Robust Secretary and Prophet Algorithms for Packing Integer Programs*

C.J. Argue† Anupam Gupta‡ Marco Molinaro§ Sahil Singla¶

Abstract

We study the problem of solving Packing Integer Programs (PIPs) in the online setting, where columns in $[0,1]^d$ of the constraint matrix are revealed sequentially, and the goal is to pick a subset of the columns that sum to at most B in each coordinate while maximizing the objective. Excellent results are known in the secretary setting, where the columns are adversarially chosen, but presented in a uniformly random order. However, these existing algorithms are susceptible to adversarial attacks: they try to "learn" characteristics of a good solution, but tend to over-fit to the model, and hence a small number of adversarial corruptions can cause the algorithm to fail.

In this paper, we give the first robust algorithms for Packing Integer Programs, specifically in the recently proposed Byzantine Secretary framework [BGSZ20]. Our techniques are based on a two-level use of online learning, to robustly learn an approximation to the optimal value, and then to use this robust estimate to pick a good solution. These techniques are general and we use them to design robust algorithms for PIPs in the prophet model as well, specifically in the Prophet-with-Augmentations framework [ISW20]. We also improve known results in the Byzantine Secretary framework: we make the non-constructive results algorithmic and improve the existing bounds for single-item and matroid constraints.

1 Introduction

Resource allocation is a central problem in online decision making: here, a set of requests for resources arrive one-by-one, each having an associated value. The goal is to accept a subset of the requests with a large total value, subject to satisfying given resource constraints. In online algorithms, we have to make these decisions sequentially and irrevocably, without the knowledge of future requests. It is common to model these problems as *packing integer programs* (PIPs) of the form:

$$\max\{\langle c, x\rangle \mid Ax \le b,\ x \in \{0,1\}^n\ \}, \tag{1.1}$$

where the columns of an unknown constraint matrix $A \in [0,1]^{d\times n}$ appear one-by-one, and the decisions of the algorithm are encoded as variables $x_t \in \{0,1\}$.

Although online packing IPs are difficult to solve in the worst-case, one of the remarkable successes has been for settings where the PIP instance is chosen adversarially, but the columns are then presented in a *uniformly random order*. If the constraints are "not-too-tight", we can get very good solutions: e.g., for a PIP where the constraint matrix $A \in [0,1]^{d\times n}$ and the entries of b are $\Omega(\varepsilon^{-2}\log d)$, we know algorithms that obtain a $(1-\varepsilon)$-approximation as long as the columns of A arrive in a uniformly random order [KRTV14, GM16, AD15]. In fact, these results can be thought of as extensions of the *multiple-secretary* problem [Kle05] (which is the case where $d = 1$ and the matrix A is the all-1s matrix), and ultimately as extensions of the classical *secretary problem* [Dyn63] where the goal is to pick a single item to maximize the value.

However, previous algorithms rely heavily on the random-order assumption, and are susceptible to worst-case corruptions: If an adversary is allowed to add in a small number of columns that arrive at specific times, most existing algorithms for the random-order setting fail disastrously. E.g., the classical single-item secretary

*CJA and AG were supported in part by NSF awards CCF-1907820, CCF1955785, and CCF-2006953. MM was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001, CNPq Bolsa de Produtividade em Pesquisa #4310516/2017-0 and FAPERJ grant Jovem Cientista do Nosso Estado.

†Carnegie Mellon University, cargue@andrew.cmu.edu.

‡Carnegie Mellon University, anupamg@cs.cmu.edu.

§PUC Rio, mmolinaro@inf.puc-rio.br.

¶Georgia Tech, ssingla@gatech.edu.

algorithm—which waits for n/e items and then picks the first item bigger than all preceding items—fails even if a single high-value item is added at the beginning. Many algorithms for solving LPs essentially try to estimate duals/thresholds, which can be skewed by a small number of adversarial items. In short, most current algorithms are non-robust, and seem to over-fit to the model. Our central motivating questions are:

> *Can we get robust algorithms for online resource allocation, and for PIPs in particular? More broadly, when we give algorithms that make assumptions on the data, how do we ensure that their performance degrades gracefully as we allow adversarial corruptions?*

The *Byzantine secretary* model [KM20, BGSZ20] is one attempt to model adversarial corruptions in the random-order model. In this model, each request (i.e., column of A) is either red (i.e., adversarial) or green (i.e., benign), where these colors are not known to the algorithm. The adversary chooses the arrival time of each red column, whereas the green columns choose their arrival times independently and uniformly at random. The benchmark is now the value of the optimal green set (sometimes with one green item removed).

In [BGSZ20], the authors give robust algorithms for the single- and multi-item secretary settings, but leave open the question of getting robust algorithms for the setting of packing integer programs. In this paper, we resolve this question positively, and also give simpler and better robust algorithms for the single-item case. Our techniques are general and also extend to robust algorithms for PIPs in the prophet model, specifically to the Prophet-with-Augmentations model of [ISW20].

1.1 Our Results PIPs in the Byzantine Secretary Model. Our first set of results give robust algorithms for packing integer programs. To state the result, let $\mathcal{G}$ denote the green columns of matrix A, and let $g_{\max}$ denote the green column with largest value. Let $\mathrm{OPT}(\mathcal{G})$ and $\mathrm{OPT}(\mathcal{G} \setminus g_{\max})$ denote the optimal value of the offline PIP (1.1) when restricted to the green columns, with and without $g_{\max}$ respectively.

THEOREM 1.1. (INFORMAL: ROBUST PIPS) *There exists an algorithm for packing integer programs with value* $\Omega(\mathrm{OPT}(\mathcal{G} \setminus g_{\max}))$ *in the Byzantine secretary model when the entries of* b *are* $\Omega(\mathrm{poly}(\log n))$.

(See §2 for formal statements.) One may ask whether we can compare to $\mathrm{OPT}(\mathcal{G})$ instead: sadly, [BGSZ20] showed instances where it is impossible to compare to $\mathrm{OPT}(\mathcal{G})$ without further assumptions. However, if we make (mild) regularity assumptions on the input, we can indeed get much more nuanced results. Indeed, suppose we are given even a very rough estimate of $\mathrm{OPT}(\mathcal{G})$—to within *polynomial* in n factors—then we can achieve the following:

THEOREM 1.2. (INFORMAL: ROBUST PIPS WITH A ROUGH ESTIMATE) *There exists algorithms for packing integer programs which take a rough estimate of the* $\mathrm{OPT}(\mathcal{G})$ *and achieve the following guarantees:*

(i) value $\mathrm{OPT}(\mathcal{G}) \cdot \Omega(1)$ *for any instance.*

(ii) value $\mathrm{OPT}(\mathcal{G}) \cdot (1 - \alpha - O(\varepsilon))$ *for instances where* α *is the fraction of red/adversarial columns and the value of any* $\approx \log d/\varepsilon^4$ *columns accounts for only an* ε*-fraction of the optimum (i.e., the optimal value is not concentrated on a small set of columns).*

Both these algorithms require that the entries of b *are at least* $B := \Omega(\log d \cdot \mathrm{loglog}\, n/\varepsilon^4)$.

(See §2 for a formal statement of the former; the latter appears in the full version of the paper.) Note that the performance of this algorithm in part (ii) approaches 1 as the fraction α of adversarial corruptions gets small. The algorithm of part (i) is the same as in Theorem 1.1, but we need new ideas for part (ii), the case of "smooth" instances where the optimal value is spread out. Theorems 1.1 and 1.2 both use our technique of *robust threshold estimation*, which is based on two conceptually clean ideas using tools from online learning: Firstly, we show that given an estimate $\gamma \approx \frac{\mathrm{OPT}(\mathcal{G})}{B}$, we can pick a set of items that achieve value close to $\mathrm{OPT}(\mathcal{G})$. This uses a low-regret online linear optimization algorithm to learn a good set of duals, which are then used to select items. Secondly, we break the time horizon into K pieces, and then use another online learning algorithm to "learn" the parameter γ. This is where we use our coarse estimate of the optimal value: it allows us to focus on a set of $O(\varepsilon^{-1} \log n)$ possible values for γ. We discuss the technical ideas in §1.2.

PIPs in the Prophet-with-Augmentations Model. We think that our approach of using online learning to get robust algorithms will be useful in other contexts as well. As an example, we consider PIPs in the *prophet*

model where the columns are known up-front, but the value of each column is independently drawn from a known distribution. In the *prophet-with-augmentations* model of [ISW20], the adversary is allowed to adaptively add arbitrary positive perturbations/augmentations to the random column values, and the algorithm has to be robust to these augmentations. The algorithm competes against the base instance, i.e., the expected value OPT_{base} of the offline optimum when all perturbations are zero.

It may appear that adding positive perturbations should only improve the algorithm's performance, but [ISW20] show that the popular 1/2-approximation median threshold algorithm [SC84] for single-item prophets can become arbitrarily bad due to augmentations. (The reason is similar to that for the Byzantine Secretary model: the adversary can present a single high-valued item in the beginning that is just above the threshold.) [ISW20] show how to avoid these problems, and design robust prophet algorithms for the single-item and uniform-matroid problems. In §4 we show robust prophet algorithms for the general setting of PIPs, where our underlying technique is again based on robust threshold estimation using tools from online learning.

THEOREM 1.3. (ROBUST PROPHET PIPS) *There exists an algorithm for packing integer programs that gets value* $\Omega(\text{OPT}_{base})$ *in the Prophet-with-Augmentations model when the right-hand sides of the PIP are* $\Omega(\log d)$.

Improved Results for Single-Item and Matroid Cases. Our next results improves on those of [BGSZ20], for the case of picking a single item or an independent set in a matroid. (Details in §3.)

THEOREM 1.4. (SINGLE-ITEM PROBABILITY MAX) *There is an algorithm for the single-item Byzantine secretary problem that picks value at least* $\text{OPT}(\mathcal{G} \setminus g_{\max})$ *with probability* $\Omega\big(1/\log n\big)$.

In the case of a single-item, observe that $\text{OPT}(\mathcal{G} \setminus g_{\max})$ is the same as the value $\mathsf{val}(g_2)$ of the 2nd-highest green item. The previous result of [BGSZ20] was non-constructive, and only showed *existence* of an algorithm with success probability $\Omega\big(1/(\log n)^2\big)$; hence our result improves on the previous results both qualitatively and quantitatively. We also improve the value maximization results of [BGSZ20].

THEOREM 1.5. (VALUE-MAXIMIZATION FOR SECRETARY PROBLEMS) *There exist algorithms for the following Byzantine secretary problems that aim to maximize the expected value of selected items:*

(i) for the single-item case, we can get expected value $\Omega\big(1/\log^* n\big) \cdot \text{OPT}(\mathcal{G} \setminus g_{\max})$, *and*

(ii) for the case of a matroid of rank r, *we can get expected value* $\Omega\big(1/\big((\log^* n)^2 \cdot \log r\big)\big) \cdot \text{OPT}(\mathcal{G} \setminus g_{\max})$

(See §5 for proofs.) The former result improves on the the previous expected value of $\Omega\big(1/(\log^* n)^2\big) \cdot \text{OPT}(\mathcal{G} \setminus g_{\max})$, and the latter result improves the previous expected value of $\Omega\big(1/\log n\big) \cdot \text{OPT}(\mathcal{G} \setminus g_{\max})$ when the rank r is sufficiently smaller than the number of items n.

1.2 Our Techniques The general idea of all our algorithms is to find *robust thresholds*. For packing IPs in the secretary and prophet settings, these robust thresholds are obtained by using a Multiplicative-Weight Updates (MWU) algorithm, and for single-item probability/value maximization the idea is to perform a robust binary search on a set of candidate thresholds as we gather more information over time. Let us now flesh these ideas out in the context of two of our results: for Byzantine PIPs and for single-item probability maximization.

Byzantine Packing IPs. At a high level, our robust algorithm looks at a Lagrangified value $c_t x_t - \gamma\langle \lambda_t, A_t x_t\rangle$ of the t^{th} item to make its decision. Here the dual λ_t (computed using MWU) puts a relative "penalty" on each constraint, with higher penalties for constraints that are more occupied. The scale parameter γ balances between the value and the occupation penalty. Given this, our algorithm $\text{Alg}(\gamma)$ picks item t if its Lagrangian value is non-negative, namely if $c_t \geq \gamma\langle \lambda_t, A_t\rangle$. Variants of this algorithm have been previously used to study packing IPs in the *stochastic* setting [AD15, GM16]. Our first algorithmic contribution is that this algorithm can be made robust in the Byzantine Secretary model, *assuming that the right* γ *is known*. Intuitively, the algorithm has a "self-correcting" nature that balances the occupation of the different constraints, and the worst-case guarantees of MWU can be used to show its robustness.

In stochastic settings, the right γ is easy to estimate from the initial items; however, a few adversarial items can bias the estimation in the Byzantine setting. So our second algorithmic contribution is to use a second layer of learning to estimate γ. We break the time horizon into K intervals, learn $\gamma_1, \gamma_2, \ldots, \gamma_K$ online and run $\text{Alg}(\gamma_i, I_i)$ in each interval I_i. The i^{th} reward of expert γ is the value the algorithm would have obtained on

interval I_i is on using γ. Since MWU algorithms choose the sequence $(\gamma_i)_i$ to do almost as well as the right γ^*, we get $\sum_i \text{Alg}(\gamma_i, I_i) \gtrsim \sum_i \text{Alg}(\gamma^*, I_i)$, which is essentially the value of the algorithm that knew the right γ^* in hindsight. One difficulty is that the additive regret typically scales with the range of the possible rewards of the experts, and the red items can make this range too big. To handle this we introduce a truncation to these rewards (note we don't know $\text{OPT}(\mathcal{G})$, so this step needs care), and also use a recent multiscale experts result of [BDHN19] to make the regret scale with the range of the reward of the *best expert*, not with all the ranges.

This idea (with some changes) extends to the case of robust prophets. Again, we define robust thresholds using MWU: we start off considering the Lagrangified value $c_t x_t - \gamma\langle \lambda_t, A_t x_t\rangle$ of the t^{th} item to make our decisions. The challenge of estimating the right γ now becomes simpler since we are given the distributions. But a new challenge arises: the expected occupation is different at each time step t (which was not the case with random-order). To handle this issue and get the eventual solution, we further refine the Lagrangian penalty function (see §4 for details).

Single-item: probability maximization. The classical secretary algorithm has two phases: *sample* items to estimate a good threshold and *select* an item above the threshold. Recall that an adversary can thwart such an algorithm by sending a red item of very large value in the sample phase (which makes the threshold too high, so the algorithm does not pick any item). However, this failure gives us information: namely, that the max value seen in the sample phase is an upper bound on the value of future items. By running $O(\log n)$ copies of the secretary algorithm with distinct sample phases, we can force the adversary to plant a high-valued red item in each sample phase, and these values must decrease over time. (Running these multiple algorithms picks up to $O(\log n)$ items, but we can subsample down to 1; this is where we lose our approximation factor.)

Now, assuming this nice structure, we can give our SEARCH procedure that takes an initial set of candidate thresholds and narrows it down as it gets more information, using a robust binary-search procedure. The algorithm uses the values of items in an initial prefix $I_0 = [0, \frac{1}{4}]$ as a candidate set Θ for the "right" threshold $\mathsf{val}(g_2)$. At each time, it maintains upper and lower estimates that is used to filter these candidates:

1. The upper bound $\widehat{u}$ is the maximum value seen in the previous interval, and the lower bound $\widehat{\ell}$ is the maximum value we have picked so far.
2. The threshold for the current interval is the median of the "surviving" candidates of Θ, namely those of value in $(\widehat{\ell}, \widehat{u}]$. These have value strictly above what we have already picked.
3. The algo picks the first item in the interval that is above this threshold.

