturally fits within the theory of optimal stopping theory which is concerned with choosing the right time to take a particular action, so as to maximize the expected reward. Two landmark examples within this theory are the secretary problem (or game of Googol) described above, and the prophet inequality. In the latter a gambler faces a finite sequence of non-negative independent random variables $X_1, \ldots, X_n$ with known distributions F_i from which iteratively a prize is drawn. After seeing a prize, the gambler can either accept the prize and the game ends, or reject the prize and the next prize is presented to her. The classical result of Krengel and Sucheston (1977, 1978) states that the gambler can obtain at least half of the expected reward that a prophet can make who knows the realizations of the prizes beforehand. That is, $\sup\{\mathbb{E}[X_t] : t \text{ stopping rule}\} \geq \frac{1}{2}\mathbb{E}\{\sup_{1\leq i\leq n} X_i\}$. Moreover, this bound is best possible. Remarkably, Samuel-Cahn (1984) showed that the bound of $1/2$ can be obtained by a single threshold rule, which stops as soon as a prize is above a fixed threshold. We refer the reader to the survey by Hill and Kertz (1992) for further classic results.

1.2 Posted price mechanisms and the prophet inequality

Posted price mechanisms are simple mechanisms used to sell one or many items among one or many buyers. The mechanism is straightforward, buyers arrive sequentially, and upon their arrival they are offered a (possibly different) price for the item(s). Based on her valuation for the item, the buyer decides whether to purchase or not. It turns out that there is a connection between posted price mechanisms and prophet inequalities. When there is one item and many buyers, the posted price mechanism has an analogy to the classic prophet inequality, where the arriving values can be seen as the valuations of the

arriving buyers and the threshold for accepting each value can be seen as the price offered to the buyers. The buyer will purchase the item if their valuation is higher than the offered price, the same way as the decision maker will stop at a certain value if it is greater than its threshold.

In recent years, motivated by the connections between optimal stopping and posted price mechanisms (Hajiaghayi et al., 2007; Chawla et al., 2010; Correa et al., 2019c), there has been a regained interest in studying algorithms for variants of the classic prophet inequalities. The survey of Lucier (2017) is a good starting point to understand the economic perspective of prophet inequalities, while the letter of Correa et al. (2019b) provides a recent overview of results for single choice prophet inequalities. Due to this regained interest, new variants of the prophet inequality setting have been studied, including problems when the gambler has to select multiple items, when the selected set has to satisfy combinatorial constraints, when the market is large, when prior information is inaccurate, among many others (Abolhassani et al., 2017; Kleinberg and Weinberg, 2012; Dutting et al., 2020; Ehsani et al., 2018; Dütting and Kesselheim, 2019).

1.3 Prophet Secretary Problem and data-driven variants

Particularly relevant to our work is the *prophet secretary* problem. In this version, a gambler faces a finite sequence of non-negative independent random variables $X_1, \ldots, X_n$ with known distributions F_i, as in the prophet inequality. However, the order in which these random variables are faced is a uniform random permutation, as in the secretary problem. As in the prophet inequality, the goal of the gambler is to maximize the expectation of the selected prize. The problem was first studied by Esfandiari et al. (2017) who found a bound of $1 - 1/e$. Later, Ehsani et al. (2018) showed that the bound of $1 - 1/e$ can even be achieved using a single threshold. The factor $1 - 1/e$ was first beaten by Azar et al. (2018) by a tiny margin, while the current best bound is 0.67 (Correa et al., 2021b). In terms of upper bounds it was unclear until very recently whether there was a separation between prophet secretary and the i.i.d prophet inequality, where the random variables are identically distributed. For the latter problem, Hill and Kertz (1982) provided the family of worst possible instances from which Kertz (1986) proved the largest possible bound one could expect is 0.7451. Correa et al. (2017) established that this value is actually tight for the i.i.d. prophet inequality. On the other hand, Correa et al. (2021b) showed that no algorithm can achieve an approximation factor better than $\sqrt{3} - 1 \approx 0.73$ for the prophet secretary problem, establishing the separation between the problems. Interestingly, Liu et al. (2020) show that this separation is somewhat *weak* by showing that if we are allowed to remove a small number of random variables from the prophet secretary problem one can recover the 0.7451 bound of the i.i.d. case.

Some recent work has started to investigate data-driven versions of the prophet inequality, since the full distributional knowledge seems quite strong for many applications. In this context, Azar et al. (2014) consider a version in which the gambler only has access to one sample from each distribution. They prove a prophet inequality with an approximation guarantee of $1/4$ for this problem and left open whether achieving $1/2$ is possible. This question was recently answered on the positive by Rubinstein et al. (2020) who uses an elegant approach to prove that just taking the maximum of the samples as threshold leads to

the optimal guarantee. If the instance is further specialized to be i.i.d., until the conference version of the current paper the best known bound was $1-1/e$, while it was known that one cannot achieve a guarantee better than $\ln(2) \approx 0.69$ (Correa et al., 2019a). In subsequent work, the lower bound of $1-1/e$ was improved to 0.67 by Correa et al. (2020).

The classic approach to tackle data-driven variants of stochastic optimization problems has two steps: first learn the distributions and then optimize the objective using the learnt distribution. The analysis then focuses on calculating the estimation error and to what extent it is propagated by the optimization procedure. Most work on data-driven variants of the prophet inequality follows this strategy. In our problem, the number of samples is too small to learn the distributions (in fact, we only have one sample per distribution). Consequently, our approach is different: Noting that the samples and the actual values are interchangeable, we exploit the combinatorial structure of the problem to optimize the objective for every possible set of realized values. Very recent work has focused on using this technique, with exciting results. Examples of this line of research are the works of Correa et al. (2020), Correa et al. (2021a), Kaplan et al. (2020), Kaplan et al. (2022) and Caramanis et al. (2022).

1.4 Connection to the two-sided game of Googol

Our problem is closely related to the single-sample version of the secretary problem and the prophet secretary problem, which combine the data-driven approach of Azar et al. (2014) and the random arrival order of Esfandiari et al. (2017). In these problems we are given n distributions, which are unknown to us, and only have access to a single sample from each. Then n values are drawn, one from each distribution, and presented to us in random order. When we get to see a value, we have to decide whether to keep it and stop, or to drop it and continue. In the single-sample secretary problem the goal again is to maximize the probability of stopping with the largest value, while in the single-sample prophet secretary problem the goal is to maximize the expectation of the value at which we stopped divided by the expectation of the maximum of the values.

It is immediate to observe that an algorithm for the two-sided game of Googol that has a guarantee of α, for either the secretary or prophet variants of the problem, readily implies the same approximation guarantee for the single-sample secretary problem and single-sample prophet secretary problem respectively, as noted in Rubinstein et al. (2020). Indeed, if we consider an instance of the two-sided game of Googol where the values on card i are independent draws from a distribution F_i we exactly obtain the single-sample secretary and the single-sample prophet secretary problems.

1.5 Our results

Most of our results come from analyzing three basic algorithms. The first is the *Open moving window* algorithm, in which we stop the first time the card just flipped is larger than all the currently visible values. The second is the *Closed moving window* algorithm, in which we additionally require that in the last flipped card the value just revealed is larger than the value that was visible before. Finally we consider the *Full window* algorithm in which we simply take the largest value initially visible as threshold (and therefore stop the first time we see a value larger than all values we have seen).

We first study the secretary variant of the two-sided game of Googol in which the goal is to maximize the probability of choosing the maximum hidden value. For this problem we prove that the closed window algorithm gets the maximum value with probability 0.4529. Of course, this value is more than $1/e$ which is the best possible for the classic secretary problem (Dynkin, 1963; Lindley, 1961; Ferguson, 1989), but it is less than 0.5801 which is the best possible guarantee for the i.i.d. full information secretary problem, i.e., when the full distribution is known (Gilbert and Mosteller, 1966). Remarkably, Nuti (2020) showed that this latter bound holds in the full information case with nonidentical random variables arriving in random order. For our two-sided game of Googol, an improved upper bound, of 0.502, for the probability of choosing the maximum hidden value can be derived from the work of Campbell and Samuels (1981).

Next, we concentrate on the main subject of this paper which is the prophet variant of the two-sided game of Googol. Here, the goal is to maximize the ratio between the expectation of the chosen value, and that of the maximum hidden value. In this case we start by observing that all three algorithms described above can only give a guarantee of $1/2$. Indeed, consider an instance with two cards. Card 1 has values 0 and 1 on each side while card 2 has the numbers ε and ε^2. Clearly the expectation of the maximum hidden value is $1/2 + O(\varepsilon)$ (the value 1 is hidden with probability $1/2$) while the open moving window algorithm gets in expectation $1/4+O(\varepsilon)$ (to get the value of 1 the algorithm needs that it is hidden and that card 1 is the first card). For the other two algorithms, consider the instance in which card 1 has the values 1 and $1-\varepsilon$ while card 2 has the values ε and 0. Clearly the expectation of the maximum hidden value is $1 - O(\varepsilon)$, but since now neither of the algorithms can stop when $1-\varepsilon$ is revealed, both algorithms obtain $1/2+O(\varepsilon)$. However, by randomizing the choice of the algorithm we significantly improve the approximation ratio.

Our main result is to show that a simple randomization of the three proposed algorithms achieves a guarantee of $0.635 > 1 - 1/e$. Interestingly, our bound surpasses that of Azar et al. (2018) which until very recently was the best known bound for (the full information) prophet secretary. Furthermore, our bound also beats the bound of $1 - 1/e$ obtained by Correa et al. (2019a) for the i.i.d. single-sample prophet inequality.[1] So our bound not only improves upon these known bounds, but also works in a more general setting. The key behind the analysis is a very fine description of the performance of each of the three basic algorithms. Indeed for each algorithm we exactly compute the probability that they obtain any of the $2n$ possible values.[2] With this performance distribution at hand it remains to set the right probabilities of choosing each so as to maximize the expectation of the obtained value.

To wrap-up the work we consider a large data-set situation which naturally applies to a slightly restricted version of the single sample prophet secretary problem. The assumption states that, with high probability, if we rank all $2n$ values from largest to smallest, the position of the maximum over all cards of the minimum of the two values in the card lies far down the list (i.e., when ranking the $2n$ values the first few values correspond to a maximum in its card). This assumption holds for instance in the single-sample i.i.d. prophet inequality,

1. This was the best known lower bound for the problem until the conference version of the paper. This bound was improved to 0.67 in Correa et al. (2020).
2. In this respect, our result for the secretary variant can be seen as a warm up for the prophet variant of the problem.

Instance	Prophet Secretary $(\mathbb{E}(ALG) \geq \alpha \mathbb{E}(OPT))$	Secretary $\mathbb{P}(ALG = OPT) \geq \alpha$
Single Sample	$\frac{1}{2} \leq 0.635 \leq \alpha \leq \ln(2)$ [R20] [*] [C19]	$\frac{1}{e} \leq 0.452 \leq \alpha \leq 0.502$ [D63] [*] [CS81]
Full information	$0.669 \leq \alpha \leq \sqrt{3} - 1$ [CSZ21]	$\alpha = 0.5801$ [N20]

Table 1: Summary of known results for prophet secretary and secretary problems in the single-sample and full information settings. Results marked [*] are proven in this work. [R20] refers to Rubinstein et al. (2020), [C19] refers to Correa et al. (2019a), [D63] refers to Dynkin (1963), [CS81] refers to Campbell and Samuels (1981), [CSZ21] refers to Correa et al. (2021b), and [N20] refers to Nuti (2020).

under the large market assumption used by Abolhassani et al. (2017), or whenever the underlying distributions of the prophet secretary instance overlap enough. For this variant we design an optimal moving window algorithm and prove that it achieves an approximation ratio of 0.642.

Table 1 summarizes our results and the previous best bounds for the problems we consider.

2. Preliminaries, basic algorithms and statement of our results

2.1 Warm-up: Why is the problem hard?

Let us start the discussion by noting that the two-sided game of Googol presents some difficulties that go much beyond those of the classic game of Googol. To illustrate these difficulties, we present the ordinal algorithms[3] that achieve the highest possible competitive ratio in the prophet variant of the problem, for $n = 2$ and $n = 3$.

In the case $n = 2$ the optimal algorithm turns out to be simple and natural. It can be explained as follows. When flipping the first card, three of the four numbers will be known. If the number that was revealed is the highest among the three numbers seen so far, then stop with that number. Otherwise, select the number hidden in the second card. A slightly tedious, but straightforward calculation shows that this algorithm achieves an approximation ratio of 3/4.

However, the situation changes dramatically when $n = 3$. This case already illustrates how complicated the optimal ordinal algorithm can be, and serves to motivate the development of simple algorithms that perform competitively. In what follows we describe the optimal ordinal algorithm in this case. This is obtained through a factor revealing linear

3. An algorithm which decides to stop or not based on the relative ranking of the observed elements, not on the numerical values of the elements.

program which is developed in detail in Appendix A. Unfortunately, the size of the linear program grows exponentially with n, and the execution needs to compute a family of coefficients via brute force, making it intractable even for $n = 5$. So let us describe some of the surprising aspects of the optimal algorithm when $n = 3$. At the moment of flipping the first card, four numbers will be observed. If the revealed number is the highest among the four numbers observed so far, and the number at the other side of the flipped card is the fourth or third lowest among the four numbers observed so far, then stop. On the contrary, if the revealed number is the highest and the number on the other side of the flipped card is second highest, then stop with probability 0.95. This randomization seems odd, but it is necessary to achieve the optimal ratio. Indeed, if we require this sole case to be decided in a deterministic way (stop with probability 0 or 1), then the approximation ratio obtained is strictly lower. The first step of the optimal algorithm is completed by noting that if the revealed number on the first card is not the highest among the four observed, then the algorithm drops that number and continues. For the second step the situation gets even messier so an exhaustive description of the algorithm can be found in Table 2.1. Finally, at step 3 the algorithm always stops.

This delicate and surprising behavior can be explained by the fact that the algorithm has to be robust not only against the numbers written in the cards, as it happens for the classic game of Googol, but also against how the $2n$ numbers are coupled among the cards. The way that the numbers are coupled has an important impact on how the game can develop, in terms of how these rankings among the visible numbers change through the process. Consequently, under different couplings the same "picture" of the game can give very different information about which action is the most convenient to take. The algorithm then must carefully balance its performance in all of the possible couplings, otherwise the adversary would simply choose the worst case coupling.

