IOE 691: Approximation & Online Algorithms

Lecture Notes: Stochastic Optimization

Instructor: Viswanath Nagarajan

In many situations involving uncertain input, there is some prior information available, for example from historical data. In such cases, using an online model may be overly pessimistic because it completely ignores the prior information and prepares for the worst possible input. Stochastic optimization is a popular approach to model such applications, where we work with a (known) probability distribution over inputs. Solutions to stochastic problems are "policies" or decision trees that map the current state (i.e., all decisions and observations so far) to the next decision.

1 Framework

In *stochastic* combinatorial optimization, some of the input parameters are *random variables* with known probability distributions. While the algorithm does know the distribution of each such random variable X, the actual outcome of X is unknown until a corresponding decision is made. We will assume that all random variables are *independent*. An algorithm makes decisions sequentially and observes realizations associated with these decisions. Crucially, the algorithm can react and be *adaptive* to the observed random outcomes. Therefore, any solution is a sequential decision process, which can be represented by a decision tree. (Solutions in the stochastic setting are also referred to as *policies*.) The goal now is to optimize the *expected* objective while satisfying some constraints. We will compare our algorithms to an optimal policy, and measure the algorithm's performance using the approximation ratio. For a minimization objective, this is:

$$\sup_{\text{instance } I} \frac{\text{expected objective of the algorithm on } I}{\text{expected objective of the optimal policy on } I}.$$

Note that an instance I for a stochastic problem also specifies the probability distributions of all its random parameters.

To formalize these concepts, consider a generic stochastic problem involving n binary decisions indexed $i \in \{1, 2, \cdots n\}$. Let O denote the space of possible random outcomes for any single decision. For example, if each decision has a random (integer) cost between 0 and C then $O = \{0, 1, \cdots C\}$. The random outcome corresponding to any decision $i \in [n]$ is only observed if and when the policy selects i (i.e., a "yes" decision is made on i). In order to define a solution/policy, we need to introduce the *state space*, which consists of the decisions made so far along with their *observed* random outcomes. So the state represents the current information at any point in the solution. The state space is denoted $\Omega = (O \cup \{*\})^n$. We say that a policy is currently in state $\omega \in \Omega$ if for each decision $i \in [n]$:

- $\omega_i = *$ if and only if decision i has not been selected so far, and
- if $\omega_i \neq *$ then the random outcome observed at decision i is ω_i.

Adaptive Policies. A solution or policy is a mapping $\pi : \Omega \to [n]$ which specifies the next selection decision to be made at any state. In particular, if the solution is currently at state ω then it will

next select decision $\pi(\omega)$. Note that this definition allows the policy to fully utilize all the observed random outcomes. So, such a policy is said to be *adaptive*. Moreover, an adaptive policy may even require exponential space to describe: this is because the number of states $|\Omega|$ is exponential in the number of decisions n. Therefore, we are interested in obtaining *efficiently computable* policies π, where given any state ω the next decision $\pi(\omega)$ can be computed in polynomial time. Note that the running time of an efficient policy is always polynomial (under any set of random outcomes). We note that the optimal policy may not be efficiently computable. Nevertheless, we will compare the performance of our efficient policies to the optimal adaptive policy.

Decision tree representation. It is often convenient (for analysis) to view an adaptive policy explicitly as a decision tree. Nodes in the decision tree represent the complete *sequence* of prior decisions and outcomes. Branches in the decision tree correspond to the observed random outcome of the policy's current decision. Note that each node corresponds to a 'state' by just ignoring the sequence in which prior decisions were made. When the policy is implemented, it follows a random path from the root of the decision tree to some leaf (based on the observed random outcomes).

Non-adaptive Policies. One drawback of adaptive policies is that they require incremental computation before each decision, which may be too slow for certain applications (even if the policy runs in polynomial time). An alternative approach is to use *non-adaptive* policies that specify upfront a static sequence σ of the n decisions. The non-adaptive policy then makes selection decisions in the *fixed* order σ until some (simple) stopping rule. The stopping rule is usually very efficient to check; for example, checking if the total reward exceeds a threshold. As minimal computation is needed between decisions, non-adaptive policies can be implemented very efficiently. For some stochastic problems, we will focus on obtaining good non-adaptive policies (relative to the optimal non-adaptive policy). The table below summarizes these two types of policies.

Table 1: Comparing adaptive and non-adaptive policies.

	optimal value	running time
Adaptive	better	worse
Non-adaptive	worse	better

Adaptivity Gap. Surprisingly, for many stochastic problems we can achieve the best of both worlds: non-adaptive policies that have objective value comparable to the optimal adaptive policy. In order to quantify the relative strength of these two types of policies, we define the *adaptivity gap* to be the worst-case ratio between the expected objectives of optimal adaptive and non-adaptive policies. For a minimization objective,

$$\text{adaptivity gap} = \sup_{\text{instance } I} \frac{\text{optimal non-adaptive value on } I}{\text{optimal adaptive value on } I}.$$

2 Series Testing

We first consider a fundamental (and simple) stochastic problem. There are n independent components in some system (e.g., car or airplane). Each component $i \in [n]$ may fail independently with probability (w.p.) p_i. We also use $q_i := 1 - p_i$ to denote the probability of working. We use the

random variable (r.v.) X_i to denote the status of each component:

$$X_i = \begin{cases} 0 & \text{if component } i \text{ working} \\ 1 & \text{if component } i \text{ failed} \end{cases}, \qquad \forall i \in [n].$$

In order to determine the status of any component i, one needs to perform a test that incurs cost c_i. The goal is to determine if there is *any* failed component. Equivalently, we want to determine whether/not $\sum_{i=1}^{n} X_i \geq 1$. This is called *series testing* because it corresponds to systems that only function when all their components are working: such a system can be viewed as one where the n components are connected in series. It is often possible to determine this without having to test all components. Therefore, a natural objective is to minimize the *expected* cost of testing.

Any policy for series testing is just a permutation of the components: tests will be performed until a failure is observed (or all components are tested). So, these are *non-adaptive* (the stopping rule is observing a failure).

Greedy policy. A simple algorithm for series testing is to perform tests in increasing order of $\frac{c_i}{p_i}$. It turns out that this is optimal.