Suppose that, as discussed above, all intervals have a high-value item; namely, the maximum-value item ρ_i of the ith interval has value $\geq \mathsf{val}(g_2)$. Then the upper bound $\widehat{u}$ never excludes the "right" threshold $\mathsf{val}(g_2)$. Moreover, assume that these max values ρ_i are non-increasing. In this case, the set of surviving candidates halves in each interval! Indeed, if we pick an item in the interval the lower bound increases to the median value, else all items in the interval were below the median and the upper bound decreases to the current median value. Since there are more than $\log n$ intervals, at some point the set of candidates becomes empty. Now if g_2 comes in interval I_0 (which happens with probability $\frac{1}{4}$), the right threshold $\mathsf{val}(g_2)$ is an initial candidate but the lower bound must have excluded it, so we have already picked an item with at least this much value.

The idea for our *value-maximization* algorithms is to iteratively refine the thresholds: we start with a polynomially-approximate threshold, but each time we pick an item, we prove that we either get good expected value, or else we get an exponentially-better threshold. Naturally, this has to be done robustly, so that the red items have a limited impact.

1.3 Further Related Work See [GS20] for general works on random-order online problems. In particular, generalizations to matroid and Packing LPs in stochastic models have been extensively studied, e.g., see [BIKK18, Lac14, FSZ15] for matroids and [KRTV14, GM16, AD15] and references therein for packing. In the last two decades, there is a long line of work extending the classical single-item $1/2$-approximation prophet inequality [KS77, SC84] to packing constraints. In particular, see [KW12] for matroids and [Rub16, RS17] for arbitrary packing constraints. For applications of prophet inequalities to pricing mechanisms and online algorithms, we suggest the tutorial [FKS21] and the survey [Luc17].

The above online algorithms for secretary and prophet models rely heavily on the stochastic assumption, and are susceptible to even slight worst-case corruptions. We believe that the robust algorithms in this paper are interesting in their own right. They bridge the gap between the (optimistic) stochastic and (pessimistic) adversarial models, which has been a topic of significant interest in both online algorithms [Mey01, MGZ12, KMZ15, KKN15,

EKM18, Mol17, KM20, GKRS20, Mol21] and online learning (see [LMPL18, GKT19] and references within).

The recent paper [Mol21] considers a Byzantine-type model with adversarial and stochastic items, and studies Online Convex Optimization and Welfare Maximization problems. A major difference from our work is that both of these problems are unconstrained. In [KM20], the authors consider the Knapsack Secretary problem (i.e., PIP's with a single constraint) in a similar Byzantine model but with the additional assumption that the adversarial items come in bursts. They obtain a $(1-\varepsilon)$-approximation when $B \gtrsim \frac{1}{\varepsilon^2}$ and there are at most $\approx \frac{n}{\sqrt{B}}$ adversarial items in bursts of size $\approx \sqrt{B}$. Finally, [GKRS20] consider streaming problems (i.e. the algorithm has limited memory) in a similar Byzantine model, and design algorithms for max-matching and submodular maximization.

2 Byzantine Packing Integer Programs

In this section we discuss how to solve PIPs given in (1.1) and get a constant factor of the expected optimal value. The approach will be to solve the its *linear programming* relaxation: since we assume that the right-hand sides are large (i.e., $\Omega(\log d)$) compared to the entries of the constraint matrix, scaling down the solution slightly and independently rounding each variable immediately give an integer solution with almost as much value with high probability.

Each item i is a pair (c_i, a_i) of a column $a_i \in [0,1]^d$ of A, and its value $c_i \in \mathbb{R}_{\geq 0}$. The n items consist of G green items (denoted by $\mathcal{G}$) and $R = n - G$ red items (denoted by $\mathcal{R}$). The arrival times of red items are chosen by an adversary. Then each green item $i \in \mathcal{G}$ chooses its arrival time independently and uniformly in the time horizon $[0,1]$.

Our algorithm breaks the time horizon $[0,1]$ into several intervals and considers the items that fall into each interval seperately. To argue about this and other objects, it is useful to define the induced LP (denoted LP(S)) for any subset S of columns/items:

$$\text{(LP}(S)\text{)} \qquad \begin{aligned} \max\ & \textstyle\sum_{i\in S} c_i x_i \\ \text{s.t.}\ & \textstyle\sum_{i\in S} a_i x_i \leq B \cdot \mathbb{1}_d \\ & \mathbf{x} \in [0,1]^n \ , \end{aligned}$$

where $\mathbb{1}_d$ is the d-dimensional all-ones vector. Let its optimal solution be $\mathbf{x}^*(S)$, having value $\langle c, \mathbf{x}^* \rangle = \text{OPT}(S)$. By rescaling rows, we can assume a common value B on the RHS. We focus on two benchmarks: OPT$(\mathcal{G})$ and OPT$(\mathcal{G} \setminus g_{\max})$ where $g_{\max}$ denoting the green item with the highest value.

2.1 Algorithm Outline

The first step of our algorithm to reduce to solving the following "smooth" instances:

Assumption 2.1. (Smooth Instance) *An instance with* $\text{OPT} := \text{OPT}(\mathcal{G})$ *is* smooth *if:*

1. *the total value of the green items of value* $> \frac{\text{OPT}}{B}$ *is at most* $\frac{\text{OPT}}{2}$.
2. *we are given an estimate* $\widehat{O}$ *for* OPT*, such that* $\widehat{O} \in [\text{OPT}/n, \text{OPT} \cdot n]$.

Lemma 2.1. (Reduction to Smooth Instances) *Suppose* $B \geq \Omega(\text{poly}(\log n))$. *Given an algorithm to solve Packing LPs in the Byzantine Secretary setting that with constant probability is* ρ*-competitive w.r.t.* OPT$(\mathcal{G})$ *for all smooth instances, we can obtain an algorithm which is* $\Omega(\rho)$*-competitive in expectation for all instances w.r.t.* OPT$(\mathcal{G} \setminus g_{\max})$.

Given this reduction (which is proved in §C), we prove our main result for smooth instances:

Theorem 2.1. (Algorithm for Smooth Instances) *Suppose* $B \geq \Omega\big(K \log(dK/\delta')\big)$ *and* $K \geq \Omega(\log\log n)$. *The solution returned by Algorithm 2 for a smooth Byzantine Secretary instance satisfies:*

i. (Feasibility) The solution always packs into the modified budget of $(B+K) \cdot \mathbb{1}_d$.

ii. (Value) The solution has value at least $\Omega(\text{OPT})$ *with probability at least* $1 - \delta'$.

We can scale down each x_t by $\frac{B}{B+K} = 1 - o(1)$ to get a feasible solution with the same value guarantee up to a constant. Then, applying Lemma 2.1 to Theorem 2.1 gives us the constant factor approximation of Theorem 1.1. We now prove Theorem 2.1 in the rest of this section: here are the main conceptual steps:

- We break the time horizon into K time intervals of equal size (we will choose $K := \Theta(\log\log n)$). For interval I, use $\mathcal{G}(I)$ and $G(I)$ to denote the green items and their number in I, respectively. Note that $\mathbb{E}[G(I)] = |I| \cdot G$, where $|I|$ is the fraction of total time $[0,1]$ covered by interval I.

- In §2.2 we give our algorithm for a single interval I. Given a parameter γ, this algorithm runs a low-regret OLO subroutine on a carefully chosen Lagrangification of the problem. Let $\mathrm{Alg}(I,\gamma)$ denote the expected value that the algorithm gets when applied to interval I with parameter γ, where the expectation is over the random arrival times of the green items. Algorithm 1 gives a lower bound on $\mathrm{Alg}(I,\gamma)$.

- Finally, in §2.3 we use a multi-scale low regret algorithm to learn the optimal choice of γ. This allows us to combine the single-interval algorithms and prove Theorem 2.1.

2.2 Algorithm for a Single Interval So in this section we fix an interval $I \subseteq [0,1]$, and give an algorithm that gets good value in this interval as long as it knows the "correct" scalar parameter γ. We use r.v.s (C_t, A_t) to denote the value and size of the t-th item that appears in this interval: these depend on which of the items from $\mathcal{G}$ actually fall into this interval, and on their locations. As mentioned before, the idea of the algorithm is to look at the Lagrangified value $C_t x_t - \gamma\langle \lambda_t, A_t x_t\rangle$ to make the decision $x_t \in \{0,1\}$ to pick or not the t-th item in the interval. The duals λ_t's are computed using an online learning algorithm and put a relative "price" on each constraint, with higher prices for constraints that are more occupied.

To make this precise, let $\triangle^{d-1} := \left\{\mathbf{p} \in [0,1]^d \mid \|\mathbf{p}\|_1 = 1\right\}$ be the probability simplex. Given the algorithm's choice $x_t \in [0,1]$ for time step t, define the linear penalty function $f_t : \triangle^{d-1} \to [0,1]$ as

$$f_t(\lambda) := \langle \lambda, A_t x_t\rangle \ . \tag{2.2}$$

The algorithm for interval I (given a γ) is then described in Algorithm 1. Note that the algorithm does not depend on the exact arrival times of the items, just on their relative arrival order and step t denotes the t-th arrival in interval I.

Algorithm 1 IntervalByzLP(I,γ)

1: **for** steps $t = 1, 2, \ldots$ in interval I **do**
2: Use the low-regret OLO algorithm from Lemma A.1 with $\varepsilon = \frac{1}{2}$ on $f_1, \ldots, f_{t-1}$ to get $\lambda_t \in \triangle^{d-1}$
3: Compute $x_t \in [0,1]$ maximizing $x \mapsto C_t x - \gamma\langle \lambda_t, A_t x\rangle$
4: **break** if the scaled budget is violated, i.e., if a coordinate of $\sum_{s\le t} A_s x_s$ exceeds $B\,|I|$

In the rest of this section, we prove a lower bound on the value that this algorithm obtains over the interval I. For that, let x^* denote the optimal solution consisting only of green items whose value is at most $\frac{OPT}{B}$. By Assumption 2.1, this solution has value at least $\frac{OPT}{2}$. Also, define for any set S of timesteps the Lagrangified value of the optimal solution using the algorithm's choices of $\gamma\lambda_t$'s as the Lagrangian multipliers: $\mathcal{L}(S,\gamma) := \sum_{t\in S}\left(C_t x_t^* - \gamma\langle \lambda_t, A_t x_t^*\rangle\right)$. Let $\mathcal{G}_{\mathrm{T}}(I) \subseteq \mathbb{N}$ denote the steps t where the t-th item in the interval I is green, which is a random set since each green item chooses its arrival times uniformly at random in $[0,1]$ whereas the red items choose their arrival times adversarially.

The first step for analyzing our algorithm is showing that it obtains value in I comparable to the Lagrangified value of the optimal solution x^* in this interval.

Lemma 2.2. (Value Comparable to Lagrangified OPT) *For any $I \subseteq [0,1]$ and $\gamma > 0$, we have*

$$\mathrm{Alg}(I,\gamma) \ \ge \ \min\left\{\tfrac{1}{2}\,|I|\gamma B \ , \ \mathcal{L}\big(\mathcal{G}_{\mathrm{T}}(I),\gamma\big)\right\} - 2\gamma\log d \ . \tag{2.3}$$

Proof. Consider the algorithm's run. Let τ be the number of items seen when the algorithm stops; that is, the smallest value τ such that $\sum_{t\le\tau} A_t x_t \not\le |I|\cdot B\cdot \mathbb{1}_d$. If the algorithm does not exhaust the budget, set τ to be the number of items $|\,\mathrm{items}(I)|$ in interval I. The value of the algorithm is exactly $\sum_{t\le\tau} C_t x_t$ and its occupation is $\sum_{t\le\tau} A_t x_t$.

Case 1 (Budget exhausted): The multiplicative-plus-additive guarantees of the low-regret algorithm from Lemma A.1 for $\varepsilon = 1/2$ gives

$$\textstyle\sum_{t\le\tau} f_t(\lambda_t) \;\ge\; \frac{1}{2}\max_{\lambda\in\Delta^{d-1}} \sum_{t\le\tau} f_t(\lambda) - 2\log d \;.$$

Substituting the definition $f_t(\lambda) = \langle \lambda, A_t x_t\rangle$, we get

$$\textstyle\sum_{t\le\tau}\langle \lambda_t, A_t x_t\rangle \;\ge\; \frac{1}{2}\max_{\lambda\in\Delta^{d-1}} \sum_{t\le\tau}\langle \lambda, A_t x_t\rangle - 2\log d \;=\; \frac{1}{2}\|A_1x_1 + \ldots + A_\tau x_\tau\|_\infty - 2\log d \;.$$

Hence, we can infer that

$$\sum_{t\le\tau} C_t x_t \;-\; \frac{\gamma}{2}\cdot\|A_1x_1+\ldots+A_\tau x_\tau\|_\infty \;\ge\; \sum_{t\le\tau} C_t x_t - \gamma\sum_{t\le\tau}\langle\lambda_t, A_t x_t\rangle - 2\gamma\log d \;\ge\; -\,2\gamma\log d, \tag{2.4}$$

where the second inequality uses that x_t is a best-response, and hence is no worse than playing $x = 0$. Moreover, if we exhaust our budget, the occupation $\|\sum_{t\le\tau} A_t x_t\|_\infty \ge |I|\cdot B$, and thus

$$\textstyle\sum_{t\le\tau} C_t x_t \;\ge\; \mathbf{1}(\text{budget exhausted})\cdot\frac{1}{2}\gamma|I|\cdot B - 2\gamma\log d \;. \tag{2.5}$$

Case 2 (Budget left): To give a lower bound on the value in the case we do not exhaust our budget, we use that for any t the value x_t^* is not a better response than x_t:

$$C_t x_t \;\ge\; C_t x_t - \gamma\langle\lambda_t, A_t x_t\rangle \;\ge\; C_t x_t^* - \gamma\langle\lambda_t, A_t x_t^*\rangle \;.$$

Summing over all times and using the non-negativity of $C_t x_t$ to drop the red items,

$$\textstyle\sum_{t\le\tau} C_t x_t \;\ge\; \sum_{t\in\mathcal{G}_{\mathrm{T}}(I),\, t\le\tau} C_t x_t \;\ge\; \sum_{t\in\mathcal{G}_{\mathrm{T}}(I),\, t\le\tau}\Big(C_t x_t^* - \gamma\langle\lambda_t, A_t x_t^*\rangle\Big) \;.$$

But when the algorithm does not exhaust its budget, the RHS is precisely $\mathcal{L}\big(\mathcal{G}_{\mathrm{T}}(I),\gamma\big)$, and so

$$\textstyle\sum_{t\le\tau} C_t x_t \;\ge\; \mathbf{1}(\text{budget left})\cdot\mathcal{L}\big(\mathcal{G}_{\mathrm{T}}(I),\gamma\big) \;. \tag{2.6}$$

Combining (2.5) and (2.6) concludes the proof. ∎

The final piece is to lower bound the Lagrangified value of the optimal solution x^* on the interval I. The proof of this lemma crucially uses the random arrival times of the green items.

