Fair Secretaries with Unfair Predictions

Eric Balkanski
Columbia University
eb3224@columbia.edu

Will Ma
Columbia University
wm2428@gsb.columbia.edu

Andreas Maggiori
Columbia University
am6292@columbia.edu

Abstract

Algorithms with predictions is a recent framework for decision-making under uncertainty that leverages the power of machine-learned predictions without making any assumption about their quality. The goal in this framework is for algorithms to achieve an improved performance when the predictions are accurate while maintaining acceptable guarantees when the predictions are erroneous. A serious concern with algorithms that use predictions is that these predictions can be biased and, as a result, cause the algorithm to make decisions that are deemed unfair. We show that this concern manifests itself in the classical secretary problem in the learning-augmented setting—the state-of-the-art algorithm can have zero probability of accepting the best candidate, which we deem unfair, despite promising to accept a candidate whose expected value is at least $\max\{\Omega(1), 1 - O(\varepsilon)\}$ times the optimal value, where ε is the prediction error. We show how to preserve this promise while also guaranteeing to accept the best candidate with probability $\Omega(1)$. Our algorithm and analysis are based on a new "pegging" idea that diverges from existing works and simplifies/unifies some of their results. Finally, we extend to the k-secretary problem and complement our theoretical analysis with experiments.

1 Introduction

As machine learning algorithms are increasingly used in socially impactful decision-making applications, the fairness of those algorithms has become a primary concern. Many algorithms deployed in recent years have been shown to be explicitly unfair or reflect bias that is present in training data. Applications where automated decision-making algorithms have been used and fairness is of central importance include loan/credit-risk evaluation [46, 36, 45], hiring [8, 13], recidivism evaluation [47, 2, 19, 11, 14], childhood welfare systems [12], job recommendations [40], and others [30, 28, 31]. A lot of work in recent years has been devoted to formally defining different notions of fairness [43, 34, 24, 21, 15, 38, 37] and designing algorithms that satisfy these different definitions [33, 32, 9, 10, 49].

While most fairness work concentrates on classification problems where the instance is known offline, we explore the problem of making fair decisions when the input is revealed in an online manner. Although fairness in online algorithms is an interesting line of research per se, fairness considerations have become increasingly important due to the recent interest in incorporating (possibly biased) machine learning predictions into the design of classical online algorithms. This framework, usually referred to as *learning-augmented algorithms* or *algorithms with predictions*, was first formalized in [44]. In contrast to classical online algorithms problems where it is assumed that no information is known about the future, learning-augmented online algorithms are given as input, possibly erroneous, predictions about the future. The main challenge is to simultaneously achieve an improved performance when the predictions are accurate and a robust performance when the

To appear at the 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

predictions are arbitrarily inaccurate. A long list of online problems have been considered in this setting and we point to [41] for an up-to-date list of papers. We enrich this active area of research by investigating how potentially biased predictions affect the fairness of decisions made by learning-augmented algorithms, and ask the following question:

Can we design fair algorithms that take advantage of unfair predictions?

In this paper, we study this question on a parsimonious formulation of the secretary problem with predictions, motivated by fairness in hiring candidates.

The problem. In the classical secretary problem, there are n candidates who each have a value and arrive in a random order. Upon arrival of a candidate, the algorithm observes the value of that candidate and must irrevocably decide whether to accept or reject that candidate. It can only accept one candidate and the goal is to maximize the probability of accepting the candidate with maximum value. In the classical formulation, only the *ordinal* ranks of candidates matter, and the algorithm of Dynkin [22] accepts the best candidate with a constant probability, that equals the best-possible $1/e$.

In the learning-augmented formulation of the problem proposed by Fujii and Yoshida [25], the algorithm is initially given a predicted value about each candidate and the authors focus on comparing the expected *cardinal* value accepted by the algorithm to the maximum cardinal value. The authors derive an algorithm that obtains expected value at least $\max\{\Omega(1), 1-O(\varepsilon)\}$ times the maximum value, where $\varepsilon \geq 0$ is the prediction error. The strength of this guarantee is that it approaches 1 as the prediction error decreases and it is a positive constant even when the error is arbitrarily large.

However, because the algorithm is now using predictions that could be biased, the best candidate may no longer have any probability of being accepted. We view this as a form of unfairness, and aim to derive algorithms that are fair to the best candidate by guaranteeing them a constant probability of being accepted (we contrast with other notions of fairness in stopping problems in Section 1.1). Of course, a simple way to be fair by this metric is to ignore the predictions altogether and run the classical algorithm of Dynkin. However, this approach would ignore potentially valuable information and lose the improved guarantee of [25] that approaches 1 when the prediction error is low.

Outline of results. We first formally show that the algorithm of [25] may in fact accept the best candidate with 0 probability. Our main result is then a new algorithm for secretary with predictions that: obtains expected value at least $\max\{\Omega(1), 1-O(\varepsilon)\}$ times the maximum value, like [25]; and ensures that, under any predictions, the probability that the best candidate is accepted is at least $1/16$. This result takes advantage of potentially biased predictions to achieve a guarantee on expected value that approaches 1 when the prediction error is small, while also providing a fairness guarantee for the true best candidate irrespective of the predictions. We note that Antoniadis et al. [4] also derive an algorithm for secretary with predictions, where the prediction is of the maximum value. This algorithm accepts the best candidate with constant probability but it does not provide a guarantee on the expected value accepted that approaches 1 as the prediction error approaches 0. Similarly, Dynkin's algorithm for the classical secretary problem accepts the best candidate with constant probability but does not make use of predictions at all. Finally, we note that the definitions of the prediction error ε differ in [25] and [4]; the former error definition uses the maximum ratio over all candidates between their predicted and true value while the latter uses the maximum absolute difference. Our techniques present an arguably simpler analysis and extend to a general family of prediction error measures that includes both of these error definitions.

We then extend our approach to the multiple choice or k-secretary problem where the goal is to accept at most k candidates and maximize the total of their values, which is the most technical part of the paper. We design an algorithm that obtains expected total value at least $\max\{\Omega(1), 1-O(\varepsilon)\}$ times the optimum (which is the sum of the k highest values), while simultaneously guaranteeing the k highest-valued candidates a constant probability of being accepted. We also have a refined guarantee that provides a higher acceptance probability for the $(1-\delta)k$ highest-valued candidates, for any $\delta \in (0,1)$.

Finally, we simulate our algorithms in the exact experimental setup of Fujii and Yoshida [25]. We find that they perform well both in terms of expected value accepted and fairness, whereas benchmark algorithms compromise on one of these desiderata.

1.1 Related work

The secretary problem. After Gardner [26] introduced the secretary problem, Dynkin [22] developed a simple and optimal stopping rule algorithm that, with probability at least $1/e$, accepts the candidate with maximum value. Due to its general and simple formulation, the problem has received a lot of attention (see, e.g., [42, 27] and references therein) and it was later extended to more general versions such as k-secretary [39], matroid-secretary [7] and knapsack-secretary [1].

Secretaries with predictions. The two works which are closest to our paper are those of Antoniadis et al. [3] and Fujii and Yoshida [25]. Both works design algorithms that use predictions regarding the values of the candidates to improve the performance guarantee of Dynkin's algorithm when the predictions are accurate while also maintaining robustness guarantees when the predictions are arbitrarily wrong. Antoniadis et al. [3] uses as prediction only the maximum value and defines the prediction error as the additive difference between the predicted and true maximum value while Fujii and Yoshida [25] receives a prediction for each candidate and defines the error as the maximum multiplicative difference between true and predicted value among all candidates.

Secretaries with distributional advice. Another active line of work is to explore how *distributional* advice can be used to surpass the $1/e$ barrier of the classical secretary problem. Examples of this line of work include the *prophet secretary problems* where each candidate draws its valuation from a known distribution [23, 17, 18, 6] and the *sample secretary problem* where the algorithm designer has only sample access to these distribution [35, 16]. We note that in the former models, predictions are either samples from distributions or distributions themselves which are assumed to be perfectly correct, while in the learning-augmented setting, we receive point predictions that could be completely incorrect. Dütting et al. [20] investigate a general model for advice where both values and advice are revealed upon a candidate's arrival and are drawn from a joint distribution $\mathcal{F}$. For example, their advice can be a noisy binary prediction about whether the current candidate is the best overall. Their main result uses linear programming to design optimal algorithms for a broad family of advice that satisfies two conditions. However, these two conditions are not satisfied by the predictions we consider. Additionally, we do not assume any prior knowledge of the prediction quality, whereas their noisy binary prediction setting assumes that the error probability of the binary advice is known.

Fairness in stopping algorithms. We say that a learning-augmented algorithm for the secretary problem is F-fair if it accepts the candidate with the maximum true value with probability at least F. In that definition, we do not quantify unfairness as a prediction property but as an algorithmic one, since the algorithm has to accept the best candidate with probability at least F no matter how biased predictions are our fairness notion is a challenging one. That notion can be characterized as an individual fairness notion similar to the *identity-independent fairness* (IIF) and *time-independent fairness* (TIF) introduced in [5]. In the context of the secretary problem, IIF and TIF try to mitigate discrimination due to a person's identity and arrival time respectively. While these are very appealing fairness notions, the fair algorithms designed in [5] fall in the classical online algorithms setting as they do not make any assumptions about the future. Consequently, their performance is upper bounded by the performance of the best algorithm in the classical worst-case analysis setting. It is also interesting to note the similarities with the *poset secretary problem* in [48]. In the latter work the set of candidates is split into several groups and candidates belonging to different groups cannot be compared due to different biases in the evaluation. In some sense, we try to do the same; different groups of candidates may have predictions that are affected by different biases making the comparison difficult. In our case though, the main challenge arises from the fact that the sets of candidates whose predictions are subject to different biases are not known in advance. Thus, even with predictions regarding the candidates' values, it is difficult to rank them against each other before their true values get revealed.

2 Preliminaries

Secretary problem with predictions. Candidates $i = 1, \dots, n$ have true values u_i and predicted values $\hat{u}_i$. The number of candidates n and their predicted values are known in advance. The candidates arrive in a uniformly random order. Every time a new candidate arrives their true value is revealed and the algorithm must immediately decide whether to accept the current candidate or reject them irrevocably and wait for the next arrival. We let $i^* = \text{argmax}_i\, u_i$ and $\hat{\imath} = \text{argmax}_i\, \hat{u}_i$ denote the indices of the candidates with the maximum true and predicted value respectively. An *instance* $\mathcal{I}$ consists of the $2n$ values $u_1, \dots, u_n, \hat{u}_1, \dots, \hat{u}_n$ which, for convenience, are assumed to be mutually distinct[1] and greater or equal to 1^2. We let $\varepsilon(\mathcal{I})$ denote its *prediction error*. For simplicity, we focus on the additive prediction error $\varepsilon(\mathcal{I}) = \max_i |\hat{u}_i - u_i|$, but we consider an abstract generalization that includes the multiplicative prediction error of [25] in Appendix A.2.

Objectives. We let $\mathcal{A}$ be a random variable denoting the candidate accepted by a given algorithm on a fixed instance, which depends on both the arrival order and any internal randomness in the algorithm. We consider the following desiderata for a given algorithm:

$$\mathbf{E}[u_{\mathcal{A}}] \geq u_{i^*} \cdot (1 - C \cdot \varepsilon(\mathcal{I}))\,,\ \forall \mathcal{I} \qquad \textit{(smoothness)}$$
$$P[\mathcal{A} = i^*] \geq F,\ \forall \mathcal{I}. \qquad \textit{(fairness)}$$

Our goal is to derive algorithms that can satisfy smoothness and fairness with constants $C, F > 0$ that do not depend on the instance $\mathcal{I}$ or the number of candidates n. Existing algorithms for secretary with predictions do not simultaneously satisfy these desiderata, as we explain in Appendix A.1.

Comparison to other objectives. Existing algorithms for secretary with predictions do satisfy a weaker notion called R*-robustness*, where $\mathbf{E}[u_{\mathcal{A}}] \geq R \cdot u_{i^*}$ for some constant $R > 0$. Our desideratum of fairness implies F-robustness and aligns with the classical secretary formulation where one is only rewarded for accepting the best candidate. Another notion of interest in existing literature is *consistency*, which is how $\mathbf{E}[u_{\mathcal{A}}]$ compares to u_{i^*} when $\varepsilon(\mathcal{I}) = 0$. Our smoothness desideratum implies 1-consistency, the best possible consistency result, and guarantees a smooth degradation as $\varepsilon(\mathcal{I})$ increases beyond 0.

3 The algorithm

We first present and analyze the ADDITIVE-PEGGING algorithm which achieves the desiderata from Section 2. Then, we mention how using a more abstract prediction error and an almost identical analysis, permits us to generalize ADDITIVE-PEGGING to PEGGING which achieves comparable guarantees for a more general class of error functions that includes the multiplicative error.

Our algorithms assume that each candidate i arrives at an independently random arrival time t_i drawn uniformly from $[0, 1]$. The latter continuous-time arrival model is equivalent to candidates arriving in a uniformly random order [29] and simplifies the algorithm description and analysis. We also write ϵ_i as shorthand for $|u_i - \hat{u}_i|$, ε as shorthand for $\varepsilon(\mathcal{I})$ (so that $\varepsilon = \max_i \epsilon_i$) and $i \prec j$ if $t_i < t_j$.