Theorem 2.1 *The greedy policy is optimal for series testing.*

Proof: Notice that any policy for series testing is merely given by a permutation π of the n components: the policy then performs tests in that sequence until some failure. Let us re-number the components so that

$$\frac{c_1}{p_1} \leq \frac{c_2}{p_2} \leq \cdots \frac{c_n}{p_n}.$$

So, the greedy ordering is just $\langle 1, 2, \cdots n \rangle$. We will prove that the greedy ordering is optimal by an exchange argument. Suppose that the optimal ordering π contains a pair of consecutive tests b, a such that $\frac{c_b}{p_b} \geq \frac{c_a}{p_a}$; i.e., $\pi = \langle I, b, a, O \rangle$ where $I \cup O = [n] \setminus \{a, b\}$. Then, consider swapping a and b to obtain ordering $\pi' = \langle I, a, b, O \rangle$. We will show that $\mathbb{E}[cost(\pi')] \leq \mathbb{E}[cost(\pi)]$. Using this argument iteratively, we can show that greedy is optimal (left as exercise). We have:

$$\begin{aligned} \mathbb{E}[cost(\pi')] - \mathbb{E}[cost(\pi)] &= c_a \left(\Pr[\pi' \text{ tests } a] - \Pr[\pi \text{ tests } a] \right) + c_b \left(\Pr[\pi' \text{ tests } b] - \Pr[\pi \text{ tests } b] \right) \\ &= c_a \prod_{i \in I} q_i \cdot (1 - q_b) \; + \; c_b \prod_{i \in I} q_i \cdot (q_a - 1) \; = \; \prod_{i \in I} q_i \cdot (c_a p_b - c_b p_a) \; \leq \; 0. \end{aligned}$$

We use the fact that a policy tests a component only if all previous components are working.

■

For any realization $X = \langle X_1, \cdots X_n \rangle$, we define

$$\phi(X) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} X_i = 0 \\ 1 & \text{if } \sum_{i=1}^{n} X_i \geq 1 \end{cases},$$

which denotes the outcome of testing. Given a policy π and realization X, we use $c(\pi, X)$ to denote the total cost of testing under realization X when policy π is used. We actually have the following stronger result.

Theorem 2.2 *The greedy policy minimizes* $\mathbb{E}[c(\pi, X) | \phi(X) = 1]$ *over all policies* π.

Proof: For any policy π, we have by conditional expectations,

$$\begin{aligned}\mathbb{E}[c(\pi,X)] &= \mathbb{E}[c(\pi,X)|\phi(X)=1]\cdot\Pr[\phi(X)=1] + \mathbb{E}[c(\pi,X)|\phi(X)=0]\cdot\Pr[\phi(X)=0] \\ &= \mathbb{E}[c(\pi,X)|\phi(X)=1]\cdot\Pr[\phi(X)=1] + \Pr[\phi(X)=0]\cdot\sum_{i=1}^{n} c_i\end{aligned}$$

The last equality uses the fact that if $\phi(X)=0$ then any policy must test all components. Notice that only the term $\mathbb{E}[c(\pi,X)|\phi(X)=1]$ depends on the chosen policy. Therefore, minimizing $\mathbb{E}[c(\pi,X)]$ is equivalent to minimizing $\mathbb{E}[c(\pi,X)|\phi(X)=1]$. The result now follows from Theorem 2.1. ■

3 Testing k-of-n Functions

We now consider a more general problem, where the goal is to identify whether/not there are at least k failed components. In particular, we want to evaluate the function:

$$\psi(X) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} X_i \le k-1 \\ 1 & \text{if } \sum_{i=1}^{n} X_i \ge k \end{cases},$$

where X denotes the realization.

For $k>1$, we can no longer assume that all policies are non-adaptive (as we did for series testing). Indeed, adaptive policies may perform strictly better that non-adaptive ones as shown next.

Adaptivity gap example. Consider $n=3$ and $k=2$ with r.v.s $X_1=1$ w.p. $\frac{1}{2}$, $X_2=1$ w.p. ϵ and $X_3=1$ w.p. $1-\epsilon$. Here, $\epsilon\to 0$. Notice that ψ is just the "majority" function in this case. The optimal adaptive policy is as follows:

1. Test X_1.
2. If $X_1=0$ then test X_2 followed by X_3 (only if $X_2=1$).
3. If $X_1=1$ then test X_3 followed by X_2 (only if $X_2=0$).

The expected cost is $2+\epsilon$.

On the other hand, any non-adaptive policy costs at least 2.5. This can be checked directly. This implies that the adaptivity gap is at least 1.25.

We first consider the (simpler) task of verifying the function value, where we condition on $\psi(X)=1$ and need to find k failed components. In this setting, it turns out that the greedy (non-adaptive) policy is still optimal. Then, we discuss the evaluation problem that does not know ψ, for which we provide an optimal adaptive policy.

Verification problem. We will show that the greedy policy that tests components in the $1,2,\cdots n$ order (which is increasing in c_i/p_i) is optimal. Formally,

Theorem 3.1 *The greedy policy minimizes* $\mathbb{E}[c(\pi,X)|\psi(X)=1]$ *over all policies* π *that verify* $\psi(X)=1$ *(i.e., find* k *failures).*

Proof: We first claim that there is an optimal policy that tests component 1 w.p. one. Let π be an optimal policy that minimizes the above conditional expectation. We *mark* all states in π where $k-1$ failures have been observed. Consider any marked state ω and let $T \subseteq [n]$ be the already-tested components at this state. The remaining task at ω is to find one failure among $[n] \setminus T$ (conditioned on some failure occurring). Using Theorem 2.2, it follows that the greedy policy on $[n] \setminus T$ minimizes the expected testing cost, conditioned on ω. We ensure this property at all marked states ω, by modifying policy π if necessary. Note that π is still optimal. Now, at any marked state ω, we either have $1 \in T$ or 1 will be the next component tested at ω. In either case, component 1 will be tested by policy π. Finally, observe that some marked state must be reached under every realization X because we conditioned on $\psi(X) = 1$. Hence, component 1 is tested by policy π w.p. one.

Given the above property, we can modify the optimal policy π by testing component 1 first. This does not increase the expected cost as π always tests component 1.