LEMMA 2.3. *Let $|I| \le \frac{1}{4}$ and $B \ge \Omega\big(\frac{\log(4d/\delta)}{|I|}\big)$. Then for any $\gamma > 0$,*

$$\Pr\Big[\mathcal{L}\big(\mathcal{G}_{\mathrm{T}}(I),\gamma\big) \;\ge\; |I|\cdot\big(\tfrac{\mathrm{OPT}}{4} - 4\gamma B\big)\Big] \ge 1-\delta \;,$$

where the probability is taken over the random arrival times of the green items.

We give the essential intuition of this lemma here (at least in expectation) and defer the details to Appendix C.1. Consider any timestep $t \in \mathcal{G}_{\mathrm{T}}(I)$:

$$\mathbb{E}\Big[C_t x_t^* - \gamma\langle\lambda_t, A_t x_t^*\rangle\Big] \;\ge\; \tfrac{\mathrm{OPT}}{2G} - \gamma\mathbb{E}\Big[\langle\lambda_t, A_t x_t^*\rangle\Big] \;, \tag{2.7}$$

where the expectation is over the random ordering. For intuition only, suppose each column A_t is an i.i.d. sample (this is not w.l.o.g.), so that $A_t x_t^*$ is independent of λ_t. Then the expectation can be pushed into the inner product; hence if there are G green items overall, the expected value $\mathbb{E}[A_t x_t^*] \le B/G\cdot\mathbb{1}_d$, and so

$$\mathbb{E}\Big[C_t x_t^* - \gamma\langle\lambda_t, A_t x_t^*\rangle\Big] \;\;\ge\;\; \tfrac{\mathrm{OPT}}{2G} - \gamma\langle\mathbb{E}\lambda_t, \tfrac{B}{G}\cdot\mathbb{1}_d\rangle \;\;\ge\;\; \tfrac{\mathrm{OPT}}{2G} - \gamma\tfrac{B}{G} \;, \tag{2.8}$$

where the last inequality uses that $\lambda_t \in \Delta^{d-1}$. Finally, $\mathbb{E}[|\,\mathcal{G}_{\mathrm{T}}(I)|] = |I|\cdot G$, so we get $\mathbb{E}[\mathcal{L}] \ge |I|(\frac{\mathrm{OPT}}{2} - \gamma B)$ to complete the proof of Lemma 2.3 in expectation. However, the reason why this is just intuition and not a proof is that we sample *without replacement*, so λ_t (which depends on the $t-1$ first items on the interval I) is correlated with $A_t x_t^*$. To handle this, in Appendix C.1 we have to argue why these correlations are small.

Combining these lemmas gives the desired guarantee for our algorithm.

LEMMA 2.4. (VALUE OF ALGORITHM 1) *Let $|I| \le \frac{1}{4}$ and $B \ge \Omega\big(\frac{\log(4d/\delta)}{|I|}\big)$. Then for any $\gamma > 0$, with probability at least $1-\delta$ we have*

$$\mathrm{Alg}(I,\gamma) \ge |I| \cdot \min\Big\{\tfrac{\gamma B}{2}\,,\ \tfrac{\mathrm{OPT}}{4} - 4\gamma B\Big\} - 2\gamma \log d.$$

As mentioned earlier, we see that a good choice of γ is $\Theta(\frac{\mathrm{OPT}}{B})$, but this requires us to know the value of OPT; we now show how another layer of online learning can learn this value well enough.

2.3 A Robust LP Algorithm via Learning the Multiplier γ We partition the time interval $[0,1]$ into K intervals $I_1, I_2, \ldots, I_K$ of equal size, and run Algorithm 1 in each interval with the value of γ learned from previous intervals via an online learning algorithm. Formally, define the gain functions:

$$\mathrm{Alg}_i(\gamma) := \text{value from Algorithm 1 with parameter } \gamma \text{ over the } i^{th} \text{ interval } I_i$$
$$\overline{\mathrm{Alg}}_i(\gamma) := \min\{\mathrm{Alg}_i(\gamma)\ ,\ \tfrac{B}{K}\gamma\} \text{ the algorithm's truncated value } .$$

Using Assumption 2.1 that we know OPT up to $\mathrm{poly}(n)$ factors, let Γ be a list of $O(\log n)$ values that contains $\frac{\mathrm{OPT}}{16B}$ within a factor of 2. The complete algorithm is the following:

Algorithm 2 ByzLP

1: **for** interval $i = 1, \ldots, K$ **do**
2: pick $\gamma_i \in \Gamma$ using the multiscale experts algorithm of Lemma A.2 on $\overline{\mathrm{Alg}}_1, \overline{\mathrm{Alg}}_2, \ldots, \overline{\mathrm{Alg}}_{i-1}$.
3: run Algorithm 1 over interval I_i with parameter γ_i, thereby getting value $\mathrm{Alg}_i(\gamma_i)$.

Proof. [Proof of Theorem 2.1] Since there are K intervals, the feasibility follows directly from the stopping rule for the algorithm, and the fact that item sizes are at most 1. For the second claim, the value of the algorithm is $\sum_i \mathrm{Alg}_i(\gamma_i)$, which is at least $\sum_i \overline{\mathrm{Alg}}_i(\gamma_i)$, so it suffices to lower bound the latter. Let γ^* be a value in Γ that is in $\big[\frac{1}{32}\frac{\mathrm{OPT}}{B}, \frac{1}{16}\frac{\mathrm{OPT}}{B}\big]$. Due to the truncation $\overline{\mathrm{Alg}}_i(\gamma^*)$ is $O(\frac{\mathrm{OPT}}{K})$, so the multiscale regret guarantee of Lemma A.2 with $\varepsilon = \Theta(\sqrt{\log|\Gamma|/K})$ gives that in every scenario,

$$\sum_i \overline{\mathrm{Alg}}_i(\gamma_i) \ \ge\ \sum_i \overline{\mathrm{Alg}}_i(\gamma^*) - O(1)\cdot\sqrt{K\log|\Gamma|}\cdot\tfrac{\mathrm{OPT}}{K} \ =\ \sum_i \overline{\mathrm{Alg}}_i(\gamma^*) - \mathrm{OPT}\cdot O(\tfrac{\sqrt{\log\log n}}{\sqrt{K}})\ . \tag{2.9}$$

Using the guarantee of Lemma 2.4 with $\delta = \frac{\delta'}{K}$, we have that with probability at least $1-\frac{\delta'}{K}$, (using $\mathrm{Alg}(I,\gamma^*)$ for the value obtained by the algorithm of the previous section, and I_i for the i-th interval)

$$\overline{\mathrm{Alg}}_i(\gamma^*) \ \ge\ \min\big\{\,\mathrm{Alg}(I_i,\gamma^*)\ ,\ \tfrac{\gamma^* B}{K}\big\} \ \ge\ \tfrac{1}{K}\cdot\min\big\{\tfrac{\gamma^* B}{2}\ ,\ \tfrac{\mathrm{OPT}}{4} - 4\gamma^* B\big\} - 2\gamma^*\log d\ ,$$

where the last inequality uses Lemma 2.2 and Lemma 2.3. Since $\gamma^* \in \big[\frac{1}{32}\frac{\mathrm{OPT}}{B}, \frac{1}{16}\frac{\mathrm{OPT}}{B}\big]$, we get

$$\overline{\mathrm{Alg}}_i(\gamma^*) \ \ge\ \Big(\tfrac{1}{K}\cdot\min\big\{\tfrac{1}{64}, \tfrac{1}{4}-\tfrac{1}{8}\big\} - \tfrac{\log d}{8B}\Big)\,\mathrm{OPT} \ \ge\ \Omega(1)\cdot\tfrac{\mathrm{OPT}}{K}\ .$$

Taking sum over all K intervals we get with probability at least $1-\delta'$ that $\sum_i \overline{\mathrm{Alg}}_i(\gamma^*) \ge \Omega(\mathrm{OPT})$. Using this on (2.9) and using $K \ge \Omega(\log\log n)$ concludes the proof of Theorem 2.1. ∎

In the full version we give an algorithm that gets an approximation approaching 1 when the number of red items gets small, thereby proving Theorem 1.2.

3 Byzantine Secretary for Single-Item Probability Maximization

We now consider the (single-item) Byzantine Secretary Problem, where the online model is exactly the same as in the previous section but now we can only pick one item: the goal is to maximize the *probability* of selecting an item of value at least $\mathrm{OPT}(\mathcal{G} \setminus g_{\max})$, i.e., the value of the second-most valuable green item. We show the following:

THEOREM 3.1. (SINGLE-ITEM PROBABILITY MAX) *There is an algorithm for the single-item Byzantine secretary problem that picks value at least $\mathrm{OPT}(\mathcal{G}\setminus g_{\max})$ with probability $\Omega\big(1/\log n\big)$.*

This improves on the algorithm of [BGSZ20], which (a) is nonconstructive, relying on the use of Yao's minimax principle, and (b) succeeds with a smaller probability of $\Theta(1/\log^2 n)$.

3.1 Algorithm Our algorithm is based on two procedures, STRUCT and SEARCH. Both pick $K = O(\log n)$ items. We will show that with constant probability at least one of STRUCT and SEARCH succeeds in picking an item with value at least $\mathrm{OPT}(\mathcal{G} \setminus g_{\max})$. By picking all items chosen by either procedure, we get an algorithm that picks $2K = O(\log n)$ items and succeeds with probability $\Omega(1)$. Now picking uniformly at random one of these $2K$ items, we get an algorithm that picks a single item of value at least $\mathrm{OPT}(\mathcal{G} \setminus g_{\max})$ with probability $\Omega(1/\log n)$. In §1.2 we outline the intuition behind these algorithms.

For both our procedures, we partition the time interval $(\frac{1}{4}, \frac{3}{4}]$ into $K = \Theta(\log n)$ intervals $I_1, \ldots, I_K$, each of equal width $\frac{1}{2K}$. Let $I_0 := [0, \frac{1}{4}]$. For all i, let ρ_i be the maximum value of a *red item* in interval I_i and let $\widehat{\mu}_i$ be the maximum value of any item in interval I_i (note that ρ_i is deterministic but unknown to the algorithm and $\widehat{\mu}_i$ depends on when the green items arrive). For an item e, let $\mathsf{time}(e)$ and $\mathsf{val}(e)$ denote the arrival time and value of e respectively. Recall that g_2 is the 2nd-most valuable green item. To simplify the notation, let $C^* := \mathrm{OPT}(\mathcal{G} \setminus g_{\max}) = \mathsf{val}(g_2)$.

The procedure STRUCT runs K independent subroutines similar to the classic single-item secretary algorithm: the i^{th} one sets a threshold $\widehat{\mu}_i$ and picks the first item after I_i that reaches this threshold.

Algorithm 3 Procedure STRUCT

for value $i = 1, 2, \ldots, K$ in parallel **do**
  $\widehat{\mu}_i :=$ maximum value of any item seen in interval I_i
  pick the first item (if any) arriving after interval I_i with value at least $\widehat{\mu}_i$.

The second procedure SEARCH maintains a *set of candidate thresholds* containing a subset of all the values $\widehat{\Theta}_1$ seen in interval $I_0 = [0, \frac{1}{4}]$. In each interval, the current candidate set is obtained by focusing on these values lying between the maximum value of an item seen in the previous interval, and the largest value item picked by this procedure so far. The threshold for the current interval is set to the median of these values. (We define $\mathrm{median}(\varnothing) = -\infty$.)

Algorithm 4 Procedure SEARCH

for interval $i = 1, 2, \ldots, K$ **do**
  $\widehat{L}_i \leftarrow$ max value of an item already picked by this procedure, $-\infty$ if no item has been picked
  $\widehat{U}_i \leftarrow \widehat{\mu}_{i-1} :=$ maximum value of any item seen in interval I_{i-1}
  $\widehat{\Theta}_i \leftarrow \{\mathsf{val}(e) \mid \mathsf{time}(e) \in [0, \frac{1}{4}] \text{ and } \widehat{L}_i < \mathsf{val}(e) \le \widehat{U}_i\}$
  pick the first item (if any) in the interval I_i having value at least $\mathrm{median}(\widehat{\Theta}_i)$

3.2 Analysis It is clear that both procedures pick at most $K = O(\log n)$ items. We show that with probability $\Omega(1)$, at least one of the two procedures picks an item with value at least C^*. The intuition is this: if the maximum value items in each interval are monotone decreasing and greater than C^*, then in each interval procedure SEARCH halves the candidate set size. If this set contains the value C^* (which happens, e.g., when g_2 arrives in the interval I_0), then we must eventually pick an item of large value. Of course, the maximum value items may not be monotone, and they may have values below C^*, but then we show that STRUCT gets large value.

We now give the proof details. Recall that ρ_i is the maximum value of any red item in interval I_i; define $\rho_i \leftarrow -\infty$ if no such items exist. We say an interval I_i (including I_0) is *high* if $\rho_i \ge C^*$, and *low* otherwise. Recall that the property of being high just depends on the location of the red items, which we assume are deterministically placed.

LEMMA 3.1. *For any value of K,* STRUCT *succeeds (i.e. picks an item of value at least C^*) with probability $\Omega(1)$ if either of the following properties fail:*

(i) If I_i and I_j are high intervals with $i < j$, then $\rho_i > \rho_j$.

(ii) There are fewer than $\frac{K}{4}$ low intervals.

Proof. Suppose property (i) fails, and let I_i and I_j be high intervals with $i < j$ and $\rho_i \le \rho_j$. The highness of I_i implies $\rho_i \ge C^*$. Now if $g_{\max}$ does not fall in I_i (which happens with probability $1 - 1/2K$), then the maximum

value $\widehat{\mu}_i$ in this interval is ρ_i, and therefore the run of STRUCT corresponding to interval i sets a threshold of $\rho_i \geq C^*$. This run would definitely select the item corresponding to ρ_j in I_j, if it has not picked an item earlier; this selected item has value at least $\rho_i \geq C^*$.

Else suppose property (ii) fails, and there are at least $K/4$ low intervals. If g_2 arrives in a low interval I_i, then we set the threshold to be $\mathsf{val}(g_2) = C^*$; now if $g_{\max}$ arrives in $(\frac{3}{4}, 1]$, the i^{th} run of STRUCT is guaranteed to pick an item. Since there are at least $\frac{K}{4}$ low intervals, this event occurs with probability at least $\frac{K}{4} \cdot \frac{1}{2K} \cdot \frac{1}{4} \geq \frac{1}{32}$. ∎

The remainder of the proof shows that if the two properties of Lemma 3.1 are indeed satisfied, then SEARCH succeeds with constant probability. Define $U_i := \min\{\rho_j \mid j < i \text{ and } I_j \text{ is high}\}$, and

$$\Theta_i := \{\mathsf{val}(e) \mid \mathsf{time}(e) \in \left[0, \tfrac{1}{4}\right] \text{ and } \widehat{L}_i < \mathsf{val}(e) \leq U_i\}.$$

The only difference from $\widehat{\Theta}_i$ is that the upper bound is U_i instead of $\widehat{U}_i$. (The intuition is that if property (i) of Lemma 3.1 holds, then loosely speaking $\widehat{\Theta}_i$ and Θ_i should behave similarly, and we can argue about the latter instead of the former.) Observe that $\widehat{L}_i$ can only increase and U_i can only decrease, so $\Theta_i \supseteq \Theta_{i+1}$.

To make this precise, define an interval I_i (for $i \geq 1$) to be *nice* if the intervals I_i and I_{i-1} are both high, and moreover the item $g_{\max}$ does not arrive in interval I_{i-1}. This property of being nice does depend on the location of the top green item $g_{\max}$, but otherwise is independent of the random locations of other green items.