Description of ADDITIVE-PEGGING . ADDITIVE-PEGGING ensures smoothness by always accepting a candidate whose value is close to $u_{\hat{\imath}}$ which, as we argue, is at least $u_{i^*} - 2\,\varepsilon$. To see this, note that $u_{\hat{\imath}} \geq \hat{u}_{\hat{\imath}} - \epsilon_{\hat{\imath}} \geq \hat{u}_{i^*} - \epsilon_{\hat{\imath}} \geq u_{i^*} - \epsilon_{i^*} - \epsilon_{\hat{\imath}} \geq u_{i^*} - 2\,\varepsilon$, where we used that $\hat{u}_{\hat{\imath}} \geq \hat{u}_{i^*}$ (by definition of $\hat{\imath}$) and $\varepsilon \geq \max\{\epsilon_{i^*}, \epsilon_{\hat{\imath}}\}$ (by definition of ε). Because $u_{i^*} - 2\,\varepsilon \geq u_{i^*}(1 - 2\varepsilon)$ from the assumption that $u_{i^*} \geq 1$, this suggests that for smoothness it suffices to focus on comparing to $u_{\hat{\imath}}$.

For the fairness desideratum, we note that Dynkin's algorithm [22] for the classical secretary problem relies on the observation that if a constant fraction of the candidates have arrived and the candidate who just arrived has the maximum true value so far, then this candidate has a constant probability of being the best overall. The same high-level intuition is used in some of the cases of our algorithm. Every time a new candidate i arrives, we check if i is the maximum so far and if $t_i > 1/2$. If these conditions hold, then in case 3 and subcases 4a and 4b of the algorithm, we accept the candidate. However, there are two crucial situations where ADDITIVE-PEGGING differs from Dynkin's algorithm.

The first such situation is when the candidate $\hat{\imath}$ with maximum predicted value arrives and we have that $\hat{\imath}$ is not the maximum so far or $t_{\hat{\imath}} \leq 1/2$. In this case, we cannot always reject $\hat{\imath}$, as Dynkin's algorithm would, because that would not guarantee smoothness. Instead, we reject $\hat{\imath}$ (subcase 2b) only if there is a future candidate whose prediction is sufficiently high compared to $u_{\hat{\imath}}$. We call I^{pegged} the set of those candidates. The main idea behind the pegged set I^{pegged} is that it contains the last candidate to arrive who can guarantee the smoothness property, which is why we accept that candidate when they arrive (*subcase 1b*).

The second situation where our algorithm departs from Dynkin's algorithm is when a candidate i arrives (with $i \neq \hat{\imath}$ and $i \neq i^{\mathsf{pegged}}$) and we have that u_i is the maximum value so far and $t_i > 1/2$.

[1]This is without loss as adding an arbitrarily small perturbation to each true and predicted value does not change the performance of our algorithms. This perturbation is convenient because it allows to never have ties.

[2]Our bounds can be expressed less elegantly when only assuming that the values are non-negative; see Appendix A

In this situation, we cannot always accept i as Dynkin's algorithm would, because that would again violate smoothness. Indeed, if $\hat{\imath}$ hasn't arrived yet and $u_i < \hat{u}_{\hat{\imath}} - \epsilon_i$, then it is possible that u_i is much smaller than $u_{\hat{\imath}}$, which is why our algorithm would enter neither subcase 4a nor 4b and end up rejecting i. Finally, we note that the running time of the algorithm is $O(n \log n)$

Algorithm 1 ADDITIVE-PEGGING

```
//* The algorithm stops when it accepts a candidate by executing A ← i. *//
while agent i arrives at time t_i do
    τ ← max_{j≺i} u_j
    case 1: i ∈ I^pegged
        subcase 1a: |I^pegged| > 1
            I^pegged ← I^pegged \ {i}
        subcase 1b: |I^pegged| = 1
            A ← i
    case 2: i = î and (t_î ≤ 1/2 or u_î < τ)
        subcase 2a: u_î ≥ max_{j≻i} û_j + ε_î
            A ← i
        subcase 2b: u_î < max_{j≻i} û_j + ε_î
            I^pegged ← {j ≻ i : u_î < û_j + ε_î}
    case 3: i = î and t_î > 1/2 and u_î > τ
        A ← i
    case 4: i ≠ î and t_i > 1/2 and u_i > τ
        subcase 4a: î has already arrived
            A ← i
        subcase 4b: u_i > û_î − ε_i
            A ← i
end while
```

Analysis of the ADDITIVE-PEGGING algorithm.

Lemma 1. ADDITIVE-PEGGING *satisfies* $u_{\mathcal{A}} \geq u_{i^*} \cdot (1 - 4\,\varepsilon(\mathcal{I}))$, $\forall \mathcal{I}$ *with probability 1.*

Proof. Let i^{pegged} denote the last arriving candidate in I^{pegged}.

We first argue that PEGGING always accepts a candidate irrespective of the random arrival times of the candidates. We focus on any instance where ADDITIVE-PEGGING does not accept a candidate until time $t_{\hat{\imath}}$. At time $t_{\hat{\imath}}$ either the conditions of *case 2* or *case 3* are true. Since in the latter case candidate $\hat{\imath}$ is accepted, we focus on the former case. The conditions of *subcase 2a* and *subcase 2b* are mutually exclusive. If conditions of *subcase 2a* are true then we accept candidate $\hat{\imath}$ and if conditions of *subcase 2b* are true then due to the definition of i^{pegged} and the conditions of *case 1* and *subcase 1b* it is guaranteed that by time $t_{i^{\text{pegged}}}$ ADDITIVE-PEGGING will accept a candidate.

We now argue that in all cases ADDITIVE-PEGGING maintains smoothness. Using ε, ϵ_i definitions and the fact that $\hat{\imath}$ is the candidate with the maximum predicted value we have: $\hat{u}_{\hat{\imath}} \geq \hat{u}_{i^*} \geq u_{i^*} - \epsilon_{i^*} \geq u_{i^*} - \varepsilon$. In *subcase 2a* and *case 3* candidate $\hat{\imath}$ is accepted and using the latter lower bound we get $u_{\hat{\imath}} \geq \hat{u}_{\hat{\imath}} - \epsilon_{\hat{\imath}} \geq u_{i^*} - \varepsilon - \epsilon_{\hat{\imath}} \geq u_{i^*} - 2\,\varepsilon$. If we accept i using *subcase 4a* note that we have $u_i \geq u_{\hat{\imath}}$ and consequently we get again $u_i \geq u_{i^*} - 2\,\varepsilon$. If i is accepted by *subcase 4b* we get that $u_i > \hat{u}_{\hat{\imath}} - \epsilon_i \geq u_{i^*} - \varepsilon - \epsilon_i \geq u_{i^*} - 2\,\varepsilon$. Finally, we need to lower bound the value $u_{i^{\text{pegged}}}$ in case our algorithm terminates accepting i^{pegged}. Note that from the way the pegged set I^{pegged} is defined in *subcase 2b* we always have $u_{\hat{\imath}} < \hat{u}_{i^{\text{pegged}}} + \epsilon_{\hat{\imath}}$ and since $u_{i^{\text{pegged}}} \geq \hat{u}_{i^{\text{pegged}}} - \epsilon_{i^{\text{pegged}}}$ we can conclude that $u_{i^{\text{pegged}}} > u_{\hat{\imath}} - \epsilon_{\hat{\imath}} - \epsilon_{i^{\text{pegged}}} \geq u_{i^*} - 4\,\varepsilon$. Finally, note that since $u_{i^*} \geq 1$ we have that $u_{i^*} - 4\,\varepsilon \geq u_{i^*}(1 - 4\,\varepsilon)$. □

Lemma 2. ADDITIVE-PEGGING *satisfies* $P[\mathcal{A} = i^*] \geq 1/16,\ \forall \mathcal{I}$.

Proof. We denote by $\tilde{\imath}$ the index of the candidate with the highest true value except possibly i^* and $\hat{\imath}$, i.e., $\tilde{\imath} = \text{argmax}_{i \neq i^*, \hat{\imath}}\, u_i$. Note that if $i^* = \hat{\imath}$ then $\tilde{\imath}$ denotes the index of the candidate with the second highest true value. To prove fairness we distinguish between two cases: either $\hat{\imath} = i^*$ or

$\hat{\imath} \neq i^*$. For each of those cases, we define an event and argue that: (1) the event happens with constant probability; and (2) if that event happens then ADDITIVE-PEGGING accepts i^*.

If $i^* = \hat{\imath}$ we define event $C = \{t_{\tilde{\imath}} < 1/2 < t_{i^*}\}$ for which $P[C] = 1/4$. C implies that our algorithm does not accept any candidate until time t_{i^*}. Indeed note that the conditions of *case 1, 2, and 3* may be true only after the arrival of $\hat{\imath}$ which is $t_{\hat{\imath}}$. In addition to that after time $1/2$ the threshold $\tau = u_{\tilde{\imath}}$ is larger than any value other than u_{i^*} and consequently the conditions of *case 4* are not met before t_{i^*}. At time t_{i^*} all conditions of *case 4* and *subcase 4a* are met and our algorithm hires i^*.

On the other hand, if $i^* \neq \hat{\imath}$ we distinguish between two subcases. We show that either $u_{\hat{\imath}} < \hat{u}_{i^*} + \epsilon_{\hat{\imath}}$ or $u_{i^*} > \hat{u}_{\hat{\imath}} - \epsilon_{i^*}$. By contradiction, assume that both inequalities do not hold, then

$$u_{\hat{\imath}} \geq \hat{u}_{i^*} + \epsilon_{\hat{\imath}} \xrightarrow{u_{i^*} > u_{\hat{\imath}}} u_{i^*} > \hat{u}_{i^*} + \epsilon_{\hat{\imath}} \Rightarrow u_{i^*} - \hat{u}_{i^*} > \epsilon_{\hat{\imath}} \xrightarrow{\epsilon_{i^*} \geq u_{i^*} - \hat{u}_{i^*}} \epsilon_{i^*} > \epsilon_{\hat{\imath}}$$

$$u_{i^*} \leq \hat{u}_{\hat{\imath}} - \epsilon_{i^*} \xrightarrow{u_{i^*} > u_{\hat{\imath}}} u_{\hat{\imath}} < \hat{u}_{\hat{\imath}} - \epsilon_{i^*} \Rightarrow \epsilon_{i^*} < \hat{u}_{\hat{\imath}} - u_{\hat{\imath}} \xrightarrow{\epsilon_{\hat{\imath}} \geq u_{\hat{\imath}} - \hat{u}_{\hat{\imath}}} \epsilon_{i^*} < \epsilon_{\hat{\imath}}$$

which is a contradiction. Thus, we define two events C_1 and C_2 which imply that i^* is always accepted whenever $\{u_{\hat{\imath}} < \hat{u}_{i^*} + \epsilon_{\hat{\imath}}\}$ and $\{u_{i^*} > \hat{u}_{\hat{\imath}} - \epsilon_{i^*}\}$ are true respectively.

If $u_{\hat{\imath}} < \hat{u}_{i^*} + \epsilon_{\hat{\imath}}$, then we define event $C_1 = \{t_{\tilde{\imath}} < 1/2\} \wedge \{t_{\hat{\imath}} < 1/2\} \wedge \{1/2 < t_{i^*}\}$ which is composed by 3 independent events and it happens with probability $P[C_1] = 1/2^3 = 1/8$. C_1 implies that at time t_{i^*} the conditions of *subcase 4a* are satisfied. Consequently, if until time t_{i^*} all candidates are rejected then candidate i^* is hired using either *case 1* or *subcase 4a*. To argue that no candidate is accepted before time t_{i^*} note that at time $t_{\hat{\imath}}$ the set $\{j \succ \hat{\imath} : u_{\hat{\imath}} < \hat{u}_j + \epsilon_{\hat{\imath}}\}$ contains i^* and that after time $1/2$ the condition $\tau > u_i$ is not met until time t_{i^*}.

If $u_{i^*} > \hat{u}_{\hat{\imath}} - \epsilon_{i^*}$, then we define $C_2 = \{t_{\tilde{\imath}} < 1/2 < t_{i^*} < t_{\hat{\imath}}\}$ which happens with probability

$$\begin{aligned}
P[C_2] &= P[t_{\tilde{\imath}} < 1/2] \cdot P[1/2 < t_{i^*} < t_{\hat{\imath}}] \\
&= P[t_{\tilde{\imath}} < 1/2] \cdot P[1/2 < \min\{t_{i^*}, t_{\hat{\imath}}\} \wedge \min\{t_{i^*}, t_{\hat{\imath}}\} = t_{i^*}] \\
&= P[t_{\tilde{\imath}} < 1/2] \cdot P[1/2 < \min\{t_{i^*}, t_{\hat{\imath}}\}] \cdot P[\min\{t_{i^*}, t_{\hat{\imath}}\} = t_{i^*}] \\
&= (1/2) \cdot (1/4) \cdot (1/2) = 1/16
\end{aligned}$$

Note that until time t_{i^*} no candidate is accepted since the conditions of all cases are not satisfied. Between times 0 and $1/2$ only $\hat{\imath}$ could have been accepted but its arrival time is after t_{i^*}, and between times $1/2$ and t_{i^*} the threshold τ is equal to $u_{\tilde{\imath}}$ and no candidate meets the condition of *case 4* to have $u_i > \tau$. Finally, note that at time t_{i^*} the conditions of *case 4b* are satisfied and i^* gets accepted. □

Theorem 3. ADDITIVE-PEGGING *satisfies smoothness and fairness with* $C = 4$ *and* $F = 1/16$.