We can now complete the proof by induction on n. If $X_1 = 0$ then policy π needs to minimize the conditional expectation of finding k failures among $[n] \setminus \{1\}$: by induction the greedy policy $\langle 2, 3, \cdots n \rangle$ is optimal for this. If $X_1 = 1$ then policy π needs to minimize the conditional expectation of finding $k-1$ failures among $[n] \setminus \{1\}$: again, $\langle 2, 3, \cdots n \rangle$ is optimal by induction. So, the greedy policy $\langle 1, 2, \cdots n \rangle$ minimizes the conditional expectation for the instance on n components. ■

Let σ denote the permutation such that:

$$\frac{c_{\sigma(1)}}{q_{\sigma(1)}} \leq \frac{c_{\sigma(2)}}{q_{\sigma(2)}} \leq \cdots \frac{c_{\sigma(n)}}{q_{\sigma(n)}}.$$

Note that this is the greedy ordering to find $n-k+1$ working components, which would prove $\psi(X) = 0$. Using the "complements" $1 - X_i$ (and appropriately changed k), Theorem 3.1 implies the following.

> The non-adaptive policy σ minimizes $\mathbb{E}[c(\pi, X) | \psi(X) = 0]$ over all policies π that verify $\psi(X) = 0$ (i.e., find $n-k+1$ working components).

Below, we refer to the order $\langle 1, 2, \cdots n \rangle$ as list L_1 as it is used to verify $\psi(X) = 1$. Similarly the order σ is called list L_0 as it verifies $\psi(X) = 0$. For any list L and integer ℓ, we use $L[\ell]$ to denote the ℓ-prefix of L, i.e., the first ℓ components in L.

The adaptive policy for evaluation. We now consider the k-of-n evaluation problem, where $\psi(X)$ must be determined. We describe the adaptive policy $\mathcal{A}([n], k)$ in a recursive manner. It combines both lists L_0 and L_1 as follows.

1. Select any $i \in L_1[k] \cap L_0[n-k+1]$ and observe X_i.
2. If $X_i = 0$ recurse on $\mathcal{A}([n] \setminus i, k)$.
3. If $X_i = 1$ recurse on $\mathcal{A}([n] \setminus i, k-1)$.

Notice that the lists do not need to be re-computed for each step: we simply drop component i from both L_0 and L_1 to obtain the lists for the recursive instances. We will show:

Theorem 3.2 *The above adaptive policy is optimal for k-of-n evaluation.*

Lemma 3.1 *For any realization X such that $\mathcal{A}$ observes k failed components (with $X_i = 1$) and*

$t \geq 0$ working components (with $X_i = 0$), the components tested by $\mathcal{A}$ are precisely $L_1[k+t]$. Hence, $\mathbb{E}[c(\mathcal{A}, X)|\psi(X) = 1] = \mathbb{E}[c(L_1, X)|\psi(X) = 1]$.

Proof: Let i denote a component tested in the adaptive policy $\mathcal{A}$. If $X_i = 0$ then the relevant prefixes for the recursive instance $\mathcal{A}([n] \setminus i, k)$ are $L_1[k+1] \setminus i$ and $L_0[n-k+1] \setminus i$: so the L_1 prefix increases by one but the L_0 prefix remains the same. Similarly, if $X_i = 1$ then the relevant prefixes for the recursive instance $\mathcal{A}([n] \setminus i, k-1)$ are $L_1[k] \setminus i$ and $L_0[n-k+2] \setminus i$: so the L_0 prefix increases by one but the L_1 prefix remains the same. Using this fact inductively, if follows that the components tested under realization X are precisely $L_1[k+t]$ because we observe $X_i = 0$ exactly t times. This proves the first statement.

For the second statement, notice that for any such realization X, we must have $\sum_{i=1}^{k+t} X_i = k$: otherwise $\mathcal{A}$ would not stop after testing $L_1[k+t]$. Moreover, $\sum_{i=1}^{k+t-1} X_i = k-1$ and $X_{k+t} = 1$, as otherwise $\mathcal{A}$ would have stopped sooner. Hence, the components tested by the L_1 greedy policy under X are exactly $L_1[k+t]$. Finally, every realization X with $\psi(X) = 1$ has the form stated in the lemma (for some $t \geq 0$). So, $\mathbb{E}[c(\mathcal{A}, X)|\psi(X) = 1] = \mathbb{E}[c(L_1, X)|\psi(X) = 1]$. ■

Similarly,

Lemma 3.2 *For any realization X such that $\mathcal{A}$ observes $n-k+1$ working components (with $X_i = 0$) and $t \geq 0$ failed components (with $X_i = 1$), the components tested by $\mathcal{A}$ are precisely $L_0[n-k+1+t]$. Hence, $\mathbb{E}[c(\mathcal{A}, X)|\psi(X) = 0] = \mathbb{E}[c(L_0, X)|\psi(X) = 0]$.*

Let π^* be an optimal policy for k-of-n. We can now complete the proof of Theorem 3.2:

$$\begin{aligned}
\mathbb{E}[c(\mathcal{A}, X)] &= \mathbb{E}[c(\mathcal{A}, X)|\psi(X) = 1] \cdot \Pr[\psi(X) = 1] + \mathbb{E}[c(\mathcal{A}, X)|\psi(X) = 0] \cdot \Pr[\psi(X) = 0] \\
&= \mathbb{E}[c(L_1, X)|\psi(X) = 1] \cdot \Pr[\psi(X) = 1] + \mathbb{E}[c(L_0, X)|\psi(X) = 0] \cdot \Pr[\psi(X) = 0] \\
&\leq \mathbb{E}[c(\pi^*, X)|\psi(X) = 1] \cdot \Pr[\psi(X) = 1] + \mathbb{E}[c(\pi^*, X)|\psi(X) = 0] \cdot \Pr[\psi(X) = 0] \\
&= \mathbb{E}[c(\pi^*, X)].
\end{aligned}$$

Here, we used that the optimal policy π^* verifies both $\psi(X) = 1$ and $\psi(X) = 0$: so it can be used in Lemma 3.1 for *both* $\psi(X) = 1$ and $\psi(X) = 0$.

4 Maximum Value Problem

In the maximum value problem (MVP), there are n non-negative random variables (r.v.s) $X_1, \cdots X_n$ with known distributions. In order to observe the value of any r.v., an algorithm needs to *probe* it. Moreover, for each probed r.v., the algorithm needs to immediately accept or reject it (after seeing its value). The algorithm may probe any subset of r.v.s, but it can accept only *one* r.v. The goal is to probe the r.v.s sequentially (possibly adaptively) and make reject/accept decisions so as to maximize the expected value of the accepted r.v. We will assume that all the r.v.s have continuous distributions. (With additional work, the results also extend to discrete distributions.)