LEMMA 3.2. *If property (i) from Lemma 3.1 holds and the interval I_i is nice, then $|\Theta_{i+1}| \leq \frac{1}{2}|\Theta_i|$.*

Proof. Property (i) of Lemma 3.1 means the ρ-values of the high intervals are in decreasing order, and the minimum in the definition of U_i is achieved at the last high interval before i. Since interval I_{i-1} is high, we have $U_i = \rho_{i-1}$. Moreover, $g_{\max}$ does not arrive in interval I_{i-1}, and therefore $\rho_{i-1} = \widehat{\mu}_{i-1}$. This in turn means that $U_i = \widehat{\mu}_{i-1} = \widehat{U}_i$, and $\widehat{\Theta}_i = \Theta_i$.

If $m_i := \text{median}(\Theta_i) = \text{median}(\widehat{\Theta}_i)$, then define $\Theta_i^+ := \{v \in \Theta_i \mid v > m_i\}$ and $\Theta_i^- := \{v \in \Theta_i \mid v < m_i\}$. Each has size at most $\frac{1}{2}|\Theta_i|$. If the SEARCH procedure chose an item in I_i, then $\widehat{L}_{i+1} \geq m_i$ and therefore $\Theta_{i+1} \subseteq \Theta_i^+$. Otherwise SEARCH did not chose an item in I_i, so all its items must have been smaller than m_i; in particular, $\rho_i < m_i$. Since I_i is high, $u_{i+1} \leq \rho_i < m_i$ and $\Theta_{i+1} \subseteq \Theta_i^-$. ∎

LEMMA 3.3. *Let $K = 2\log_2 n + 4$. Then with probability $\Omega(1)$, at least one of* STRUCT *and* SEARCH *picks an item of value at least C^*.*

Proof. By Lemma 3.1, if either of its properties (i) or (ii) fails then STRUCT succeeds with constant probability. So suppose both properties hold. Now condition on the location of item $g_{\max}$; this decides on the niceness of the intervals. Property (ii) being satisfied means there are at least $K - \frac{K}{4} - \frac{K}{4} - 1 = K/2 - 1 > \log_2 n$ nice intervals. (Indeed, we may discard at most $K/4$ intervals because I_{i-1} is bad, $K/4$ others because I_i is bad, and one more because $g_{\max}$ falls in I_{i-1}.)

Now let us condition on the event that g_2 arrives in the time interval $[0, \frac{1}{4}]$, which happens with probability $1/4$. Since $|\Theta_1| \leq n$, applying Lemma 3.2 (which relies on property (i)) to each of the nice intervals implies that after $\log_2 n$ nice intervals, we get to some index i for which Θ_i is empty, and interval I_i is nice. The upper bound U_i for Θ_i is at least C^*, by definition. Since item g_2 arrived in $[0, \frac{1}{4}]$ but yet $C^* = \mathsf{val}(g_2) \notin \Theta_i$, the only reason would be that $\widehat{L}_i \geq C^*$. This means that SEARCH must have already chosen an item of value at least C^*. ∎

Proof. [Proof of Theorem 1.4] Since STRUCT and SEARCH pick at most K items each, the final algorithm is to run a random one of these two algorithms, and to randomly output one of the $K = O(\log n)$ items picked by that algorithm. This gives an item of value at least C^* with probability $\Omega(1/\log n)$. ∎

4 Prophet-with-Augmentations for Packing Integer Programs

To show the power of our robust threshold selection idea from §2, we use it to give robust algorithms for PIPs in the *prophet model* as well. Recall that in the classic prophets model, the inputs are drawn from independent (but possibly non-identical) distributions. Here we consider the *Prophets-with-Augmentations* model [ISW20], in which an adversary is allowed to perturb the values by adding non-negative values. We now show how to make PIP algorithms robust to such perturbations.

4.1 Model and Notation We are given a *base prophet* instance $((a_1, \mathcal{D}_1), \ldots, (a_n, \mathcal{D}_n), B)$ with budget $B \cdot \mathbb{1}_d$ and n items whose values will be perturbed by an adversary. The t^{th} item has a known deterministic size vector $a_t \in [0,1]^d$ and an initially unknown value $V_t \geq 0$ that is drawn independently from a known distribution $\mathcal{D}_t$. Items values are revealed one-by-one, and before the t-th item's value is revealed, an adversary adds an unknown perturbation $R_t \geq 0$ to V_t. This perturbation may depend on the history up to (and including) time t, as well as the algorithm's decisions up to time $t-1$. The player then sees $C_t := V_t + R_t$ and has to immediately pick or reject the item. The goal is to pick a set of items that pack into the known budget $B \cdot \mathbb{1}_d$, in order to maximize the sum of seen values C_t of the picked items.

The algorithm competes against the base instance, i.e., the expected offline optimum when all perturbations R_t are zero. Let OPT_{base} be the value of this expected offline optimum. The main result of this section is as follows.

THEOREM 4.1. (ROBUST PROPHET PIPS) *There exists an algorithm for packing integer programs that gets value $\Omega(\mathrm{OPT}_{base})$ in the Prophet-with-Augmentations model when the right-hand sides of the PIP are $\Omega(\log d)$.*

To make our proofs simpler, we assume w.l.o.g. that there are no "large values". Here we sketch the proof; the full proof is deferred to §D.1.

ASSUMPTION 4.1. *Each distribution $\mathcal{D}_t$ is supported on values that are at most $\frac{\mathrm{OPT}_{base}}{20}$.*

Proof. [Proof Sketch] Consider running simultaneously an algorithm that obtains a constant approximation under Assumption 4.1 and also an algorithm that picks the first item that takes a value above $\frac{\mathrm{OPT}_{base}}{20}$. Intuitively, in the scenarios where all items come up with value at most $\frac{\mathrm{OPT}_{base}}{20}$ the approximation guarantee of the first algorithm kicks in, and in the remaining scenarios the second algorithm already guarantees value at least $\frac{\mathrm{OPT}_{base}}{20}$; overall, we should get a constant approximation. Running both algorithm may lead to budget occupation up to $(B+1) \cdot \mathbb{1}_d$, but rescaling the solution by $\frac{B}{B+1} > 1 - o(1)$ restores feasible while maintaining the same approximation guarantee. ∎

4.2 Algorithm The idea of the algorithm is similar to that used for Byzantine PIPs in §2: to make decisions $x_t \in \{0,1\}$ based on the Lagrangified value $C_t x_t - \gamma \langle \lambda_t, a_t x_t \rangle$, for a scaling factor γ. As before, the "right" value for γ is $\approx \frac{\mathrm{OPT}_{base}}{B}$. Previously we learned γ over multiple intervals using online learning (since OPT was not known), we can now directly compute it using the known value distributions $\mathcal{D}_t$'s. However, new challenges arise. Firstly, we will again need to bound a Lagrangified value of the offline optimum with *good probability* (as in Lemma 2.3), but since the optimal solution's decisions to pick items depend on the outcomes of the values of *all* other items, we are not guaranteed to have any concentration. (Previously OPT was such that picking the t-th item only depended on the identity of that item.) To fix this, we compare not against the optimal solution, but against a surrogate $\psi_t(V_t) \in \{0,1\}$ that makes decisions about item t based only on its base value V_t. Another challenge is that the expected occupation of such a solution is different in each time step t (which was not the case in random-order, see Equation (2.8)): this makes it harder to bound the quantity $\langle \lambda_t, a_t \, \psi(V_t) \rangle$. To fix this, we define the Lagrangian in terms of the modified penalty function $f_t(\lambda) := \langle \lambda, a_t x_t - a_t \, \mathbb{E}\psi_t(V_t) \rangle$. On a technical note, since this penalty can be negative, we consider λ values in the "full-dimensional simplex" (including the 0 vector), namely $\blacktriangle^d := \{\lambda \in [0,1]^d : \sum_i \lambda_i \leq 1\}$.

To make this precise, we start with the existence of the good solution $\psi_1(V_1), \ldots, \psi_n(V_n)$ for the base prophet instance. Similar solutions algorithms have been previously designed for related problems (e.g., see [Ala14, AHL12]), and we defer the proof to Appendix D.2.

LEMMA 4.1. *Given a base prophet instance $((a_1, \mathcal{D}_1), \ldots, (a_n, \mathcal{D}_n), B)$, there are functions $\psi_1, \ldots, \psi_n$, where each ψ_t maps the value V_t to a decision in $\{0,1\}$ such that:*

1. *Total expected value $\mathbb{E}[\sum_t V_t \, \psi_t(V_t)] \geq \frac{\mathrm{OPT}_{base}}{4}$.*
2. *Total expected utilization $\mathbb{E}[\sum_t a_t \, \psi_t(V_t)] \leq \frac{B}{4} \cdot \mathbb{1}_d$.*

We now describe our robust algorithm for the prophet-with-augmentations model. To simplify notation, define $x_t^* := \mathbb{E}[\psi_t(V_t)]$, that is, the probability that this solution picks the t-th item. Our algorithm requires

these x_t^*'s, but since they only depend on the distributions of the V_t's thay can be computed a priori. It also needs to know OPT_{base} to set the value of γ, which can also be computed a priori (and as the proof shows, a constant-factor approximation to this value suffices).

Algorithm 5 Procedure PROPHET-WITH-AUGMENTATIONS

1: **for** steps $t = 1, 2, \cdots$ **do**
2: Compute $\lambda_t \in \blacktriangle^d$ by using the low-regret algorithm of Lemma A.1 with $\varepsilon = \frac{1}{2}$ on the functions $f_1, \ldots, f_{t-1}$, where $f_t(\lambda) := \langle \lambda, a_t x_t - a_t x_t^* \rangle$.
3: Compute $x_t \in [0,1]$ maximizing $x_t \mapsto C_t x_t - \gamma \langle \lambda_t, a_t x_t - a_t x_t^* \rangle$, where $\gamma = \frac{\mathrm{OPT}_{base}}{B}$.
4: **break** if the budget is exhausted, i.e., if a coordinate of $\sum_{s \le t} a_s x_s$ exceeds B. Let τ denote this stopping time step, where $\tau = n$ if the budget is never exhausted.

4.3 Analysis We now analyze the above algorithm, proving that it attains the guarantee stated in Theorem 1.3 under Assumption 4.1 (without loss of generality). First, the algorithm violates the budget by at most +1, but this is easily fixed by rescaling or subsampling, so we ignore this issue henceforth. To prove that it gets high value, fix a scenario of V_t and R_t. In this scenario, since our decision x_t is a best response, it is at least as good as $\psi_t(V_t)$. In other words, the best-response ensures

$$\underbrace{\sum_{t \le \tau} C_t x_t}_{=\mathrm{Alg}} - \gamma \underbrace{\sum_{t \le \tau} \langle \lambda_t, a_t (x_t - x_t^*) \rangle}_{=: M_{lhs}} \;\ge\; \sum_{t \le \tau} C_t\, \psi_t(V_t) - \gamma \underbrace{\sum_{t \le \tau} \langle \lambda_t, a_t (\psi_t(V_t) - x_t^*) \rangle}_{=: M_{rhs}}.$$

Rewriting, we get

$$\mathrm{Alg} \;\ge\; \sum_{t \le \tau} C_t\, \psi_t(V_t) + \gamma M_{lhs} - \gamma M_{rhs}. \tag{4.10}$$

To lower bound the value of Alg, we first lower bound M_{lhs}. From the regret guarantee in Lemma A.1, we can compare against the action $\lambda = 0$ to infer

$$M_{lhs} \;\ge\; -O(\log d). \tag{4.11}$$

Similarly, comparing against the action $\lambda = e_i$ we get (using $|\langle e_i, a_t(x_t - x_t^*)\rangle| \le \langle e_i, a_t x_t \rangle + \langle e_i, a_t x_t^* \rangle$)

$$M_{lhs} \;\ge\; \frac{1}{2}\langle e_i, \textstyle\sum_{t \le \tau} a_t x_t \rangle - \frac{3}{2}\langle e_i, \textstyle\sum_{t \le \tau} a_t x_t^* \rangle - O(\log d),$$

which then implies

$$\begin{aligned} M_{lhs} \;&\ge\; \frac{1}{2}\Big\| \sum_{t \le \tau} a_t x_t \Big\|_\infty - \frac{3}{2}\Big\| \sum_{t \le \tau} a_t x_t^* \Big\|_\infty - O(\log d) \\ &\ge\; \frac{1}{2}\Big\| \sum_{t \le \tau} a_t x_t \Big\|_\infty - \frac{3B}{8} - O(\log d), \end{aligned} \tag{4.12}$$

where the last inequality uses that $\| \sum_{t \le \tau} a_t x_t^* \|_\infty \le B/4$ due to Item 2 in Lemma 4.1 and the definition $x_t^* = \mathbb{E}[\psi_t(V_t)]$.

For the scenario when $\tau = n$, (4.10) and (4.11) give that

$$\mathrm{Alg} \;\ge\; \sum_{t} C_t\, \psi_t(V_t) - \gamma \cdot O(\log d) - \gamma M_{rhs}.$$

Note that the sum is now over all t, not only $t \le \tau$. For the other scenario, when $\tau < n$ (i.e., we exhaust the budget), we know that $\| \sum_{t \le \tau} a_t x_t \|_\infty > B$, and so using (4.10) and (4.12) we get

$$\mathrm{Alg} \;\ge\; \frac{\gamma B}{8} - \gamma \cdot O(\log d) - \gamma M_{rhs}.$$

Taking a minimum of both the scenarios $\tau = n$ and $\tau < n$, we get the following bound that holds for every scenario:

$$\text{Alg} \;\geq\; \min\Big\{ \sum_t C_t\ \psi_t(V_t)\ ,\ \frac{\gamma B}{8} \Big\}\ - \gamma \cdot O(\log d)\ - \gamma M_{rhs}.$$

To calculate $\mathbb{E}[M_{rhs}]$, notice that both $\mathbf{1}(\tau \geq t)$ and λ_t depend only on the history up to time $t-1$. (This is where we crucially use that the augmentations R_t do not depend on the future.) So taking conditional expectation,

$$\begin{aligned}\mathbb{E}_{t-1}[\mathbf{1}(\tau \geq t) \cdot \langle \lambda_t, a_t(\psi_t(V_t) - x_t^*)\rangle] \;&=\; \mathbf{1}(\tau \geq t) \cdot \langle \lambda_t, a_t\, \mathbb{E}_{t-1}[\psi_t(V_t) - x_t^*]\rangle \\ &=\; \mathbf{1}(\tau \geq t) \cdot \langle \lambda_t, a_t\, \mathbb{E}[\psi_t(V_t) - x_t^*]\rangle \;=\; 0,\end{aligned}$$

where the second equality follows from the fact that ψ_t decides based only on V_t, and hence is independent of the past. Taking expectations and summing over t gives $\mathbb{E}[M_{rhs}] = 0$. Hence,

$$\mathbb{E}[\text{Alg}] \geq \mathbb{E}\min\Big\{ \sum_t C_t\ \psi_t(V_t)\ ,\ \frac{\gamma B}{8} \Big\}\ - \gamma \cdot O(\log d). \tag{4.13}$$

To prove Theorem 1.3, it suffices to show that the first term in the minimization is $\Omega(\text{OPT}_{base})$ with constant probability. Notice that this term is always non-negative, and that the second term in the minimization is $\Omega(\text{OPT}_{base})$, since we set $\gamma = \Theta(\frac{\text{OPT}_{base}}{B})$.

LEMMA 4.2. *With probability at least* 0.6 *it holds that* $\sum_t C_t\ \psi_t(V_t) \geq \Omega(\text{OPT}_{base})$.