2.2 Problem statement

Formal problem statement. We are given $2n$ different and arbitrary positive numbers, organized into n pairs that we denote as $\{(a_i, b_i)\}_{i=1}^n$, representing the numbers written in both sides of each card. These pairs of numbers are shuffled into sets $\mathcal{U}$ and $\mathcal{D}$: for each $i = 1, ..., n$ an independent unbiased coin is tossed, if the coin lands head, then $a_i = U_i \in \mathcal{U}$ and $b_i = D_i \in \mathcal{D}$, otherwise $b_i = U_i \in \mathcal{U}$ and $a_i = D_i \in \mathcal{D}$. The set $\mathcal{U} = \{U_1, ..., U_n\}$ represents the numbers that landed facing up, and the numbers in $\mathcal{D} = \{D_1, ..., D_n\}$ are those facing down. The numbers in $\mathcal{U}$ are revealed, and a random uniform permutation $\sigma \in \Sigma_n$ is drawn. Then the elements in $\mathcal{D}$ are revealed in n steps: at each step $i \in [n]$, the value $D_{\sigma(i)}$ and the index $\sigma(i)$ are revealed. After this, we must decide whether to stop or to continue to step $i + 1$.

We study algorithms (stopping rules) for the two variants. Let τ be the step at which we stop. In the secretary variant the objective is to maximize $\mathbb{P}(D_{\sigma(\tau)} = \max \mathcal{D})$, and in the prophet variant the objective is to maximize $\mathbb{E}(D_{\sigma(\tau)})/\mathbb{E}(\max \mathcal{D})$. Note that the latter is equivalent to just maximizing $\mathbb{E}(D_{\sigma(\tau)})$ since the algorithm does not control $\mathbb{E}(\max \mathcal{D})$.

Ranking and couples. Our analyses rely of the ranking of the $2n$ values of the instance, so let us denote by $Y_1 > Y_2 > \cdots > Y_{2n}$ the $2n$ numbers in $\mathcal{U} \cup \mathcal{D}$ ordered from the largest to the smallest. Without loss of generality we can assume that there are no ties among the

Card 1 ranks	Card 2 ranks	Card 3 ranks	Prob. of stopping
(1,4)	(2,3)	(5,?)	0.15
(1,5)	(2,3)	(4,?)	0.15
(2,3)	(4,1)	(5,?)	1
(2,3)	(5,1)	(4,?)	1
(2,4)	(3,1)	(5,?)	1
(2,4)	(5,1)	(3,?)	1
(2,5)	(4,1)	(3,?)	1
(2,5)	(3,1)	(4,?)	1
(3,2)	(4,1)	(5,?)	1
(3,2)	(5,1)	(4,?)	1
(3,4)	(1,2)	(5,?)	1
(3,4)	(2,1)	(5,?)	1
(3,4)	(5,1)	(2,?)	1
(3,5)	(1,2)	(4,?)	1
(3,5)	(2,1)	(4,?)	0.1
(3,5)	(4,1)	(2,?)	1
(4,2)	(1,3)	(5,?)	0.3
(4,3)	(2,1)	(5,?)	1
(4,3)	(5,1)	(2,?)	1
(4,5)	(1,2)	(3,?)	1
(4,5)	(2,1)	(3,?)	1
(4,5)	(3,1)	(2,?)	1
(4,5)	(3,2)	(1,?)	1
(5,3)	(1,2)	(4,?)	1
(5,3)	(2,1)	(4,?)	1
(5,3)	(4,1)	(2,?)	1
(5,4)	(1,2)	(3,?)	0.3
(5,4)	(2,1)	(3,?)	1
(5,4)	(2,3)	(1,?)	0.5
(5,4)	(3,1)	(2,?)	1

Table 2: Optimal policy at the second card for $n = 3$. The last column shows the probability of stopping with the second card in each possible scenario. The table is read as follows. The three cards are represented by an order pair (a, b), where a (respectively, b) represents the rank, among the five numbers seen so far, of the side of the card that was originally facing up (respectively, down). The ? sign in the third card simply represents that it has not been flipped yet. The first row, thus, means that if the initially hidden numbers from cards 1 and 2 are fifth and first highest seen so far, respectively, and the numbers on the other side of these cards are second and fourth, respectively, then stop with probability 1. In all configurations not shown, we do not stop with the second card and continue to the third card.

$2n$ values. Indeed, their difference can be arbitrarily small and we can modify our threshold algorithms so they break ties at random while maintaining the probabilities of selecting each value. We say that indices i and j are a couple, and denote it by $i \sim j$ if Y_i and Y_j are written in the two sides of the same card, i.e., if $(Y_i, Y_j) = (a_\ell, b_\ell)$ or $(Y_j, Y_i) = (a_\ell, b_\ell)$ for some $\ell \in [n]$.

Parameter k. We introduce k, an instance-dependent parameter that plays a key role in our analyses. We define k to be the smallest index so that k is the couple of some $k' < k$. In particular, $Y_1, Y_2, \ldots, Y_{k-1}$ are all written on different cards. Note that k and k' only depend on numbers in the cards[4], i.e., on the numbers $\{(a_i, b_i)\}_{i=1}^n$, and not on the coin tosses or on the permutation σ. We will restrict all our analysis to the numbers $Y_1, ..., Y_k$, which is easily justified by the observation that $\max \mathcal{D}$ always lies in the set $\{Y_1, ..., Y_k\}$.

Random arrival times. For most of our analyses it will be useful to consider the following reinterpretation of the randomness of both the coins tosses (hidden sides) and the random permutation (flipping order). Consider that each of the $2n$ numbers arrives in a uniform random time in the interval $(-1, 1]$ as follows. For every index ℓ smaller than its couple ℓ' (i.e. $\ell \sim \ell', Y_\ell > Y_{\ell'}$) an independent random arrival time θ_ℓ uniform in $(-1, 1]$ is sampled and ℓ' receives opposite arrival time $\theta_{\ell'} = C(\theta_\ell)$, where

$$C(x) = \begin{cases} x - 1, & \text{if } x > 0, \\ x + 1, & \text{otherwise.} \end{cases}$$

The reinterpretation can the be done as follows. For each $j \in [2n]$, if $\theta_j > 0$, the number Y_j is facing down, i.e., $Y_j \in \mathcal{D}$, and if $\theta_j \leq 0$ then Y_j is facing up, i.e., $Y_j \in \mathcal{U}$. Therefore, we get to see all values whose corresponding arrival time is negative and the order in which we scan the hidden values $Y_j \in \mathcal{D}$ is increasing in θ_j.

Throughout this work we use the term *value* to refer to the numbers written in the cards $Y_1, Y_2, \ldots, Y_{2n}$, while we use the term *element* to refer to a side of a particular card, that is an element in $\{U_1, \ldots, U_n, D_1, \ldots, D_n\}$. Of course these sets are the same so the point is the way they are ordered: the i-th value corresponds to Y_i, whereas the i-th element corresponds to the i-th number in the list $U_{\sigma(1)}, \ldots, U_{\sigma(n)}, D_{\sigma(1)}, \ldots, D_{\sigma(n)}$.

Once the random coins for all cards and the random uniform permutation are all selected, we are left with the following situation. A list $E = (e_1, \ldots, e_{2n}) = (U_{\sigma(1)}, \ldots, U_{\sigma(n)}, D_{\sigma(1)}, \ldots, D_{\sigma(n)})$ of $2n$ different positive elements are presented one-by-one to an algorithm. The algorithm must observe and skip the first n elements of the list, since they correspond to the values that landed facing up on the cards. The next n elements correspond to values that landed facing down (in such a way that the s-th and the $s + n$-th elements form a couple). The algorithm must decide immediately after observing the s-th element whether to select it and stop, or to continue with the next element.

2.3 Basic algorithms

We now present generic *moving window* algorithms for selecting one of the last n elements of a given list $E = (e_1, \ldots, e_{2n})$. This family of algorithms form the basis of the algorithms employed in this work, which are described next.

4. This notation was introduced by Rubinstein et al. (2020)

Generic moving window algorithm. For every $s \in \{n+1, \ldots, 2n\}$, the algorithm first specifies, possibly in a random or implicit way, a **window** of elements

$$W_s = \{e_r : r \in \{\ell_s, \ldots, s-1\}\} \tag{1}$$

scheduled to arrive immediately before s (the window may be empty). Then the algorithm proceeds as follows. It observes the first n elements without selecting them. For every $s \geq n+1$, the algorithm observes the s-th element, e_s, and selects it if e_s is larger than every element in its window[5], i.e. if $e_s > \max W_s$. Otherwise, the algorithm rejects it and continues.

We now present three basic moving window algorithms for deciding when to stop. Each of them can be described by a sequence of left extremes $(\ell_s)_{s=n+1}^{2n}$ that will define the elements inside the window W_s.

Open moving window algorithm ($\mathrm{ALG_o}$). This algorithm corresponds to the following strategy: flip the next unflipped card and accept its value x if and only if x is the largest of the n values that are currently visible (i.e., the i-th element is accepted if and only if it is larger than the $n-i$ currently unflipped cards, and is also larger than the $i-1$ previously flipped cards). In terms of the generic moving window, the left extreme of W_s is $\ell_s = s - n + 1$ for $n+1 \leq s \leq 2n$.

Closed moving window algorithm ($\mathrm{ALG_c}$). This algorithm works similarly to the previous one, with a slight difference. In this strategy, the element on the back of the last flipped card, the one that was facing up, is also required to be smaller than the element to be selected. In terms of the generic moving window, the left extreme of W_s is $\ell_s = s - n$ for $n+1 \leq s \leq 2n$.

Full window algorithm ($\mathrm{ALG_f}$). This algorithm corresponds to the algorithm presented by Rubinstein et al. (2020). It stops when the revealed element is larger than all element seen so far. Note that this is equivalent to stop with the first element that is larger than all elements that landed facing up. In terms of the generic moving window, the left extreme of W_s is $\ell_s = 1$ for $n+1 \leq s \leq 2n$.

Randomizing between these three algorithms, we can obtain algorithm ALG_r^*, which picks $\mathrm{ALG_o}$ or $\mathrm{ALG_c}$ with probability $(1-r)/2$ each, and $\mathrm{ALG_f}$ with probability r.

2.4 Statement of our results

Our main results are:

Theorem 1 *For any instance of the two-sided game of Googol,*

$$\mathbb{P}(ALG_c = \max \mathcal{D}) \geq \ln 2 \left(1 - \frac{\ln 2}{2}\right) \approx 0.45292.$$

Theorem 2 *There exists a value $r^* \in [0,1]$ such that for any instance of the two-sided game of Googol, $\mathbb{E}(\mathrm{ALG}_{r^*}^*) \geq \alpha \mathbb{E}(\max \mathcal{D})$ with $\alpha = \frac{4-5\ln 2}{5-6\ln 2} \approx 0.635184$.*

5. If $W_s = \emptyset$ then we say that e_s is trivially selected.

The moving windows can also be seen from the perspective of the random arrival times. For ALG_o, the window for an element that arrives at time $\theta > 0$ can be defined as all elements arriving in $(\theta - 1, \theta)$. For ALG_c, this changes to $[\theta - 1, \theta)$, while for ALG_f it becomes $[-1, \theta)$. Therefore, we can define threshold functions[6] for $\mathrm{ALG_o}$, $\mathrm{ALG_c}$, and $\mathrm{ALG_f}$ as $T_o(\theta) = \max\{Y_j : \theta_j \in (\theta - 1, \theta)\}$, $T_c(\theta) = \max\{Y_j : \theta_j \in [\theta - 1, \theta)\}$ and $T_f(\theta) = \max\{Y_j : \theta_j \in [-1, \theta)\}$, respectively. This motivates the definition of a fourth algorithm, $\mathrm{ALG}(t)$, for which the window for an element that arrives at time θ is defined as all elements that arrive in $[\max\{-1, \theta - 1 - t\}, \theta)$.

Our third result shows that for large k, this type of algorithms perform better.

Theorem 3 *For $t^* \approx 0.1128904$, and any instance of the two-sided game of Googol, as k goes to infinity, $\mathbb{E}[\mathrm{ALG}(t^*)]/\mathbb{E}[\max \mathcal{D}]$ is at least $R_\infty(t^*) \approx 0.6426317$.*

The assumption that k goes to infinity holds, for instance, in the single-sample i.i.d. prophet inequality, under the large market assumption used by Abolhassani et al. (2017), or whenever the underlying distributions of a large prophet secretary instance overlap enough. Moreover, the guarantee of $\mathrm{ALG}(t)$ for not so large values of k is already very close to $R_\infty(t^*)$. For example, the guarantee of $\mathrm{ALG}(t)$ for $k \geq 20$ is at least $R_\infty(t^*) - 0.00031$.

2.5 Preliminary lemmas

To wrap up this section, we establish two basic lemmas that will be useful throughout the work. The first is used to compute the probability of selecting any element arriving before a given element e_s. It is clear to see that all our algorithms satisfy the conditions.

Lemma 4 (Moving window lemma) *Suppose the windows' left extremes $(\ell_s)_{s\in\{n+1,\dots 2n\}}$ specified by a moving window algorithm satisfy*

$$\ell_{n+1} \leq \ell_{n+2} \leq \cdots \leq \ell_{2n} \leq n + 1 \tag{2}$$

Let s be an index with $n + 1 \leq s$ and let $e_M = \max\{e_r \colon n + 1 \leq r \leq s\}$ be the maximum value arriving between positions $n + 1$ and s. The algorithm selects some element in the set $\{e_r \colon n + 1 \leq r \leq s\}$ if and only if e_M is larger than every element in its window, i.e. $e_M > \max W_M$. Moreover, if the window W_s of s is nonempty and $e_J = \max W_s$ then the algorithm does not stop strictly before e_s if and only if $J \leq n$ or ($J \geq n + 1$ and $e_J < \max W_J$).

Proof. Suppose the algorithm selects some e_r with $n + 1 \leq r \leq s$. Note that e_r cannot arrive after e_M because in that case $e_M > e_r$ would be in the window W_r of e_r contradicting that e_r was selected. Then e_r must be e_M or an element arriving before e_M. In any case, by condition (2), $\ell_r \leq \ell_M$. Since e_r is selected, $e_r \geq \max\{e_t \colon \ell_r \leq t \leq r - 1\} \geq \max\{e_t \colon \ell_M \leq t \leq n\}$. But then, as e_M is the largest element arriving between $n + 1$ and s, we also conclude that $e_M > \max\{e_t \colon \ell_M \leq t \leq M - 1\} = \max W_M$. For the converse suppose that $e_M > \max W_M$ then the algorithm will pick e_M unless it has already selected another

6. These thresholds specify the minimum value under which the algorithm stops for each possible arrival time.

element e_r arriving before e_M. In any case, the algorithm stops by selecting some element e_r, with $n+1 \leq r \leq s$.