A non-adaptive policy involves probing r.v.s in a fixed order π, along with a stopping rule that decides when to accept. An adaptive policy may vary the order of probing based on observed outcomes. It can be shown that:

Theorem 4.1 *The adaptivity gap of the maximum value problem is one.*

So, it suffices to focus on non-adaptive policies. We will use the following simple *upper bound* on

the optimal value of any policy:

$$E^* := \mathbb{E}[X_{max}] = \mathbb{E}\left[\max_{i=1}^{n} X_i\right].$$

Note that E^* is the best value achievable even in the "clairvoyant" setting where one is allowed to observe *all* the r.v.s before accepting. For this reason, E^* is also called the *prophet* bound. We will provide non-adaptive policies that achieve a good approximation relative to E^*. Such relations are called *prophet inequalities*: they bound the performance of a sequential algorithm relative to the prophet.

Furthermore, we will focus on "single threshold" policies. These are non-adaptive policies (i.e., given by some permutation π) with a threshold τ that corresponds to a very simple stopping rule: accept the first r.v. in π that exceeds τ.

Arbitrary probing order. Our first policy actually works for *any* probing order, say $1, 2, \cdots n$. The algorithm doesn't even need to know the order. This policy uses threshold τ such that:

$$\Pr[X_{max} > \tau] = \frac{1}{2}. \tag{1}$$

Theorem 4.2 *The non-adaptive policy that considers r.v.s in an arbitrary order and uses threshold τ has expected value at least $\frac{1}{2} \cdot E^*$.*

Proof: We re-number the r.v.s so that they are probed in the order $X_1, X_2, \cdots X_n$. Let $\mathcal{E}_i$ denote the event that $\max_{j=1}^{i-1} X_j \leq \tau$, which corresponds to probing r.v. X_i. We can write the policy's objective as:

$$\begin{aligned}
\mathbb{E}[\mathsf{ALG}] &= \tau \cdot \Pr[\mathsf{ALG} \text{ accepts some r.v.}] + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+ \wedge \mathbf{1}(\mathcal{E}_i)\right] \\
&= \tau \cdot \Pr[X_{max} > \tau] + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+ \wedge \mathbf{1}(\mathcal{E}_i)\right] \\
&= \tau \cdot \Pr[X_{max} > \tau] + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+\right] \cdot \Pr[\mathcal{E}_i] \qquad (2) \\
&\geq \tau \cdot \Pr[X_{max} > \tau] + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+\right] \cdot \Pr\left[X_{max} \leq \tau\right] \qquad (3) \\
&= \frac{1}{2}\left(\tau + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+\right]\right). \qquad (4)
\end{aligned}$$

Above, (2) uses the fact that $\mathcal{E}_i$ is independent of X_i and (3) uses $\Pr[\mathcal{E}_i] \geq \Pr[X_{max} \leq \tau]$. Finally, (4) uses the choice of threshold τ from (1). We can now relate to E^* using:

$$E^* \leq \tau + \mathbb{E}\left[(X_{max} - \tau)^+\right] \leq \tau + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+\right]. \tag{5}$$

The theorem now follows. ∎

It can also be shown that if the ordering is fixed (i.e., cannot be changed by the algorithm) then no policy can achieve expected value more than $\frac{1}{2} \cdot E^*$.

Random probing order. We now consider the setting where the policy probes r.v.s in a uniformly random order. We will show that one can obtain a better approximation ratio. Again, the policy uses a single threshold, this time given by τ such that

$$\Pr[X_{max} > \tau] = 1 - \frac{1}{e}. \tag{6}$$

Theorem 4.3 *The non-adaptive policy that probes r.v.s in a uniformly random order with threshold τ from (6) has expected value at least $(1 - \frac{1}{e}) \cdot E^*$.*

Proof: Let σ denote a u.a.r. permutation of $[n]$. We use $j \prec i$ to denote that r.v. X_j appears before r.v. X_i in the order σ. Let $\mathcal{E}_i$ denote the event that $\max_{j \prec i} X_j \leq \tau$, which corresponds to probing r.v. X_i when we use order σ. By (2),

$$\mathbb{E}[\mathsf{ALG}] = \tau \cdot \Pr[X_{max} > \tau] + \sum_{i=1}^{n} \mathbb{E}\left[(X_i - \tau)^+\right] \cdot \Pr[\mathcal{E}_i].$$

We will show that $\Pr[\mathcal{E}_i] \geq 1 - \frac{1}{e}$ for each $i \in [n]$. Combined with (5) and (6), this would imply the theorem.

We now focus on bounding $\Pr[\mathcal{E}_i]$. It will be convenient to represent permutation σ as follows. Each $j \in [n]$ picks a number $t_j \in [0, 1]$ u.a.r. and independently. (There are no ties with probability one.) Then, σ orders the r.v.s by increasing t_j values. We will show that

$$\Pr[\mathcal{E}_i | t_i = t] \geq e^{-t}, \quad \forall t \in [0, 1], \quad \forall i \in [n]. \tag{7}$$

This would imply

$$\Pr[\mathcal{E}_i] = \int_{t=0}^{t} \Pr[\mathcal{E}_i | t_i = t] dt \geq \int_{t=0}^{t} e^{-t} dt = 1 - \frac{1}{e},$$

as desired.

To prove (7), we fix $i \in [n]$ and $t \in [0, 1]$, and condition on $t_i = t$. Then, we can write the event

$$(\mathcal{E}_i | t_i = t) = \wedge_{j \neq i} \left((X_j \leq \tau) \vee (t_j > t)\right).$$

Letting $a_j := \Pr[X_j \leq \tau]$, it follows that

$$\begin{aligned} \Pr[\mathcal{E}_i | t_i = t] &= \prod_{j \neq i} (1 - t \cdot a_j) \geq \prod_{j=1}^{n} (1 - t \cdot a_j) = e^{\sum_{j=1}^{n} \ln(1 - t a_j)} \\ &\geq e^{t \sum_{j=1}^{n} \ln(1 - a_j)} = e^{-t}. \end{aligned}$$

Above, the inequality uses the fact that $\ln(1 - t \cdot y) \geq t \cdot \ln(1 - y)$ for all $t, y \in [0, 1]$. The last equality uses $\sum_{j=1}^{n} \ln(1 - a_j) = -1$, which follows from the choice of threshold (6) and the fact that $\Pr[X_{max} > \tau] = 1 - \prod_{j=1}^{n} (1 - a_j) = 1 - e^{\sum_{j=1}^{n} \ln(1 - a_j)}$. This completes the proof of (7) and hence the theorem. ∎

5 Constrained Maximum Value (Non Adaptive)

We now consider a variant where there is a limit on the number of *probed* r.v.s. Formally, the input consists of n independent non-negative r.v.s $X_1, \cdots X_n$ and a bound $k \leq n$. We will assume that

all r.v.s X_i are discrete and described explicitly via a list of values along with their probabilities. With some extra work, the results can be extended to continuous distributions as well.