Proof. First, we always have $\sum_t C_t\ \psi_t(V_t) \geq \sum_t V_t\ \psi_t(V_t)$, since $R_t \geq 0$. By the upper bound in Assumption 4.1 on V_t, and the fact that $\psi_t(V_t) \in [0,1]$ and that for any random variable $X \in [0,\alpha]$ its variance satisfies $\text{Var}(X) \leq \alpha\mathbb{E}[X]$, we have

$$\text{Var}(V_t\, \psi_t(V_t)) \;\leq\; \frac{\text{OPT}_{base}}{20}\, \mathbb{E}[V_t\, \psi_t(V_t)].$$

Using the fact that $\psi_t(V_t)$ are independent and Item 1 in Lemma 4.1, we have

$$\text{Var}\Big(\sum_t V_t \psi_t(V_t)\Big) \;\leq\; \frac{\text{OPT}_{base}}{20}\, \mathbb{E}\Big[\sum_t V_t\, \psi_t(V_t)\Big] \;\leq\; \frac{1}{10}\Big(\mathbb{E}\Big[\sum_t V_t\, \psi_t(V_t)\Big]\Big)^2.$$

Applying Chebychev's inequality, we have

$$\Pr\Big(\sum_t V_t\, \psi_t(V_t) \leq \tfrac{1}{2}\mathbb{E}\Big[\sum_t V_t\, \psi_t(V_t)\Big]\Big) \;\leq\; 0.4.$$

Finally, using $\mathbb{E}[\sum_t V_t\, \psi_t(V_t)] \geq \frac{\text{OPT}_{base}}{2}$ by Item 1 in Lemma 4.1 concludes the proof. ∎

Combining Lemma 4.2 with (4.13) shows the expected value of the algorithm is at least a constant fraction of the OPT_{base}, and hence proves Theorem 1.3.

5 Byzantine Secretary for Value Maximization

In this section we consider two classical secretary problems in the Byzantine framework: that of (i) picking a single item, and (ii) picking an independent set in a matroid, to maximize the expected value of picked items. The main results of this section are:

THEOREM 5.1. (VALUE-MAXIMIZATION FOR SECRETARY PROBLEMS) *There exist algorithms for the following Byzantine secretary problems that aim to maximize the expected value of selected items:*

(i) for the single-item case, we can get expected value $\Omega\big(1/\log^* n\big) \cdot \text{OPT}(\mathcal{G} \setminus g_{\max})$, *and*

(ii) for the case of a matroid of rank r, *we can get expected value* $\Omega\big(1/\big((\log^* n)^2 \cdot \log r\big)\big) \cdot \text{OPT}(\mathcal{G} \setminus g_{\max})$

The first claim is proved in §5.1, and improves upon the previous factor of $\Omega\big(1/(\log^* n)^2\big)$ of [BGSZ20]. The second claim is then proved in §5.2 and improves upon the previous factor of $\Omega\big(1/\log n\big)$ when the rank r is sufficiently smaller than the total number of items n, also from [BGSZ20].

Note that these packing problems are related to the results we saw in §2 : both the current problems can also be modeled as PIPs, but we cannot assume that the capacities (i.e., the right-hand sides of the PIP) are large, so we cannot apply results we proved earlier. The single-item *probability maximization* result from §3 also gives an $\Omega(1/\log n)$-approximation for the value-maximization setting, which is much weaker than the result here.

5.1 Single Item Value Maximization For the value-maximization problem, we give three procedures, each of which picks at most $K = O(\log^* n)$ items. By picking all items chosen by any of the three procedures, we get an algorithm that picks $O(\log^* n)$ items and gets expected value $\Omega(C^*)$. Then picking one of these items uniformly at random, we get a single item with expected value $\Omega(C^*/\log^* n)$.

The main idea of this algorithm is break the time horizon into K intervals, and to iteratively get a better estimate of the optimum value each time we fail to get a large expected value. As in §3.1, we divide time into K intervals, and let $\widehat{\mu}_i$ be the largest value of any item in interval I_i. For the sake of intuition, assume that $nC^* \geq \widehat{\mu}_0 \geq \widehat{\mu}_1 \geq \cdots \geq \widehat{\mu}_K \geq C^*$. Suppose we know that $\widehat{\mu}_i$ is an α-approximation to C^*. Pick k randomly from $[\log \alpha]$ and set the threshold $2^{-k}\widehat{\mu}_i$ for interval I_{i+1}. Since we know that $\widehat{\mu}_{i+1}$ is contained in the interval $[2^{-k}\widehat{\mu}_i, 2^{-(k-1)}\widehat{\mu}_i]$ for some $k \in [\log n]$, choosing this value of k yields value $\approx \widehat{\mu}_{i+1}$ with probability at least $\frac{1}{\log \alpha}$. If $\widehat{\mu}_{i+1} \geq C^* \log \alpha$, then we get expected value C^*. Otherwise, we know that $\widehat{\mu}_{i+1}$ is a $\log \alpha$-approximation to C^*. Since $\widehat{\mu}_0$ is an n-approximation to C^*, we find inductively that each $\widehat{\mu}_i$ is a $\log^{(i)} n$-approximation to C^*. This implies that we need consider only $K = O(\log^* n)$ many intervals. This gives the main idea: the actual procedure needs to consider situations where the interval maxima are not monotonically decreasing, so we need more care.

Formally, the structural procedures are Struct—defined in Algorithm 3 of §3—and a second procedure which simply picks one random item.

Algorithm 6 Procedure Sample

1: Choose one item uniformly at random

To explain the third and main search procedure, let us recall useful notation from §3: let ρ_i be the maximum value of any red item in interval I_i; define $\rho_i \leftarrow -\infty$ if no such items exist. An interval I_i (including I_0) is *high* if $\rho_i \geq C^*$, and *low* otherwise. An interval I_i (for $i \geq 1$) is *nice* if the intervals I_i and I_{i-1} are both high, and moreover the item $g_{\max}$ does not arrive in interval I_{i-1}.

Now in interval I_i, the new search procedure determines the location of the previous high intervals, assuming that I_{i-1} is high. The threshold for interval I_i is chosen from among $\log^{(\ell+1)} n$ exponentially-spaced values centered around the $\widehat{\mu}$ value of the previous nice interval, where ℓ is roughly the number of previous nice intervals.

Algorithm 7 Procedure SearchII

1: **for** interval $i = 1, 2, \ldots, K$ **do**
2: $\quad \widehat{\mathcal{N}}_i \leftarrow \{j \in [i-1] : \widehat{\mu}_j \geq \widehat{\mu}_{i-1}$ and $\widehat{\mu}_{j-1} \geq \widehat{\mu}_{i-1}\}$
3: $\quad \widehat{\ell}_i \leftarrow |\widehat{\mathcal{N}}_i| + 1$
4: $\quad$ **if** $\widehat{\ell}_i > 1$ **then**
5: $\quad\quad \hat{\jmath}_i \leftarrow \max(\widehat{\mathcal{N}}_i)$
6: $\quad$ **else**
7: $\quad\quad \hat{\jmath}_i \leftarrow i - 1$
8: $\quad \widehat{k}_i \leftarrow$ uniformly random integer in $[-\log^{(\widehat{\ell}_i)} n, \log^{(\widehat{\ell}_i)} n]$
9: $\quad$ In interval I_i, pick the first item (if any) with value at least $\tau_i := 2^{\widehat{k}_i} \cdot \widehat{\mu}_{\hat{\jmath}_i}$.

Intuitively, $\widehat{\mathcal{N}}_i$ is the index set of nice intervals among $I_1, \ldots, I_{i-1}$, the number $\widehat{\ell}_i$ is one more than the number of previous nice intervals, and $\hat{\jmath}_i$ is the index of the most recent nice interval. Indeed, these statements are all true when I_{i-1} is high (Lemma 5.2).

The Analysis. It is clear that each procedure picks at most $K = O(\log^* n)$ items.

Lemma 5.1. Struct *has expected value* $\Omega(C^*)$ *unless the following properties both hold:*
(i) If I_i *and* I_j *are high intervals with* $i < j$*, then* $\rho_i > \rho_j$*.*
(ii) There are at most $\frac{K}{4}$ *low intervals.*
Sample *has expected value* C^* *unless the following property holds:*
(iii) The maximum value of any item is less than nC^**.*

Proof. By Lemma 3.1, if either of (i) or (ii) fails then Struct picks an item of value at least C^* with probability $\Omega(1)$, and hence has expected value $\Omega(C^*)$. The second claim is immediate. ∎

Let $j_1, \ldots, j_m$ denote the indices of the nice intervals. Let $N_\ell := I_{j_\ell}$ denote the ℓ-th nice interval and let $\widehat{\mu}(\widehat{\mathcal{N}}_\ell) := \widehat{\mu}_{j_\ell}$ denote its most valuable item. Let $\mathcal{N}_i = \{j_\ell \mid j_\ell < i\}$ denote the set of indices of nice intervals before interval I_i.

LEMMA 5.2. *Suppose that item $g_{\max}$ arrives in $(\frac{3}{4}, 1]$ and property (i) of Lemma 5.1 holds. If interval I_{i-1} is high, then $\widehat{\mathcal{N}}_i = \mathcal{N}_i$. In particular, if I_i is nice, then $I_i = N_{\widehat{\ell}_i}$, and $I_{\hat{\jmath}_i} = N_{\widehat{\ell}_i - 1}$.*

Proof. Since $g_{\max}$ arrives in $(\frac{3}{4}, 1]$, for each j it holds that $\rho_j = \widehat{\mu}_j$. Since I_{i-1} is high, property (i) implies that $\widehat{\mu}_{i-1}$ is minimal among the $\widehat{\mu}$ values of high intervals seen so far. Thus, for $j \le i-1$, interval I_j is nice if and only if $\widehat{\mu}_j \ge \widehat{\mu}_{i-1}$. It follows from definition that $\widehat{\mathcal{N}}_i = \mathcal{N}_i$. ∎

It is convenient to let N_0 denote the (high) interval immediately preceding N_1, so that for all $\ell = 1, \ldots, |\mathcal{N}_i|$, in interval $I_{j_\ell} = N_\ell$, SEARCHII's choice of $\hat{\jmath}_{j_\ell}$ satisfies $I_{\hat{\jmath}_{j_\ell}} = N_{\ell-1}$.

LEMMA 5.3. *Let $K = 2\log^* n + 2$. Suppose that (i)-(iii) of Lemma 5.1 all hold. Then* SEARCHII *has expected value $\Omega(C^*)$.*

Proof. By property (*ii*), the number of nice intervals is at least

$$K - \tfrac{K}{4} - \tfrac{K}{4} - 1 \;=\; \tfrac{K}{2} - 1 \;\ge\; \log^* n.$$

Let ℓ be the minimal number $l \ge 1$ satisfying $\widehat{\mu}(N_l) \ge C^* \log^{(l)} n$. (This is well defined—the inequality is always satisfied by $l = \log^* n$.) We claim that

$$C^* \;\le\; \widehat{\mu}(N_{\ell-1}) \;<\; C^* \log^{(\ell-1)} n.$$

The first inequality follows from the fact that N_ℓ is nice (in particular, it is high). When $\ell = 1$, the second inequality holds by property (*iii*); when $\ell > 1$, it holds by the minimality of ℓ. Therefore, there is some $k \in [-\log^{(\ell)} n, \log^{(\ell)} n]$ such that

$$2^k \widehat{\mu}(N_\ell) \;\le\; C^* \log^{(\ell)} n \;<\; 2^{k+1} \widehat{\mu}(N_\ell).$$

For ease of notation, let $t = j_\ell$ so that $I_t = N_\ell$. Condition on $g_{\max}$ arriving in $(\frac{3}{4}, 1]$—which happens with probability $\frac{1}{4}$—and consider the choice of SEARCHII in interval N_ℓ. Since property (*i*) also holds, Lemma 5.2 applies. In particular, $\widehat{\ell}_t = \ell$ and $\widehat{\mu}_{\hat{\jmath}_t} = \widehat{\mu}_{j_{\ell-1}} = \widehat{\mu}(N_{\ell-1})$. Therefore $\widehat{k}_t$ is drawn from $[-\log^{(\ell)} n, \log^{(\ell)} n]$, and the threshold $\tau_t = 2^{\widehat{k}_t} \cdot \widehat{\mu}(N_{\ell-1})$. With probability $\Omega\Big(\frac{1}{\log^{(\ell)} n}\Big)$, the algorithm chooses $\widehat{k}_t = k$. In this case, since $\widehat{\mu}_t = \widehat{\mu}(N_\ell) \ge C^* \log^{(\ell)} n \ge \tau_t$, SEARCHII will pick an item in I_t. Thus SEARCHII gets expected value at least

$$\tfrac{1}{4} \cdot \Omega(1) \cdot \tfrac{1}{\log^{(\ell)} n} \cdot C^* \log^{(\ell)} n \;=\; \Omega(C^*).$$

Hence the proof. ∎

Proof. [Proof of Theorem 1.5(i)] We now run one of the three algorithms STRUCT, SAMPLE, and SEARCHII uniformly at random. By Lemmas 5.1 and 5.3 the picked algorithm gives expected value at least C^* with probability 1/3, while picking at most $K = O(\log^* n)$ items. Actually selecting a random one of these K items proves the claim. ∎

5.2 Byzantine Secretary for Matroids We consider the setting where the items are the ground set of an arbitrary matroid and the algorithm can choose any independent set of items. The algorithm is given the rank r of the matroid and the number n of items, but only learns the matroid structure as the items arrive. The main result of this section is Theorem 1.5(ii). The factor of $O((\log^* n)^2 \cdot \log r)$ improves upon the previous factor of $O(\log n)$ if the rank of the matroid r is much smaller than the number of items n. For the rest of the section, we assume the benchmark $C^* := \mathrm{OPT}(\mathcal{G} \setminus g_{\max})$ satisfies $C^* \ge 2\mathsf{val}(g_2)$. This is without loss of generality: indeed, we can run the algorithm from §5.1 with probability $\frac{1}{2}$ (which suffices when $C^* \le 2\mathsf{val}(g_2)$) and run the algorithm below with the remaining probability.

The Algorithm. The idea of our algorithm is to use the approach from §5.1 that, at each step, it either gets items of high value, or gets a better estimate of the highest-value items. Define intervals I_0 and I_i, and values ρ_i and $\widehat{\mu}_i$ for each interval as in §3. Again the algorithm will pick one of the following 3 procedures uniformly at random and use it for the whole instance.

Procedure 1. Run the single-item value-maximization algorithm from §5.1.

Procedure 2. The second procedure is adapted from the $O(\log n)$-competitive algorithm of [BGSZ20]. The intuition is as follows: most of the value in OPT comes from items with value in $[\mathsf{val}(g_2)/2r, \mathsf{val}(g_2)]$. If this value interval is partitioned into exponentially-separated value-levels, an average level contains a $\frac{1}{\log r}$ fraction of the value of OPT. The procedure estimates $\mathsf{val}(g_2)$ by the max-value item in a random interval, uses this estimate to choose a random-value interval, and sets a threshold to get the value of this interval. More precisely, the procedure is the following:

Algorithm 8 Procedure I-BGSZ

1: $i \leftarrow$ uniformly random integer in $[K]$
2: $k \leftarrow$ uniformly random integer in $[\log(2r^2)]$
3: $\widehat{\mu}_i =$ max value of an item arriving in I_i
4: After I_i, greedily chose items with value at least $2^{-k}\widehat{\mu}_i$.