Let us show the second statement of the lemma. Suppose that the window of s is nonempty and let J be the index of the maximum element in its window. Let also M' be the largest element arriving between steps $n+1$ and $s-1$. Since $e_{M'}$ is also in the window of s, we conclude that either $J \geq n+1$ and $M' = J$, or $J \leq n$ and $e_J > e_{M'}$. Suppose $J \geq n+1$. By the first part of the lemma, not stopping strictly before e_s is equivalent to $e_{M'} < \max W_{M'}$, and since $M' = J$, $e_J < \max W_J$. If on the other hand $J \leq n$ then $e_J > e_{M'}$ and by assumption (2), $\ell_{M'} \leq \ell_s \leq J \leq n+1 \leq M' \leq s-1$. We conclude that e_J is also in the window of M' and therefore $\max W_{M'} \geq e_J > e_{M'}$, which is equivalent to not stopping strictly before e_s. ■

As a final ingredient for our results we need the following lemma about independent uniform random variables in $(-1, 1]$.

Lemma 5 *Let $X_0, X_1, \ldots, X_m$ be i.i.d. random variables distributed uniformly in $(-1, 1]$. Define the event $E : X_0 > 0$ and, for all $i \in \{1, \ldots, m\}, X_i \in [X_0 - 1, X_0)$. Conditioned on event E, $\{1 - X_0, X_1 + 1 - X_0, X_2 + 1 - X_0, \ldots, X_m + 1 - X_0\}$ is a family of $m+1$ i.i.d. random variables distributed uniformly in $[0, 1)$.*

Proof. Conditioned on E and on the realization $a \in (0, 1]$ of the variable X_0, the family $\{X_i + 1 - a\}_{i \in \{1,\ldots,m\}}$ is mutually independent. Since this is true for every realization a of X_0, we can uncondition on a and deduce that the family $\{1 - X_0, X_1 + 1 - X_0, X_2 + 1 - X_0, \ldots, X_m + 1 - X_0\}$ is mutually independent only conditioned on E. Furthermore, since under E, X_0 distributes uniformly in $(0, 1]$ we conclude that $1 - X_0$ is uniform in $[0, 1)$. Finally, under E, for $i > 0$, X_i distributes uniformly in $[X_0 - 1, X_0)$, and therefore, $X_i + 1 - X_0$ distributes uniformly in $[0, 1)$. ■

3. Maximizing the probability of picking the maximum

In this section we present a lower bound for the secretary variant of the two-sided game of Googol. Recall that the objective is to maximize the probability of stopping at the maximum value that landed facing down. If we were maximizing the probability of stopping the overall maximum value (the classical secretary problem), the best strategy would be to skip a constant number of elements (roughly $1/e$) and then select the first element larger than all previous ones (Lindley (1961); Dynkin (1963)). Since we cannot select any number from the first half, we would be tempted to try the full window algorithm ALG_f that selects the first element in the second half that is larger than all previous ones. Unfortunately, this algorithm does not select any element if the top one, Y_1, appears in the first half. It turns out that the best algorithm among the three basic algorithms presented is the closed moving window algorithm ALG_c that we analyze below.

Theorem 1 *For any instance of the two-sided game of Googol,*

$$\mathbb{P}(ALG_c = \max \mathcal{D}) \geq \ln 2 \left(1 - \frac{\ln 2}{2}\right) \approx 0.45292.$$

To prove Theorem 1 the interpretation using random arrival times over $(-1, 1]$ will be of use. Recall that k is the smallest index so that k is the couple of some $k' < k$. The idea of the proof is to compute, for a fixed value k, the probability of winning, that is, selecting the largest number in $\mathcal{D}$. To compute this, we partition the event of winning by identifying which value Y_i is picked and which element Y_j is the largest inside its window. We find out that the worst case is when $k \to \infty$ and conclude the bound.

Define the events E_1: Y_j is the largest element inside Y_i's window, E_2: $Y_i = \max \mathcal{D}$, E_3: the algorithm selects Y_i. Define Q_{ij} as the event of E_1, E_2 and E_3 happening simultaneously. The following lemma will be of use to compute the probability of winning.

Lemma 6 *For $1 \leq i \leq k-1$,H*

$$\mathbb{P}(Q_{ij}) = \begin{cases} \frac{1}{2^j}\frac{1}{i}\left(1-\frac{1}{j}\right) & i+1 \leq j \leq k-1, \\ \frac{1}{2^{k-1}}\frac{1}{i} & j = k, \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. If $j < i$ and E_1 holds, Y_i will not be selected, therefore $\mathbb{P}(Q_{ij}|j < i) = 0$. Also, note that a window of length 1 will contain either Y_k or $Y_{k'}$, where k' is the couple of k, so if $j > k$, E_1 can not hold and $\mathbb{P}(Q_{ij}|j > k) = 0$. So assume that $i + 1 \leq j \leq k$.

Observe now that E_1 is equivalent to the event that $C(\theta_1), \ldots, C(\theta_{i-1})$, $C(\theta_{i+1}), \ldots, C(\theta_{j-1}), \theta_j$ are inside the interval $[\theta_i - 1, \theta_i)$. As this interval has length 1 and all variables are independent and uniformly chosen from $[-1, 1)$, $\mathbb{P}(E_1 \,|\, \theta_i > 0, j < k) = 1/2^{j-1}$. The case $j = k$ is slightly different, because $\theta_k = C(\theta_{k'})$. If $i = k'$ then $\theta_k = \theta_i - 1 \in [\theta_i - 1, \theta_i)$, and if $i \neq k'$ then $\theta_k = C(\theta_l)$ for some $l \in \{1, \ldots, j-1\} \setminus \{i\}$. In both cases, we need to impose only $j - 2$ variables in $[\theta_i - 1, \theta_i)$, resulting in $\mathbb{P}(E_1 \,|\, \theta_i > 0, j = k) = 1/2^{k-2}$.

Define the auxiliary events $F_1 : \max\{0, C(\theta_1), \ldots, C(\theta_{i-1})\} = 0$ and $F_2 : \max\{0, C(\theta_1), \ldots, C(\theta_{i-1}), C(\theta_{i+1}), \ldots, C(\theta_{j-1}), \theta_j)\} \neq \theta_j$. Note that given E_1, event E_2 is equivalent to $\theta_1, \ldots, \theta_{i-1} \leq 0 < \theta_i$, which in turn is equivalent to $\theta_i > 0$ and F_1. From here E_1 and E_2 happening simultaneously is equivalent to E_1, F_1 and $\theta_i > 0$. Also, by Lemma 4, E_3 is equivalent to either $\theta_j \leq 0$, or $\theta_j > 0$ and Y_j is smaller than some element arriving in $[\theta_j - 1, \theta_j)$.

We claim first that for $j = k$, Q_{ik} is equivalent to E_1, F_1 and $\theta_i > 0$ (which is equivalent to E_1, E_2). Indeed, suppose that E_1, E_2 hold. If $j = k$ we distinguish 2 cases. If $i \sim k$, then we have that $\theta_k \leq 0$. If $i \not\sim k$ and $\theta_k > 0$ then the couple k' of k, will satisfy that $Y_{k'} > Y_k$ and $\theta_{k'} \in [\theta_j - 1, \theta_j]$. In both cases, E_3 holds.

Now we claim that for $1 \leq j \leq k-1$, Q_{ij} is equivalent to all events E_1, F_1, F_2 and $\theta_i > 0$ happening simultaneously. For that, suppose that E_1, and E_2 hold. Under that conditioning, E_3 holds in two cases, either $\theta_j \leq 0$, or $\theta_j > 0$ and for some $\ell \in \{1, \ldots, j-1\} \setminus \{i\}$, $\theta_j - 1 \leq \theta_\ell < \theta_j$, the latter being the same as $C(\theta_\ell) \in [-1, \theta_j - 1) \cup [\theta_j, 1]$. But recall that by E_1, every $\ell \in \{1, \ldots, j-1\} \setminus \{i\}$ satisfies $C(\theta_\ell) \in [\theta_i - 1, \theta_i)$. Therefore, E_3 holds if and only if $\theta_j \leq 0$ or for some $\ell \in \{1, \ldots, j-1\} \setminus \{i\}$, $C(\ell) \in [\theta_j, \theta_i)$. This is equivalent to event F_2, concluding the claim.

We are ready to compute the probability of Q_{ij}. Note that θ_j being or not the maximum of the set $\{0, C(\theta_1), \ldots, C(\theta_{i-1}), C(\theta_{i+1}), \ldots, C(\theta_{j-1}), \theta_j\}$ does not depend on the inner ordering of the set $\{0, C(\theta_1), \ldots, C(\theta_{i-1})\} = 0$. This implies that F_1 and F_2 are independent

events. Since F_1 is equivalent to $\max\{1-\theta_i, C(\theta_1)+1-\theta_i, \ldots, C(\theta_{i-1})+1-\theta_i\} = 1-\theta_i$, from Lemma 5, $\mathbb{P}(F_1|E_1, \theta_i > 0) = 1/i$, and since F_2 is equivalent to $\max\{1-\theta_i, C(\theta_1)+1-\theta_i, \ldots, C(\theta_{i-1})+1-\theta_i, C(\theta_{i+1})+1-\theta_i, \ldots, C(\theta_{j-1})+1-\theta_i, \theta_j+1-\theta_i\} \neq \theta_j+1-\theta_i$, $\mathbb{P}(F_2|E_1, \theta_i > 0) = 1-1/j$. Putting all together and using independence, $\mathbb{P}(Q_{ik}) = \mathbb{P}(E_1, \theta_i > 0, F_1) = \frac{1}{2}\frac{1}{2^{k-2}}\frac{1}{i}$, and for $1 \leq j \leq k-1$ $\mathbb{P}(Q_{ij}) = \mathbb{P}(E_1, \theta_i > 0, F_1, F_2) = \frac{1}{2}\frac{1}{2^{j-1}}\frac{1}{i}\left(1-\frac{1}{j}\right)$. This concludes the proof of the lemma. ■

With Lemma 6, we can compute the probability of winning as:

$$\begin{aligned}\mathbb{P}(\text{Win}) &= \sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\mathbb{P}(Q_{ij}) = \sum_{i=1}^{k-1}\left(\sum_{j=i+1}^{k-1}\frac{1}{2^j}\frac{1}{i}\left(1-\frac{1}{j}\right)+\frac{1}{2^{k-1}}\frac{1}{i}\right)\\ &= \frac{H_{k-1}}{2^{k-1}} + \sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1}\frac{1}{2^j}\frac{1}{i}\frac{j-1}{j},\end{aligned}$$

where $H_s = \sum_{i=1}^{s}\frac{1}{i}$ is the s-th harmonic number. Note that the probability of winning only depends on k and denote it $F(k)$. The following lemma helps to find the worst case scenario for our algorithm.

Lemma 7 *$F(k)$ is non-increasing in k.*

Proof. To prove the lemma we note that for $k \geq 2$:

$$\begin{aligned}F(k+1) - F(k) &= \frac{H_k}{2^k} - \frac{H_{k-1}}{2^{k-1}} + \sum_{i=1}^{k-1}\frac{1}{2^k}\frac{1}{i}\left(1-\frac{1}{k}\right)\\ &= \frac{H_k}{2^k} + \frac{H_{k-1}}{2^k}\left(1-\frac{1}{k}\right) - \frac{H_{k-1}}{2^{k-1}} = \frac{1-H_{k-1}}{k2^k} \leq 0.\end{aligned}$$

■

Proof of Theorem 1. To prove the main theorem of the section we use the previous results to give a lower bound for the probability of winning. The worst case happens for $k \to \infty$:

$$\begin{aligned}\mathbb{P}(\text{Win}) &\geq \min_{k\geq 2} F_k = \lim_{k\to\infty} F_k = \sum_{i=1}^{\infty}\frac{1}{i}\sum_{j=i+1}^{\infty}\frac{1}{2^j}\left(1-\frac{1}{j}\right)\\ &= \sum_{i=1}^{\infty}\frac{1}{i}\sum_{j=i+1}^{\infty}\left(\frac{1}{2^j} - \int_0^{1/2}x^{j-1}dx\right) = \sum_{i=1}^{\infty}\frac{1}{i}\left(\frac{1}{2^i} - \int_0^{1/2}\frac{x^i}{1-x}dx\right)\\ &= \sum_{i=1}^{\infty}\int_0^{1/2}x^{i-1}dx - \int_0^{1/2}\frac{1}{1-x}\int_0^x\sum_{i=1}^{\infty}y^{i-1}dydx\\ &= \int_0^{1/2}\frac{1}{1-x}dx - \int_0^{1/2}\frac{1}{1-x}\int_0^x\frac{1}{1-y}dydx\\ &= \ln 2\left(1-\frac{\ln 2}{2}\right) \approx 0.45292.\end{aligned}$$

■

Notice that F_k is exactly the probability that ALG_c wins. This implies that this bound is the best we can hope for using this specific algorithm.

4. Maximizing the expected value

In this section we present a lower bound for the prophet variant of the two-sided game of Googol, in which the objective is to maximize $\mathbb{E}(D_{\sigma(\tau)})/\mathbb{E}(\max \mathcal{D})$. We analyze a combination of the three basic algorithms presented in Section 2. More precisely, for a fixed $r \in [0,1]$, we define ALG_r^* as follows. With probability $(1-r)/2$ run ALG_o, with probability $(1-r)/2$ run ALG_c, and with probability r run ALG_f. This combination allows us to obtain the main result of the section.

Theorem 2 *There exists a value $r^* \in [0,1]$ such that for any instance of the two-sided game of Googol, $\mathbb{E}(\mathrm{ALG}_{r^*}^*) \geq \alpha \mathbb{E}(\max \mathcal{D})$ with $\alpha = \frac{4-5\ln 2}{5-6\ln 2} \approx 0.635184$.*

This ratio is better than $1 - 1/e \approx 0.632$ obtained in Correa et al. (2019a), which until the conference version of this paper was the best know bound for for the i.i.d. prophet secretary problem with samples, which is a particular case of our problem.

Recall that none of the 3 basic algorithms can guarantee a ratio better than $\frac{1}{2}$ by itself, even in very simple cases. Indeed, as shown in the introduction, for the instance $\{(1,0),(\varepsilon,\varepsilon^2)\}$ we have that $\mathbb{E}(\max \mathcal{D}) = 1/2 + O(\varepsilon)$ and $\mathbb{E}(\mathrm{ALG_o}) = 1/4 + O(\varepsilon)$. Furthermore, if we consider the instance $\{(1,1-\varepsilon),(\varepsilon,0)\}$ then $\mathbb{E}(\max \mathcal{D}) = 1 - O(\varepsilon)$ but $\mathbb{E}(\mathrm{ALG_c}) = \mathbb{E}(\mathrm{ALG_f}) = 1/2 + O(\varepsilon)$.