An algorithm is allowed to probe at most k r.v.s. The goal is to maximize the expected maximum value among the probed r.v.s. Here, we focus on the non-adaptive setting, where the set of probed r.v.s is fixed upfront. In contrast, an adaptive solution may select the next r.v. to probe based on the outcomes of previous r.v.s: we will consider adaptive policies later. So, the non-adaptive problem can be expressed as:

$$\max_{S\subseteq[n]:|S|\leq k} \quad \mathbb{E}\left[\max_{i\in S} X_i\right]. \tag{8}$$

Unlike the unconstrained maximum value problem (seen earlier), here we do not require the algorithm to make immediate accept/reject decisions on each probed r.v. So, the objective value is just the maximum over all probed r.v.s.

A naive approach to solving (8) is to probe the k r.v.s with largest means $\mathbb{E}[X_i]$. However, as the following example shows, the approximation ratio of this algorithm is at least $\Omega(k)$. Consider k r.v.s each of which has value k w.p. $\frac{1}{k}$ (and value 0 otherwise), and k other r.v.s that have value 2 (w.p. 1). The mean of the first type r.v.s is 1 whereas the mean of the second type r.v.s is 2. The above algorithm probes all k r.v.s of the second type, and obtains objective value 2. On the other hand, the solution that probes all k r.v.s of the first type has objective $k\cdot\left(1-(1-1/k)^k\right) \geq (1-\frac{1}{e})\cdot k$.

Our approach is to handle the stochastic objective in (8) by re-formulating it as a different (and more complex) deterministic problem. In particular, we define a set function $f : 2^{[n]} \to \mathbb{R}_+$ where

$$f(S) := \mathbb{E}\left[\max_{i\in S} X_i\right], \quad \forall S \subseteq [n]. \tag{9}$$

Observe that the stochastic aspect is completely captured in the definition of f, and (8) reduces to $\max_{S:|S|\leq k} f(S)$. Crucially, the value $f(S)$ can be calculated exactly for any subset S.

Lemma 5.1 *For any $S \subseteq [n]$, the value $f(S)$ can be computed in time polynomial in n and the maximum support-size of the r.v.s $\{X_i\}_{i=1}^n$.*

Proof: Let $\mathcal{G}$ denote the union of the support of all r.v.s $\{X_i\}_{i=1}^n$. Let $0 < v_1 < v_2 < \cdots v_g$ denote the (sorted) values in $\mathcal{G}$; also let $v_0 = 0$. Note that $g = |\mathcal{G}|$ is polynomial in n and the maximum support-size. For each r.v. X_i and value v_j, let $q_{ij} := \Pr[X_i < v_j]$. For any $S \subseteq [n]$, we have

$$\begin{aligned} f(S) &= \mathbb{E}\left[\max_{i\in S} X_i\right] = \sum_{j=1}^{g}(v_j - v_{j-1})\Pr\left[\exists i \in S : X_i \geq v_j\right] \\ &= \sum_{j=1}^{g}(v_j - v_{j-1})\left(1 - \prod_{i\in S} q_{ij}\right). \end{aligned} \tag{10}$$

Clearly, this can be computed in time polynomial in n and g. ∎

We now show that function f is monotone and submodular.

Lemma 5.2 *For any subsets $A \subseteq B \subseteq [n]$, we have $f(A) \leq f(B)$.*

Proof: This follows directly from the definition (9) because all r.v.s are non-negative. ∎

Lemma 5.3 *For any subsets $A \subseteq B \subseteq [n]$ and $e \in [n]$, we have*

$$f(A \cup e) - f(A) \geq f(B \cup e) - f(B).$$

Proof: We make use of the description of f in (10). Recall the definitions of values $\{v_j\}_{j=1}^{g}$ and the probabilities q_{ij} for $i \in [n]$ and $j \in [g]$. For any $S \subseteq [n]$, we have

$$f(S) = \sum_{j=1}^{g} (v_j - v_{j-1}) \left(1 - \prod_{i \in S} q_{ij}\right) = \sum_{j=1}^{g} (v_j - v_{j-1}) \phi_j(S),$$

where $\phi_j(S) := 1 - \prod_{i \in S} q_{ij}$ for all j and $S \subseteq [n]$. We will show that for each j, function ϕ_j satisfies the property in the lemma, i.e.,

$$\phi_j(A \cup e) - \phi_j(A) \geq \phi_j(B \cup e) - \phi_j(B), \quad \forall A \subseteq B, \ \forall e \in [n]. \tag{11}$$

By taking a non-negative linear combination of these inequalities (with coefficient $v_j - v_{j-1}$ for ϕ_j), this would prove the lemma.

In order to prove (11), fix any j, subsets $A \subseteq B \subseteq [n]$ and $e \in [n]$. Let $a = \prod_{i \in A} q_{ij}$, $b = \prod_{i \in B} q_{ij}$ and $q = q_{ej}$. As $A \subseteq B$ and all the $q_{ij} \in [0, 1]$, we obtain $a \geq b$. Hence,

$$\begin{aligned} &\phi_j(A \cup e) - \phi_j(A) - \phi_j(B \cup e) + \phi_j(B) = a - aq + bq - b \\ &= (a - b)(1 - q) \ \geq \ 0, \end{aligned}$$

where the inequality uses $a \geq b$ and $q \leq 1$. ■

Using the above properties of f, we can apply the greedy algorithm for maximizing a monotone submodular function over a cardinality constraint, to obtain:

Theorem 5.1 *There is a $1 - \frac{1}{e}$ approximation for non-adaptive constrained maximum value.*

6 Constrained Maximum Value (Adaptive)

We now consider the *adaptive* constrained maximum value problem and obtain a constant-factor approximation algorithm. Through this, we will also introduce a general technique for solving *adaptive* stochastic optimization problems. The idea is to write a linear program (LP) relaxation for adaptive policies, using variables of the form $x_i = \Pr[\text{policy chooses decision } i]$. Although such an LP does not exactly represent adaptive policies, it suffices to obtain approximately optimal policies for many problems. Interestingly, we can often obtain non-adaptive policies by "rounding" such LPs, which also provides a bound on the adaptivity gap.