Procedure 3. The third procedure attempts to capture the value of large red items. It is simpler than SearchII from §5.1, but similar in spirit:

Algorithm 9 Procedure SearchIII

1: $i \leftarrow$ uniformly random integer in $[K]$
2: $k \leftarrow$ uniformly random integer in $[-\log^{(i)}(n), \log^{(i)}(n)]$
3: $\widehat{\mu}_{i-1} =$ max value of an item arriving in I_{i-1}
4: Pick the first item arriving in (or after) I_i with value at least $2^k\widehat{\mu}_{i-1}$. If no such item arrives, pick nothing.

The Analysis. Similarly to §3, we analyze the algorithm by breaking into different cases based on the following structural properties of the instance.

(i) No item has value more than $n \cdot r \cdot \mathsf{val}(g_2)$.
(ii) $C^* \geq 4\mathsf{val}(g_2)$.
(iii) For every i, it holds that $\rho_i \geq r \cdot \mathsf{val}(g_2)$.

Recall that ρ_i denotes the maximum value of a red item arriving in interval I_i.

Lemma 5.4. *If either of properties (i) or (ii) fail, then the algorithm of §5.1 gets expected value* $\Omega(\frac{C^*}{\log^* n})$.

Proof. Suppose property (i) fails. With probability $\Omega(\frac{1}{\log^* n})$, the algorithm of §5.1 picks a uniformly random item. This item has expected value at least $r \cdot \mathsf{val}(g_2) \geq C^*$, so the algorithm of §5.1 has expected value $\Omega(\frac{C^*}{\log^* n})$.

Now suppose property (ii) fails. Then as proved in §5.1, the algorithm of §5.1 has expected value $\Omega(\frac{\mathsf{val}(g_2)}{\log^* n}) \geq \Omega(\frac{C^*}{\log^* n})$. ∎

Lemma 5.5. *If properties (i) and (ii) hold and property (iii) fails, then* I-BGSZ *has expected value at least* $\Omega(\frac{C^*}{K^2 \log r})$.

Proof. Fix i such that $\rho_i \leq r \cdot \mathsf{val}(g_2)$. Condition on the event that (a) I-BGSZ chooses this i in line 1 and (b) g_2 arrives in I_i but $g_{\max}$ does not, which happens with probability $\frac{1}{K} \cdot \frac{1}{2K}(1 - \frac{1}{2K}) \geq \frac{1}{3K^2}$. Now $\widehat{\mu}_i = \max\{\rho_i, \mathsf{val}(g_2 2)\}$, which implies that

$$\mathsf{val}(g_2) \;\leq\; \widehat{\mu}_i \;\leq\; r \cdot \mathsf{val}(g_2). \tag{5.14}$$

Let OPT denote the items in the optimal solution of green items excluding $g_{\max}$.

Let $L^* = \left\{x \in OPT \mid \mathsf{val}(x) > \frac{\mathsf{val}(g_2)}{2r}\right\}$. Since $|OPT| \leq r$, the items in $OPT \setminus L^*$ have total value at most $\frac{\mathsf{val}(g_2)}{2} \leq \frac{C^*}{2}$. Thus $\mathsf{val}(L^*) \geq \frac{C^*}{2}$. Given the conditioning, all items in $L^* \setminus \{g_2\}$ arrive at independent uniformly

random times. In particular, since $I_i \subseteq [\frac{1}{4}, \frac{3}{4}]$, the expected value of items in L^* that arrive after I_i is at least $\frac{1}{4}(\frac{C^*}{2} - \mathsf{val}(g_2)) \geq \frac{C^*}{16}$ (using property (ii)). The value of each of these items is in $[\frac{\mathsf{val}(g_2)}{2r}, \mathsf{val}(g_2)]$. By (5.14), each of these items has value in some value-level $[2^{-k}\widehat{\mu}_i, 2^{-k+1}\widehat{\mu}_i)$ with $k \in [\log(2r^2)]$. An average level therefore contains at least $\Omega(\frac{C^*}{\log r})$ value from OPT. The greedy algorithm on one such value-level gets at least half of OPT's value in that level, so I-BGSZ gets expected value $\Omega(\frac{C^*}{\log r})$ as well. Since this holds when we condition on an event of probability $\Omega(\frac{1}{K^2})$, the result follows. ∎

LEMMA 5.6. *Let $K = \log^*(n)$. If properties (ii) and (iii) hold, then procedure* SEARCHIII *gets expected value* $\Omega(\frac{C^*}{\log^* n})$

Proof. Fix the minimal index $i \geq 0$ for which $\mu_i \geq r \cdot \log^{(i)} n \cdot \mathsf{val}(g_2)$. Since property (iii) holds, the index $i = \log^*(n)$ satisfies the desired constraint, hence i is well-defined. Furthermore, since property (i) holds, $i \geq 1$. Then we have the following inequalities:

$$r \cdot \mathsf{val}(g_2) \;\leq\; \mu_{i-1} \;\leq\; \log^{(i-1)}(n) \cdot r \cdot \mathsf{val}(g_2),$$

the first by property (ii) and the second by minimality of i. It follows that there is a $k \in [\pm \log^{(i)}(n)]$ such that the value level defined by $2^k\mu_{i-1}$ contains an item of value at least $r \cdot \log^{(i)} n \cdot \mathsf{val}(g_2)$. With probability $\frac{1}{K \cdot 2\log^{(i)}(n)}$, Procedure 3 picks interval i and level k. In this case, it gets value at least

$$r \cdot \log^{(i)} n \cdot \mathsf{val}(g_2) \;\geq\; \log^{(i)} n \cdot C^*.$$

Thus, SEARCHIII gets expected value $\Omega(\frac{C^*}{K}) = \Omega(\frac{C^*}{\log^* n})$. ∎

We now complete the proof of Theorem 1.5(ii).

Proof. [Proof of Theorem 1.5(ii)] Consider the algorithm that uses $K = \log^* n$ and runs one of the above three procedures, each with probability $\frac{1}{3}$. By Lemmas 5.4 to 5.6, at least one of these procedures has expected value $\Omega\big(\frac{C^*}{(\log^* n)^2 \cdot \log r}\big)$. With probability $\frac{1}{3}$, the algorithm picks the right procedure. Hence, the same value guarantee applies up to a constant. ∎

Appendix

A Useful Prior Results

Let $\triangle^{d-1} := \big\{\mathbf{p} \in [0,1]^d \mid \|\mathbf{p}\|_1 = 1\big\}$ be the *probability simplex* in $\mathbb{R}^d$, and $\blacktriangle^d := \big\{\mathbf{p} \in [0,1]^d \mid \|\mathbf{p}\|_1 \leq 1\big\}$ be the *full-dimensional* probability simplex. We recall the full-information online linear optimization (OLO) low regret bound; see, e.g., [AHK12].

LEMMA A.1. (OLO) *Fix $\varepsilon \in (0, 1/2]$. The experts algorithm considers a setting with d experts. At each time the algorithm plays a probability distribution $p^t \in \triangle^{d-1}$ and receives a linear reward function $f_t : \triangle^{d-1} \to [-1, 1]$. For any time τ, let $p^* = \operatorname{argmax}_{p \in \triangle^{d-1}} \sum_{t \leq \tau} f_t(p)$ be fixed action that gives the best reward over the entire input sequence. Until any time τ, the following holds:*

$$\sum_{t \leq \tau} f_t(p^*) - \sum_{t \leq \tau} f_t(p^t) \leq \varepsilon \sum_{t \leq \tau} |f_t(p^*)| + \frac{\log d}{\varepsilon} \quad . \tag{A.1}$$

We will also need the following generalization from Bubeck et al. [BDHN19, Theorem 1] for the full-information multiscale online learning problem.

LEMMA A.2. (MULTI-SCALE REGRET OF BUBECK ET AL. [BDHN19]) *Fix $\varepsilon \in (0, 1]$. The multi-scale experts algorithm considers a setting with M experts. At each time the algorithm plays a probability distribution $p^t \in \triangle_M$ and receives a reward vector r^t with each $r_i^t \in [0, c_i]$; moreover, c_i is known in advance. Let $R_i := \max_i \sum_t r_i^t$ be the reward of action i over the entire input sequence. The following holds for each $i \in [M]$:*

$$R_i - \sum_t \langle r^t, p^t \rangle \leq \varepsilon R_i + O\Big(\frac{c_i \log M}{\varepsilon}\Big) \quad . \tag{A.2}$$

If we pick a random action in $[M]$ at each timestep t independently from the distribution p^t, then the above theorem gives a guarantee for the expected regret against oblivious adversaries as well.

The following concentration inequality is classical: see, e.g., [Ver18, Theorem 2.8.4]:

LEMMA A.3. (BERNSTEIN'S INEQUALITY) *For $X_1, X_2, \ldots, X_n$ independent mean-zero random variables such that $|X_i| \leq M$ for all i, and any $t \geq 0$,*

$$\Pr\Big[\Big|\sum_i X_i\Big| > t\Big] \leq 2\exp\Big(\frac{t^2/2}{\sigma^2 + Mt/3}\Big),$$

where $\sigma^2 = \sum_i \mathbb{E}[X_i^2]$ is the variance of the sum.

Our algorithm for Packing Integer Programs will use the following result about Byzantine Knapsacks (and hence about the multiple-item Byzantine secretary) from [BGSZ20, Theorem 2 and Lemma 7] in case most of the value is concentrated on a small number of items.

LEMMA A.4. *There is an algorithm for Byzantine secretary on knapsacks with size at least $B \geq poly(\varepsilon^{-1}\log n)$ (and items of at most unit size) that is $(1-\varepsilon)$-competitive with the benchmark $\mathrm{OPT}(\mathcal{G} \setminus g_{\max})$. Moreover, there is an algorithm that given an estimate that is at least $\mathrm{OPT}(\mathcal{G})$ and at most $\mathrm{poly}(n)$ times as much, and knapsack size $B \geq poly(\varepsilon^{-1}\log n)$, prodices a solution that is $(1-\varepsilon)$-competitive with the benchmark $\mathrm{OPT}(\mathcal{G})$.*

B Handling Correlations due to Sampling Without Replacement

In contrast to i.i.d. arrivals, to handle correlations due to sampling without replacement for secretary problems, we will use the following general lemma.

LEMMA B.1. *Consider a set of vectors $\{y^1, \ldots, y^m\} \in [0,1]^d$ and let $Y^1, \ldots, Y^k$ be sampled without replacement from this set. Let $Z^1, \ldots, Z^k$ be random vectors in $\blacktriangle^d$ such that Z^j is a (possibly random) function of $Y^1, \ldots, Y^{j-1}$ for all j. Let τ be a stopping time for the sequence $((Y^t, Z^t))_t$ such that $\tau \leq \frac{m}{2}$. Then for any $\varepsilon \in (0, \frac{1}{10}]$ and $\delta \in (0,1]$, with probability at least $1-\delta$ we have*

$$\sum_{j \leq \tau} \langle Z^j, Y^j \rangle \;\leq\; (1+4\varepsilon) \sum_{j \leq \tau} \langle \mathbb{E}_{j-1} Z^j, \mathbb{E} Y^j \rangle \;+\; \frac{O(\log d/\delta)}{\varepsilon} \;,$$

where $\mathbb{E}_{j-1} Z^j = \mathbb{E}[Z^j \mid (Y_1, Z_1), \ldots, (Y_{j-1}, Z_{j-1})]$.

Special cases of the above lemma have appeared before in the literature, e.g. of [GM16, Lemma 5]. We will need the following convenient concentration inequality for "martingales with drift".

LEMMA B.2. (LEMMA 2.2 OF [BAN19]) *Let $X_1, X_2, \ldots, X_k$ be a sequence of (possibly dependent) random variables with values in $(-\infty, 1]$ and such that there is $\alpha \in (0,1)$ such that*

$$\mathbb{E}[X_j \mid X_1, \ldots, X_{j-1}] \leq -\alpha \mathbb{E}[X_j^2 \mid X_1, \ldots, X_{j-1}]$$

for all j. Then for all $\lambda \geq 0$

$$\Pr(X_1 + \ldots + X_k > t) \leq e^{-\alpha\lambda}.$$

We also need a maximal Bernstein's inequality for sampling without replacement.It follows by applying Lemma 1 of [GM16] to the scaled random variables $\frac{X_i}{M} \in [0,1]$ and using the fact that $\mathrm{Var}(X) \leq \mathbb{E}X$ for every random variable in $[0,1]$ (the last inequality follows from the inequality $\frac{a}{b+c} \geq \min\{\frac{a}{2b}, \frac{a}{2c}\}$, valid for all non-negative reals a, b, c).

LEMMA B.3. (LEMMA 1 OF [GM16]) *Consider a set of real values $x_1, \ldots, x_m$ in $[0, M]$, and let $X_1, \ldots, X_k$ be sampled without replacement from this collection. Assume $k \leq m/2$. Let $S_i = X_1 + \ldots X_i$. Also let $\mu = \frac{1}{m}\sum_i x_i$ and $\sigma^2 = \frac{1}{m}\sum_i (x_i - \mu)^2$. Then for every $\alpha > 0$*

$$\Pr\left(\max_{i \leq k} |S_i - i\mu| \geq \alpha\right) \leq 30\exp\left(-\frac{(\alpha/24)^2}{M(2k\mu + (\alpha/24))}\right) \leq 30\exp\left(-\min\left\{\frac{(\alpha/24)^2}{4k\mu M} \;,\; \frac{\alpha}{48M}\right\}\right)$$

Let $\mathcal{F}_j$ be the σ-algebra generated by $Y^1, \ldots, Y^j$ and $Z^1, \ldots, Z^j$, i.e., the history up to time j. We use $\mathbb{E}_{j-1}[\cdot] := \mathbb{E}[\,\cdot \mid \mathcal{F}_{j-1}]$ to denote expectation conditioned on the history up to time $j-1$.

LEMMA B.4. *Consider $i \in [d]$. Then with probability at least $1 - \frac{\delta}{d}$ we have $\mathbb{E}_{j-1} Y_i^j \le (1+2\varepsilon)\mathbb{E} Y_i^j + \frac{O(\log d/\delta)}{m\varepsilon}$ for all $j \le \frac{m}{2}$ simultaneously.*

Proof. Let $\mu = \frac{1}{m}\sum_{j \le m} y_i^j$, which is the expected value of Y_i^j. Moreover, the conditional expectation $\mathbb{E}_{j-1} Y_i^j$ is the average of the y_i^t's that have not appeared up until time $j-1$, namely

$$\mathbb{E}_{j-1} Y_i^j = \frac{\sum_t y_i^t - \sum_{t \le j-1} Y_i^t}{m - (j-1)} = \frac{m\mu - \sum_{t \le j-1} Y_i^t}{m - (j-1)}. \tag{B.3}$$

We then bound the last term uniformly for all $j \le \frac{m}{2}$ using the maximal Bernstein's inequality Lemma B.3.