The intuition of our result is that the situations where each of the three algorithms perform poorly are very different. As we are able to compute exactly the distribution of the outcome of each algorithm, we can balance their distributions, so that in all cases ALG_r^* performs well. We can state this in terms of the ordered values $Y_1, ..., Y_k$. On the one hand, since $\mathrm{ALG_f}$ is very restrictive, it has good chances of stopping at Y_1 when it is in $\mathcal{D}$. On the other hand, $\mathrm{ALG_c}$ has higher probability of stopping at Y_i than $\mathrm{ALG_f}$, for $1 < i < k$. But none of the two algorithms can stop at Y_k, whereas $\mathrm{ALG_o}$ stops at Y_k with positive probability. This comes at the cost of sacrificing a bit of the better elements, but as showed in the examples, it is very important for the case when k is small.

Roughly speaking, the proof of Theorem 2 goes as follows. For each of the basic algorithms we compute in lemmas 8, 9 and 10 the probability that they stop with the value Y_j for each $j < k$, and express it as a series truncated in k and an extra term that depends on k. Then in Lemma 11 we prove that in the combined formula, for certain values of r, the extra term can be replaced with the tail of the series, so the formula does not depend on k. Finally, for a specific value of r, we show in lemmas 12 and 13 an approximate stochastic dominance and conclude, i.e., we make use of the following general observation.

Observation 1 *If $\min_{j \leq k} \frac{\mathbb{P}(\mathrm{ALG} \geq Y_j)}{\mathbb{P}(\max \mathcal{D} \geq Y_j)} \geq \alpha$, then $\mathbb{E}(\mathrm{ALG}) \geq \alpha \mathbb{E}(\max \mathcal{D})$.*[7]

4.1 Distribution of the maximum

As noted before, $\mathbb{P}(\max \mathcal{D} = Y_i) = 0$ if $i > k$. For the remaining values, Rubinstein et al. (2020) did a simple analysis for the distribution of $\max \mathcal{D}$.

For Y_i to be the maximum of $\mathcal{D}$ we need: (1) $Y_i \in \mathcal{D}$ and (2) $Y_j \in \mathcal{U}$ for all $1 \leq j < i$. If $1 \leq i < k$ all these events are independent with probability $\frac{1}{2}$, so we have $\mathbb{P}(\max \mathcal{D} =$

7. Recall that $\max \mathcal{D}$ can only take values in $\{Y_1, ..., Y_k\}$, so the implication follows immediately.

$Y_i) = (\frac{1}{2})^i$. If $i = k$, the fact that Y_k is facing downwards implies that its couple is facing upwards, so we need one less coin toss for the events to happen simultaneously. Putting all together:

$$\mathbb{P}(\max \mathcal{D} = Y_i) = \begin{cases} \frac{1}{2^i} & i < k \\ \frac{1}{2^{k-1}} & i = k \\ 0 & i > k \end{cases}$$

Therefore we have that, $\mathbb{P}(\max \mathcal{D} \geq Y_i) = 1 - \frac{1}{2^i}$ if $1 \leq i < k$ and $\mathbb{P}(\max \mathcal{D} \geq Y_k) = 1$.

4.2 Analysis of the basic algorithms

In this section we precisely derive the distribution of the obtained value of each of the basic algorithms. In the next section we combine these distributions to find an improved randomized algorithm. In what follows, we will denote by k' the couple of k. Note that the identities of k and k' depend only on the instance, and not on the realizations of the random coins or the random permutation.

Lemma 8 *For every* $1 \leq i \leq k-1$,

$$\mathbb{P}(\mathrm{ALG_c} = Y_i) = \frac{1}{2^{k-1}} + \sum_{j=i+1}^{k-1} \frac{1}{2^j} \left(1 - \frac{1}{j}\right)$$

and

$$\mathbb{P}(\mathrm{ALG_c} \geq Y_k) = \sum_{j=1}^{k-1} \frac{1}{2^j j}.$$

Proof. Intuitively, for each $j > i$, we condition on that Y_j is the maximum value in the window of Y_i and show that $\mathrm{ALG_c}$ stops in Y_i with probability $(1 - 1/j)$. More precisely, fix an index $i \leq k-1$. For $j \leq i, j \neq i$, denote by F_j the event that $\theta_i \geq 0$ (or equivalently $Y_i \in \mathcal{D}$) and that j is the smallest index such that $\theta_j \in [\theta_i - 1, \theta_i)$. If $j < k$ then $\mathbb{P}(F_j) = 1/2^j$ because the variables $\{\theta_\ell\}_{\ell \leq j}$ are all independent and uniformly distributed in $(-1, 1]$, and for any given $\theta_i \geq 0$, the interval $[\theta_i - 1, \theta_i)$ has length 1. For F_k we have that $\theta_k \in [\theta_i - 1, \theta_i)$ if and only if $\theta_{k'} = C(\theta_k) \notin [\theta_i - 1, \theta_i)$. Since the variables $\{\theta_\ell\}_{\ell \leq k, \ell \neq k'}$ are all independent, $\mathbb{P}(F_k) = 1/2^{k-1}$.

Note that if for some $j < i$, $\theta_j \in [\theta_i - 1, \theta_i)$, then $\mathrm{ALG_c} \neq Y_i$, because its threshold $T_c(\theta_i)$ will be at least $Y_j > Y_i$. Also note that either θ_k or $\theta_{k'}$ is in $[\theta_i - 1, \theta_i)$, so the events $(F_j)_{j=i+1}^k$ (that are pairwise disjoint) completely cover the event $\{\mathrm{ALG_c} = Y_i\}$. Thus we have the identity

$$\mathbb{P}(\mathrm{ALG_c} = Y_i) = \frac{1}{2^{k-1}} \mathbb{P}(\mathrm{ALG_c} = Y_i | F_k) + \sum_{j=i+1}^{k-1} \frac{1}{2^j} \cdot \mathbb{P}(\mathrm{ALG_c} = Y_i | F_j). \tag{3}$$

Now, for $j > i$, conditional on F_j, $\mathrm{ALG_c} = Y_i$ if and only if $\mathrm{ALG_c}$ does not stop before observing Y_i. Using lemmas 4 and 5 we will show that $\mathbb{P}(\mathrm{ALG_c} = Y_i | F_k) = 1$ and that $\mathbb{P}(\mathrm{ALG_c} = Y_i | F_j) = (1 - 1/j)$ if $i < j < k$, and conclude the first formula of the lemma.

Conditional on F_k, the maximum value in the window of Y_i is Y_k, but Y_k is not larger than all elements in its window, because $Y_{k'}$ is in it (it is on the other side of its card). So Lemma 4 implies that $\mathbb{P}(\mathrm{ALG_c} = Y_i | F_k) = 1$.

For the case $i < j < k$, we have that conditional on F_j, Y_j is the maximum value in the window of Y_i, so Lemma 4 implies that $\mathrm{ALG_c}$ stops in $(0, \theta_i)$ if and only if $\theta_j \geq 0$ and Y_j is larger than all elements in its window, i.e., $\theta_\ell \notin [\theta_j - 1, \theta_j)$, for all $\ell < j$. This is equivalent to $\max\{0, C(\theta_1), ..., C(\theta_{i-1}), C(\theta_{i+1}), ..., C(\theta_{j-1}), \theta_j\} = \theta_j$. Now, Lemma 5 implies that conditional on F_j, the random variables $\{0 - (\theta_i - 1), C(\theta_1) - (\theta_i - 1), ..., C(\theta_{i-1}) - (\theta_i - 1), C(\theta_{i+1}) - (\theta_i - 1), ..., C(\theta_{j-1}) - (\theta_i - 1), \theta_j - (\theta_i - 1)\}$ are independent and uniformly distributed in $(0, 1)$. Hence, conditional on F_j, $\mathrm{ALG_c}$ stops in $(0, \theta_i)$ with probability $1/j$, so $\mathbb{P}(\mathrm{ALG_c} = Y_i | F_j) = (1 - 1/j)$. This proves the first formula.

For computing $\mathbb{P}(\mathrm{ALG_c} \geq Y_k)$, which is the probability that $\mathrm{ALG_c}$ stops, define as E_j the event that j is the smallest index such that $\theta_j > 0$ (or equivalently, that $Y_j = \max \mathcal{D}$). It is clear that for $j < k$, $\mathbb{P}(E_j) = 1/2^j$, that $\mathbb{P}(E_k) = 1/2^{k-1}$, and that $\mathbb{P}(E_j) = 0$ if $j > k$. For $j < k$, conditional on E_j, Lemma 4 implies that $\mathrm{ALG_c}$ stops if and only if $\max\{C(\theta_1), C(\theta_2), ..., C(\theta_{j-1}), \theta_j\} = \theta_j$. But this happens with probability $1/j$. Conditional on E_k, $\mathrm{ALG_c}$ never stops. This concludes the proof of the lemma. ■

Lemma 9 *For every* $1 \leq i \leq k - 1$,

$$\mathbb{P}(\mathrm{ALG_o} = Y_i) = \frac{1}{2^{k-1}} \left(1 - \frac{\mathbb{1}_{\{i \neq k'\}}}{k-1}\right) + \sum_{j=i+1}^{k-1} \frac{1}{2^j} \left(1 - \frac{1}{j}\right)$$

and

$$\mathbb{P}(\mathrm{ALG_o} \geq Y_k) = \frac{1}{2^{k-1}(k-1)} + \sum_{j=1}^{k-1} \frac{1}{2^j j}.$$

Proof. Recall that $\mathrm{ALG_o}$ cannot select a value Y_j with $j > k$. Suppose now that we run $\mathrm{ALG_o}$ and $\mathrm{ALG_c}$ in the same instance and realization. Since $\mathrm{ALG_c}$ is more restrictive than $\mathrm{ALG_o}$, if $\mathrm{ALG_o}$ stops, it stops earlier than $\mathrm{ALG_c}$. Nevertheless, note that Y_k is the only value that can be accepted by $\mathrm{ALG_o}$ but not by $\mathrm{ALG_c}$. Hence, for $1 \leq i \leq k - 1$, if $\mathrm{ALG_o} = Y_i$ then $\mathrm{ALG_c} = Y_i$, and if $\mathrm{ALG_c} = Y_i$ then either $\mathrm{ALG_o} = Y_i$ or $\mathrm{ALG_o} = Y_k$. Thus, we can write the following identity for the case $1 \leq i \leq k - 1$.

$$\mathbb{P}(\mathrm{ALG_o} = Y_i) = \mathbb{P}(\mathrm{ALG_c} = Y_i) - \mathbb{P}(\mathrm{ALG_c} = Y_i \text{ and } \mathrm{ALG_o} = Y_k)\,. \tag{4}$$

Thus, we just need to compute the negative term in Equation (4). In order to have $\mathrm{ALG_o} = Y_k$ we need that $\theta_k > 0$ and that $\theta_j \notin (\theta_k - 1, \theta_k]$, for all $1 \leq j \leq k - 1$, with $j \neq k'$. Note that this is also a sufficient condition, because values that are smaller than Y_k cannot be accepted by $\mathrm{ALG_o}$. Thus, $\mathbb{P}(\mathrm{ALG_o} = Y_k) = 1/2^{k-1}$. If $i = k'$, then $\theta_i = \theta_{k'} = \theta_k - 1$, which is negative if θ_k is positive, so $\mathbb{P}(\mathrm{ALG_c} = Y_{k'} \text{ and } \mathrm{ALG_o} = Y_k) = 0$.

Assume now that $i \neq k'$. Conditional on $\mathrm{ALG_o} = Y_k$, we have that $\mathrm{ALG_c} = Y_i$ if Y_i arrives after Y_k and all other values in $\{Y_1, ..., Y_k\}$ arrive either before the window of Y_k or after Y_i, i.e., if $\theta_k < \theta_i$ and $\theta_j \in (-1, \theta_k - 1) \cup (\theta_i, 1]$, for all $j \leq k-1$, with $j \notin \{i, k'\}$. This is equivalent to say that $C(\theta_i) = \min\{0, C(\theta_1), ..., C(\theta_{k'-1}), C(\theta_{k'+1}), ..., C(\theta_{k-1})\}$. Note that Lemma 5 implies that conditional on $\mathrm{ALG_o} = Y_k$, the variables $\{0 - (\theta_k - 1), C(\theta_1) - (\theta_k -$

$1), ..., C(\theta_{k'-1}) - (\theta_k - 1), C(\theta_{k'+1}) - (\theta_k - 1), ..., C(\theta_{k-1}) - (\theta_k - 1)\}$ are independent and uniformly distributed in $[0, 1]$. Then, $\mathbb{P}(\mathrm{ALG_c} = Y_i | \mathrm{ALG_o} = Y_k) = 1/(k-1)$.

We conclude the first formula by replacing the just computed probability and the formula of Lemma 8 in Equation (4). For the second one, we have that $\mathbb{P}(\mathrm{ALG_o} \geq Y_k) = \sum_{i=1}^{k} \mathbb{P}(\mathrm{ALG_o} = Y_i) = \mathbb{P}(\mathrm{ALG_o} = Y_k) - (k-2)\frac{1}{2^{k-1}(k-1)} + \sum_{i=1}^{k-1} \mathbb{P}(\mathrm{ALG_c} = Y_i) = \frac{1}{2^k} - (k-2)\frac{1}{2^{k-1}(k-1)} + \mathbb{P}(\mathrm{ALG_c} \geq Y_k) = \mathbb{P}(\mathrm{ALG_c} \geq Y_k) + \frac{1}{2^{k-1}(k-1)}$. ■

Lemma 10 *For every* $1 \leq i \leq k-1$,

$$\mathbb{P}(\mathrm{ALG_f} = Y_i) = \frac{1}{2^{k-1}} \frac{1}{k-1} + \sum_{j=i+1}^{k-1} \frac{1}{2^j (j-1)}$$

and

$$\mathbb{P}(\mathrm{ALG_f} \geq Y_k) = \frac{1}{2}.$$

Proof. Let F_j be the event that Y_j is the maximum value (j is the minimum index) such that $\theta_j < 0$. Roughly speaking, we condition on each F_j and show that $\mathrm{ALG_f} = Y_i$ if Y_i is the value with earliest arrival time from $\{Y_1, ..., Y_{j-1}\}$.