Theorem 3 follows directly from Lemmas 1 and 2. We note that Lemma 1 actually implies a stronger notion of smoothness that holds with probability 1.

The general PEGGING algorithm. In Appendix A.2 we generalize the ADDITIVE-PEGGING algorithm to the PEGGING algorithm to provide fair and smooth algorithms for different prediction error definitions. ADDITIVE-PEGGING is an instantiation of PEGGING when the prediction error is defined as the maximum absolute difference between true and predicted values among candidates. To further demonstrate the generality of PEGGING , we also instantiate it over the same prediction error definition $\varepsilon(\mathcal{I}) = \max_i |1 - \hat{u}_i/u_i|$ as in [25] and recover similar smoothness bounds as in [25] while also ensuring fairness. We name the latter instantiation MULTIPLICATIVE-PEGGING and present its guarantees in Theorem 4.

Theorem 4. *If* $\varepsilon(\mathcal{I}) = \max_i |1 - \hat{u}_i/u_i|$, *then* MULTIPLICATIVE-PEGGING *satisfies smoothness and fairness with* $C = 4$ *and* $F = 1/16$.

Fujii and Yoshida [25] define the prediction error as in Theorem 4 and design an algorithm that accepts a candidate with expected value at least $u_{i^*} \cdot \max\{(1-\varepsilon)/(1+\varepsilon), 0.215\}$. Since $(1-\varepsilon)/(1+\varepsilon) \geq 1 - 2\,\varepsilon$ the latter algorithm satisfies the smoothness desideratum of Section 2, but, as we prove in Appendix A.1 it violates the fairness desideratum.

4 Extension: k-Secretary problem with predictions

We consider the generalization to the k-secretary problem, where $k \geq 1$ candidates can be accepted. To simplify notation we label the candidates in decreasing order of predicted value, so that $\hat{u}_1 >$

$\cdots > \hat{u}_n$ and denote r_i to be the index of the candidate with the i'th highest true value so that $u_{r_i} > \cdots > u_{r_n}$. The prediction error is again defined as $\varepsilon(\mathcal{I}) := \max_i |u_i - \hat{u}_i|$ and we let S denote the random set of candidates accepted by a given algorithm on a fixed instance. The extension of our two objectives to this setting is

$$\mathbf{E}\left[\sum_{i\in S} u_i\right] \geq (1 - C\cdot\varepsilon(\mathcal{I}))\sum_{i=1}^{k} u_{r_i},\ \forall\mathcal{I} \qquad \textit{(smoothness for k-secretary)}$$

$$P[r_i \in S] \geq F_i,\ \forall i = 1,\ldots,k,\ \ \forall\mathcal{I}. \qquad \textit{(fairness for k-secretary)}$$

The smoothness desideratum compares the expected sum of true values accepted by the algorithm to the sum of the k highest true values that could have been accepted. The fairness desideratum guarantees each of these candidates in the top k to be accepted with probability F_i. The k-secretary problem with predictions has been studied by Fujii and Yoshida [25], who derive an algorithm satisfying $\mathbf{E}\left[\sum_{i\in S} u_i\right] \geq \max\{1 - O(\log k/\sqrt{k}), 1 - O(\max_i |1 - \hat{u}_i/u_i|)\}\sum_{i=1}^{k} u_{r_i}$ but without any fairness guarantees. We derive an algorithm K-PEGGING that satisfies the following.

Theorem 5. K-PEGGING *satisfies smoothness and fairness for k-secretary with $C = 4$ and $F_i = \max\left\{(1/3)^{k+5}, \frac{1-(i+13)/k}{256}\right\}$ for all $i = 1,\ldots,k$.*

Assuming k is a constant, C and $F_1,\ldots,F_k$ in Theorem 5 are constants that do not depend on n or the instance $\mathcal{I}$. For large values of k the first term in F_i is exponentially decaying, but the second term still guarantees candidate r_i a probability of acceptance that is independent of k as long as i/k is bounded away from 1. More precisely, for any constant $\delta > 0$ and $i \leq (1-\delta)k - 13$, candidate r_i is accepted with probability at least $\delta/256$. For large values of k, we can apply this fact with $\delta = 1/2$ to see that all candidates in the top quarter are accepted with probability at least $1/512$, getting a "robustness" constant that is independent of k: $\mathbf{E}\left[\sum_{i\in S} u_i\right] \geq \frac{1}{2048}\sum_{i=1}^{k} u_{r_i}$.

The algorithm. While we defer the proof of Theorem 5 to Appendix B, we present the intuition and the main technical difficulties in the design of K-PEGGING . K-PEGGING maintains in an online manner the following sets: (1) the solution set S which contains all the candidates that have already been accepted; (2) a set H that we call the hopeful set and contains the $k - |S|$ future candidates with the highest predicted values; (3) a set B that we call the blaming set. B contains a subset of already arrived candidates that pegged a future candidate; and (4) the set P of pegged elements which contains all candidates that have been pegged by a candidate in B. In addition, we use function $peg : \{1,...,k\} \to [n]$ to store the "pegging responsibility", i.e., if $peg(i) = j$ then at time t_i, $i \in \{1,...,k\}$: (a) was not accepted; and (b) forced j to be pegged. We use $peg^{-1}(j) = i$ to denote that j was pegged by i.

To satisfy the fairness property, we check if the current candidate i has arrived at time $t_i \geq 1/2$ and if u_i larger than the k^{th} highest value seen so far. We refer to these two conditions as the fairness conditions. If $i \in P$ (*case 1*) or $i \in H$ and the fairness conditions hold (*case 3*), then we accept i. If the fairness conditions hold but $i \notin H$ then we accept only if there is a past candidate in B (*subcase 4a*) or a future candidate in H (*subcase 4b*) whose value is close to u_i.

The main technical challenge arises in order to generalize the pegging idea when a candidate $i \in H$ arrives but one of the fairness conditions does not hold. Indeed, it is not clear if we should reject i and peg a future candidate or accept i. Consider an instance where the prediction error is high enough so that when i arrives there is always a future candidate that can be pegged and consider the case where $t_i < 1/2$. If we accept then we may decrease our budget too aggressively until time $1/2$ and do not have enough space in our solution to accept candidates not in $[k]$ in the second half of the time. However, if we do not accept then we do not give i the possibility of being accepted in the first half of the time and we may decrease its probability of being accepted too much. K-PEGGING balances these two objectives while achieving smoothness by accepting a solution set S with values that are pairwise "close" to the values of candidates in $\{1, 2, \ldots, k\}$. We prove the latter in lemma 8 by defining an injective function $m(\cdot)$ from set S to $\{1, 2, \ldots, k\}$ such that for each $j \in S$, $u_j \approx u_{m(j)}$. Finally, we note that the running time of this algorithm is also $O(n \log n)$.

Algorithm 2 K-PEGGING

$H \leftarrow [k], S \leftarrow \emptyset, P \leftarrow \emptyset, B \leftarrow \emptyset$
while agent i arrives at time t_i **do**
  $\tau \leftarrow k^{th}$ highest value seen before time t_i, $\varepsilon_{t_i} \leftarrow \max_{j:t_j \leq t_i} |\hat{u}_j - u_j|$
  case 1: $i \in P$
    Add i to S, remove i from P and remove $peg^{-1}(i)$ from B.
  case 2: $i \in H$ **and** ($t_i \leq 1/2$ **or** $u_i \leq \tau$)
    subcase 2a: $\{j \succ i : u_i < \hat{u}_j + \varepsilon_{t_i}\} \setminus (P \cup [k]) = \emptyset$ **or** ($C_i = 0$ with $C_i \sim Bernoulli(1/2)$)
      Add i to S, remove i from H
    subcase 2b: Otherwise
      Add $\text{argmin}_{j \in \{j \succ i : u_i < \hat{u}_j + \varepsilon_{t_i}\} \setminus (P \cup [k])} \hat{u}_j$ to P, add i to B, remove i from H
  case 3: $i \in H$ **and** ($t_i > 1/2$ **and** $u_i > \tau$)
    Add i to S and remove i from H
  case 4: $i \notin H$ **and** ($t_i > 1/2$ **and** $u_i > \tau$)
    subcase 4a: $\{j \in B : u_i > u_j\} \neq \emptyset$
      Add i to S, remove $\text{argmax}_{j \in B : u_i > u_j} u_j$ from B, and remove $peg(j)$ from P
    subcase 4b: $\{j \in H : u_i > \hat{u}_j - \varepsilon_{t_i}\} \neq \emptyset$
      Add i to S and remove $\text{argmax}_{j \in H : u_i > \hat{u}_j - \varepsilon_{t_i}} u_j$ from H
end while

5 Experiments

We simulate our ADDITIVE-PEGGING and MULTIPLICATIVE-PEGGING algorithms in the exact experimental setup of Fujii and Yoshida [25], to test its average-case performance.

Experimental Setup. Fujii and Yoshida [25] generate various types of instances. We follow their Almost-constant, Uniform, and Adversarial types of instances, and also create the Unfair type of instance to further highlight how slightly biased predictions can lead to very unfair outcomes. All these instance types are parameterized by a scalar $\varepsilon \in [0, 1)$ which controls the prediction error. Setting $\varepsilon = 0$ creates instances with perfect predictions and setting a higher value of ε creates instances with more erroneous predictions. Almost-constant models a situation where one candidate has a true value of $1/(1-\varepsilon)$ and the rest of the candidates have a value of 1. All predictions are set to 1. In Uniform we sample each u_i independently from the exponential distribution with parameter 1. The exponential distribution generates a large value with a small probability and consequently models a situation where one candidate is significantly better than the rest. All predicted values are generated by perturbing the actual value with the uniform distribution, i.e., $\hat{u}_i = \delta_i \cdot u_i$, where δ_i is sampled uniformly and independently from $[1-\varepsilon, 1+\varepsilon]$. In Adversarial the true values are again independent samples from the exponential distribution with parameter 1. The predictions are "adversarially" perturbed while maintaining the error to be at most ε in the following manner: if i belongs to the top half of candidates in terms of true value, then $\hat{u}_i = (1-\varepsilon) \cdot u_i$; if i belongs to the bottom half, then $\hat{u}_i = (1+\varepsilon) \cdot u_i$. Finally, in Unfair all candidates have values that are at most a $(1+\varepsilon)$ multiplicative factor apart. Formally, u_i is a uniform value in $[1-\varepsilon/4, 1+\varepsilon/4]$, and since $(1+\varepsilon/4)/(1-\varepsilon/4) \leq (1+\varepsilon)$ we have that the smallest and largest value are indeed very close. We set $\hat{u}_i = u_{n-r(i)+1}$ where $r(i)$ is the rank of u_i, i.e., predictions create a completely inverted order.

We compare ADDITIVE-PEGGING and MULTIPLICATIVE-PEGGING against LEARNED-DYNKIN [25], HIGHEST-PREDICTION which always accepts the candidate with the highest prediction, and the classical DYNKIN algorithm which does not use the predictions. Following [25], we set the number of candidates to be $n = 100$. We experiment with all values of ε in $\{0, 1/20, 2/20, \ldots, 19/20\}$. For each type of instance and value of ε in this set, we randomly generate 10000 instances, and then run each algorithm on each instance. For each algorithm, we consider instance-wise the ratio of the true value it accepted to the maximum true value, calling the average of this ratio across the 10000 instances its *competitive ratio*. For each algorithm, we consider the fraction of the 10000 instances on which it successfully accepted the candidate with the highest true value, calling this fraction its *fairness*. We report the competitive ratio and fairness of each algorithm, for each type of instance and each value of ε, in Figure 1. Our code is written in Python 3.11.5 and we conduct experiments on an M3 Pro CPU with 18 GB of RAM. The total runtime is less than 5 minutes.

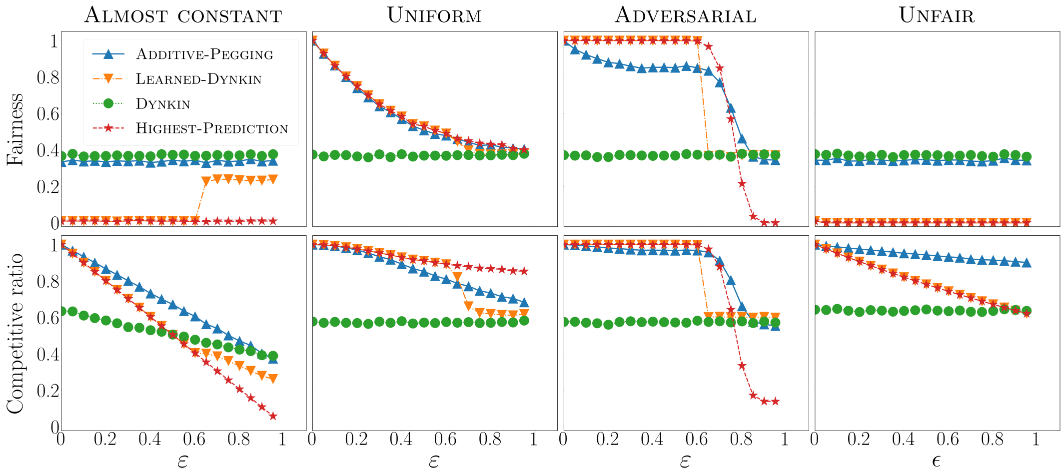

Figure 1: Competitive ratio and fairness of different algorithms, for each instance type and level of ε.