For that let $\sigma^2 := \frac{1}{m}\sum_t (y_i^t - \mu)^2$ and notice that

$$\sigma^2 \;=\; \frac{1}{m}\sum_t (y_i^t)^2 - \mu^2 \;\le\; \frac{1}{m}\sum_t y_i^t \;=\; \mu,$$

where the inequality uses $y_i^t \in [0,1]$. Applying Lemma B.3 with

$$\alpha := \varepsilon m\mu + \frac{2 \cdot (24)^2}{\varepsilon}\left(\log d/\delta + \log 30\right)$$

we get, since $\alpha^2 \ge 4m\mu\,(24)^2\,(\log d/\delta + \log 30)$,

$$\begin{aligned}\Pr\left(\max_{j \le m/2} |\textstyle\sum_{t \le j} Y_i^j - j\mu| \ge \alpha\right) &\le 30 \exp\left(-\min\left\{\frac{4m\mu(\log d/\delta + \log 30)}{2m\mu}\;,\; \frac{2(24)^2(\log d/\delta + \log 30)/\varepsilon}{48}\right\}\right)\\ &\le 30 e^{-(\log d/\delta + \log 30)} \;\le\; \frac{\delta}{d}.\end{aligned}$$

Finally, whenever this event holds, equation (B.3) gives that for all $j \le m/2$

$$\mathbb{E}_{j-1} Y_i^j \le \frac{(m-(j-1))\,\mu + \alpha}{m-(j-1)} \le \mu + \frac{\varepsilon m\mu + O(\frac{\log d/\delta}{\varepsilon})}{m-(j-1)} \le (1+2\varepsilon)\mu + \frac{O(\log d/\delta)}{m\varepsilon},$$

the last inequality using $j \le \frac{m}{2}$. This concludes the proof. ∎

Proof. [Proof of Lemma B.1] Since Z^t and Y^j are independent conditioned on $\mathcal{F}_{j-1}$, we have

$$\mathbb{E}_{j-1}\langle Z^j, Y^j\rangle \;=\; \langle \mathbb{E}_{j-1} Z^j, \mathbb{E}_{j-1} Y^j\rangle$$

Moreover, applying a union bound on Lemma B.4 over all coordinates i, with probability at least $1 - \frac{\delta}{2}$ for all $j \le \frac{m}{2}$ (in particular for all $j \le \tau$) we have $\langle \mathbb{E}_{j-1} Z^j, \mathbb{E}_{j-1} Y^j\rangle \le (1+2\varepsilon)\langle \mathbb{E}_{j-1} Z^j, \mathbb{E} Y^j\rangle + \frac{O(\log d/\delta)}{m\varepsilon}$. Adding over all $j \le \tau$ we get that

$$\sum_{j \le \tau} \mathbb{E}_j\langle Z^j, Y^j\rangle \le (1+2\varepsilon)\sum_{j \le \tau}\langle \mathbb{E}_{j-1} Z^j, \mathbb{E} Y^j\rangle + \frac{O(\log d/\delta)}{\varepsilon} \qquad \text{with probability} \ge 1 - \frac{\delta}{2}. \tag{B.4}$$

We now show using Lemma B.2 that with good probability the desired quantity $\sum_{j \le \tau}\langle Z^j, Y^j\rangle$ is close to $\sum_{j \le \tau}\mathbb{E}_j\langle Z^j, Y^j\rangle$. Define the stopped random variable

$$X_j := \mathbf{1}(\tau \ge j) \cdot \Big[(1-\varepsilon)\langle Z^j, Y^j\rangle - \mathbb{E}_j\langle Z^j, Y^j\rangle\Big].$$

Recall that by definition of stopping time, the event $\mathbf{1}(\tau \geq j)$ is $\mathcal{F}_{j-1}$-measurable, and hence $\mathbb{E}_j X_j = \mathbf{1}(\tau \geq j) \cdot (-\varepsilon \, \mathbb{E}_j \langle Z^j, Y^j \rangle)$. Moreover,

$$\begin{aligned} \mathbb{E}_j X_j^2 &= \mathbf{1}(\tau \geq j) \cdot \Big[(1-\varepsilon)^2 \, \mathbb{E}_j \underbrace{\langle Z^j, Y^j \rangle^2}_{\leq \langle Z^j, Y^j \rangle} - \underbrace{(2(1-\varepsilon)-1)}_{\geq 0} (\mathbb{E}_j \langle Z^j, Y^j \rangle)^2 \Big] \\ &\leq \mathbf{1}(\tau \geq j) \cdot \mathbb{E}_j \langle Z^j, Y^j \rangle, \end{aligned}$$

where the first underbrace is because $\langle Z^j, Y^j \rangle \leq 1$ and the second because $\varepsilon \in (0, \frac{1}{2}]$. Together, these observations give

$$\mathbb{E}_j X_j \leq -\varepsilon \, \mathbb{E}_j X_j^2.$$

Then applying Lemma B.2 to the sequence $(X_j)_j$ with $\lambda = \frac{\log 1/2\delta}{\varepsilon}$ we obtain

$$\Pr\left((1-\varepsilon) \sum_{j \leq \tau} \langle Z^j, Y^j \rangle \;>\; \sum_{j \leq \tau} \mathbb{E}_j \langle Z^j, Y^j \rangle + \frac{\log 1/2\delta}{\varepsilon} \right) \;\leq\; \frac{\delta}{2}.$$

Then by union bound with (B.4), with probability at least $1 - \delta$ we have

$$\sum_{j \leq \tau} \langle Z^j, Y^j \rangle \;\leq\; \frac{(1+2\varepsilon)}{(1-\varepsilon)} \sum_{j \leq \tau} \langle \mathbb{E}_{j-1} Z^j, \mathbb{E} Y^j \rangle + \frac{O(\log d/\delta)}{\varepsilon}.$$

Verifying that $\frac{(1+2\varepsilon)}{(1-\varepsilon)} \leq 1 + 4\varepsilon$ for all $\varepsilon \in (0, \frac{1}{10}]$ then proves Lemma B.1.

∎

C Missing Proofs from Section 2

LEMMA 2.1. (REDUCTION TO SMOOTH INSTANCES) *Suppose $B \geq \Omega(\mathrm{poly}(\log n))$. Given an algorithm to solve Packing LPs in the Byzantine Secretary setting that with constant probability is ρ-competitive w.r.t.* $\mathrm{OPT}(\mathcal{G})$ *for all smooth instances, we can obtain an algorithm which is $\Omega(\rho)$-competitive in expectation for all instances w.r.t.* $\mathrm{OPT}(\mathcal{G} \setminus g_{\max})$.

Proof. The desired $O(\rho)$-approximation for general instances is given by choosing uniformly at random and running one of the following 3 algorithms:

1. Pick one of the n items uniformly at random
2. Run the algorithm given by Lemma A.4 aiming at picking the best B items
3. See the largest value $c_{\max}^{1/2}$ of an item with arrival time in $[0, \frac{1}{2})$ (do not pick any) and on the remaining times $[\frac{1}{2}, 1]$ run an algorithm that is ρ-competitive in a smooth instance with constant probability using $c^{1/2}$ as estimate for $\mathrm{OPT}(\mathcal{G})$ (still using budget B).

By construction this procedure always produces a feasible solution, and we show it has good value in expectation.

Let $c_{\max}$ be the maximum value of over all (green and red) items. First, if $c_{\max} \geq \frac{n}{2} \mathrm{OPT}(\mathcal{G} \setminus g_{\max})$ then the procedure has expected value at least $\frac{1}{3} \mathrm{OPT}(\mathcal{G})$ just from the first algorithm that it may run, and the result follows. Also, if the B top valued green items (excluding $g_{\max}$) have combined value at least $\frac{\mathrm{OPT}(\mathcal{G} \setminus g_{\max})}{4}$, then the procedure gets expected value at least $\Omega(\mathrm{OPT}(\mathcal{G} \setminus g_{\max}))$ just from the second algorithm that it may run, and the result also follows.

So consider the "remaining situation" where neither of these cases happen. Further, condition on the event where $g_{\max}$ shows up at a time $[0, \frac{1}{2})$ and $\mathrm{OPT}(\mathcal{G} \cap [\frac{1}{2}, 1])$ (the optimal value considering only green items on times $[\frac{1}{2}, 1]$) it at least $\frac{1}{2} \mathrm{OPT}(\mathcal{G} \setminus g_{\max})$, which happens with probability at least $\frac{1}{4}$. In this case the instance over times $[\frac{1}{2}, 1]$ satisfies both items of the smoothness Assumption 2.1 with $\widehat{O} := c_{\max}^{1/2}$ because:

1. Item 1: The B top valued green items in $[\frac{1}{2}, 1]$ have combined value at most $\frac{\mathrm{OPT}(\mathcal{G} \setminus g_{\max})}{4} \leq \frac{\mathrm{OPT}(\mathcal{G} \cap [\frac{1}{2}, 1])}{2}$. Since this set of items contains all items in $[\frac{1}{2}, 1]$ of value at least $\frac{\mathrm{OPT}(\mathcal{G} \cap [\frac{1}{2}, 1])}{B}$, it satisfies Item 1 of the assumption.

2. Item 2: Using $c(g_{\max})$ to denote the value of $g_{\max}$,

$$c_{\max}^{1/2} \;\geq\; c(g_{\max}) \;\geq\; \frac{1}{n}\,\mathrm{OPT}(\mathcal{G}) \;\geq\; \frac{1}{n}\,\mathrm{OPT}(\mathcal{G} \cap [\tfrac{1}{2}, 1])$$
$$c_{\max}^{1/2} \;\leq\; \frac{n}{2}\,\mathrm{OPT}(\mathcal{G} \setminus g_{\max}) \;\leq\; n\,\mathrm{OPT}(\mathcal{G} \cap [\tfrac{1}{2}, 1]).$$

Thus, under this event, with probability $\frac{1}{3}$ the third algorithm within the procedure is run and with further constant probability it is guaranteed to obtain expected value at least $\Omega(\rho\,\mathrm{OPT}(\mathcal{G} \cap [\frac{1}{2}, 1])) \geq \Omega(\rho\,\mathrm{OPT}(\mathcal{G} \setminus g_{\max}))$. Overall, in this "remaining situation" the procedure obtains expected value at least $\Omega(\rho\,\mathrm{OPT}(\mathcal{G} \setminus g_{\max}))$, thus concluding the proof. ∎

C.1 Proof of Lemma 2.3 Recall that $\mathcal{G}_{\mathrm{T}}(I) \subseteq \mathbb{N}$ denotes the steps t where the t-th item in the interval I is green. We first argue that the value of the green items in any interval I is large, with high probability. Recall that x^* is the optimal solution consisting only of green items each of whose value is at most $\frac{OPT}{B}$, and has total value at least $\frac{OPT}{2}$.

Claim C.1. *If $|I| = 1/K$ and $B \geq K \log 1/\delta$, then with probability at least $1 - \frac{\delta}{2}$ we have*

$$\sum_{t \in \mathcal{G}_{\mathrm{T}}(I)} C_t x_t^* \;\geq\; \frac{1}{4}\frac{OPT}{K}.$$

Proof. The LHS is $\sum_{i \in \mathcal{G}} c_i x_i^* \cdot \mathbf{1}(i \in I)$ and has expectation at least $\frac{1}{2}\frac{\mathrm{OPT}}{K}$ and variance

$$\mathrm{Var}\left(\sum_{i \in \mathcal{G}} c_i x_i^* \cdot \mathbf{1}(i \in I)\right) \;\leq\; \frac{1}{K}\sum_{i \in \mathcal{G}} \left(c_i x_i^*\right)^2 \;\leq\; \frac{\mathrm{OPT}}{BK}\sum_{i \in \mathcal{G}} c_i x_i^* \;\leq\; \frac{\mathrm{OPT}^2}{BK}\;,$$

where the second inequality uses that $x_i^* \in [0, 1]$ and that, by definition, $x_i^* > 0$ only when item i has value $c_i \leq \frac{OPT}{B}$. Then applying Bernstein's Inequality (Lemma A.3) to $X_i := c_i x_i^*(\mathbf{1}(i \in I) - |I|)$,

$$\Pr\left(\sum_{i \in \mathcal{G}} c_i x_i^* \cdot \mathbf{1}(i \in I) \leq \frac{1}{2}\frac{\mathrm{OPT}}{K} - \frac{1}{4}\frac{\mathrm{OPT}}{K}\right) \;\leq\; 2e^{-\frac{3B}{64K}}\;,$$

and the result follows from the assumption $B \geq \Omega(K \log 1/\delta)$. ∎

Next we argue that the total cost in the Lagrangified value is not large. The proof of this claim goes by first conditioning on the stochastic times of the green items inside interval I. This fixes the order in which we see all the items, and also fixes the number of greens in I. Now at each stochastic time in I, we draw an item from the remaining greens (without replacement), and use Lemma B.1 to bound the effect of sampling without replacement.

Claim C.2. *If $|I| = \frac{1}{K} \leq \frac{1}{4}$, and $B \geq \Omega(K \log(4d/\delta))$, then with probability at least $1 - \frac{\delta}{2}$,*

$$\sum_{t \in \mathcal{G}_{\mathrm{T}}(I)} \langle \lambda_t, A_t x_t^* \rangle \;\leq\; \frac{4B}{K}\;.$$

Proof. Recall that $\mathcal{G}(I)$ and $G(I)$ denote the set and the number of green items, respectively, that fall in interval I. Let t_j be the position of the j^{th} green item to appear in interval I. Then $\sum_{t \in \mathcal{G}_{\mathrm{T}}(I)} \langle \lambda_t, A_t x_t^* \rangle = \sum_{j \leq G(I)} \langle \lambda_{t_j}, A_{t_j} x_{t_j}^* \rangle$. Notice that the t_j's are random.

We condition on $G(I) = k$, say, and on their positions $(t_1, \ldots, t_k) =: t_{\le k}$. Notice that under this conditioning the stochastic items (C_{t_j}, A_{t_j}) are still sampled without replacement from the green items. Moreover, notice that λ_{t_j} is a function of the green items before the j^{th} green item in the interval, and therefore on $(C_{t_1}, A_{t_1}), \ldots, (C_{t_{j-1}}, A_{t_{j-1}})$. (It also depends on the red items in I, but these are deterministic.)

In order to upper bound $\sum_{j \le G(I)} \langle \lambda_{t_j}, A_{t_j} x^*_{t_j} \rangle$ with high-probability (conditioned on $G(I) = k$ and $t_{\le k}$) we use Lemma B.1. For that, we first notice that due to the feasibility of x^* we have $\mathbb{E}[A_{t_j} x^*_{t_j} \mid G(I) = k,\, t_{\le k}] \le \frac{B \cdot \mathbb{1}_d}{G}$ for all j, and hence

$$\sum_{j \le k} \Big\langle \lambda_{t_j}, \mathbb{E}\big[A_{t_j} x^*_{t_j} \;\big|\; G(I) = k, t_{\le k}\big] \Big\rangle \le \sum_{j \le k} \langle \lambda_{t_j}, \tfrac{B \cdot \mathbb{1}_d}{G} \rangle = \frac{B}{G} k.$$

Then applying Lemma B.1 conditionally (setting $Z^j := \lambda_{t_j}|_{G(I)=k, t_{\le k}}$, $Y^j := A_{t_j} x^*_{t_j}|_{G(I)=k, t_{\le k}}$, and $\varepsilon = \frac{1}{10}$) gives that if $k \le \frac{G}{2}$ then

$$\Pr\left(\sum_j \langle \lambda_{t_j}, A_{t_j} x^*_{t_j} \rangle \ge \frac{3B}{2G}\, k \;+\; \Omega(\log {}^{4d}\!/_{\delta}) \;\middle|\; G(I) = k,\, t_{\le k} \right) \;\le\; {}^{\delta}\!/_{4} \;.$$

Taking expectation over the $t_{\le k}$'s and then over the values of k that are at most $\frac{2G}{K}$ (which is at most ${}^{G}\!/_{2}$ by our assumption on $K \ge 4$), we get

$$\Pr\left(\sum_j \langle \lambda_{t_j}, A_{t_j} x^*_{t_j} \rangle \ge \frac{3B}{K} \;+\; \Omega(\log {}^{4d}\!/_{\delta}) \;\middle|\; G(I) \le {}^{2G}\!/_{K} \right) \;\le\; {}^{\delta}\!/_{4} \;. \tag{C.5}$$

Observe that ${}^{3B}\!/_{K} + \Omega(\log {}^{4d}\!/_{\delta}\}) \le {}^{4B}\!/_{K}$ by our assumption $B \ge \Omega(K \log(4d/\delta))$. Next we show that the conditioning holds with high probability. Indeed, $\mathbb{E}[G(I)] = {}^{G}\!/_{K}$, so by Bernstein's inequality

$$\Pr(G(I) > {}^{2G}\!/_{K}) \;\le\; e^{-\frac{G}{2K}} \;\le\; e^{-\frac{B}{4K}} \;\le\; {}^{\delta}\!/_{4d} \;\le\; {}^{\delta}\!/_{4} \;, \tag{C.6}$$

where the second inequality uses Assumption 2.1 that the green items with value at most $\frac{\mathrm{OPT}}{B}$ contain a solution of value at least $\frac{\mathrm{OPT}}{2}$ (hence there are at least $\frac{B}{2}$ green items to obtain the remaining value $\frac{\mathrm{OPT}}{2}$) and the third inequality uses the assumption that $B \ge \Omega(K \log(4d/\delta))$. Combining (C.5) and (C.6),

$$\Pr\left(\sum_j \langle \lambda_{t_j}, A_{t_j} x^*_{t_j} \rangle \ge \frac{4B}{K} \right) \;\le\; \delta/2 \;,$$

which completes the proof of Claim C.2. ∎

Finally, using Claims C.1 and C.2 and taking a union bound concludes the proof of Lemma 2.3.