Note that the events F_j are pairwise disjoint, and since either θ_k or $\theta_{k'}$ is negative, $\mathbb{P}(F_j) = 0$ if $j > k$. Thus, the events F_j for $1 \leq j \leq k$ form a partition of the probability space. Note also that conditional on F_j, by definition $\mathrm{ALG_f}$ cannot accept any value smaller than Y_j. Therefore $\mathbb{P}(\mathrm{ALG_f} = Y_i) = \sum_{j=i+1}^{k} \mathbb{P}(\mathrm{ALG_f} = Y_i | F_j) \mathbb{P}(F_j)$.

For $j \leq k-1$ the arrival times $\theta_1, ..., \theta_j$ are all independent and uniform in $(-1, 1]$, so $\mathbb{P}(F_j) = 1/2^j$. Conditional on F_j we have that $\theta_1, ..., \theta_{j-1}$ are independent and uniform in $[0, 1]$. Moreover, conditional on F_j, $\mathrm{ALG_f}$ simply accepts the element in $\{Y_1, ..., Y_{j-1}\}$ with earliest arrival time, as they are exactly the ones larger than $\max \mathcal{U} = Y_j$. Hence, $\mathbb{P}(\mathrm{ALG_f} = Y_i | F_j) = 1/(j-1)$.

The case of F_k is slightly different. F_k is equivalent to the event that $\theta_1, ..., \theta_{k-1} > 0$, because $\theta_{k'} > 0$ implies that $\theta_k < 0$, so $\mathbb{P}(F_k) = 1/2^{k-1}$. Again $\mathrm{ALG_f}$ simply accepts the first element larger than Y_k, so it accepts Y_i with probability $1/(k-1)$. This concludes the proof of the first formula.

For computing $\mathbb{P}(\mathrm{ALG_f} \geq Y_k)$, note that this is simply the probability that $\theta_1 > 0$, which is $1/2$. This is because Y_1 is the overall largest value, so if it arrives before 0, nothing can be accepted, and if it arrives after 0, then Y_1 itself (and possibly other values with earlier arrival times) can be accepted. ■

4.3 The combined algorithm

We now use the distributions obtained in the last section in order to obtain an improved randomized algorithm. To this end we combine lemmas 9, 8 and 10 to conclude that for

$1 \le i \le k-1$,

$$\mathbb{P}(\mathrm{ALG}_r^* = Y_i) = \frac{1-r}{2^{k-1}} + \frac{r - \mathbb{1}_{\{i \neq k'\}} \cdot (1-r)/2}{2^{k-1}(k-1)} + \sum_{j=i+1}^{k-1} \frac{1}{2^j}\left((1-r) - \frac{(1-r)}{j} + \frac{r}{j-1}\right)$$
$$\ge \frac{1-r}{2^{k-1}} + \frac{r-(1-r)/2}{2^{k-1}(k-1)} + \sum_{j=i+1}^{k-1} \frac{1}{2^j}\left((1-r) - \frac{(1-r)}{j} + \frac{r}{j-1}\right). \tag{5}$$

The inequality comes from the fact that $-\mathbb{1}_{\{i \neq k'\}} \ge -1$. Now we use the following lemma to complete the summation in Equation (5) using the extra term.

Lemma 11 *For every $\frac{3-4\ln(2)}{5-6\ln(2)} \le r \le 2/3$, and any $k \ge 2$,*

$$\sum_{j \ge k} \frac{1}{2^j}\left(1 - r - \frac{1-r}{j} + \frac{r}{j-1}\right) \le \frac{1-r}{2^{k-1}} + \frac{r-(1-r)/2}{2^{k-1}(k-1)}.$$

Proof. Rearranging terms, we obtain the following.

$$\begin{aligned}\sum_{j \ge k} \frac{1}{2^j}\left(1 - r - \frac{1-r}{j} + \frac{r}{j-1}\right) &= \frac{1-r}{2^{k-1}} - \sum_{j \ge k} \frac{1-r}{2^j j} + \sum_{j \ge k-1} \frac{r/2}{2^j j} \\ &= \frac{1-r}{2^{k-1}} + \frac{r/2}{2^{k-1}(k-1)} - \sum_{j \ge k} \frac{1-3r/2}{2^j j} \\ &= \frac{1-r}{2^{k-1}} + \frac{r/2}{2^{k-1}(k-1)} - \frac{(1-3r/2)}{2^{k-1}(k-1)} \sum_{j \ge 1} \frac{(k-1)}{2^j(j+k-1)}.\end{aligned} \tag{6}$$

Therefore, it is enough to prove that $-(1-3r/2)\sum_{j\ge1} \frac{(k-1)}{2^j(j+k-1)} \le r - 1/2$. Since $r \le 2/3$, the term $(1-3r/2)$ is positive. Note that for $k \ge 2$, we have $\sum_{j\ge1} \frac{(k-1)/(j+k-1)}{2^j} \ge \sum_{j\ge1} \frac{1}{2^j(j+1)} = 2\ln(2) - 1$. Thus, it would be enough to prove that $-(1-3r/2)(2\ln 2 - 1) \le r - 1/2$. Rearranging the terms in the last expression and multiplying by -1, we obtain the equivalent condition $3 - 4\ln 2 \le r(5 - 6\ln 2)$. ∎

In what follows, we apply Lemma 11 to derive a bound on the distribution of the accepted element that does not depend on k. Then, we select a specific value for r that balances these bounds, and allows us to approximate the distribution of $\max \mathcal{D}$. We define now the function

$$a(r) := \sum_{j \ge 2} \frac{1}{2^j}\left((1-r) - \frac{(1-r)}{j} + \frac{r}{j-1}\right) = 1 - \ln 2 + \frac{r(3\ln 2 - 2)}{2}. \tag{7}$$

This function appears in the next lemma we prove as the approximation factor of the distribution of $\max \mathcal{D}$.

Lemma 12 *For $r^* = \frac{3-4\ln 2}{5-6\ln 2}$ we have that for all $1 \le i \le k-1$,*

$$\mathbb{P}(ALG_{r^*}^* = Y_i) \ge \frac{1}{2^i} 2a(r^*). \tag{8}$$

Proof. Applying Lemma 11 on Equation (5) we can write the following for $r^* \leq r \leq 2/3$.

$$\mathbb{P}(ALG_r^* = Y_i) \geq \sum_{j \geq i+1} \frac{1}{2^j} \left((1-r) - \frac{(1-r)}{j} + \frac{r}{j-1} \right). \tag{9}$$

Note that this immediately gives the desired bound for $i = 1$ and any r in the interval $[r^*, 2/3]$. We will prove directly that it also holds for $i = 2$ and will proceed by induction for $i \geq 3$. For $i = 2$, Equation (9) combined with the definition of $a(r)$ gives that

$$\begin{aligned} \mathbb{P}(ALG_r^* = Y_2) &\geq a(r) - \frac{1}{4} \left((1-r) - \frac{(1-r)}{2} + r \right) \\ &= a(r) - \frac{1+r}{8}. \end{aligned}$$

Therefore, it is enough to prove that $a(r^*) - \frac{1+r^*}{8} \geq a(r^*)/2$, which is equivalent to $4a(r^*) \geq 1 + r^*$. If we replace $a(r^*)$ with the explicit formula in the right hand of Equation (7) and rearrange terms, we obtain the inequality $3 - 4\ln(2) \geq r^*(5 - 6\ln(2))$. Note that this is satisfied with equality by r^*.

For $i \geq 3$ we simply prove that if $r \leq 1/3$, then

$$\sum_{j \geq i+1} \frac{1}{2^j} \left((1-r) - \frac{(1-r)}{j} + \frac{r}{j-1} \right) \geq \frac{1}{2} \sum_{j \geq i} \frac{1}{2^j} \left((1-r) - \frac{(1-r)}{j} + \frac{r}{j-1} \right). \tag{10}$$

In fact, note that we can change the index in the right-hand side of Equation (10) to get the same range as in the sum of the left-hand side. So when we write the inequality for each term of the two summations, we obtain

$$\begin{aligned} \frac{1}{2^j} \left((1-r) - \frac{1-r}{j} + \frac{r}{j-1} \right) \geq \frac{1}{2^j} \left((1-r) - \frac{1-r}{j-1} + \frac{r}{j-2} \right) \quad &\Leftrightarrow \\ \frac{1}{j-1} - \frac{1}{j} \geq r \left(\frac{1}{j-2} - \frac{1}{j} \right) \quad &\Leftrightarrow \\ \frac{j-2}{2(j-1)} \geq r. \end{aligned}$$

which holds whenever $j \geq 4$ and $r \leq 1/3$. Since $r^* = \frac{3-4\ln 2}{5-6\ln 2} \approx 0.270$, we conclude that Equation (10) holds for r^*, and therefore, Equation (8) holds for all $1 \leq i \leq k-1$. ■

Lemma 13 *For any $k \geq 2$, we have that $\mathbb{P}(ALG_{r^*}^* \geq Y_k) \geq 2a(r^*)$.*

Proof. Using lemmas 8, 9 and 10 we get the following inequality.

$$\mathbb{P}(ALG_{r^*}^* \geq Y_k) = (1-r^*) \left(\frac{1}{2^k(k-1)} + \sum_{j=1}^{k-1} \frac{1}{2^j j} \right) + \frac{r^*}{2}. \tag{11}$$

Define now the function $G(k) = \frac{1}{2^k(k-1)} + \sum_{j=1}^{k-1} \frac{1}{2^j j}$. We minimize G to obtain a general lower bound for Equation (11). We have that

$$\begin{aligned} G(k+1) - G(k) &= \frac{1}{2^k k} + \frac{1}{2^{k+1}k} - \frac{1}{2^k(k-1)} \\ &= \frac{2(k-1) + (k-1) - 2k}{2^{k+1}k(k+1)} \\ &= \frac{k-3}{2^{k+1}k(k+1)}\,. \end{aligned}$$

Hence, $G(2) \geq G(3)$, and $G(k) \leq G(k+1)$ for $k \geq 3$, so G is minimized when $k = 3$. Thus,

$$\begin{aligned} \mathbb{P}(ALG^*_{r^*} \geq Y_k) &\geq (1 - r^*)\left(\frac{1}{16} + \frac{1}{2} + \frac{1}{8}\right) + \frac{r^*}{2} \\ &= \frac{11 - 3r^*}{16} \\ &= \frac{55 - 66\ln 2 - 9 + 12\ln 2}{16(5 - 6\ln 2)} \\ &= \frac{16(4 - 5\ln 2)}{16(5 - 6\ln 2)} + \frac{26\ln 2 - 18}{16(5 - 6\ln 2)} \\ &\geq 2a(r^*). \end{aligned}$$

The last inequality comes from the fact that $\frac{26\ln 2 - 18}{16(5 - 6\ln 2)} \approx 0.00162 \geq 0$ and that

$$\begin{aligned} a(r^*) &= 1 - \ln 2 + \frac{r(3\ln 2 - 2)}{2} \\ &= \frac{(2 - 2\ln 2)(5 - 6\ln 2) + (3 - 4\ln 2)(3\ln 2 - 2)}{2(5 - 6\ln 2)} \\ &= \frac{4 - 5\ln 2}{2(5 - 6\ln 2)}. \end{aligned}$$

■

With the last two lemmas we are ready to prove the main theorem of this section.

Proof of Theorem 2. Summing the lower bound in Lemma 12 it follows that for all $1 \leq i \leq k-1$,

$$\begin{aligned} \mathbb{P}(ALG^*_{r^*} \geq Y_i) &\geq 2a(r^*) \cdot \left(1 - \frac{1}{2^i}\right) \\ &= 2a(r^*)\mathbb{P}(\max \mathcal{D} \geq Y_i)\,. \end{aligned}$$

From Lemma 13 we get that also $\mathbb{P}(ALG^*_{r^*} \geq Y_k) \geq 2a(r^*) \cdot \mathbb{P}(\max \mathcal{D} \geq Y_k)$. Therefore, as $\max \mathcal{D} \in \{Y_1, ..., Y_k\}$ with probability 1, we conclude that $\mathbb{E}(ALG^*_{r^*}) \geq 2a(r^*) \cdot \mathbb{E}(\max \mathcal{D})$.

■

5. Large data sets (large k)

In this section we consider the case in which k is large and show that for that case one can obtain better guarantees for the prophet two-sided game of Googol. This case appears very often, for instance, in the i.i.d. prophet secretary problem with unknown (continuous) distributions, all permutations of $[2n]$ are equally likely to be the ordering of the $2n$ values in the cards. The probability that k is at least k_0 equals the probability that the top t elements of the list are written in different cards. Note that for any $\ell = o(\sqrt{n})$,

$$\begin{aligned}\mathbb{P}(k \geq \ell) &= \left(1 - \frac{1}{2n-1}\right) \cdot \left(1 - \frac{2}{2n-2}\right) \cdots \left(1 - \frac{\ell-1}{2n-(\ell-1)}\right) \geq \left(1 - \frac{\ell}{2n-\ell}\right)^{\ell} \\ &\geq 1 - \frac{\ell^2}{2n-\ell} \geq 1 - o(1).\end{aligned}$$

Our algorithm uses a moving window of length strictly larger than 1, as outlined in section 2.

Moving window algorithm of length $1+t$ (ALG(t)). We first draw n uniform variables in $[0,1]$ and we sort them from smallest to largest[8] as $0 < \tau_1 < \tau_2 < \cdots < \tau_n < 1$. We interpret τ_j as the *arriving time* of the s-th hidden value that we reveal, and therefore, $\tau_s - 1 = C(\tau_j)$ is the arriving time of the corresponding s-th face up value. The algorithm will accept the s-th face up value $x = D_{\sigma(s)}$ if and only if x is larger than any value arrived in the previous $1+t$ time units, i.e. if and only if x is larger than all elements arriving in $[\max(\tau_s - 1 - t, -1), \tau_s]$

Observe that ALG(0) and ALG(1) are exactly the algorithms $\mathrm{ALG_c}$ and $\mathrm{ALG_f}$ defined in previous sections. Below we analyze the performance of ALG(t).