Recall the constrained maximum value problem. Given n independent r.v.s $X_1, \cdots X_n$ (non-negative and discrete valued) and a bound $k \leq n$, the goal is to probe at most k r.v.s so as to maximize the expected maximum value among the probed r.v.s. Here, we focus on the *adaptive* setting, where the solution may use the outcomes of previously probed r.v.s to decide on the next r.v. to probe. This differs from the non-adaptive setting studied in the previous section. It is also known that the adaptivity gap is at least some constant $c > 1$.

Let $\mathcal{G}$ denote the set of all possible values taken by the r.v.s. As the r.v.s are discrete, $\mathcal{G}$ is a finite set. For any $i \in [n]$ and $v \in \mathcal{G}$ let $p_{iv} = \Pr[X_i = v]$. Our algorithms will require running time polynomial in n and $|\mathcal{G}|$.

Linear program relaxation for adaptive policies. Representing an arbitrary adaptive policy requires exponential space because the state space is exponential in n. (The state space corresponds to the subset of already-probed r.v.s along with their observed values.) Nevertheless, we can write a polynomial-size relaxation that uses variables corresponding to the *marginal probabilities* of various decisions. In particular, let

$$x_i = \Pr[\text{adaptive policy probes r.v. } X_i], \quad \forall i \in [n].$$

For any $i \in [n]$ and value $v \in \mathcal{G}$, let

$$y_{iv} = \Pr\left[\text{policy observes } X_i = v \bigwedge X_i \text{ is the maximum observed value}\right].$$

In case of ties in the maximum observed value, we break ties arbitrarily so that exactly one X_i has the maximum observed value.

We now have the following linear program (LP) relaxation.

$$\max \sum_{i=1}^{n} \sum_{v \in \mathcal{G}} v \cdot y_{iv} \tag{12}$$

$$\sum_{i=1}^{n} x_i \leq k, \tag{13}$$

$$y_{iv} \leq p_{iv} \cdot x_i, \quad \forall i \in [n], \forall v \in \mathcal{G} \tag{14}$$

$$\sum_{i=1}^{n} \sum_{v \in \mathcal{G}} y_{iv} \leq 1, \tag{15}$$

$$0 \leq x \leq 1,\ y \geq 0.$$

Lemma 6.1 *The optimal value of the above LP is at least the optimal adaptive value.*

Proof: Let $\mathcal{A}$ denote an optimal adaptive policy. Define the following r.v.s associated with the policy. For each $i \in [n]$ and $v \in \mathcal{G}$,

$$Z_i := \mathbf{1}[\mathcal{A} \text{ probes } i] \ \text{ and } \ Y_{iv} := \mathbf{1}[\mathcal{A} \text{ probes } i, \text{ observes } X_i = v \text{ and } i \text{ is the maximizer}].$$

We say that i is the maximizer if X_i is the maximum value among all probed r.v.s (we break ties arbitrarily, say by prioritizing larger indexed r.v.s). Consider the fractional solution $x_i = \mathbb{E}[Z_i]$ and $y_{iv} = \mathbb{E}[Y_{iv}]$ for all i and v. Note that the expected objective of $\mathcal{A}$ is just $\mathbb{E}[\sum_i \sum_v v \cdot Y_{iv}]$ which is the LP objective. We will show that it is feasible in the above LP.

Clearly, $\sum_{i=1}^{n} Z_i \leq k$ as $\mathcal{A}$ cannot probe more that k r.v.s. Moreover, $\sum_i \sum_v Y_{iv} \leq 1$ as only one index is declared to be the maximizer. Taking expectations, it follows that LP constraints (13) and (15) are satisfied. Finally, $Y_{iv} \leq Z_i \cdot \mathbf{1}[X_i = v]$, which yields

$$y_{iv} = \mathbb{E}[Y_{iv}] \leq \mathbb{E}[Z_i \cdot \mathbf{1}[X_i = v]] = \mathbb{E}[Z_i] \cdot \Pr[X_i = v] = x_i \cdot p_{iv},$$

where we use the fact that Z_i (decision to probe i) is independent of X_i: this is because the decision to probe i only depends on the values of *previously* probed r.v.s. This shows that LP constraint (14) is satisfied and completes the proof. ■

Our algorithm first solves this LP to obtain a fractional solution (x, y). Then, it "rounds" the fractional solution into a policy for the stochastic problem. In fact, we will obtain a (randomized)

non-adaptive policy. Such a policy considers r.v.s in some fixed order π. When r.v. X_i is considered, the policy randomly decides whether/not to probe it. At the end of the policy, its objective is the maximum probed value. In order to be feasible, the number of probed r.v.s must always be at most k. We first provide a policy that considers r.v.s in an arbitrary order. Then, we provide a better policy that considers r.v.s in a uniformly random order.

6.1 Arbitrary order policy

Let $\alpha \in (0, 1]$ denote a constant value that will be fixed later. For each $i = 1, 2, \cdots n$:

If less than k r.v.s have been probed so far, then probe X_i w.p. $\alpha \cdot x_i$.

We will show that:

Theorem 6.1 *The above algorithm is a $\frac{1}{8}$-approximation algorithm for adaptive constrained maximum value.*

A key part of the analysis is in lower bounding the maximum value of the probed r.v.s, which is not a linear function. To this end, we explicitly *mark* one probed r.v. and use the value of this marked r.v. as a lower-bound on the maximum value. Algorithm 1 describes the marking process along with the rest of the algorithm.

Algorithm 1 Adaptive constrained maximum value

for each $i \in [n]$ **do**
  set $R_i \leftarrow 1$ w.p. αx_i (and $R_i \leftarrow 0$ otherwise)
  if less than k r.v.s have been probed **and** $R_i = 1$, probe X_i
  let $X_i = v$ be the observed value (happens w.p. p_{iv})
  set $S_{iv} \leftarrow 1$ w.p. $\frac{y_{iv}}{x_i p_{iv}}$ (and $S_{iv} \leftarrow 0$ otherwise)
  if no r.v. has been marked **and** $S_{iv} = 1$ then **mark** $\langle X_i, v\rangle$

Clearly, the expected value of the algorithm is at least the expected *marked* value, i.e.,

$$\mathsf{ALG} \geq \sum_{i=1}^{n} \sum_{v \in \mathcal{G}} v \cdot \Pr\left[\langle X_i, v\rangle \text{ marked}\right].$$

We will prove that

$$\Pr\left[\langle X_i, v\rangle \text{ marked}\right] \;\geq\; \alpha(1 - 2\alpha) \cdot y_{iv}, \quad \forall i, v. \tag{16}$$

Setting $\alpha = \frac{1}{4}$, this would imply Theorem 6.1.