Results. The results are summarized in figure 1. Since ADDITIVE-PEGGING and MULTIPLICATIVE-PEGGING achieve almost the same competitive ratio and fairness for all instance types and values of ε we only present ADDITIVE-PEGGING in figure 1 but include the code of both in the supplementary material. Our algorithms are consistently either the best or close to the best in terms of both competitive ratio and fairness for all different pairs of instance types and ε values. Before discussing the results of each instance type individually it is instructive to mention some characteristics of our benchmarks. While DYNKIN does not use predictions and is therefore bound to suboptimal competitive ratios when predictions are accurate, we note that it accepts the maximum value candidate with probability at least $1/e$, i.e., it is $1/e$-fair. When predictions are non-informative this is an upper bound on the attainable fairness for any algorithm whether it uses predictions or not. HIGHEST-PREDICTION is expected to perform well when the highest prediction matches the true highest value candidate and poorly when the latter is not true. In Almost-constant for small values of ε all candidates have very close true values and all algorithms except DYNKIN have a competitive ratio close to 1. DYNKIN may not accept any candidate and this is why its performance is poorer than the rest of the algorithms. Note that as ε increases both our algorithms perform significantly better than all other benchmarks.

In terms of fairness, predictions do not offer any information regarding the ordinal comparison between candidates' true values and this is why for small values of ε the probability of HIGHEST-PREDICTION and LEARNED-DYNKIN of accepting the best candidate is close to $1/100 = 1/n$, i.e., random. Here, the fairness of our algorithms and DYNKIN is similar and close to $1/e$. In both Uniform and Adversarial we observe that for small values of ε the highest predicted candidate is the true highest and ADDITIVE-PEGGING , LEARNED-DYNKIN and HIGHEST-PREDICTION all accept that candidate having a very close performance both in terms of fairness and competitive ratio. For higher values of ε the fairness of those algorithms deteriorates similarly and it approaches again $0.37 \simeq 1/e$. In Unfair our algorithms outperform all other benchmarks in terms of competitive ratio for all values of ε and achieve a close to optimal fairness. This is expected as our algorithms are particularly suited for cases where predictions may be accurate but unfair.

Overall, our algorithms are the best-performing and most robust. The HIGHEST-PREDICTION algorithm does perform slightly better on Uniform instances and Adversarial instances under most values of ε, but performs consistently worse on Almost-constant and Unfair instances, especially in terms of fairness. Our algorithms perform better than LEARNED-DYNKIN in almost all situations.

6 Limitations

We study a notion of fairness that is tailored to the secretary problem with predictions and build our algorithms based on this notion. However, there are alternative notions of fairness one could consider in applications such as hiring, as well as variations of the secretary problem that capture other features in these applications. While our model allows for arbitrary bias in the predictions we assume that the true value of a candidate is fully discovered upon arrival, and define fairness based on hiring the best candidate (who has the highest true value) with a reasonable probability. Thus, we ignore considerations such as bias in how we get the true value of a candidate (e.g., via an interview process). In addition, as noted in Section 1, we use an *individual fairness* notion which does not model other natural desiderata like hiring from underprivileged populations or balance the hiring probabilities across different populations. These are considerations with potentially high societal impact which our algorithms do not consider and are interesting directions for future work on fair selection with predictions.

References

[1] Susanne Albers and Leon Ladewig. New results for the k-secretary problem. In *Proceedings of the 30th Annual International Symposium on Algorithms and Computation (ISAAC)*, pages 18:1–18:19, 2019.

[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica*, May 23 2016. URL `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

[3] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 345–355, 2020. URL `https://proceedings.mlr.press/v119/antoniadis20a.html`.

[4] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 7933–7944, 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/5a378f8490c8d6af8647a753812f6e31-Paper.pdf`.

[5] Makis Arsenis and Robert Kleinberg. Individual fairness in prophet inequalities. In *Proceedings of the 23rd ACM Conference on Economics and Computation (EC)*, page 245, 2022.

[6] Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Prophet secretary: Surpassing the $1 - 1/e$ barrier. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pages 303–318, 2018.

[7] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 16–28, 2007.

[8] Miranda Bogen and Aaron Rieke. Help wanted: an examination of hiring algorithms. *Equity, and Bias, Upturn (December 2018)*, 2018.

[9] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with fairness constraints. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.

[10] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvtiskii. Matroids, matchings, and fairness. In *Proceedings of the 22ndInternational Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2212–2220, 2019. URL `https://proceedings.mlr.press/v89/chierichetti19a.html`.

[11] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, June 2017.

[12] Alexandra Chouldechova, Diana Benavides-Prado, Oleksandr Fialko, and Rhema Vaithianathan. A case study of algorithm-assisted decision making in child maltreatment hotline screening decisions. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency (FAccT)*, pages 134–148, 2018. URL `http://proceedings.mlr.press/v81/chouldechova18a.html`.

[13] Lee Cohen, Zachary C. Lipton, and Yishay Mansour. Efficient candidate screening under multiple tests and implications for fairness. In *Proceedings of the 1st Symposium on the foundations of responsible computing (FORC)*, page 1–20, 2019.

[14] COMPAS (software). `https://en.wikipedia.org/wiki/COMPAS_(software)`.

[15] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (KDD)*, page 797–806, 2017. URL `https://doi.org/10.1145/3097983.3098095`.

[16] José R. Correa, Andrés Cristi, Laurent Feuilloley, Tim Oosterwijk, and Alexandros Tsigonias-Dimitriadis. The secretary problem with independent sampling. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2047–2058, 2021.

[17] José R. Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted price mechanisms and optimal threshold strategies for random arrivals. *Mathematics of Operations Research*, 46(4):1452–1478, 2021.

[18] José R. Correa, Raimundo Saona, and Bruno Ziliotto. Prophet secretary through blind strategies. *Mathematical Programming*, 190(1):483–521, 2021.

[19] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science Advances*, 4(1), 2018. URL `https://www.science.org/doi/abs/10.1126/sciadv.aao5580`.

[20] Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. Secretaries with advice. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*, page 409–429, 2021. URL `https://doi.org/10.1145/3465456.3467623`.

[21] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, page 214–226, 2012. URL `https://doi.org/10.1145/2090236.2090255`.

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

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

[24] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining (KDD)*, page 259–268, 2015. URL `https://doi.org/10.1145/2783258.2783311`.

[25] Kaito Fujii and Yuichi Yoshida. The secretary problem with predictions. In *Mathematics of Operations Research*, 2023.

[26] Martin Gardner. Mathematical games. *Scientific American*, 202(3):172–186, 1960.

[27] John P. Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. *Journal of the American Statistical Association*, 61:35–73, 1966.

[28] Lev Grossman. Are face-detection cameras racist? *Time*, 2010. URL `http://content.time.com/time/business/article/0,8599,1954643,00.html`.

[29] Anupam Gupta and Sahil Singla. *Random-Order Models*, page 234–258. Cambridge University Press, 2021.

[30] Alex Hern. Artificial intelligence beauty contest doesn't like black people. *The Guardian*, September 8 2016. URL `https://www.theguardian.com/technology/2016/sep/08/artificial-intelligence-beauty-contest-doesnt-like-black-people`.

[31] A Howard and J Borenstein. The ugly truth about ourselves and our robot creations: The problem of bias and social inequity. *Science and Engineering Ethics*, 24(5):1521–1536, October 2018.

[32] Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/eb163727917cbba1eea208541a643e74-Paper.pdf.

[33] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *Proceedings of the 11thInternational Conference on Data Mining Workshops (ICDMW)*, pages 643–650, 2011.

[34] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, volume 7524, 2012.

[35] Haim Kaplan, David Naori, and Danny Raz. Competitive analysis with a sample and the secretary problem. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2082–2095, 2020.

[36] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010. URL https://www.sciencedirect.com/science/article/pii/S0378426610002372.

[37] Jon Kleinberg and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, 2017.

[38] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human Decisions and Machine Predictions. *The Quarterly Journal of Economics*, 133(1): 237–293, 2017. URL https://doi.org/10.1093/qje/qjx032.

[39] Robert D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–631, 2005.

[40] Anja Lambrecht and Catherine Tucker. Algorithmic bias? an empirical study of apparent gender-based discrimination in the display of stem career ads. *Management Science*, 65(7): 2966–2981, 2019.

[41] Alexander Lindermayr and Nicole Megow. Algorithms with predictions. https://algorithms-with-predictions.github.io/, 2024.

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

[43] Binh Thanh Luong, Salvatore Ruggieri, and Franco Turini. k-nn as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (KDD)*, page 502–510, 2011. doi: 10.1145/2020408.2020488. URL https://doi.org/10.1145/2020408.2020488.

[44] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3302–3311, 2018. URL http://proceedings.mlr.press/v80/lykouris18a.html.

[45] Rashmi Malhotra and D. K. Malhotra. Evaluating consumer loans using neural networks. *Omega*, 31(2):83–96, 2003. URL https://EconPapers.repec.org/RePEc:eee:jomega:v:31:y:2003:i:2:p:83-96.

[46] Amitabha Mukerjee, Rita Biswas, Kalyanmoy Deb, and Amrit P. Mathur. Multi–objective evolutionary algorithms for the risk–return trade–off in bank loan management. *International Transactions in Operational Research*, 9(5):583–597, 2002. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1475-3995.00375.

[47] David B. Mustard. Reexamining criminal behavior: The importance of omitted variable bias. *The Review of Economics and Statistics*, 85(1):205–211, 2003.

[48] Jad Salem and Swati Gupta. Secretary problems with biased evaluations using partial ordinal information. *Management Science*. URL `https://doi.org/10.1287/mnsc.2023.4926`.

[49] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29thInternational Conference on Scientific and Statistical Database Management (SSDBM)*, SSDBM '17, 2017. URL `https://doi.org/10.1145/3085504.3085526`.

A Additional discussion and missing analysis for single secretary

A.1 Unfair outcomes in previous work

In this section we present the learning-augmented algorithms of [3] and [25], and argue that they fail to satisfy simultaneously the smoothness and fairness desiderata described in section 2. We follow the same notation as in the main paper where the $i^*, \hat{\imath}$ denote the index of the candidate with maximum true and predicted value respectively. Since the algorithm in [3] requires only the prediction about the maximum value but not the identity of that candidate, we use the symbol $\hat{u}^*$ to denote such value.

Algorithm 3 LEARNED-DYNKIN [25]

$\theta \leftarrow 0.646$, $t \leftarrow 0.313$, mode $\leftarrow$ Prediction
while agent i arrives at time t_i **do**
  $\tau \leftarrow \max_{j \prec i} u_j$
  if $|1 - \hat{u}_i/u_i| > \theta$ **then**
    mode $\leftarrow$ Secretary.
  end if
  if mode = Prediction and $i = \hat{\imath}$ **then**
    $\mathcal{A} \leftarrow i$.
  end if
  if mode = Secretary and $t_i > t$ and $u_i > \tau$ **then**
    $\mathcal{A} \leftarrow i$.
  end if
end while

Algorithm 4 VALUE-MAXIMIZATION SECRETARY [3]

$t^* \leftarrow \exp\{W^{-1}(-1/(ce))\}$, $t^{**} \leftarrow \exp\{W^0(-1/(ce))\}$
while agent i arrives at time t_i **do**
  $\tau_I \leftarrow \max_{j:t_j<t^*} u_j$
  if $t^* < t_i < t^{**}$ **and** $u_i > \max\{\tau^*, \hat{u}^* - \lambda\}$ **then**
    $\mathcal{A} \leftarrow i$
  end if
  $\tau^{**} \leftarrow \max_{j:t_j<t^{**}} u_j$
  if $t_i \geq t^{**}$ **and** $u_i > \tau^{**}$ **then**
    $\mathcal{A} \leftarrow i$
  end if
end while

LEARNED-DYNKIN of Fujii and Yoshida [25] receives a predicted valuation for all candidates and defines the prediction error of candidate i as $|1 - \hat{u}_i/u_i|$. If the prediction error of a candidate is higher than θ then it switches to Secretary mode where it mimics the classical DYNKIN algorithm where all candidates are rejected for a constant fraction of the stream and after that rejection phase the first candidate whose valuation is the maximum overall is hired. Note that if all candidates have very low prediction error then LEARNED-DYNKIN remains in the Prediction mode and the candidate with the higher prediction is hired. One instance where LEARNED-DYNKIN never accepts candidate i^* is the following: the true valuations are $\{1 + \theta', 1, u_3, \ldots, u_n\}$ and the predicted valuations are $\{1+\theta', 1/(1-\theta'), u_3/(1-\theta'), \dots, u_n/(1-\theta')\}$ where $\theta' < \theta = 0.646$ and $u_3, \dots, u_n$ are pairwise distinct numbers in $(0,1)$. Note that $1/(1-\theta') > 1+\theta'$ and the prediction error is as most $\theta' < \theta = 0.616$ for all candidates. Thus, LEARNED-DYNKIN does not switch to Prediction mode, it never accepts the candidate with valuation $1+\theta'$ and does not satisfy our Fairness desideratum.