Lemma 14 *For every $1 \leq i \leq k-1$,*

$$\mathbb{P}(\mathrm{ALG}(t) = Y_i) = b(k,t) + \sum_{j=i+1}^{k-1} \frac{a(j,t)}{2^j}, \tag{12}$$

$$\text{where} \quad a(j,t) = \frac{1-(1-t)^{j-1}(1+t)}{j-1} + (1-t)^{j-1}(1+t) - \frac{(1-t)^j}{j},$$

$$b(k,t) = \frac{1}{2^{k-1}}\left(\frac{1-(1-t)^{k-1}}{k-1} + (1-t)^{k-1}\right),$$

and

$$\mathbb{P}(\mathrm{ALG}(t) \geq Y_k) = \frac{1}{2} + \sum_{i=2}^{k-1} \frac{(1-t)^j}{2^j j}.$$

Proof. Fix an index $i \leq k-1$. Recall that $\theta_i \in [-1,1)$ denotes the arriving time of Y_i. Let J be the random variable denoting the index of the largest element arriving in the window of Y_i, or equivalently, the smallest index such that $\theta_j \in [\theta_i - 1 - t, \theta_i)$. Since $[\max(\theta_i - 1 - t, -1), \theta_i)$ has length at least 1, it always contains θ_k or $C(\theta_k) = \theta_{k'}$ (or both), we conclude that $J \leq k$.

8. We assume that the drawn values are all distinct as this happens with probability 1

Note that the algorithm selects Y_i if and only if the next three events occur: (I) $\theta_i > 0$, (II) $J > i$, as otherwise, $Y_J > Y_i$ would be inside the window of Y_i, and (III) No element is selected before Y_i's arrival. By Lemma 4, the third event occurs if and only if the maximum element[9] X arriving in $[0, \theta_i)$ is smaller than some element in its own window. Note that this happens in two cases: either $\theta_J < 0$ (and therefore Y_J is in the window of X), or $\theta_J \geq 0$ and Y_J is smaller than some element arriving in its own window.

In what follows fix some j with $i + 1 \leq j \leq k - 1$. We will compute the probability of both selecting Y_i and $J = j$.

Case 1: $0 \leq \theta_i \leq t$**.** In order to impose that $J = j$ we need that all $j - 2$ arrival times $(\theta_l : l \in [j-1] \setminus \{i\})$ are outside the interval $[-1, \theta_i]$ and θ_j is inside. Furthermore, in this case every element arriving in $[0, \theta_j)$ is strictly smaller than Y_j. Therefore in order to impose that no element is selected before Y_i's arrival we need that $\theta_j < 0$. It follows that

$$\begin{aligned}\mathbb{P}(\mathrm{ALG}(t) = Y_i, J = j, \theta_i \in [0,t]) &= \frac{1}{2}\int_0^t \mathbb{P}(\theta_j < 0) \prod_{l \in [j-1]\setminus\{i\}} \mathbb{P}(\theta_l > 1 - x)dx \\ &= \frac{1}{2}\int_0^t \frac{1}{2}\left(\frac{1-x}{2}\right)^{j-2} dx = \frac{1}{2^j} \cdot \frac{1-(1-t)^{j-1}}{j-1}.\end{aligned}$$

Case 2: $t < \theta_i \leq 1$ **and** $\theta_j < 0$**.** We now need that all $j-2$ arrival times $\theta_l, l \in [j-1] \setminus \{i\}$ fall outside $[\theta_i - t - 1, \theta_i]$ and θ_j is inside. Since we are imposing $\theta_j < 0$, we actually require that $\theta_j \in [\theta_i - t - 1, 0]$. Since $\theta_J < 0$ this is enough to guarantee that Y_i will be selected. Therefore,

$$\begin{aligned}&\mathbb{P}(\mathrm{ALG}(t) = Y_i, J = j, \theta_i \in (t,1] \wedge \theta_j < 0) \\ &= \frac{1}{2}\int_t^1 \mathbb{P}(\theta_j \in [x - t - 1, 0]) \prod_{l \in [j-1]\setminus\{l\}} \mathbb{P}(\theta_l \notin [x - t - 1, x])dx \\ &= \frac{1}{2}\int_t^1 \frac{1 + t - x}{2}\left(\frac{1-t}{2}\right)^{j-2} dx = \frac{(1-t)^{j-1}(1+t)}{2^{j+1}}.\end{aligned}$$

Case 3: $t < \theta_i \leq 1$ **and** $0 \leq \theta_j < t$**.** As before we need that all $j - 2$ arrival times $\theta_l, l \in [j-1] \setminus \{i\}$ fall outside $[\theta_i - t - 1, \theta_i]$ and $\theta_j \in [0, t)$. Since $\theta_j \geq 0$ we also need that at least one of $Y_l, l \in [j-1] \setminus \{j\}$ arrives in Y_j's window $[-1, \theta_j)$ and since they must be outside Y_i's window this reduces to the event that not all $j - 2$ arrival times fall in the interval $(\theta_i, 1]$. Therefore,

$$\begin{aligned}&\mathbb{P}(\mathrm{ALG}(t) = Y_i, J = j, \theta_i \in (t,1] \wedge 0 \leq \theta_j < t) \\ &= \frac{1}{2}\int_t^1 \mathbb{P}(\theta_j \in [0,t))\mathbb{P}(\forall l \in [j-1] \setminus \{i\}, \theta_l \notin [x - t - 1, x] \text{ and not all in } (x,1])dx \\ &= \frac{1}{2}\int_t^1 \frac{t}{2}\left(\left(\frac{1-t}{2}\right)^{j-2} - \left(\frac{1-x}{2}\right)^{j-2}\right) dx = \frac{t(1-t)^{j-1}}{2^j} - \frac{t(1-t)^{j-1}}{2^j(j-1)}.\end{aligned}$$

9. If no element arrives in $[0, \theta_i)$ then event (III) also occurs

Case 4: $t < \theta_i \leq 1$ **and** $t \leq \theta_j < \theta_i$**.** Once again we need that all $j-2$ arrival times $\theta_l, l \in [j-1] \setminus \{i\}$ fall outside $[\theta_i - t - 1, \theta_i]$ and $\theta_j \in [t, \theta_i)$. Since $\theta_j \geq 0$ we also need that at least one of $Y_l, l \in [j-1] \setminus \{j\}$ arrives in Y_j's window $[\theta_j - 1 - t, \theta_j)$ and since they must be outside Y_i's window this reduces to the event that not all $j-2$ arrival times fall in the set $[-1, \theta_j - t - 1) \cup (\theta_i, 1]$. Therefore,

$$\begin{aligned}
&\mathbb{P}(\mathrm{ALG}(t) = Y_i \wedge J = j \wedge \theta_i \in (t,1] \wedge t \leq \theta_j < x) \\
&= \frac{1}{2}\int_t^1 \mathbb{P}(\theta_j \in [t,x))\mathbb{P}(\forall l \in [j-1]\setminus\{i\}, \theta_l \notin [x-t-1, x] \\
&\qquad\qquad \text{and not all in } [-1, \theta_j - t - 1) \cup (x, 1])dx \\
&= \frac{1}{2}\int_t^1 \int_t^x \left(\left(\frac{1-t}{2}\right)^{j-2} - \left(\frac{1-x+y-t}{2}\right)^{j-2}\right) dydx \\
&= \frac{1}{2^j}\int_t^1 (1-t)^{j-2}(x-t) - \frac{(1-t)^{j-1} - (1-x)^{j-1}}{j-1} dx = \frac{(1-t)^j}{2^{j+1}} - \frac{(1-t)^j}{2^j j}.
\end{aligned}$$

Putting all cases together we get that for all $1 \leq i \leq k-1$ and $i+1 \leq j \leq k-1$,

$$\mathbb{P}(\mathrm{ALG}(t) = Y_i \wedge J = j) = \frac{1}{2^j} \cdot \left(\frac{1-(1-t)^{j-1}(1+t)}{j-1} + (1-t)^{j-1}(1+t) - \frac{(1-t)^j}{j}\right). \tag{13}$$

Now let us consider the case $J = k$. In this case we only need to impose that all $k-2$ arrival times $\theta_l, l \in [k-1] \setminus \{i\}$ fall outside the window of Y_i. No matter who the couple of k is, the previous condition implies that k is inside the window of Y_i. Therefore,

$$\begin{aligned}
\mathbb{P}(\mathrm{ALG}(t) = Y_i \wedge J = k) &= \frac{1}{2}\int_0^t \left(\frac{1-x}{2}\right)^{k-2} dx + \left(\frac{1-t}{2}\right)^{k-1} \\
&= \frac{1}{2^{k-1}}\left(\frac{1-(1-t)^{k-1}}{k-1} + (1-t)^{k-1}\right).
\end{aligned} \tag{14}$$

By combining (13) and (14) together we finish the proof of the first equality of this lemma, (12).

We now compute the probability that $\mathrm{ALG}(t)$ selects one of the top k values $Y_1, \ldots, Y_k$. Observe first that the algorithm never selects a value Y_j with $j > k$. This is because the window of Y_j always contains θ_k or $C(\theta_k) = \theta_{k'}$, and both Y_k and $Y_{k'}$ are larger than Y_j. By this observation, $\mathbb{P}(\mathrm{ALG}(t) \geq Y_k)$ equals the probability that the algorithm stops by selecting something.

Let M be the random variable denoting the index of the largest value arriving after 0, i.e. $\theta_M \geq 0$ and $\theta_i < 0$, for all $i < M$. By Lemma 4, $\mathrm{ALG}(t)$ stops by selecting a value if and only if Y_M is larger than every value arriving in its window $[\max(\theta_M - 1 - t, 1), \theta_M))$. Observe that if $M = k$, this is impossible because the couple of k is always in that interval. Therefore,

$$\begin{aligned}\mathbb{P}(\mathrm{ALG}(t) \geq Y_k) &= \sum_{j=1}^{k} \mathbb{P}(\mathrm{ALG}(t) \geq Y_k, M = j) \\ &= \sum_{j=1}^{k} \frac{1}{2} \int_0^1 \mathbb{P}(\forall i \in [j-1], \theta_i \in [-1, x-1-t) dx. \\ &= \frac{1}{2} + \sum_{j=2}^{k-1} \frac{1}{2} \int_t^1 \left(\frac{x-t}{2}\right)^{j-1} dx = \frac{1}{2} + \sum_{j=2}^{k-1} \frac{(1-t)^j}{2^j j}.\end{aligned}$$

■

Algorithm $\mathrm{ALG}(t)$ has a very poor performance for $k = 2$. For example, in the instance $\{(1, 1-\varepsilon), (\varepsilon, 0)\}$, we have $Y_1 = 1$, $Y_2 = 1-\varepsilon$ and $k = 2$. By Lemma 14, $\mathbb{P}(\mathrm{ALG}(t) \geq Y_2) = 1/2$. Therefore $\mathbb{E}(\mathrm{ALG}(t)) \leq 1/2$, while $\mathbb{E}(\max \mathcal{D}) \geq 1-\varepsilon$.

We will now study the behaviour of $\mathrm{ALG}(t)$ when k is large. For that, define

$$R_k(i,t) = \frac{\mathbb{P}(\mathrm{ALG}(t) \geq Y_i)}{\mathbb{P}(\max \mathcal{D} \geq Y_i)} = \frac{1}{1-(1/2)^i} \sum_{l=1}^{i} \sum_{j=l+1}^{k-1} b(k,t) + \frac{a(j,t)}{2^j}.$$

$$R_k(t) = \mathbb{P}(\mathrm{ALG}(t) \text{ stops}) = \frac{1}{2} + \sum_{j=2}^{k-1} \frac{(1-t)^j}{2^j j}.$$

Consider also their limits as $k \to \infty$, which since $kb(k,t) \to 0$ can be computed as

$$R_\infty(i,t) = \frac{1}{1-(1/2)^i} \sum_{l=1}^{i} \sum_{j=l+1}^{\infty} \frac{a(j,t)}{2^j}.$$

$$R_\infty(t) = \frac{1}{2} + \sum_{j=2}^{\infty} \frac{(1-t)^j}{2^j j} = \frac{t}{2} + \ln\left(\frac{2}{1+t}\right).$$

Recall that the competitiveness of $\mathrm{ALG}(t)$ is at least $\min_{i \geq 1} R_\infty(i,t)$, and we are looking for the value t that maximizes this minimum. In Figure 1 we plot a few of these $R_\infty(i,t)$, for small values of i, together with function $R_\infty(t)$. From the plot we observe that the t^* maximizing the minima of all the curves satisfies $R_\infty(1,t^*) = R_\infty(t^*)$. By simplifying the sums, we obtain that $t^* \approx 0.1128904$ is the only root of:

$$R_\infty(1,t) - R_\infty(t) = 2 - (5t)/2 - (3+t)\ln 2 + (4+t)\ln(1+t).$$

To formally prove that $R_\infty(t^*) \approx 0.6426317$ is the sought guarantee we need the following lemma.

Lemma 15 *Let $a(j,t)$ be defined as in Lemma 14. For all $j \geq 3$, we have $a(j,t^*) \geq a(j+1,t^*)$*

Proof. Below, we tabulate a few values of $a(j,t^*)$ with $t^* \approx 0.1128904$, and we observe that $a(j,t^*) \geq a(j+1,t^*)$ for all $j \in \{3,\ldots,9\}$:

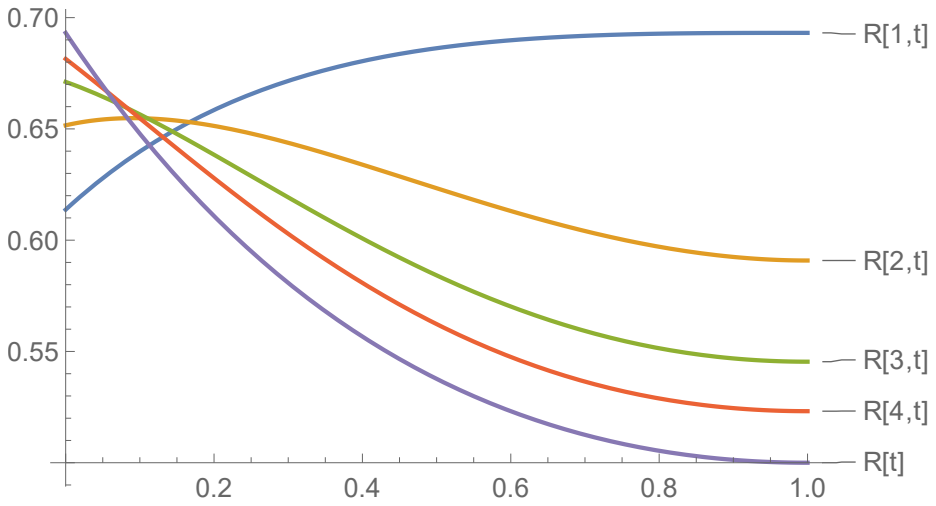

Figure 1: Auxiliary functions to analyze $\mathrm{ALG}(t)$, as k tends to ∞.

j	3	4	5	6	7	8	9	10
$a(j,t^*)$	0.705194	0.696462	0.65704	0.607906	0.556898	0.50734	0.460684	0.417513

Let us prove that the inequality also holds for $j \geq 10$. By rearranging terms, $a(j,t) \geq a(j+1,t)$ is equivalent to

$$\frac{1}{(1-t)^{j-1}}\left(\frac{1}{j-1}-\frac{1}{j}\right) \geq -t(1+t)-\frac{t(1-t)}{j}-\frac{(1-t)^2}{j+1}+\frac{1+t}{j-1}.$$

But note that the right hand side, evaluated at $t^* \geq 1/9$ is:

$$\begin{aligned}-t^*(1+t^*)-\frac{t^*(1-t^*)}{j}-\frac{(1-t^*)^2}{j+1}+\frac{1+t^*}{j-1} &\leq (1+t^*)\left(\frac{1}{j-1}-t^*\right)\\ &\leq (1+t^*)\left(\frac{1}{j-1}-\frac{1}{9}\right),\end{aligned}$$

which is nonpositive for all $j \geq 10$. Then, it is smaller than the left hand side which is positive. ■

Theorem 3 *For $t^* \approx 0.1128904$, and any instance of the two-sided game of Googol, as k goes to infinity, $\mathbb{E}[\mathrm{ALG}(t^*)]/\mathbb{E}[\max \mathcal{D}]$ is at least $R_\infty(t^*) \approx 0.6426317$.*

Proof. We will prove that for all $i \geq 1$, $R_\infty(i,t^*) \geq R_\infty(t^*)$. Denote for all $i \geq 1$, $P_i = \lim_{k\to\infty} \mathbb{P}(\mathrm{ALG}(t^*) = Y_i) = \sum_{j=i+1}^{\infty} a(j,t^*)/2^j$. By choice of t^*, $R_\infty(1,t^*) = R_\infty(t^*) \approx 0.6426317$. Furthermore we we can numerically evaluate $R_\infty(2,t^*) = \frac{P_1+P_2}{1-1/4} \approx 0.6547 \geq R_\infty(t^*)$ and $R_\infty(3,t^*) = \frac{P_1+P_2+P_3}{1-1/8} \approx 0.654331 \geq R_\infty(t^*)$.