Proof of (16). Fix any $i \in [n]$ and $v \in \mathcal{G}$. Let $\mathcal{B}_i$ denote the event that at least k r.v.s have been probed before i is considered. Let $\mathcal{M}_i$ be the even that some r.v. has been marked before i is considered. Note that $\langle X_i, v\rangle$ is marked exactly when the following events occur:

$$R_i = 1, \quad X_i = v, \quad S_{iv} = 1, \quad \neg\mathcal{B}_i, \quad \neg\mathcal{M}_i.$$

We now have

$$\begin{aligned}
\Pr\left[\langle X_i, v\rangle \text{ marked}\right] \;&=\; \Pr\left[R_i = 1 \wedge X_i = v \wedge S_{iv} = 1 \wedge \neg\mathcal{B}_i \wedge \neg\mathcal{M}_i\right] \\
&= \alpha x_i \cdot p_{iv} \cdot \frac{y_{iv}}{x_i p_{iv}} \cdot \Pr\left[\neg\mathcal{B}_i \wedge \neg\mathcal{M}_i \,|\, R_i = 1,\, X_i = v,\, S_{iv} = 1\right] && (17) \\
&= \alpha y_{iv} \cdot \Pr\left[\neg\mathcal{B}_i \wedge \neg\mathcal{M}_i\right] && (18) \\
&\geq \alpha y_{iv} \cdot \Pr\left[\neg\mathcal{B} \wedge \neg\mathcal{M}\right]. && (19)
\end{aligned}$$

Above, $\mathcal{B} = \mathcal{B}_n$ corresponds to the event that at least k r.v.s are probed by the end of the algorithm. Similarly, $\mathcal{M} = \mathcal{M}_n$ is the event that some r.v. is marked in the algorithm. (17) uses the facts that $\Pr[R_i = 1] = \alpha x_i$, $\Pr[X_i = v] = p_{iv}$ and $\Pr[S_{iv} = 1] = \frac{y_{iv}}{x_i p_{iv}}$. (18) follows from the fact that events $\mathcal{B}_i$ and $\mathcal{M}_i$ are independent of R_i, X_i and S_{iv}. Finally, (19) uses the fact that $\mathcal{B}_n \supseteq \mathcal{B}_i$ and $\mathcal{M}_n \supseteq \mathcal{M}_i$.

Bounding $\Pr[\mathcal{B}]$ and $\Pr[\mathcal{M}]$. Observe that event $\mathcal{B}$ corresponds to $\sum_{i=1}^{n} R_i \geq k$. Using $\mathbb{E}[R_i] = \alpha x_i$ and Markov's inequality,

$$\Pr[\mathcal{B}] \;\leq\; \frac{1}{k}\sum_{i=1}^{n} \mathbb{E}[R_i] = \frac{\alpha}{k}\sum_{i=1}^{n} x_i \leq \alpha, \tag{20}$$

where the last inequality uses the LP constraint (13).

We bound $\Pr[\mathcal{M}]$ in a similar manner. For any $i \in [n]$ and $v \in \mathcal{G}$,

$$\Pr[\langle X_i, v\rangle \text{ marked}] \leq \Pr[X_i = v \wedge R_i = 1 \wedge S_{iv} = 1] = \alpha y_{iv}.$$

By a union bound,

$$\Pr[\mathcal{M}] \;\leq\; \sum_{i=1}^{n}\sum_{v\in\mathcal{G}} \Pr[\langle X_i, v\rangle \text{ marked}] \leq \alpha \sum_{i=1}^{n}\sum_{v\in\mathcal{G}} y_{iv} \leq \alpha, \tag{21}$$

where the last inequality uses the LP constraint (15).

Wrapping up. Combining (20) and (21), we get $\Pr[\mathcal{B} \vee \mathcal{M}] \leq 2\alpha$. Plugging this into (19), we obtain (16).

6.2 Random order policy

Let σ be a uniformly random permutation on $[n]$; we use $j \prec i$ to denote that j appears before i in σ. Let $\alpha \in (0, 1]$ denote a constant value that will be fixed later. Our algorithm considers r.v.s in the order given by σ. When r.v. X_i is considered, we do the following:

> If less than k r.v.s have been probed so far, then probe X_i w.p. $\alpha \cdot x_i$.

We will show that:

Theorem 6.2 *The above algorithm is a $\frac{1}{4}$-approximation algorithm for adaptive constrained maximum value.*

Again, we will use the marking process in Algorithm 1 to lower-bound the maximum value. The only difference is that we now consider r.v.s in the order σ (rather than an arbitrary order). Here, we will show:

$$\Pr\left[\langle X_i, v\rangle \text{ marked}\right] \;\geq\; \alpha(1-\alpha)\cdot y_{iv}, \quad \forall i, v. \tag{22}$$

We note that the above probability is over all the randomness in the algorithm (including the permutation σ). Setting $\alpha = \frac{1}{2}$, this would imply Theorem 6.2.

Proof of (22)**.** Fix any $i \in [n]$ and $v \in \mathcal{G}$. As before, let $\mathcal{B}_i$ denote the event that at least k r.v.s have been probed before i is considered, and $\mathcal{M}_i$ be the even that some r.v. has been marked before i is considered. Using (18) from the previous analysis, we have

$$\Pr\left[\langle X_i, v\rangle \text{ marked}\right] \geq \alpha y_{iv} \cdot \Pr\left[\neg\mathcal{B}_i \wedge \neg\mathcal{M}_i\right] \tag{23}$$

Bounding $\Pr[\mathcal{B}_i]$ **and** $\Pr[\mathcal{M}_i]$**.** Observe that event $\mathcal{B}_i$ corresponds to $\sum_{j\prec i} R_j \geq k$. For any $j \in [n]$, the event $j \prec i$ is independent of R_j. Now, by Markov's inequality,

$$\Pr[\mathcal{B}_i] \;\leq\; \frac{1}{k}\sum_{j=1}^{n} \Pr[j \prec i] \cdot \mathbb{E}[R_i] = \frac{\alpha}{2k}\sum_{i=1}^{n} x_i \leq \frac{\alpha}{2}, \tag{24}$$

where the equality uses $\Pr[j \prec i] = \frac{1}{2}$ and $\mathbb{E}[R_i] = \alpha x_i$; the last inequality uses LP constraint (13).