VALUE-MAXIMIZATION SECRETARY of [3] receives only one prediction regarding the maximum value $\hat{u}^*$ and the prediction error is defined as $\varepsilon = |u_{i^*} - \hat{u}^*|$. The latter algorithm is parametrized by $\lambda \geq 0$ and $c \geq 1$ which control the relationship between the robustness and smoothness bounds. VALUE-MAXIMIZATION SECRETARY has three distinct phases defined by the time ranges $[0, t^*]$, (t^*, t^{**}) and $[t^{**}, 1]$ respectively, where t^*, t^{**} are defined using the Lambert functions $W^{-1}$and W^0. The first phase is used as an "exploration" phase where all candidates are rejected and at the end of the phase a threshold τ_I is computed.

Let $\mathcal{A}$ be the random variable denoting the candidate which is accepted by VALUE-MAXIMIZATION SECRETARY . For any $\varepsilon \in (0,1)$ we define an instance with predicted maximum value $\hat{u}^* = 1-\varepsilon$ and true values $\{u_1, u_2, \dots, u_n\}$ where all numbers are distinct, $u_1 = 1$ and $u_i \in [0, \varepsilon]$, $\forall i \in \{2, \dots, n\}$. The prediction error of that instance is ε. Note that if $i^* = 1$ arrives in the first phase then the maximum value of a candidate that VALUE-MAXIMIZATION SECRETARY can accept in the second and third phases is at most ε. Thus, we can upper bound the expected value of candidate $\mathcal{A}$ as follows: $\mathbf{E}[u_{\mathcal{A}}] \leq P[t_{i^*} \geq t^*] \cdot u_{i^*} + P[t_{i^*} < t^*] \cdot \varepsilon = (1-t^*)u_{i^*} + t^* \varepsilon$.

We emphasize that in the learning-augmented setting, there is no assumption regarding the quality of the prediction, thus the parameters c and λ cannot depend on the prediction error. For any setting with $c > 1$ we have that $t^* > 0$ is a constant that is bounded away from 0. Towards a contradiction, assume that VALUE-MAXIMIZATION SECRETARY satisfies the smoothness desideratum described in section 2 for some parameter C. Then, we have that $\mathbf{E}[u_{\mathcal{A}}] \geq (1 - C \cdot \varepsilon)u_{i^*}$. Consequently, $(1-t^*)u_{i^*} + t^* \varepsilon \geq (1-C\cdot\varepsilon)u_{i^*} \xRightarrow{u_{i^*}=1} (1-t^*) + t^* \varepsilon \geq (1-C\cdot\varepsilon)$ which leads to contradiction when we let $\varepsilon \to 0$.

A.2 The PEGGING algorithm

In this subsection we generalize the ADDITIVE-PEGGING algorithm so as to provide fair and smooth algorithms for different prediction error definitions. We use the direct sum symbol $\oplus$ to abstractly denote an operation, e.g., addition or multiplication, and $\ominus$ for its inverse operation, i.e., subtraction or division. To model the "difference" between predictions and true values we use two functions $d^+ : \Re_{\geq 0} \times \Re_{\geq 0} \to \Re$ and $d^- : \Re_{\geq 0} \times \Re_{\geq 0} \to \Re$. The natural properties that we require from d^+, d^- with respect to operations $\oplus$ and $\ominus$ are described in assumption 6. We write $d_i^+ = d^+(u_i, \hat{u}_i)$, $d_i^- = d^-(u_i, \hat{u}_i)$ and define two prediction errors as follows: $\varepsilon^+ = \max_i d_i^+$ and $\varepsilon^- = \min_i d_i^-$. While the notation is left abstract to highlight the generality of theorem 7, we provide examples on how to instantiate those functions and operators for natural prediction errors as the absolute difference that we used in ADDITIVE-PEGGING and a multiplicative prediction error function that it is used by Fujii and Yoshida [25].

The only properties that we require from those functions with respect to operations $\oplus$ and $\ominus$ are described by assumption 6.

Assumption 6. *For all $u, \hat{u}, v, \hat{v}, x, y \in \Re_{\geq 0}$*

$$u \oplus d^+(u,\hat{u}) \geq \hat{u} \geq u \oplus d^-(u,\hat{u}) \quad (1)$$

$$d^+(u,\hat{u}) \geq d^+(v,\hat{v}) \Leftrightarrow d^-(u,\hat{u}) \leq d^-(v,\hat{v}) \quad (2)$$

$$x \oplus d^-(u,\hat{u}) \geq y \Leftrightarrow x \geq y \ominus d^-(u,\hat{u}) \textit{ and } x \oplus d^+(u,\hat{u}) \geq y \Leftrightarrow x \geq y \ominus d^+(u,\hat{u}) \quad (3)$$

$$x \ominus d^-(u,\hat{u}) \geq y \Leftrightarrow x \geq y \oplus d^-(u,\hat{u}) \textit{ and } x \ominus d^+(u,\hat{u}) \geq y \Leftrightarrow x \geq y \oplus d^+(u,\hat{u}) \quad (4)$$

The first property permits us to upper and lower bound the prediction of a candidate with respect to its true value, the second property relates the functions d^+ and d^- and the last two properties permit us to manipulate inequalities between true and predicted valuations.

Using the new abstract notation we describe how one can generalize ADDITIVE-PEGGING to PEGGING by only changing the conditions of *subcases 2a, 2b and 4b*. In *subcase 2a and 2b* we do not have a strong indication that $\hat{\imath}$ is the true highest value candidate. Thus we "peg" a future candidate only if it is possible to maintain smoothness, or equivalently, if there is a future candidate whose predicted value is higher than u_{i^*} *scaled down* by the prediction error. Thus, we change the condition of *subcase 2a* to $u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} > \max_{j \succ i} \hat{u}_j$ and *subcase 2b* in the same manner. In *subcase 4b*, we have a strong indication of $i \neq \hat{\imath}$ being the true highest value candidate and we accept if u_i *scaled up* by the prediction error is higher than $\hat{u}_{\hat{\imath}}$. Using our abstract notation this is equivalent to checking if $u_i \oplus d^+_i > \hat{u}_{\hat{\imath}}$.

Algorithm 5 PEGGING

while agent i arrives at time t_i **do**
  $\tau \leftarrow \max_{j \prec i} u_j$
  case 1: $i = i^{\text{pegged}}$
    $\mathcal{A} \leftarrow i$
  case 2: $i = \hat{\imath}$ **and** $(t_{\hat{\imath}} \leq 1/2$ **or** $u_{\hat{\imath}} < \tau)$
    subcase 2a: $u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} \geq \max_{j \succ i} \hat{u}_j$
      $\mathcal{A} \leftarrow \hat{\imath}$
    subcase 2b: $u_{\hat{\imath}} \oplus d^-_{i} < \max_{j \succ i} \hat{u}_j$
      $i^{\text{pegged}} \leftarrow \max_{\succ}\{j : u_{\hat{\imath}} \oplus d^-_{i} < \hat{u}_j\}$
  case 3: $i = \hat{\imath}$ **and** $t_{\hat{\imath}} > 1/2$ **and** $u_{\hat{\imath}} > \tau$
    $\mathcal{A} \leftarrow \hat{\imath}$
  case 4: $i \neq \hat{\imath}$ **and** $t_i > 1/2$ **and** $u_i > \tau$
    subcase 4a: $\hat{\imath}$ has already arrived
      $\mathcal{A} \leftarrow i$
    subcase 4b: $u_i \oplus d^+_i > \hat{u}_{\hat{\imath}}$
      $\mathcal{A} \leftarrow i$
end while

Note: $\max_{\succ}\{j : u_{\hat{\imath}} \oplus d^-_{i} < \hat{u}_j\}$ denotes the last candidate j arriving and having its prediction lower bounded by $u_{\hat{\imath}} \oplus d^-_{i}$. We cannot assign i^{pegged} at time $t_{\hat{\imath}}$ since we do not know when the last candidate with this property will arrive. We abuse notation $\leftarrow$ to say that as long as there is a future candidate with that property, we update i^{pegged}. We assume that every time a candidate arrives at most one case and one subcase is executed and the algorithm terminates whenever we execute a $\mathcal{A} \leftarrow i$ command, i.e., a candidate is accepted.

We proceed stating and proving Theorem 3.

Theorem 7. PEGGING *accepts the maximum value candidate with probability at least* $\frac{1}{16}$ *and its expected value is at least* $\max\{u_{i^*} \oplus \varepsilon^- \oplus \varepsilon^- \ominus \varepsilon^+ \ominus \varepsilon^+, \frac{1}{16}u_{i^*}\}$

Proof. We first lower bound $\hat{u}_{\hat{\imath}}$ using the right-hand side of (1) from assumption 6 and the fact that $\hat{\imath}$ is the candidate with the maximum predicted valuation: $\hat{u}_{\hat{\imath}} \geq \hat{u}_{i^*} \geq u_{i^*} \oplus d^-_{i^*}$.

We also lower bound $u_{\hat{\imath}}$ as follows: from the left-hand side of (1) from assumption 6 we get that $u_{\hat{\imath}} \oplus d^+_{\hat{\imath}} \geq \hat{u}_{\hat{\imath}} \Rightarrow u_{\hat{\imath}} \geq \hat{u}_{\hat{\imath}} \ominus d^+_{\hat{\imath}}$. Thus, also using the lower bound $\hat{u}_{\hat{\imath}} \geq u_{i^*} \oplus d^-_{i^*}$ we have:

$$\begin{aligned} u_{\hat{\imath}} &\geq \hat{u}_{\hat{\imath}} \ominus d^+_{\hat{\imath}} \\ &\geq u_{i^*} \oplus d^-_{i^*} \ominus d^+_{\hat{\imath}} \end{aligned}$$

In *subcase 2a* and *case 3* candidate $\hat{\imath}$ is accepted and in *subcase 4a* a candidate i with $u_i \geq u_{\hat{\imath}}$ is accepted. Consequently, all these cases accept a candidate with value at least $u_{\hat{\imath}} \geq u_{i^*} \oplus d^-_{i^*} \ominus d^+_{\hat{\imath}}$.

If i is accepted by *subcase 4b* we get that $u_i \oplus d^+_i > \hat{u}_{\hat{\imath}} \Rightarrow u_i > \hat{u}_{\hat{\imath}} \ominus d^+_i$ and again using that $\hat{u}_{\hat{\imath}} \geq u_{i^*} \oplus d^-_{i^*}$ we have $u_i > u_{i^*} \oplus d^-_{i^*} \ominus d^+_i$.

Finally, we need to lower bound the value of a pegged candidate in case our algorithm terminates accepting i^{pegged}. Note that from the way a pegged candidate is defined in *subcase 2b*, we always have $u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} < \hat{u}_{i^{\text{pegged}}}$ and again using the left-hand side of (1) from assumption 6 we have that

$\hat{u}_{\imath^{\text{pegged}}} \leq u_{\imath^{\text{pegged}}} \oplus d^+_{\imath^{\text{pegged}}} \Rightarrow u_{\imath^{\text{pegged}}} \geq \hat{u}_{\imath^{\text{pegged}}} \ominus d^+_{\imath^{\text{pegged}}}$. Thus, we get that $u_{\imath^{\text{pegged}}} > u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} \ominus d^+_{\imath^{\text{pegged}}}$ and using the lower bound on $u_{\hat{\imath}}$ we conclude that $u_{\imath^{\text{pegged}}} > u_{i^*} \oplus d^-_{i^*} \ominus d^+_{\hat{\imath}} \oplus d^-_{\hat{\imath}} \ominus d^+_{\imath^{\text{pegged}}}$.

Combining all the lower bounds on the valuation of the accepted candidate deduce the first part of the lower bound.

We proceed proving fairness and, consequently robustness. We denote by $\tilde{\imath}$ the index of the candidate with the highest true valuation except possibly i^* and $\hat{\imath}$, i.e., $\tilde{\imath} = \operatorname{argmax}_{i \neq i^*, \hat{\imath}} u_i$. Note that if $i^* = \hat{\imath}$ then $\tilde{\imath}$ denotes the index of the candidate with the second highest true valuation. To prove fairness we distinguish between two cases: either $\hat{\imath} = i^*$ or $\hat{\imath} \neq i^*$. For each of those cases, we define an event and argue that: (1) the event happens with constant probability; and (2) if that event happens then Algorithm 5 accepts i^*.

If $i^* = \hat{\imath}$ we define event $C = \{t_{\tilde{\imath}} < 1/2 < t_{i^*}\}$ for which $P[C] = 1/4$. C implies that our algorithm does not accept any candidate until time t_{i^*}. Indeed note that the conditions of *case 1, 2, and 3* all happen after the arrival of $\hat{\imath}$ which is t_{i^*}. In addition to that after time $1/2$ the threshold $\tau = u_{\tilde{\imath}}$ is larger than any valuation other than u_{i^*} and consequently the conditions of *case 4* are not met before t_{i^*}. At time t_{i^*} all conditions of *case 4* and *subcase 4a* are met and our algorithm hires i^*.