To finish the proof, we will show that for all $i \geq 4$, $R_\infty(i,t^*) \geq R_\infty(i+1,t^*)$, and therefore, for all $i \geq 4$, $R_\infty(i,t^*) \geq \lim_{j\to\infty} R_\infty(j,t^*) = R_\infty(t^*)$. For this, we will also need

the inequality $P_1 \approx 0.3213158 \geq 0.304065 \approx 8P_4$. By Lemma 15, for all $i \geq 2$,

$$P_i = \sum_{j=i+1}^{\infty} \frac{a(j,t^*)}{2^j} \geq \sum_{j=i+1}^{\infty} \frac{a(j+1,t^*)}{2^j} = 2P_{i+1}.$$

Therefore,

$$\sum_{\ell=1}^{i} P_\ell \geq 8P_4 + \sum_{\ell=2}^{i} P_\ell \geq 2^i P_{i+1} + P_{i+1} \sum_{\ell=2}^{i} 2^{i+1-\ell} = P_{i+1}(2^{i+1}-2).$$

We conclude that,

$$R(i+1,t^*) = \frac{P_{i+1} + \sum_{\ell=1}^{i} P_\ell}{1 - 1/2^{i+1}} \leq \frac{\left(1 + \frac{1}{2^{i+1}-2}\right) \sum_{\ell=2}^{i} P_l}{\frac{2^{i+1}-1}{2^{i+1}}} = \frac{\sum_{\ell=2}^{i} P_l}{\frac{2^{i+1}-2}{2^{i+1}}} = R_\infty(i,t^*).$$

■

To conclude this section we observe that the guarantee obtained in the $k = \infty$ case is still useful for moderately high values of k. Indeed, note that for all $j \geq 1$, and all $t \in (0,1)$,

$$|a(j,t)| \leq 2, |b(k,t)| \leq \frac{2}{2^{k-1}}.$$

Therefore, for all $1 \leq i \leq k-1$ and all $k \geq 3$.

$$\left| \mathbb{P}(\mathrm{ALG}(t) = Y_i) - \sum_{j=i+1}^{\infty} \frac{a(j,t)}{2^j} \right| = \left| b(k,t) - \sum_{j=k}^{\infty} \frac{a(j,t)}{2^j} \right| \leq \frac{2}{2^{k-1}} + 2\sum_{j=k}^{\infty} \frac{1}{2^j} = \frac{4}{2^{k-1}} = \frac{1}{2^{k-3}}.$$

Therefore,

$$|R_k(i,t) - R_\infty(i,t)| = \frac{1}{1-1/2^k} \left| \sum_{\ell=1}^{i} \left(\mathbb{P}(\mathrm{ALG}(t) = Y_\ell) - \sum_{j=\ell+1}^{\infty} \frac{a(j,t)}{2^j} \right) \right| \leq \frac{2i}{2^{k-3}} \leq \frac{16k}{2^k}.$$

We also have that

$$|R_k(t) - R_\infty(t)| = \left| \sum_{j=k}^{\infty} \frac{(1-t)^j}{2^j j} \right| \leq \frac{1}{2^{k-1}} \leq \frac{16k}{2^k}.$$

From here,

$$\min(R_k(t^*), \min_{1 \leq i \leq k-1} R_k(i,t^*)) \geq \min(R_\infty(t^*), \min_{1 \leq i \leq k-1} R_k(i,t^*)) - \frac{16k}{2^k} = R_\infty(t^*) - \frac{16k}{2^k}.$$

Thus the guarantee of $\mathrm{ALG}(t)$ for not so large values of k is already very close to $R_\infty(t^*) \approx 0.6426317$. For example, the guarantee of $\mathrm{ALG}(t)$ for $k \geq 20$ is at least $R_\infty(t^*) - 0.00031$.

6. Concluding remarks

The two-sided game of Googol has close connections with optimal stopping theory and our results have direct implications for the Prophet Secretary problem with samples and the Secretary problem with samples. This connection immediately suggests other variants of the problem. We state here two that we consider of interest, together with some initial thoughts on the challenges they present if one tries to apply the techniques of this paper.

The d-sided game of Googol. In this variant our "cards" have $d > 2$ sides. More precisely, we are given n dice (instead of cards) with d sides each. The dice have arbitrary non-negative numbers on all sides, so we have $d \cdot n$ different numbers in total. The dice are shuffled and tossed on a table so that, for each die independently, any of its d faces lands face-down with probability $1/d$. For all dice, we observe all $d-1$ faces that did not land face-down. In uniform random order we get to flip one die at a time, revealing the hidden number. As in the two-sided game of Googol, we must decide when to stop in order to maximize either the Secretary objective or the Prophet objective. Naturally, the case where the adversary chooses a distribution for each die and writes i.i.d. samples on each face models the Prophet Secretary problem and the Secretary problem with $d-1$ samples for each distribution, instead of only one.

Our analysis of the two-sided game of Googol relies on the parameter k, the smallest index of a number (in the decreasing ordering of the $2n$ numbers) such that it is paired with a larger number. The importance of k comes from the fact that $\max \mathcal{D}$ is always one of the largest k numbers out of the $2 \cdot n$ numbers. This parameter is determined by the decisions of the adversary, so we consider all possible cases. To do a similar analysis in the d-sided version, we would have to define several parameters to account for all possible groupings of the sequence of $d \cdot n$ numbers.

Free-order two-sided game of Googol. A natural variant of our two-sided game of Googol is to allow the decision-maker to choose in which order the cards will be flipped. Certainly, our algorithm can also be applied to this setting, as flipping in uniform random order is one possible option. However, it is not obvious whether we can take advantage of the free order. On one hand, the adversary could pair the $2n$ numbers randomly, deeming useless the extra power of the decision-maker. On the other hand, when the pairing is random, the problem appears to be much easier. In fact, for the resulting problem a 0.671-approximation is possible, as shown by Correa et al. (2020).

Acknowledgments

We gratefully acknowledge support from the Center for Mathematical Modeling, CMM (ANID contracts ACE210010 and FB210005), ANID grant ACT210005 and ANID FONDECYT Regular 1181180.

Appendix A. Factor revealing LP

In this section we develop a linear program whose solution describes the optimal ordinal[10] algorithm for instances of size n of the two-sided game of Googol. This linear program is based on the techniques used in Buchbinder et al. (2014). Before we describe the linear program we must introduce some additional notation. Afterwards we develop the linear program formulation and close by presenting numerical results for $n = 2$ and $n = 3$.

A.1 Additional notation

Recall that we denote by $Y_1 > \cdots > Y_{2n}$ the $2n$ numbers of a particular instance. We define a *setting* s as a particular coupling of the values $Y_1 > \cdots > Y_{2n}$ of the instance in the n cards. For example, if we have $n = 2$ cards then we have three possible settings: $\{(Y_1, Y_2), (Y_3, Y_4)\}$, $\{(Y_1, Y_3), (Y_2, Y_4)\}$, and $\{(Y_1, Y_4), (Y_2, Y_3)\}$. We denote the set of all possible settings by $\mathcal{S}$. The different settings can be seen as different matching between the values $Y_1, \ldots, Y_{2n}$. For any setting, let k_s be the smallest index so that k is the couple of some $k' < k$. In particular, $Y_1, Y_2, \ldots, Y_{k_s-1}$ are all written on different cards.

We now turn our focus on the information available whenever we have to decide whether to stop the game or not. At step i we know exactly $n + i$ elements out of the $2n$ of the instance. At this step we can characterize all the information available (for an ordinal algorithm) by what we define as a *history* h, which consists of two tuples $(d_1, \ldots, d_i)$ and $(u_1, \ldots, u_i)$. Here d_j represents the ranking of element $D_{\sigma(j)}$ among the $n + i$ elements seen so far. Similarly, u_j represents the ranking of element $U_{\sigma(j)}$. We denote by H_i to the set of possible histories we can face at step i and define $\mathcal{H} = \cup_{i=1}^{n} H_i$ to be the set of all possible histories we can encounter at some step of the game. Notice that at any step, the history contains all the information an ordinal algorithm can utilize. Indeed, the ranking of the remaining $n - i$ numbers facing upwards cannot be mapped to the remaining numbers $D_{\sigma(i+1)}, \ldots, D_{\sigma(n)}$ facing downwards, as we do not know the order in which we will explore them. Also, the history at a given step contains the history of previous steps, as we can compute the rankings without considering the most recently observed elements. For a given history h that occurs at step i denote by C_h the set of histories seen from step 1 to $i - 1$ before seeing h at step i.

A.2 Linear program formulation

Our LP variables have the form x_h for every $h \in \mathcal{H}$. They are to be interpreted as the probability of stopping when faced to a history h, given that the randomness of the game (coin tosses and random permutation) is such that we will face h in the corresponding step of the game. We denote the event that the randomness of the game is such that we will face h in the corresponding step of the game by B_h. We clarify that B_h does not imply reaching the step in which we observe history h, only that the randomness is such that if we reach the corresponding step, we would observe h. The linear program is as follows:

10. An algorithm which decides to stop or not based on the relative ranking of the observed elements, not on the numerical values of the elements.

$$
\begin{aligned}
\mathrm{LP}_n = \max \quad & \alpha \\
\text{s.t.} \quad & x_h \le 1 - \sum_{j \in C_h} x_j && \forall h \in \mathcal{H}, && (15)\\
& \alpha \le \frac{\sum_{j=1}^{k} \sum_{i=1}^{n} \sum_{j \in H_i} x_h \lambda_{hjs} \frac{1}{2n}}{1 - 2^{-k}} && \forall s \in \mathcal{S}, \forall k \in [k_s - 1] && (16)\\
& \alpha \le \sum_{j=1}^{k_s} \sum_{i=1}^{n} \sum_{h \in H_i} x_h \lambda_{hjs} \frac{1}{2n} && \forall s \in \mathcal{S} && (17)\\
& x \ge 0.
\end{aligned}
$$

Constraints (15) are called the *feasibility constraints* which manage to capture all ordinal algorithms. Constraints (16) and (17) are called *stochastic dominance constraints*, which relate the objective function with the worst case performance of the algorithm.

Feasibility constraints. This set of constraints forms a polytope that captures how any ordinal online algorithm behaves. We first present Lemma 16, which establishes that any ordinal algorithm has a representation which satisfies the feasibility constraints.

Lemma 16 *Consider any ordinal algorithm ALG. Define x_h^{ALG} as the probability that the algorithm will stop facing history h given B_h, then x^{ALG} satisfies the feasibility constraints.*

Proof. We have

$$
\begin{aligned}
x_h^{\mathrm{ALG}} &= \mathbb{P}(\text{Stop facing history } h | B_h) \\
&\le \mathbb{P}(\text{Face history } h | B_h) \\
&= 1 - \mathbb{P}(\text{Stop before facing history } h | B_h) \\
&= 1 - \sum_{j \in \mathcal{C}_h} \mathbb{P}(\text{Stop facing history } j | B_h) \\
&= 1 - \sum_{j \in \mathcal{C}_h} \mathbb{P}(\text{Stop facing history } j | B_j) && (18) \\
&= 1 - \sum_{j \in \mathcal{C}_h} x_j^{\mathrm{ALG}}.
\end{aligned}
$$

Equality (18) holds because the algorithm does not see the future, so the decision to stop facing j or before does not depend on what happens after j. That being said, j can lead to histories different than h, but whatever history is sampled they will be identical up to j, so the behavior of the algorithm up to seeing history j does not change if we condition of B_h or B_j. ■

This establishes that any ordinal algorithm satisfies the feasibility constraint. In Lemma 17 we establish the other direction, that is, any solution to the linear program can be used to construct a corresponding ordinal algorithm.

Lemma 17 *Let (x, α) be a feasible solution to the linear program. Define the algorithm* ALG *as follows: when facing history h, stop with probability*

$$\frac{x_h}{1 - \sum_{j \in \mathcal{C}_h} x_j}.$$

Then this algorithm satisfies $\mathbb{P}(\text{Stop facing history } h | B_h) = x_h$.