D Missing Proofs from Section 4

D.1 Proof that Assumption 4.1 is WLOG The following lemma formalizes the idea that Assumption 4.1 can be made without loss of generality.

LEMMA D.1. *Let $\widetilde{\mathrm{Alg}}$ be an algorithm for packing linear programs in the Prophets with Augmentations model. Suppose that $\widetilde{\mathrm{Alg}}$ achieves expected value at least $\Omega(\mathrm{OPT}_{base})$ on instances where each distribution is supported on values that are at most $\frac{\mathrm{OPT}_{base}}{20}$. Then there is an algorithm that achieves expected value $\Omega(\mathrm{OPT}_{base})$ on arbitrary instances.*

Proof. Throughout this section, we use $V_1, \ldots, V_n$ to denote the outcome of the value of the items of the original instance $((a_t, \mathcal{D}_t)_t, B)$, $R_1, \ldots, R_n$ as the augmentations performed by the adversary, and $C_t = V_t + R_t$ the final value revealed to the algorithm.

For notational convenience let $M := \frac{\mathrm{OPT}_{base}}{40}$. The idea is to run two algorithms, one over the items that have value at most M and one over items of value above M. To make this precise, define the operation $\mathsf{trunc}(v) := \min\{v, M\}$ that truncates a value at M. Let $\widetilde{\mathcal{D}}_t$ be the distribution of the truncated random variable $\mathsf{trunc}(V_t)$. Then we consider the algorithm that flips an unbiased coin runs one of the following procedures on the (augmented version of) the original instance $((a_t, \mathcal{D}_t)_t, B)$:

1. Alg_{low}: It sends the chopped instance $((a_t, \widetilde{\mathcal{D}})_t, B)$ to the algorithm $\widetilde{\text{Alg}}$ to obtain a selection policy, and apply this policy to the sequence of truncated values $\mathsf{trunc}(C_1), \ldots, \mathsf{trunc}(C_n)$ to decide which items to take. Let $X_t \in \{0,1\}$ denote the indicator whether this policy picked the t^{th} item.

2. Alg_{high}: It picks the first items with value C_t above M, if any. Let $\tau \in [n]$ be the index of the item picked ($\tau = \infty$ if did not pick any).

We claim that either Alg_{low} or Alg_{high} has value at least $\Omega(\text{OPT}_{base})$, which then proves the lemma. For that, let $p := \Pr(\tau < \infty)$ be the probability that some item has value above M. If $p \geq \frac{1}{2}$, then Alg_{high} already has expected value least $pM \geq \Omega(\text{OPT}_{base})$. We henceforth assume that $p < \frac{1}{2}$.

Let OPT_{trunc} be the optimal value of the truncated instance $((a_t, \widetilde{\mathcal{D}})_t, B)$, namely

$$\text{OPT}_{trunc} = \mathbb{E} \max_x \left\{ \sum_t \mathsf{trunc}(V_t) \cdot x_t \ : \ \sum_t a_t x_t \leq B \cdot \mathbb{1}_d \,,\, x \in \{0,1\}^n \right\}.$$

We consider two cases:

Case 1: $\text{OPT}_{trunc} \geq \frac{\text{OPT}_{base}}{2}$**.** In every scenario the value of Alg_{low} is

$$\text{Alg}_{low} = \sum_t C_t X_t \geq \sum_t \mathsf{trunc}(C_t) \cdot X_t. \tag{D.7}$$

Moreover, notice that the sequence $\mathsf{trunc}(C_1), \ldots, \mathsf{trunc}(C_n)$ can be seen as an augmented version of the truncated instance $((a_t, \widetilde{\mathcal{D}})_t, B)$, where the adversary performed the augmentation $\widetilde{R}_t := \mathsf{trunc}(C_t) - \mathsf{trunc}(V_t)$ that is non-negative and only depends on V_t. Moreover, in this Case 1 we have that all values in the truncated instance are at most $M = \frac{\text{OPT}_{base}}{40} \leq \frac{\text{OPT}_{trunc}}{20}$. Therefore, the approximation guarantee of $\widetilde{\text{Alg}}$ holds in this case hence

$$\mathbb{E} \sum_t \mathsf{trunc}(C_t) \cdot X_t \geq \Omega(\text{OPT}_{trunc}) \geq \Omega(\text{OPT}_{base}),$$

where the last inequality again uses the assumption of Case 1. Combined with Equation (D.7) this gives that Alg_{low} has expected value at least $\Omega(\text{OPT}_{base})$ as desired.

Case 2: $\text{OPT}_{trunc} < \frac{\text{OPT}_{base}}{2}$**.** Let $X^* \in \{0,1\}^n$ be the optimal solution for the original base instance $((a_t, \mathcal{D}_t)_t, B)$, namely $\text{OPT}_{base} = \mathbb{E} \sum_t V_t X_t^*$. Since

$$V_t \ = \ \mathsf{trunc}(V_t) + (V_t - \mathsf{trunc}(V_t)) \cdot \mathbf{1}(V_t > M),$$

we get

$$\begin{aligned}
\text{OPT}_{base} &= \mathbb{E} \sum_t \mathsf{trunc}(V_t) \cdot X_t^* + \mathbb{E} \sum_t (V_t - \mathsf{trunc}(V_t)) \cdot \mathbf{1}(V_t > M) \cdot X_t^* \\
&\leq \text{OPT}_{trunc} + \mathbb{E} \sum_t V_t \cdot \mathbf{1}(V_t > M) \\
&= \text{OPT}_{trunc} + \sum_t \mathbb{E}[V_t \mid V_t > M] \ \Pr(V_t > M),
\end{aligned}$$

where the inequality follows from the fact that X^* is a feasible solution for OPT_{trunc}. Moreover, since we are in Case 2, the second term in the RHS must contribute to at least half of OPT_{base}, namely

$$\text{OPT}_{base} \ \leq \ 2 \sum_t \mathbb{E}[V_t \mid V_t > M] \ \Pr(V_t > M). \tag{D.8}$$

Now notice that we can express the value of Alg_{high} in every scenario as

$$\text{Alg}_{high} = C_\tau \geq V_\tau = \sum_t \mathbf{1}(\tau \geq t) \cdot \mathbf{1}(V_t > M) \cdot V_t,$$

so using the fact that τ is a stopping time (so $\mathbf{1}(\tau \geq t)$ is defined by the history up to time $t-1$) and that the V_t's are independent, the expected value becomes

$$\mathbb{E}\,\mathrm{Alg}_{high} = \sum_t \Pr(\tau \geq t) \cdot \mathbb{E}[V_t \mid V_t > M]\ \Pr(V_t > M) \geq \frac{1}{2} \sum_t \mathbb{E}[V_t \mid V_t > M]\ \Pr(V_t > M),$$

the inequality following because the probability of $\tau < t$ is at most the probability p that any item takes value above M, and because we have assumed $p \leq \frac{1}{2}$. Comparing with (D.8) we see that $\mathbb{E}\,\mathrm{Alg}_{high} \geq \Omega(\mathrm{OPT}_{base})$ as desired. This concludes the proof. □

D.2 Proof of Lemma 4.1 We consider the following the concave relaxation to the base prophet instance, where intuitively x_t denotes the fraction of times item t is picked by the optimal offline algorithm:

$$\begin{aligned} \max_{x_1,\ldots,x_n} \quad & \sum_t x_t \cdot \mathbb{E}[V_t \mid V_t \text{ is in its top } x_t\text{-quantile of } \mathcal{D}_t] \\ \text{s.t.} \quad & \sum_t a_t x_t \leq {}^{B}\!/_{4} \cdot \mathbb{1}_d \\ & 0 \leq x_t \leq 1 \ . \end{aligned}$$

We assume that the distributions are continuous, so that the quantiles are well-defined; this is without loss of generality (see, e.g., [RWW20, §2]).

Let x_t^* denote the optimal solution of the relaxation, and let $\mathsf{val}_t := \mathbb{E}[V_t \mid V_t \text{ is in its top } x_t^*\text{-quantile}]$. To prove that this relaxation's objective value $\sum_t x_t^* \mathsf{val}_t$ is at least $\frac{\mathrm{OPT}_{base}}{4}$, observe that if an item t is picked for x_t fraction of times by the offline optimal solution, its contribution to the offline objective is at most $\mathbb{E}[V_t \mid V_t \text{ is in its top } x_t\text{-quantile}]$. Note that although the relaxation only allows budget ${}^{B}\!/_{4} \cdot \mathbb{1}_d$ (instead of $B \cdot \mathbb{1}_d$), this only hurts the relaxation's objective by at most a factor of 4.

Next we design the desired solution $\psi_1(V_1), \ldots, \psi_n(V_n)$ for the base prophet instance with expected: We pick item t whenever its value V_t is in the top x_t^*-quantile of $\mathcal{D}_t$, that is, $\psi_t(V_t) = \mathbf{1}(V_t \text{ is in the top } x_t^*\text{-quantile of } \mathcal{D}_t)$. The expected budget consumed by such an algorithm is at most $\sum_t a_t x_t^* \leq {}^{B}\!/_{4} \cdot \mathbb{1}_d$, which proves Item 2. To prove Item 1, note that the algorithm's expected value is precisely the objective value $\sum_t x_t^* \mathsf{val}_t$ of the relaxation above, which is at least $\frac{\mathrm{OPT}_{base}}{4}$.

References

[AD15] Shipra Agrawal and Nikhil R. Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of SODA*, pages 1405–1424, 2015.

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[AHL12] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce, EC*, pages 18–35, 2012.

[Ala14] Saeed Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. *SIAM Journal on Computing*, 43(2):930–972, 2014.

[Ban19] Nikhil Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1125–1135, 2019.

[BDHN19] Sébastien Bubeck, Nikhil R. Devanur, Zhiyi Huang, and Rad Niazadeh. Multi-scale online learning: Theory and applications to online auctions and pricing. *J. Mach. Learn. Res.*, 20:62:1–62:37, 2019.

[BGSZ20] Domagoj Bradac, Anupam Gupta, Sahil Singla, and Goran Zuzic. Robust algorithms for the secretary problem. In *11th Innovations in Theoretical Computer Science Conference, ITCS*, pages 32:1–32:26, 2020.

[BIKK18] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Matroid secretary problems. *J. ACM*, 65(6):35:1–35:26, 2018.

[Dyn63] Eugene B Dynkin. The optimum choice of the instant for stopping a markov process. In *Soviet Math. Dokl*, volume 4, 1963.

[EKM18] Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Allocation with traffic spikes: Mixing adversarial and stochastic models. *ACM Trans. Econ. Comput.*, 6(3-4), 2018.

[FKS21] Michal Feldman, Thomas Kesselheim, and Sahil Singla. Tutorial on "Prophet Inequalities and Implications to Pricing Mechanisms and Online Algorithms". `http://www.thomas-kesselheim.de/tutorial-prophet-inequalities`, EC, 2021.

[FSZ15] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple $O(\log \log(\text{rank}))$-competitive algorithm for the matroid secretary problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1189–1201, 2015.

[GKRS20] Paritosh Garg, Sagar Kale, Lars Rohwedder, and Ola Svensson. Robust algorithms under adversarial injections. In *47th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 56:1–56:15, 2020.

[GKT19] Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. In *Proceedings of the Thirty-Second Conference on Learning Theory, COLT*, pages 1562–1578, 2019.

[GM16] Anupam Gupta and Marco Molinaro. How the experts algorithm can help solve lps online. *Math. Oper. Res.*, 41(4):1404–1431, 2016.

[GS20] Anupam Gupta and Sahil Singla. Random-order models. *Beyond the Worst-Case Analysis of Algorithms*, page 234, 2020.

[ISW20] Nicole Immorlica, Sahil Singla, and Bo Waggoner. Prophet inequalities with linear correlations and augmentations. In *21st ACM Conference on Economics and Computation, EC*, pages 159–185, 2020.

[KKN15] Thomas Kesselheim, Robert D. Kleinberg, and Rad Niazadeh. Secretary problems with non-uniform arrival order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 879–888, 2015.

[Kle05] Robert D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2005.

[KM20] Thomas Kesselheim and Marco Molinaro. Knapsack secretary with bursty adversary. In *47th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 72:1–72:15, 2020.

[KMZ15] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 889–898. ACM, 2015.

[KRTV14] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal beats dual on online packing LPs in the random-order model. In *Proceedings of Symposium on Theory of Computing, STOC*, pages 303–312, 2014.

[KS77] Ulrich Krengel and Louis Sucheston. Semiamarts and finite values. *Bull. Am. Math. Soc*, 1977.

[KW12] Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 123–136, 2012.

[Lac14] Oded Lachish. O(log log rank) competitive ratio for the matroid secretary problem. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 326–335, 2014.

[LMPL18] Thodoris Lykouris, Vahab S. Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 114–122, 2018.

[Luc17] Brendan Lucier. An economic view of prophet inequalities. *SIGecom Exch.*, 16(1):24–47, 2017.

[Mey01] Adam Meyerson. Online facility location. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 426–431. IEEE, 2001.

[MGZ12] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1690–1701, 2012.

[Mol17] Marco Molinaro. Online and random-order load balancing simultaneously. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1638–1650, 2017.

[Mol21] Marco Molinaro. Robust algorithms for online convex problems via primal-dual. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2078–2092, 2021.

[RS17] Aviad Rubinstein and Sahil Singla. Combinatorial prophet inequalities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.

[Rub16] Aviad Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 324–332, 2016.

[RWW20] Aviad Rubinstein, Jack Z. Wang, and S. Matthew Weinberg. Optimal single-choice prophet inequalities from samples. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS*, pages 60:1–60:10, 2020.

[SC84] Ester Samuel-Cahn. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *the Annals of Probability*, pages 1213–1216, 1984.

[Ver18] Roman Vershynin. *High-dimensional probability*, volume 47 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2018.