If $i^* \neq \hat{\imath}$ we further distinguish between two cases. To that end, note that $\{u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} < \hat{u}_{i^*}\} \vee \{d^+_{i^*} \oplus u_{i^*} > \hat{u}_{\hat{\imath}}\}$ is always true. Indeed if $\{u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} > \hat{u}_{i^*}\}$ we have that:

$$
\begin{aligned}
u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} &> \hat{u}_{i^*} \\
u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} &> u_{i^*} \oplus d^-_{i^*} \\
d^-_{\hat{\imath}} \ominus d^-_{i^*} &> u_{i^*} \ominus u_{\hat{\imath}} \\
d^-_{\hat{\imath}} \ominus d^-_{i^*} &> \mathbf{0} \\
d^-_{\hat{\imath}} &> d^-_{i^*} \\
d^+_{\hat{\imath}} &< d^+_{i^*}
\end{aligned}
$$

where $\mathbf{0}$ is the neutral element associated with operation $\oplus$, from the first to the second line we used the right-hand side of (1) from assumption 6, from the third to the fourth line we used that $u_{i^*} \geq u_{\hat{\imath}}$ by the definition of i^* and from the fifth to the sixth line we used (2) of assumption 6. Consequently:

$$
d^+_{i^*} \oplus u_{i^*} > d^+_{\hat{\imath}} \oplus u_{i^*} \geq d^+_{\hat{\imath}} \oplus u_{\hat{\imath}} \geq \hat{u}_{\hat{\imath}}
$$

where we again used that $u_{i^*} \geq u_{\hat{\imath}}$ and in the last inequality we used the right-hand side of (1) from assumption 6.

Thus, we define two events C_1 and C_2 which imply that i^* is always accepted whenever $\{u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} < \hat{u}_{i^*}\}$ and $\{d^+_{i^*} \oplus u_{i^*} > \hat{u}_{\hat{\imath}}\}$ are true respectively.

If $\{u_{\hat{\imath}} \oplus d^-_{\hat{\imath}} < \hat{u}_{i^*}\}$ is true then we define event $C_1 = \{t_{\tilde{\imath}} < 1/2\} \wedge \{t_{\hat{\imath}} < 1/2\} \wedge \{1/2 < t_{i^*}\}$ which is composed by 3 independent events and it happens with probability $P[C_1] = 1/2^3 = 1/8$. C_1 implies that at time t_{i^*} the conditions of *subcase 4a* are satisfied. Consequently, if until time t_{i^*} all candidates are rejected then candidate i^* is hired using either *case 1* or *subcase 4a*. To argue that no candidate is hired before time t_{i^*} note that at time $t_{\hat{\imath}}$ the set $\max_{\succ}\{j : u_{\hat{\imath}} < \hat{u}_j + \epsilon_{\hat{\imath}}\}$ contains i^* and that after time $1/2$ the condition $\tau > u_i$ is not met until time t_{i^*}.

If $\{d^+_{i^*} \oplus u_{i^*} > \hat{u}_{\hat{\imath}}\}$ is true then we define $C_2 = \{t_{\tilde{\imath}} < 1/2 < t_{i^*} < t_{\hat{\imath}}\}$ which happens with probability

$$
\begin{aligned}
P[C_2] &= P[t_{\tilde{\imath}} < 1/2] \cdot P[1/2 < t_{i^*} < t_{\hat{\imath}}] = \\
&= P[t_{\tilde{\imath}} < 1/2] \cdot P[1/2 < \min\{t_{i^*}, t_{\hat{\imath}}\} \wedge \min\{t_{i^*}, t_{\hat{\imath}}\} = t_{i^*}] = \\
&= P[t_{\tilde{\imath}} < 1/2] \cdot P[1/2 < \min\{t_{i^*}, t_{\hat{\imath}}\}] \cdot P[\min\{t_{i^*}, t_{\hat{\imath}}\} = t_{i^*}] = \\
&= (1/2) \cdot (1/4) \cdot (1/2) = \\
&= 1/16
\end{aligned}
$$

Note that until time t_{i^*} no candidate is accepted since the conditions of all cases are not satisfied. Indeed, between times 0 and $1/2$ only $\hat{\imath}$ could have been accepted but its arrival time is after t_{i^*}, and between times $1/2$ and t_{i^*} the threshold τ is equal to $u_{\tilde{\imath}}$ and no candidate meets the condition of *case 4* to have $u_i > \tau$. Finally, note that at time t_{i^*} the conditions of *case 4b* are satisfied and i^* gets accepted. □

Note that by instantiating $\oplus, \ominus$ to the usual scalar addition and subtraction and $d^+(u, \hat{u}) = |u - \hat{u}|$, $d^-(u, \hat{u}) = -|u - \hat{u}|$ we get that $d_i^+ = \epsilon_i$, $d_i^- = -\epsilon_i$. Consequently, $\varepsilon^+ = \max_i \epsilon_i$ and $\varepsilon^- = -\max_i \epsilon_i$. Substituting ε^+ and ε^- to the guarantees of theorem 7 we recover theorem 3.

Moreover, in theorem 4, we further demonstrate the generality of PEGGING by instantiating $\oplus, \ominus, d^+, d^-$ and recovering similar smoothness and robustness bounds as [25] while also ensuring fairness. Fujii and Yoshida [25] define the prediction error as $\varepsilon = \max_i |1 - \hat{u}_i/u_i|$ and design an algorithm which accepts a candidate i whose expected value is at least $u_{i^*} \max\{(1-\varepsilon)/(1+\varepsilon), 0.215\}$. Since $(1-\varepsilon)/(1+\varepsilon) \geq 1 - 2\,\varepsilon$ the latter algorithm satisfies the Smoothness desideratum of section 2, but, as we prove in appendix A.1 it violates the Fairness desideratum.

To that end, let MULTIPLICATIVE-PEGGING be the instantiation of PEGGING when $\oplus, \ominus$ denote the classical multiplication and division operations, $d^+(u, \hat{u}) = 1 + |1 - \hat{u}/u|$ and $d^-(u, \hat{u}) = \max\{1 - |1 - \hat{u}/u|, 0\}$.

Theorem 4. *If* $\varepsilon(\mathcal{I}) = \max_i |1 - \hat{u}_i/u_i|$*, then* MULTIPLICATIVE-PEGGING *satisfies smoothness and fairness with* $C = 4$ *and* $F = 1/16$*.*

Proof. Functions d^+, d^- satisfy the properties of assumption 6. Note that $\varepsilon^+ = 1 + \varepsilon$ and $\varepsilon^- = \max\{1 - \varepsilon, 0\}$. Consequently, using theorem 7 MULTIPLICATIVE-PEGGING accepts a candidate whose expected value is at least $u_{i^*} \max\left\{(1-\varepsilon)^2/(1+\varepsilon)^2, 1/16\right\} \geq u_{i^*} \max\{1 - 4\,\varepsilon, 1/16\}$, where we used the inequalities $1/(1+\varepsilon) \geq (1-\varepsilon)$ and $(1-\varepsilon)^4 \geq 1 - 4\,\varepsilon$. □

B Missing analysis for the k-secretary pegging algorithm

In Lemma 8 and Corollary 9 we prove that K-PEGGING satisfies the smoothness desideratum.

Lemma 8. $\sum_{j \in S} u_j \geq \sum_{i=1}^{k} u_{r_i} - 4k\,\varepsilon(\mathcal{I})$ *,* $\forall \mathcal{I}$ *with probability* 1*.*

Proof. Similarly to the single choice secretary problem we proceed in two steps, first we prove that:

$$\sum_{i \in [k]} u_{r_i} - \sum_{i \in [k]} u_i \leq 2k\,\varepsilon$$

and then we prove that

$$\sum_{i \in [k]} u_i - \sum_{j \in S} u_j \leq 2k\,\varepsilon$$

Note that combining those two inequalities is enough to prove the current lemma.

The first inequality is proven as follows:

$$\sum_{i=1}^{k} u_{r_i} \leq_{(1)} \sum_{i=1}^{k} (\hat{u}_{r_i} + \varepsilon) \leq_{(2)} k\,\varepsilon + \sum_{i=1}^{k} \hat{u}_i \leq_{(3)} k\,\varepsilon + \sum_{i=1}^{k} (u_i + \varepsilon) = 2k\,\varepsilon + \sum_{i=1}^{k} u_i$$

where (1) and (3) are by definition of ε and (2) since $\hat{u}_i$ is i^{th} largest predicted value.

We proceed to argue that:

$$\sum_{i \in [k]} u_i - \sum_{j \in S} u_j \leq 2k\,\varepsilon$$

We now define an injective function $m : [k] \to S$ for which we have:

$$u_i - u_{m(i)} \leq 2\,\varepsilon, \ \forall i \in [k]$$

Note that the existence of such a function implies the desired $\sum_{i\in[k]} u_i - \sum_{j\in H} u_j \le 2k\,\varepsilon$ inequality.

Note that each candidate $j' \in [k]$ is initially added to H. During the execution of our algorithm candidate j' may be either (1) deleted from H without being added to B or (2) added to B.

In the first case, j' is deleted from H without being added to B, which occurs either in case $2a, 3$, or $4b$ of the algorithm. Let i be the current candidate at the time t_i where the latter happens. If *case 2a or 3* happens then $j' = i$, we define $m(j') = j'$ and $u_{j'} - u_{m(j')} = 0 \le 2\,\varepsilon$. If *case 4b* happens then we have that at that time t_i: $j' \in \{j \in H : u_i > \hat{u}_j - \varepsilon_{t_i}\}$ and we define $m(i) = j'$. Consequently, we conclude that $u_i - u_{m(i)} = u_i - u_{j'} \le u_i - \hat{u}_{j'} + \varepsilon_{t_{j'}} \le \varepsilon_i + \varepsilon_{t_{j'}} \le 2\,\varepsilon$.

We now consider the cases where j' is added to B during the execution of our algorithm. Note that for that to happen j must be added to B at time t_j via *case 2b*. In that case, candidate $peg(j')$ either remains in P until time $t_{peg(j')}$ and it is added to S at that time or it is deleted from P earlier. In both cases j' is removed from B at the respective time. Thus, we conclude that j' gets deleted from B at time $t_{peg(j')}$ or before. If the deletion happens at time $t_{peg(j)}$ then it must happen through *case 1*, we define $m(j') = peg(j')$ and we have that $u_{j'} - u_{m(j')} = u_{j'} - u_{peg(j')} \le \hat{u}_{peg(j')} + \varepsilon_{t_{j'}} - u_{peg(j')} = (\hat{u}_{peg(j')} - u_{peg(j')}) + \varepsilon_{t_{j'}} \le \varepsilon_{peg(j')} + \varepsilon_{t_{j'}} \le 2\,\varepsilon$, where in the first inequality we used that $\hat{u}_{peg(j')} > u_{j'} - \varepsilon_{t_{j'}}$ since $peg(j')$ was pegged by j' at time $t_{j'}$. If j' is deleted from B before time $t_{peg(j')}$ it must happen via *subcase 4a* due to the arrival of a candidate l that is added to S. In that case we define $m(j') = l$ and from the condition of *subcase 4a* we have that $j \in E$ at time $t_{j'}$ we have that $u_{j'} - u_{m(j')} = u_{j'} - u_l < 0 \le 2\,\varepsilon$. □

Corollary 9. *If* $u_{r_i} \ge 1\ \forall i \in \{1, ..., k\}$ *then:*

$$\sum_{j\in S} u_j \ge \sum_{i=1}^{k} u_{r_i}(1 - 4\,\varepsilon(\mathcal{I}))\ ,\forall \mathcal{I} \textit{ with probability } 1.$$

Proof. The proof follows from lemma 8 and noting that $u_{r_i} \ge 1$ implies that $\sum_{i=1}^{k} u_{r_i} \ge k$ and consequently $\sum_{i=1}^{k} u_{r_i} - 4k\,\varepsilon(\mathcal{I}) \ge \sum_{i=1}^{k} u_{r_i}(1 - 4\,\varepsilon(\mathcal{I}))$. □

We now move to prove the fairness desideratum.

Lemma 10. *For all* $i \in [k]$*:* $P[r_i \in S] \ge (1/3)^{k+5}$

Proof. We start arguing that if $r_i \in [k]$, i.e., if candidate r_i is among the k-highest prediction candidates, then $P[r_i \in S] \ge 1/4$. We define event $\mathcal{C} = \{t_{r_i} < 1/2\} \wedge \{C_{r_i} = 0\}$ for which $P[\mathcal{C}] = P[t_{r_i} < 1/2] \cdot P[C_{r_i} = 0] = (1/2) \cdot (1/2) = 1/4$ Note that at time 0 we have that $r_i \in H$. During the execution of Algorithm 2 from time 0 to time t_{r_i}, candidate r_i may be removed from H without being added from S only if $t_{r_i} \ge 1/2$. Consequently, $\mathcal{C}$ implies that at time $t_{r_i} < 1/2$ the conditions of *case 2* and *subcase 2a* are true and candidate r_i is added to S. Thus, $P[r_i \in S] \ge P[\mathcal{C}] = 1/4$.