Proof. Denote by R_h the event of seeing history h at some point (i.e. not stopping before getting to see history h) and by S_h the event of stopping when seeing history h. We will show by induction that $\mathbb{P}(S_h|B_h) = x_h$ for every $h \in \mathcal{H}$. The base case is for $h \in H_1$ (histories seen in the first step). For these h we will get to face h with probability 1, and we also have that $\mathcal{C}_h = \emptyset$. With this in mind we compute

$$\begin{aligned}\mathbb{P}(S_h|B_h) &= \frac{\mathbb{P}(S_h \wedge B_h)}{\mathbb{P}(B_h)} \cdot \frac{\mathbb{P}(S_h \wedge B_h \wedge R_h)}{\mathbb{P}(S_h \wedge B_h \wedge R_h)} \cdot \frac{\mathbb{P}(R_h \wedge B_h)}{\mathbb{P}(R_h \wedge B_h)} \\ &= \mathbb{P}(S_h|B_h \wedge R_h)\mathbb{P}(R_h|B_h)\frac{1}{\mathbb{P}(R_h|B_h \wedge S_h)}.\end{aligned}$$

Clearly $\mathbb{P}(R_h|B_h \wedge S_h) = 1$, as stopping at h implies we reached it, and as we discussed $\mathbb{P}(R_h|B_h) = 1$ because $h \in H_1$ will always be reached. Finally,

$$\mathbb{P}(S_h|B_h \wedge R_h) = \frac{x_h}{1 - \sum_{j \in \mathcal{C}_h} x_j} = x_h,$$

as $\mathcal{C}_h = \emptyset$. Now assume this holds for all $h \in H_j$, $j < i$. Pick $h \in H_i$ and recall

$$\mathbb{P}(S_h|B_h) = \mathbb{P}(S_h|B_h \wedge R_h)\mathbb{P}(R_h|B_h)\frac{1}{\mathbb{P}(R_h|B_h \wedge S_h)}$$

It still holds that $\mathbb{P}(R_h|B_h \wedge S_h) = 1$. By the definition of the algorithm, we have

$$\mathbb{P}(S_h|B_h \wedge R_h) = \frac{x_h}{1 - \sum_{j \in \mathcal{C}_h} x_j},$$

and we have

$$\begin{aligned}\mathbb{P}(R_h|B_h) &= 1 - \mathbb{P}(\text{Stop before } h|B_h) \\ &= 1 - \sum_{j \in \mathcal{C}_h} \mathbb{P}(S_j|B_h) \\ &= 1 - \sum_{j \in \mathcal{C}_h} \mathbb{P}(S_j|B_j) \\ &= 1 - \sum_{j \in \mathcal{C}_h} x_j,\end{aligned}$$

where the last equality holds by our inductive hypothesis. Putting all together we get

$$\mathbb{P}(S_h|B_h) = \frac{x_h}{1 - \sum_{j \in \mathcal{C}_h} x_j} \cdot \left(1 - \sum_{j \in \mathcal{C}_h} x_j\right) \cdot 1 = x_h.$$

■

This establishes the equivalence between the polyhedron defined by the feasibility constraint and the set of non-adaptive ordinal algorithms.

Stochastic dominance constraints. The set of stochastic dominance constraints connects the objective function of the linear program with the ratio between the expectation of the value selected by the algorithm and the maximum hidden value. We develop constraints of the form

$$\frac{\mathbb{P}(\max \mathcal{D} \geq Y_j)}{\mathbb{P}(\text{ALG} \geq Y_j)} \geq \alpha \quad \forall j \in \{1, \ldots, 2n\}. \tag{19}$$

With this we can multiply both sides of the above inequality by $\mathbb{P}(\text{ALG} \geq Y_j)$ and integrate from 0 to infinity to obtain the ratio in expectation. This bound is also tight (assuming that the solution α is such that one of the constraints is tight). To see this, assume that the ratio between the expectation of the value chosen by the algorithm and the expectation of the highest hidden value is β with $\beta > \alpha$. choose j^* such that the constraint is tight (i.e. $\mathbb{P}(\text{ALG} \geq Y_{j^*}) = \alpha \mathbb{P}(\max \mathcal{D} \geq Y_{j^*})$) and construct an instance with $Y_i = 1$ for $1 \leq i \leq j^*$ and $Y_i = 0$ for $i > j$. Then we have $\mathbb{E}(\text{ALG}) = \mathbb{P}(\text{ALG} \geq Y_{j^*})$ and $\mathbb{E}(\max \mathcal{D}) = \mathbb{P}(\max \mathcal{D} \geq Y_{j^*})$, contradicting $\beta > \alpha$.

We now compute $\mathbb{P}(\max \mathcal{D} \geq Y_j)$. Notice that this probability distribution depends on the particular setting s, as $\max \mathcal{D}$ can only take values in $\{Y_1, \ldots, Y_{k_s}\}$. Through exactly the same argument as the beginning of Section 4, for any setting s we have

$$\mathbb{P}(\max \mathcal{D} = Y_j) = \begin{cases} \frac{1}{2^j} & j < k_s \\ \frac{1}{2^{k_s - 1}} & j = k_s \\ 0 & i > k_s. \end{cases}$$

This implies

$$\mathbb{P}(\max \mathcal{D} \geq Y_j) = \begin{cases} 1 - \frac{1}{2^j} & j < k_s \\ 1 & j \geq k_s. \end{cases} \tag{20}$$

We also need to compute $\mathbb{P}(\text{ALG} \geq Y_j)$ (characterized by LP variables x_h) for any setting s. This is established in the following Lemma.

Lemma 18 *For any setting $s \in \mathcal{S}$ and $j \in \{1, \ldots, 2n\}$,*

$$\mathbb{P}_s(\text{ALG} = Y_j) = \sum_{i=1}^{n} \sum_{h \in H_i} x_h \lambda_{hjs} \frac{1}{2n},$$

where $\lambda_{hjs} = \mathbb{P}_s(B_h | Y_{\pi(i)} = Y_j)$ under setting s.

Proof. Use $\mathbb{P}_s(\cdot)$ to denote the probability under a setting s and again use B_h to denote the event that the randomness of the game is such that history h would appear eventually

(if the algorithm does not stop until the corresponding step). We develop

$$\begin{aligned}
\mathbb{P}_s(\text{ALG} = Y_j) &= \sum_{i=1}^{n} \mathbb{P}(\text{Stop at step } i \wedge Y_{\pi(i)} = Y_j) \\
&= \sum_{i=1}^{n} \sum_{h \in H_i} \mathbb{P}_s(\text{Stop at step } i \wedge D_i = Y_j \wedge B_h) \\
&= \sum_{i=1}^{n} \sum_{h \in H_i} \mathbb{P}_s(\text{Stop at step } i | D_i = Y_j \wedge B_h) \mathbb{P}_s(Y_{\pi(i)} = Y_j \wedge B_h) \\
&= \sum_{i=1}^{n} \sum_{h \in H_i} \mathbb{P}_s(\text{Stop at step } i | B_h) \mathbb{P}_s(D_i = Y_j \wedge B_h) \qquad (21) \\
&= \sum_{i=1}^{n} \sum_{h \in H_i} x_h \mathbb{P}_s(B_h | Y_{\pi(i)} = Y_j) \mathbb{P}_s(D_i = Y_j) \\
&= \sum_{i=1}^{n} \sum_{h \in H_i} x_h \lambda_{hjs} \frac{1}{2n},
\end{aligned}$$

where in the last equality we implicitly define $\lambda_{hjs} = \mathbb{P}_s(B_h | Y_{\pi(i)} = Y_j)$ and used that $\mathbb{P}_s(Y_{\pi(i)} = Y_j) = 1/(2n)$ for any setting s. The only subtle step here is equation (21), where we use the fact that the whole behavior of the algorithm until step i is completely determined by history h (which includes all histories seen until step i). This means that we can discard $Y_{\pi(i)} = Y_j$ from the conditional. ■

Although we do not have closed expressions for coefficients λ_{hjs}, they can be computed by enumeration for small values of n. Constraints (16) and (17) are obtained by replacing in equation (19) the expressions obtained in (20) and Lemma 18, for every $s \in \mathcal{S}$.

With all of these results we have established the following theorem:

Theorem 19 *For instances of size n, the optimal algorithm for the two-sided game of Googol has worst case ratio between the expected selected value and the expected maximum hidden value equal to LP_n.*

A.3 Numerical results for small n

To close this section we present numerical results for the cases $n = 2$ and $n = 3$. We find that for $n = 2$ the optimal algorithm is essentially the full window algorithm, with the exception that it picks the second cards if it does no select the first one. The algorithm can be described as follows: if the first revealed number (D_1) is the highest among the three numbers seen so far (U_1, U_2 and D_1), then select it. Otherwise, select D_2. This algorithm achieves a ratio of 3/4, and all settings s have at least one tight stochastic dominance constraint.

For $n = 3$ the result is far more interesting, as the algorithm is sensitive with small changes in histories, and requires randomization. The algorithm is detailed in Table 2.1 and achieves a ratio of approximately 0.704. The table details the probability of stopping when facing each history, where we omit all histories with probability 0 of stopping and we

also omit step 3, where we stop with probability 1 no matter what history we face. The first interesting behavior is that the algorithm will stop with probability 1 when the first revealed number (D_1) is the highest number seen so far, except if the number on the back of the first card (U_1) is the second highest among the numbers seen so far. In that case, stop with probability 0.95. This randomization is necessary, as forcing the probability to be 1 (0 respectively) decreases the ratio to 0.702 (0.583 respectively). The reason of this randomization is necessary because of the setting $s = \{(Y_1, Y_6), (Y_2, Y_3), (Y_4, Y_5)\}$. In this setting, whenever we see that D_1 is best so far and U_1 is second best so far, it is the case that $D_1 = Y_2$. Moreover, whenever we skip these situations in this setting, we end up selecting Y_1. With the optimal solution and for setting s, the stochastic dominance constraints are tight for $j = 1$ and $j = 3$, and there is slack for the case $j = 2$. Thus, if we were to increase the probability of stopping when facing this history, we would increase the probability of choosing Y_2 in exchange of decreasing the probability of choosing Y_1, which in turn reduces the competitive ratio, as the constraint for $j = 1$ is already tight in this setting. Further randomization is also necessary, as seen in the table.

References

Melika Abolhassani, Soheil Ehsani, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Robert Kleinberg, and Brendan Lucier. Beating 1-1/e for ordered prophets. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing*, pages 61–71, 2017.

Pablo Azar, Robert Kleinberg, and S Matthew Weinberg. Prophet inequalities with limited information. In *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1358–1377, 2014.

Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Prophet secretary: Surpassing the 1-1/e barrier. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 303–318, 2018.

Niv Buchbinder, Kamal Jain, and Mohit Singh. Secretary problems via linear programming. *Mathematics of Operations Research*, 39(1):190–206, 2014.

Gregory Campbell and Stephen Samuels. Choosing the best of the current crop. *Advances in Applied Probability*, 13(3):510–532, 1981.

Constantine Caramanis, Paul Dütting, Matthew Faw, Federico Fusco, Philip Lazos, Stefano Leonardi, Orestis Papadigenopoulos, Emmanouil Pountourakis, and Rebecca Reiffenhäuser. Single-sample prophet inequalities via greedy-ordered selection. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1298–1325, 2022.

Shuchi Chawla, Jason Hartline, David Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. In *Proceedings of the forty-second ACM Symposium on Theory of Computing*, pages 311–320, 2010.

José Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted price mechanisms for a random stream of customers. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 169–186, 2017.

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

Jose Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Recent developments in prophet inequalities. *ACM SIGecom Exchanges*, 17(1):61–70, 2019b.

José Correa, Patricio Foncea, Dana Pizarro, and Victor Verdugo. From pricing to prophets, and back! *Operations Research Letters*, 47(1):25–29, 2019c.

José Correa, Andrés Cristi, Boris Epstein, and José Soto. Sample-driven optimal stopping: From the secretary problem to the iid prophet inequality. *arXiv preprint arXiv:2011.06516*, 2020.

José Correa, Andrés Cristi, Laurent Feuilloley, Tim Oosterwijk, and Alexandros Tsigonias-Dimitriadis. *The Secretary Problem with Independent Sampling*, pages 2047–2058. 2021a.

Jose Correa, Raimundo Saona, and Bruno Ziliotto. Prophet secretary through blind strategies. *Mathematical Programming*, 2021b.

Paul Dütting and Thomas Kesselheim. Posted pricing and prophet inequalities with inaccurate priors. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 111–129, 2019.

Paul Dutting, Michal Feldman, Thomas Kesselheim, and Brendan Lucier. Prophet inequalities made easy: Stochastic optimization by pricing nonstochastic inputs. *SIAM Journal on Computing*, 49(3):540–582, 2020.

Evgenii Borisovich Dynkin. The optimum choice of the instant for stopping a markov process. *Soviet Mathematics*, 4:627–629, 1963.

Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet secretary for combinatorial auctions and matroids. In *Proceedings of the twenty-ninth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 700–714, 2018.

Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet secretary. *SIAM Journal on Discrete Mathematics*, 31(3):1685–1701, 2017.

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

John Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. *Journal of the American Statistical Association*, 61(313):35–73, 1966.

Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *Proceeding of the twenty-second Conference on Artificial Intelligence*, pages 58–65, 2007.

Theodore Hill and Robert Kertz. Comparisons of stop rule and supremum expectations of iid random variables. *The Annals of Probability*, 10(2):336–345, 1982.

Theodore Hill and Robert Kertz. A survey of prophet inequalities in optimal stopping theory. *Contemporary Mathematics*, 125:191–207, 1992.

Haim Kaplan, David Naori, and Danny Raz. Competitive analysis with a sample and the secretary problem. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, pages 2082–2095, 2020.

Haim Kaplan, David Naori, and Danny Raz. Online weighted matching with a sample. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms*, 2022.

Robert Kertz. Stop rule and supremum expectations of iid random variables: a complete comparison by conjugate duality. *Journal of Multivariate Analysis*, 19(1):88–112, 1986.

Robert Kleinberg and Matthew Weinberg. Matroid prophet inequalities. In *Proceedings of the forty-fourth annual ACM Symposium on Theory of Computing*, pages 123–136, 2012.

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

Ulrich Krengel and Louis Sucheston. On semiamarts, amarts, and processes with finite value. *Advances in Probability and Related Topics*, 4:197–266, 1978.

Denis Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 10(1):39–51, 1961.

Allen Liu, Renato Paes Leme, Martin Pal, Jon Schneider, and Balasubramanian Sivan. Competing optimally against an imperfect prophet. *arXiv preprint arXiv:2004.10163*, 2020.

Brendan Lucier. An economic view of prophet inequalities. *ACM SIGecom Exchanges*, 16 (1):24–47, 2017.

Pranav Nuti. On the best-choice prophet secretary problem. *arXiv preprint arXiv:2012.02888*, 2020.

Aviad Rubinstein, Jack Wang, and Matthew Weinberg. Optimal single-choice prophet inequalities from samples. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, pages 60:1–60:10, 2020.

Ester Samuel-Cahn. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *The Annals of Probability*, 12(4):1213–1216, 1984.