We bound $\Pr[\mathcal{M}_i]$ in a similar manner. For any $j \in [n]$ and $u \in \mathcal{G}$,

$$\Pr[\langle X_j, u\rangle \text{ marked}] \leq \Pr[X_j = u \wedge R_j = 1 \wedge S_{ju} = 1] = \alpha y_{ju}.$$

By a union bound,

$$\begin{aligned}\Pr[\mathcal{M}_i] \;&\leq\; \sum_{j=1}^{n}\sum_{v\in\mathcal{G}} \Pr[j \prec i \bigwedge \langle X_j, u\rangle \text{ marked}] \\ &\leq\; \frac{\alpha}{2}\sum_{j=1}^{n}\sum_{u\in\mathcal{G}} y_{ju} \;\leq\; \frac{\alpha}{2},\end{aligned} \tag{25}$$

where the last inequality uses the LP constraint (15).

Combining (24) and (25), we get $\Pr[\mathcal{B}_i \vee \mathcal{M}_i] \leq \alpha$. Plugging this into (23), we obtain (22).

6.3 Further Improvement

We now obtain an even better approximation ratio:

Theorem 6.3 *For any constant $\epsilon > 0$, there is a $(\frac{1}{2} - \epsilon)$-approximation algorithm for adaptive constrained maximum value.*

First, we will show that when $k \geq \frac{1}{\epsilon}$, the random-order probing algorithm achieves an improved $\frac{1}{2} - \epsilon$ approximation. Then, we will show that an optimal adaptive policy can be found (by dynamic programming) when $k \leq \frac{1}{\epsilon}$.

Algorithm for large k**.** Recall that the random-order algorithm considers r.v.s in the order σ and probes each X_i w.p. $\alpha \cdot x_i$ (if less than k r.v.s have been probed so far). Here, we set $\alpha = 1$. As before, we will use the marking process in Algorithm 1 to lower-bound the maximum value. The key step in the analysis is to show:

$$\Pr\left[\langle X_i, v\rangle \text{ marked}\right] \;\geq\; \left(\frac{1}{2} - e^{-k/6}\right) \cdot y_{iv}, \quad \forall i, v. \tag{26}$$

This implies a $\frac{1}{2} - \epsilon$ approximation ratio for $k \geq 6 \log \frac{1}{\epsilon}$.

We now prove this for any fixed i and v. Again, using (23), it suffices to upper bound $\Pr[\mathcal{B}_i]$ and $\Pr[\mathcal{M}_i]$. We use the previous bound $\Pr[\mathcal{M}_i] \leq \frac{\alpha}{2} = \frac{1}{2}$ from (25). The new ingredient is a better bound on $\Pr[\mathcal{B}_i]$, shown next.

For any $j \in [n]$, let $I_j = \mathbf{1}_{j \prec i}$ denote the indicator that j appears before i in the order σ. Note that the r.v.s $\{I_j, R_j\}_{j \neq i}$ are mutually independent. So, r.v.s $Z_j := I_j \cdot R_j$ (for $j \neq i$) are also independent. Moreover, event $\mathcal{B}_i$ corresponds to $\mathcal{Z} := \sum_{j \neq i} Z_j \geq k$. Note that $\mathbb{E}[Z_j] = \frac{1}{2} \cdot \mathbb{E}[R_j] = \frac{x_j}{2}$; so $\mathbb{E}[\mathcal{Z}] = \frac{1}{2} \sum_{j \neq i} x_j \leq \frac{k}{2}$, where we used LP constraint (13). As the Z_js are $0-1$ valued and independent, applying Chernoff bound, we obtain $\Pr[\mathcal{Z} \geq k] \leq e^{-k/6}$. Hence, $\Pr[\mathcal{B}_i] \leq e^{-k/6}$ and $\Pr[\mathcal{B}_i \vee \mathcal{M}_i] \leq \frac{1}{2} + e^{-k/6}$. Combined with (23), we obtain (26).

Algorithm for small k. Here, we simply solve the constrained maximum value problem as a stochastic dynamic program. Such an approach suffices because the size of the state-space remains polynomial for constant k. Recall that the state space Ω represents the current decisions made by a policy (along with the corresponding random outcomes). For constrained maximum value, the state space is $\Omega \subseteq (\mathcal{G} \cup \{*\})^n$. A state ω represents the following for each $i \in [n]$:

- if $\omega_i = *$ then r.v. X_i has not been probed yet.
- if $\omega_i \neq *$ then r.v. X_i has already been probed, and we observed $X_i = \omega_i$.

Moreover, the maximum number of allowed probes is k. So, the number of feasible states $|\Omega| = \sum_{j=1}^{k} \binom{n}{j} \cdot |\mathcal{G}|^j \leq 2n^k |\mathcal{G}|^k$, which is polynomial for $k \leq 6 \log \frac{1}{\epsilon} = O(1)$.

We now describe a dynamic program that finds the *optimal* adaptive value. For any state ω, let $k(\omega) = |\{i \in [n] : \omega_i \neq *\}|$ be the number of already-probed r.v.s; note that $0 \leq k(\omega) \leq k$. Define the "value function" $V(\omega)$ to be the maximum expected max-value from probed r.v.s *given* state ω subject to probing at most k r.v.s in total. Observe that the optimal adaptive value is exactly $V(\mathbf{s})$, where $\mathbf{s} = (*, \cdots *)$ is the initial state (with zero probed r.v.s). We will compute $V(\omega)$ recursively, as follows. The "base case" involves states with k probed r.v.s, and

$$V(\omega) = \max_{i : \omega_i \neq *} \omega_i, \quad \forall \omega \in \Omega \,:\, k(\omega) = k.$$

Then, for each $\ell \leq k-1$, we compute:

$$V(\omega) \quad = \quad \max_{j : \omega_j = *} \sum_{v \in \mathcal{G}} p_{jv} \cdot V(\omega_{jv}), \qquad \forall \omega \in \Omega \,:\, k(\omega) = \ell,$$

where ω_{jv} is the state that has value v in coordinate j and coincides with ω on all coordinates except j. The above recursion is well-defined because $k(\omega_{jv}) = k(\omega) + 1$. To summarize:

Theorem 6.4 *There is a polynomial-time algorithm that finds an optimal adaptive policy for constrained maximum value for constant k.*

Using this algorithm when $k \leq 6 \log \frac{1}{\epsilon}$ and the $\frac{1}{2} - \epsilon$ approximation above for $k \geq 6 \log \frac{1}{\epsilon}$, we obtain Theorem 6.3.