In the rest of the proof, we focus on the case where $r_i \notin [k]$. Note that since $i \in [k]$ and $r_i \notin [k]$ then $\exists j \in [k]$ such that $u_{r_i} \ge u_{r_k} > u_j$, i.e., j has a true value which is not among the k-highest true values. We now argue that $\{u_j < \hat{u}_{r_i} + \varepsilon_j\} \vee \{u_{r_i} > \hat{u}_j - \varepsilon_{r_i}\}$ is always true. Similarly to the proof of lemma 2, assume towards a contradiction that both inequalities can be inverted and hold at the same time, then we end up in a contradiction as follows:

$$u_j \ge \hat{u}_{r_i} + \varepsilon_j \xrightarrow{u_{r_i} > u_j} u_{r_i} > \hat{u}_{r_i} + \varepsilon_j \Rightarrow u_{r_i} - \hat{u}_{r_i} > \varepsilon_j \xrightarrow{\varepsilon_{r_i} \ge u_{r_i} - \hat{u}_{r_i}} \varepsilon_{r_i} > \varepsilon_j$$
$$u_{r_i} \le \hat{u}_j - \varepsilon_{r_i} \xrightarrow{u_j < u_{r_i}} u_j < \hat{u}_j - \varepsilon_{r_i} \Rightarrow \varepsilon_{r_i} < \hat{u}_j - u_j \xrightarrow{\varepsilon_j \ge \hat{u}_j - u_j} \varepsilon_{r_i} < \varepsilon_j$$

For each of those cases, i.e., whether $\{u_j < \hat{u}_{r_i} + \varepsilon_j\}$ or $\{u_{r_i} > \hat{u}_j - \varepsilon_{r_i}\}$ is true we define an event which implies that r_i is added to the solution set S.

If $\{u_{r_i} > \hat{u}_j - \varepsilon_{r_i}\}$ is true then we define the following event:

$$\mathcal{C} = \bigwedge_{(l):l\in[k+2]\setminus\{r_i\cup j\}} \{t_l < 1/2\} \wedge \{1/2 < t_{r_i} < t_j\}$$

Note that:

$$\begin{aligned}
P[\mathcal{C}] &= P[1/2 < t_{r_i} < t_j] \cdot \prod_{(l):l\in[k+2]\setminus\{r_i\cup j\}} P[t_l < 1/2] \\
&= P[1/2 < \min\{t_{r_i}, t_j\} \wedge \{t_{r_i} < t_j\}] \cdot \prod_{(l):l\in[k+2]\setminus\{r_i\cup j\}} (1/2) \\
&= P[1/2 < \min\{t_{r_i}, t_j\}] \cdot P[t_{r_i} < t_j] \cdot \prod_{(l):l\in[k+2]\setminus\{r_i\cup j\}} (1/2) \\
&\geq P[1/2 < \min\{t_{r_i}, t_j\}] \cdot P[t_{r_i} < t_j] \cdot (1/2)^{k+2} \\
&= (1/4) \cdot (1/2) \cdot (1/2)^{k+2} \\
&= (1/2)^{k+5}
\end{aligned}$$

We now argue that $\mathcal{C}$ implies that r_i is added to S.

The first literal of $\mathcal{C}$ ensures that $\tau \geq u_{r_{k+1}}$ after time $1/2$ and, consequently, the only candidate with a true value higher than the threshold τ at time t such that $1/2 \leq t \leq t_{r_i}$ is r_i. That observation implies that conditions of *case 4* are true only for r_i.

We now argue that: (1) before time t_{r_i} less than k candidates are added to S; and (2) the conditions of *subcase 4b* are true for r_i at time t_{r_i}.

For (1) first note that: (a) initially we have $|S| = |B| = 0$, $|H| = k$; (b) at all times our algorithm maintain the invariant $B \cap H = \emptyset$, $B \cup H \subseteq [k]$; and (c) every time a candidate is added to the solution S then a candidate is deleted from either B, as in *case 1* and *subcase 4a*, or H as in *subcase 2a*, *case 3* and *subcase 4b*. Thus, at all times $|B| + |H| + |S|$ remains constant and since initially is equal to k we conclude that at all times $|B| + |H| + |S| = k$. In addition, a candidate not yet arrived may be removed from H only through *subcase 4b*. Since we argued that conditions of *case 4* are true only for r_i, we have that right before r_i's arrival $j \in H$ and $|S| = k - |B| - |H| \leq k - |H| \leq k - 1 < k$. For (2), to argue that conditions of *subcase 4b* are met at time t_{r_i}, it is enough to prove that $j \in \{j' : u_{r_i} > \hat{u}_{j'} - \varepsilon_{t_{r_i}}\}$. To see that note that by the definition of $\varepsilon_{t_{r_i}}$ it holds that $\varepsilon_{t_{r_i}} \geq \varepsilon_{r_i}$ and consequently, $\{j' : u_{r_i} > \hat{u}_{j'} - \varepsilon_{t_{r_i}}\} \supseteq \{j' : u_{r_i} > \hat{u}_{j'} - \varepsilon_{r_i}\} \ni j$.

Before proceeding to the second case we introduce the following notation:

$$t_{peg(j)} = \begin{cases} t_l & \text{if } \exists l : l = peg(j) \\ \infty & \text{otherwise} \end{cases}$$

that is, if at time t_j candidate l is added to the pegging set P then we use $t_{peg(j)}$ to denote the arrival time of candidate l. However, if at time t_j no candidate is added to set P then we define $t_{peg(j)}$ to be equal to ∞ so that the literal $\{t_{peg(j)} > x\}$ is true for every $x \in \Re$.

We now analyze the case where $\{u_j < \hat{u}_{r_i} + \varepsilon_j\}$ is true and define the following event:

$$\mathcal{C} = \bigwedge_{l\in[k+1]\setminus\{r_i,j\}} \{t_{r_l} < 1/3\} \wedge \{1/3 < t_j < 1/2\} \wedge \{t_{r_i} > 1/2\} \wedge \{C_j = 1\} \wedge \{t_{peg(j)} \geq t_{r_i}\}$$

To simplify notation, let P_j be the random variable denoting the pegging set at time t_j before the execution of the while loop because of j's arrival. We let $F_j = \{j' \succ j : u_j < \hat{u}_{j'} + \varepsilon_{t_j}\} \setminus (P_j \cup [k])$ be the random variable which contains all candidates that could be "pegged" at time t_j.

In addition we define event $\mathcal{T}$ as follows:

$$\mathcal{T} = \bigwedge_{l\in[k+1]\setminus\{r_i,j\}} \{t_{r_l} < 1/3\} \wedge \{1/3 < t_j < 1/2\} \wedge \{t_{r_i} > 1/2\} \wedge \{C_j = 1\}$$

Before lower bounding the probability of event $\mathcal{C}$ we argue that:

$$P\big[t_{peg(j)} \geq t_{r_i} \mid \mathcal{T}\big] \geq 2/3$$

Let $\mathcal{F}_j$ denote the set of all non-empty subsets of $[n]$ such that $P[F_j = f_j \mid \mathcal{T}] > 0$. Note that $P[F_j = \emptyset \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j} P[F_j = f_j \mid \mathcal{T}] = 1$. In addition, to alleviate notation we denote $l_{f_j} = \operatorname{argmin}_{j' \in f_j} \hat{u}_{j'}$.

From the law of total probability we have:

$$\begin{aligned}
&P\big[t_{peg(j)} \geq t_{r_i} \mid \mathcal{T}\big] && (1)\\
&= P\big[\{t_{peg(j)} \geq t_{r_i}\} \wedge \{F_j = \emptyset\} \mid \mathcal{T}\big] + P\big[\{t_{peg(j)} \geq t_{r_i}\} \wedge \{F_j \neq \emptyset\} \mid \mathcal{T}\big] && (2)\\
&= P[F_j = \emptyset \mid \mathcal{T}] + P\big[\{t_{peg(j)} \geq t_{r_i}\} \wedge \{F_j \neq \emptyset\} \mid \mathcal{T}\big] && (3)\\
&= P[F_j = \emptyset \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j} P\big[\{t_{peg(j)} \geq t_{r_i}\} \wedge \{F_j = f_j\} \mid \mathcal{T}\big] && (4)\\
&= P[F_j = \emptyset \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j} P\big[t_{peg(j)} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\big] \cdot P[F_j = f_j \mid \mathcal{T}] && (5)\\
&= P[F_j = \emptyset \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j} P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] && (6)
\end{aligned}$$

where from (2) to (3) we use that if $F_j = \emptyset$ then the condition of *subcase 2b* is false, thus no candidate is "pegged" and consequently $t_{peg(j)} = \infty$. From (3) to (4) we used that $\{F_j \neq \emptyset\} = \bigvee_{f_j \in \mathcal{F}_j} \{F_j = f_j\}$. From (5) to (6) we used that event $\{F_j = f_j\} \wedge \mathcal{T}$ implies that conditions of *case 2* and *subcase 2b* are true and consequently $peg(j) = l_{f_j}$ by the definition of l_{f_j}.

We now focus on lower bounding the summation term

$$\begin{aligned}
&\sum_{f_j \in \mathcal{F}_j} P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] =\\
&\sum_{f_j \in \mathcal{F}_j : l_{f_j} = r_i} P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] +\\
&+ \sum_{f_j \in \mathcal{F}_j : l_{f_j} \neq r_i} P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] =\\
&\sum_{f_j \in \mathcal{F}_j : l_{f_j} = r_i} P[t_{r_i} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}] \cdot P[F_j = f_j \mid \mathcal{T}] +\\
&+ \sum_{f_j \in \mathcal{F}_j : l_{f_j} \neq r_i} P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] =\\
&\sum_{f_j \in \mathcal{F}_j : l_{f_j} = r_i} 1 \cdot P[F_j = f_j \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j : l_{f_j} \neq r_i} P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}]
\end{aligned}$$

We proceed lower bounding the term $P\Big[t_{l_{f_j}} \geq t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big]$ for all $f_j \in \mathcal{F}_j : l_{f_j} \neq r_i$. Note that the conditioning $\{F_j = f_j\} \wedge \mathcal{T}$ changes the distribution of random variables $t_{l_{f_j}}, t_{r_i}$ as follows: t_{r_i} is uniformly drawn from $[1/2, 1]$ and $t_{l_{f_j}}$ is uniformly drawn from $[z, 1]$ for some $z \in [1/3, 1/2]$ which equals the realization of the random variable t_j. We define a random variable $\tilde{t}_{l_{f_j}}$ which is stochastically dominated by $t_{l_{f_j}}$ and is drawn uniformly from $[1/3, 1]$ as follows: let $\tilde{t}$ be uniformly drawn from $[1/3, z]$ and $B \sim Bernoulli((z - 1/3)/(1/2 - 1/3))$ then we define:

$$\tilde{t}_{l_{f_j}} = B \cdot \tilde{t} + (1 - B) \cdot t_{l_{f_j}}$$

Note that since $\tilde{t} \le t_{l_{f_j}}$ then also $\tilde{t}_{l_{f_j}} \le t_{l_{f_j}}$ holds almost surely.

Therefore we have:

$$\begin{aligned} P\Big[t_{l_{f_j}} \ge t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] &\ge P\Big[\tilde{t}_{l_{f_j}} \ge t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \\ &\ge 2/3 \end{aligned}$$

and we proceed bounding the initial summation as follows:

$$\begin{aligned} &\sum_{f_j \in \mathcal{F}_j} P\Big[t_{l_{f_j}} \ge t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] = \\ &= \sum_{f_j \in \mathcal{F}_j : l_{f_j} = r_i} 1 \cdot P[F_j = f_j \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j : l_{f_j} \neq r_i} P\Big[t_{l_{f_j}} \ge t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] \\ &\ge \sum_{f_j \in \mathcal{F}_j : l_{f_j} = r_i} 1 \cdot P[F_j = f_j \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j : l_{f_j} \neq r_i} (2/3) \cdot P[F_j = f_j \mid \mathcal{T}] \\ &\ge (2/3) \sum_{f_j \in \mathcal{F}_j} P[F_j = f_j \mid \mathcal{T}] \end{aligned}$$

We then have:

$$\begin{aligned} &P\big[t_{peg(j)} \ge t_{r_i} \mid \mathcal{T}\big] && (7) \\ &= P[F_j = \emptyset \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j} P\Big[t_{l_{f_j}} \ge t_{r_i} \mid \{F_j = f_j\} \wedge \mathcal{T}\Big] \cdot P[F_j = f_j \mid \mathcal{T}] && (8) \\ &\ge P[F_j = \emptyset \mid \mathcal{T}] + (2/3) \sum_{f_j \in \mathcal{F}_j} P[F_j = f_j \mid \mathcal{T}] && (9) \\ &\ge (2/3) \cdot \left(P[F_j = \emptyset \mid \mathcal{T}] + \sum_{f_j \in \mathcal{F}_j} P[F_j = f_j \mid \mathcal{T}] \right) && (10) \\ &= 2/3 && (11) \end{aligned}$$

We are now ready to lower bound the probability of event $\mathcal{C}$ as follows

$$\begin{aligned} P[\mathcal{C}] &= P[\mathcal{T}] \cdot P\big[t_{peg(j)} \ge t_{r_i} \mid \mathcal{T}\big] \\ &\ge P[\mathcal{T}] \cdot (2/3) \\ &= \prod_{(l): l \in [k+1] \setminus \{r_i, j\}} P[t_l < 1/3] \cdot P[1/3 < t_j < 1/2] \cdot (2/3) \\ &\ge (1/3)^k \cdot (1/2 - 1/3) \cdot (2/3) \\ &= (1/3)^{k+2} \end{aligned}$$

Similar to the analysis of the first case the first literal of $\mathcal{C}$ ensures that the only candidate which may be accepted after time $1/2$ without being at any point in time in the pegging set P is r_i. In addition, since $t_j < 1/2$ then we have that j remains in H until at least time t_j. Indeed, a candidate in H that has not arrived yet may be removed from set H only through *case 4b* which happens exclusively after time $1/2$. We now analyze $\mathcal{C}$'s implications regarding the execution of our algorithm at j's arrival by distinguishing between two mutually exclusive cases, that is whether r_i is in P before time t_j or not.

If r_i is not in the pegging set exactly before time t_j then we have that the conditions of *case 2b* are true. Indeed note that $\{C_j = 1\}$ is a literal of $\mathcal{C}$ and since $\varepsilon_j \le \varepsilon_{t_j}$ we have:

$$\{j' \succ j : u_j < \hat{u}_{j'} + \varepsilon_{t_j}\} \setminus (P \cup [k]) \supseteq \{j' \succ j : u_j < \hat{u}_{j'} + \varepsilon_j\} \setminus (P \cup [k]) \ni r_i$$

Thus, we are in the case where at time t_j a candidate (which may be r_i) is added to the pegging set, candidate j is added to B and $t_{peg(j)} < \infty$. Due to the literal $t_{peg(j)} \geq t_{r_i}$ of $\mathcal{C}$ and the fact that the only candidate which may be accepted after time $1/2$ without being at any point in time in the pegging set P is r_i, we can deduce that at time t_{r_i} j is still in B. Thus, at time t_{r_i} the conditions of *subcase 4a* are true and r_i is added to S. If r_i is in the pegging set exactly before time t_j then since the conditions of *case 4* are false for any candidate except possibly r_i we can deduce that at time t_{r_i}, candidate r_i is still in the pegging set P and is added to the solution through *case 1*.

Combining all the different lower bounds on $P[r_i \in S]$ we conclude the lemma. □

Lemma 11. *For all* $i \in \{1, 2, \dots, k\}$*:* $P[r_i \in S] \geq \frac{1-\frac{i+13}{k}}{256}$

Proof. Let $\delta' > 12/k$ be such that $i < (1-\delta')k - 1$. We now argue that proving $P[r_i \in S] \geq \delta'/256$ suffices to prove the lemma.

First, we underline that such a δ' exists only for $i < k - 13$. Indeed,

$$i < (1-\delta')k - 1 \Rightarrow \delta' < 1 - \frac{i+1}{k} \xRightarrow{\delta' > 12/k} i < k - 13$$

For all $i < k - 13$ we have:

$$P[r_i \in S] \geq \delta'/256 > \frac{1-\frac{i+1}{k}}{256} > \frac{1-\frac{i+13}{k}}{256}$$

For $i \geq k - 13$ the statement of the lemma is vacuous, since:

$$P[r_i \in S] \geq 0 \geq \frac{1-\frac{i+13}{k}}{256}$$

Consequently, from now on we focus on proving that $P[r_i \in S] \geq \delta'/256$. We do so by defining an event $\mathcal{C}$ for which $P[\mathcal{C}] \geq \delta'/256$ and argue that $\mathcal{C}$ implies r_i being accepted.

Before defining $\mathcal{C}$ we need to introduce some auxiliary notation. We call replacement set and denote by R the set of indexes initially in H with value lower than u_{r_i}, i.e., $R = \{j : u_{r_i} > u_j\} \cap [k]$ and by $j^{worse} = \text{argmax}_{j \in R} \varepsilon_j$ the index of the candidate with the highest error in R. For any $t, t' \in [0,1]$ we define the random variable $A_{t,t'} = \{j : t \leq t_j \leq t'\} \setminus \{r_i, j^{worse}\}$ which contains all indexes except r_i and j^{worse} of candidates arrived between times t and t'. Also, for $x \in [n]$ we define the set function $L_x : 2^{[n]} \to 2^{[n]}$, such that for any subset $Y \subseteq [n]$, $L_x(Y)$ contains the x indexes with highest true value in Y. For $\delta \in (0, 1/2)$ let (a) $R_1 = R \cap A_{0,1/2+\delta}$ and $R_2 = R \cap A_{1/2+\delta,1}$ be the random variables denoting all candidates of $R \setminus \{j^{worse}\}$ arriving before and after time $1/2 + \delta$ respectively; and (b) let $M = L_{\lfloor (1+4\delta)k \rfloor}(A_{0,1/2+\delta}) \cap A_{1/2,1/2+\delta}$ denote the random variable containing candidates which arrived between times $1/2$ and $1/2 + \delta$ with the $\lfloor (1+4\delta)k \rfloor$ higher true value among the ones arrived before time $1/2 + \delta$ (excluding r_i and j^{worse}).

We now define event $\mathcal{C}$.

$$\mathcal{C} = \{|R_2| - |M| > 1\} \wedge \{1/2 < t_{r_i} < 1/2 + \delta\} \wedge \{t_{j^{worse}} < t_{r_i}\}$$

Note that M contains all candidates not in H that may be added to our solution between times $1/2$ and $1/2 + \delta$. In addition, each candidate in M through *subcase 4b* may delete from H at most one candidate with arrival time after $1/2 + \delta$.

Consequently, the number of candidates in R that are in H until time $1/2+\delta$ is at least $|R_2| - |M| > 1$. We now argue that conditions of *subcase 4b* are true at time t_{r_i}. Indeed, note that since $t_{j^{worse}} < t_{r_i}$ we have that $\varepsilon_{t_{r_i}} > \varepsilon_{j^{worse}}$ and consequently, for every candidate j of R we have that

$$u_{r_i} > u_j \geq \hat{u}_j - \varepsilon_j \geq \hat{u}_j - \varepsilon_{j^{worse}} \geq \hat{u}_j - \varepsilon_{t_{r_i}}$$

where the first inequality comes from the definition of set R, the second from the definition of the error, the third from the definition of j^{worse} and the last from the fact that $\varepsilon_{t_{r_i}} > \varepsilon_{j^{worse}}$. Since $R_2 \subseteq R$, we have that at time t_{r_i} there is at least one candidate j in H for which $u_{r_i} \geq \hat{u}_j - \varepsilon_{t_{r_i}}$ and the conditions of *subcase 4a* are therefore true.

We now proceed lower bounding the probability of event $\mathcal{C}$. Let $\delta = \delta'/16$ we first argue that $\{|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\}$ and $|M| < 4\delta k$ implies that $|R_2| - |M| > 1$. Indeed note that:

$$
\begin{aligned}
|R_2| - |M| &> \frac{1/2-2\delta}{1-\delta}\delta' k - 4\delta k \\
&> \frac{1}{3}\delta' k - 4\delta k \\
&> (\delta'/3 - 4\delta)k \\
&\geq (\delta'/3 - 4\delta'/16)k \\
&\geq (\delta'/12)k \\
&> 1
\end{aligned}
$$

where in the second inequality we used that $\delta = \delta'/16 \leq 1/16 < 1/10$.

We now lower bound the probability of the intersection of events $\{|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\}$, $\{|M| < 4\delta k\}$, $\{1/2 < t_{r_i} < 1/2+\delta\}$ and $\{t_{j^{worse}} < 1/2\}$ since their intersection implies event $\mathcal{C}$.

Since R_2 and M do not contain neither r_i nor j^{worse} we have that events $\{|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\}$, $\{|M| < 4\delta k\}$ are independent of the time arrival of j^{worse} and r_i. Thus,

$$
\begin{aligned}
P[\mathcal{C}] &= P\left[\{|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\} \wedge \{|M| < 4\delta k\}\right] \cdot P[1/2 < t_{r_i} < 1/2+\delta] \cdot P[t_{j^{worse}} < 1/2] \\
&= P\left[\{|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\} \wedge \{|M| < 4\delta k\}\right] \cdot \delta \cdot (1/2) \\
&= (\delta'/32) \cdot P\left[\{|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\} \wedge \{|M| < 4\delta k\}\right] \\
&= (\delta'/32) \cdot P\left[\{|R_1| < |R| - \frac{1/2-2\delta}{1-\delta}\delta' k\} \wedge \{|M| < 4\delta k\}\right]
\end{aligned}
$$

We continue by lower bounding the second term of the last expression. We do so by defining a random variable $\mathcal{M}$ which is independent of R_1 and it is such that $\mathcal{M}$ stochastically dominates $|M|$. We prove the stochastic dominance of $\mathcal{M}$ using a coupling argument.

Note that every candidate i accepts an arrival time t_i uniformly at random from $[0,1]$. We now describe an equivalent procedure to create the arrival times t_i. Each candidate i draws three independent random variables $B_i \sim Bernoulli(1/2+\delta)$, $t_i^1 \sim Uniform([0,1/2+\delta])$ and $t_i^2 \sim Uniform([1/2+\delta,1])$. Note that we can construct random variables t_i using B_i, t_i^1 and t_i^2 as follows:

$$
t_i = B_i \cdot t_i^1 + (1-B_i) \cdot t_i^2
$$

Let $l = |L_{(1+4\delta)k}(A_{0,1/2+\delta})|$ and denote by $h_1,\ldots,h_l$ the set of candidates in $L_{(1+4\delta)k}(A_{0,1/2+\delta})$. We define random variables $\tilde{t}_1,\ldots,\tilde{t}_{(1+4\delta)k}$ as follows: For each $j \in \{1,\ldots,l\}$ we define $\tilde{t}_j = t_{h_j}^1$ and for each $j \in \{l+1,\ldots,1+4\delta k\}$ we define $\tilde{t}_j \sim Uniform([0,1/2+\delta])$. We define $\mathcal{M} = \sum_{i=1}^{\lfloor(1+4\delta)k\rfloor} \mathbb{I}\left\{\tilde{t}_j > 1/2\right\}$ and since $\mathcal{M} - |M| = \sum_{i=l+1}^{\lfloor(1+4\delta)k\rfloor} \mathbb{I}\left\{\tilde{t}_j > 1/2\right\}$ we have that $\mathcal{M} \geq |M|$ almost surely. Note that random variables $\tilde{t}_j$ and random variables B_i are independent. Consequently, since $|R_1| = \sum_{i\in R} B_i$ we have that events $\{\mathcal{M} < y\}$ and $\{|R_1| < x\}$ are independent. Combining these observations we have:

$$\begin{aligned}
&P\left[\{|R_1| < |R| - \frac{1/2-2\delta}{1-\delta}\delta' k\} \wedge \{|M| < 4\delta k\}\right] \\
\geq &P\left[\{|R_1| < |R| - \frac{1/2-2\delta}{1-\delta}\delta' k\} \wedge \{\mathcal{M} < 4\delta k\}\right] \\
\geq &P\left[|R_1| < |R| - \frac{1/2-2\delta}{1-\delta}\delta' k\right] \cdot P[\mathcal{M} < 4\delta k] \\
\geq &P\left[|R_2| \geq \frac{1/2-2\delta}{1-\delta}\delta' k\right] \cdot P[\mathcal{M} < 4\delta k]
\end{aligned}$$

To upper bound $P[\mathcal{M} < 4\delta k]$ note that $\mathbf{E}[\mathcal{M}] = \lfloor (1+4\delta)k \rfloor \cdot \frac{\delta}{1/2+\delta} < 3\delta k$, where the last inequality holds for any $\delta < 1/2$. From Markov's inequality, we have that:

$$\begin{aligned}
P[\mathcal{M} > 4\delta k] &= P[\mathcal{M} > (4/3) \cdot 3\delta k] \\
&\leq P[\mathcal{M} > (4/3) \cdot \mathbf{E}[\mathcal{M}]] \\
&\leq (3/4)
\end{aligned}$$

Consequently

$$P[\mathcal{M} < 4\delta k] > (1/4)$$

Note that since $i < (1-\delta')k - 1$ we have that $|R| \geq \delta' k + 1$.

$$\mathbf{E}[|R_2|] = |R \setminus \{j^{worse}\}|(1 - 1/2 - \delta) \geq \delta' k(1/2 - \delta)$$

From Markov's inequality, we have that:

$$\begin{aligned}
P\left[|R_2| > \frac{1/2-2\delta}{1-\delta}\delta' k|\right] &\geq P\left[|R_2| > \frac{1/2-2\delta}{(1-\delta)\cdot(1/2-\delta)} \cdot \mathbf{E}[|R_2|]\right] \\
&\geq \frac{(1-\delta)\cdot(1/2-\delta)}{1/2-2\delta} \\
&= \frac{(1-\delta)\cdot(1-2\delta)}{1-4\delta} \\
&\geq (1-\delta)
\end{aligned}$$

Consequently,

$$\begin{aligned}
P[\mathcal{C}] &\geq (\delta'/32) \cdot (1/4) \cdot (1-\delta) \\
&\geq (\delta'/32) \cdot (1/4) \cdot (1/2) \\
&\geq (\delta'/256)
\end{aligned}$$

□

Theorem 5. K-PEGGING *satisfies smoothness and fairness for k-secretary with $C = 4$ and $F_i = \max\left\{(1/3)^{k+5}, \frac{1-(i+13)/k}{256}\right\}$ for all $i = 1, \ldots, k$.*

Proof. The theorem follows directly from corollary 9 and lemmas 10 and 11. □