A PTAS for a Class of Stochastic Dynamic Programs

Hao Fu
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijng, China
fu-h13@mails.tsinghua.edu.cn

Jian Li
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijng, China
Corresponding author lijian83@mail.tsinghua.edu.cn

Pan Xu
Department of Computer Science, University of Maryland, College Park, USA
panxu@cs.umd.edu

Abstract

We develop a framework for obtaining polynomial time approximation schemes (PTAS) for a class of stochastic dynamic programs. Using our framework, we obtain the first PTAS for the following stochastic combinatorial optimization problems:

1. *Probemax* [19]: We are given a set of n items, each item $i \in [n]$ has a value X_i which is an independent random variable with a known (discrete) distribution π_i. We can *probe* a subset $P \subseteq [n]$ of items sequentially. Each time after probing an item i, we observe its value realization, which follows the distribution π_i. We can *adaptively* probe at most m items and each item can be probed at most once. The reward is the maximum among the m realized values. Our goal is to design an adaptive probing policy such that the expected value of the reward is maximized. To the best of our knowledge, the best known approximation ratio is $1 - 1/e$, due to Asadpour *et al.* [2]. We also obtain PTAS for some generalizations and variants of the problem.
2. *Committed Pandora's Box* [24, 22]: We are given a set of n boxes. For each box $i \in [n]$, the cost c_i is deterministic and the value X_i is an independent random variable with a known (discrete) distribution π_i. Opening a box i incurs a cost of c_i. We can adaptively choose to open the boxes (and observe their values) or stop. We want to maximize the expectation of the realized value of the last opened box minus the total opening cost.
3. *Stochastic Target* [15]: Given a predetermined target $\mathbb{T}$ and n items, we can adaptively insert the items into a knapsack and insert at most m items. Each item i has a value X_i which is an independent random variable with a known (discrete) distribution. Our goal is to design an adaptive policy such that the probability of the total values of all items inserted being larger than or equal to $\mathbb{T}$ is maximized. We provide the first bi-criteria PTAS for the problem.
4. *Stochastic Blackjack Knapsack* [16]: We are given a knapsack of capacity $\mathbb{C}$ and probability distributions of n independent random variables X_i. Each item $i \in [n]$ has a size X_i and a profit p_i. We can adaptively insert the items into a knapsack, as long as the capacity constraint is not violated. We want to maximize the expected total profit of all inserted items. If the capacity constraint is violated, we lose all the profit. We provide the first bi-criteria PTAS for the problem.

2012 ACM Subject Classification Theory of computation → Stochastic approximation

Keywords and phrases stochastic optimization, dynamic program, markov decision process, block policy, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.56

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 56; pp. 56:1–56:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version A full version of the paper is available at https://arxiv.org/abs/1805.07742.

Funding This research is supported in part by the National Basic Research Program of China Grant 2015CB358700, the National Natural Science Foundation of China Grant 61772297, 616320-16, 61761146003, and a grant from Microsoft Research Asia.

Acknowledgements We would like to thank Anupam Gupta for several helpful discussions during the various stages of the paper. Jian Li would like to thank the Simons Institute for the Theory of Computing, where part of this research was carried out. Hao Fu would like to thank Sahil Singla for useful discussions about Pandora's Box problem. Pan Xu would like to thank Aravind Srinivasan for his many useful comments.

1 Introduction

Consider an online stochastic optimization problem with a finite number of rounds. There are a set of tasks (or items, boxes, jobs or actions). In each round, we can choose a task and each task can be chosen at most once. We have an initial "state" of the system (called the value of the system). At each time period, we can select a task. Finishing the task generates some (possibly stochastic) feedback, including changing the value of the system and providing some profit for the round. Our goal is to design a strategy to maximize our total (expected) profit.

The above problem can be modeled as a class of stochastic dynamic programs which was introduced by Bellman [3]. There are many problems in stochastic combinatorial optimization which fit in this model, *e.g.,* the stochastic knapsack problem [9], the Probemax problem [19]. Formally, the problem is specified by a 5-tuple $(\mathcal{V}, \mathcal{A}, f, g, h, T)$. Here, $\mathcal{V}$ is the set of all possible values of the system. $\mathcal{A}$ is a finite set of items or tasks which can be selected and each item can be chosen at most once. This model proceeds for at most T rounds. At each round $t \in [T]$, we use $I_t \in \mathcal{V}$ to denote the current value of the system and $\mathcal{A}_t \subseteq \mathcal{A}$ the set of remaining available items. If we select an item $a_t \in \mathcal{A}_t$, the value of the system changes to $f(I_t, a_t)$. Here f may be stochastic and is assumed to be independent for each item $a_t \in \mathcal{A}$. Using the terminology from Markov decision processes, the state at time t is $s_t = (I_t, \mathcal{A}_t) \in \mathcal{V} \times 2^{\mathcal{A}}$. [1] Hence, if we select an item $a_t \in \mathcal{A}_t$, the evolution of the state is determined by the state transition function f:

$$s_{t+1} = (I_{t+1}, \mathcal{A}_{t+1}) = (f(I_t, a_t), \mathcal{A}_t \setminus a_t) \quad t = 1, \ldots, T. \tag{1}$$

Meanwhile the system yields a random profit $g(I_t, a_t)$. The function $h(I_{T+1})$ is the terminal profit function at the end of the process.

We begin with the initial state $s_1 = (I_1, \mathcal{A})$. We choose an item $a_1 \in \mathcal{A}$. Then the system yields a profit $g(I_1, a_1)$, and moves to the next state $s_2 = (I_2, \mathcal{A}_2)$ where I_2 follows the distribution $f(I_1, a_1)$ and $\mathcal{A}_2 = \mathcal{A} \setminus a_1$. This process is iterated yielding a random sequence

$$s_1, a_1, s_2, a_2, s_3, \ldots, a_T, s_{T+1}.$$

The profits are accumulated over T steps. [2] The goal is to find a policy that maximizes the

[1] This is why we do not call I_t the state of the system.

[2] If less than T steps, we can use some special items to fill which satisfy that $f(I, a) = I$ and $g(I, a) = 0$ for any value $I \in \mathcal{V}$.

expectation of the total profits $\mathbb{E}\Big[\sum_{t=1}^{T} g(I_t, a_t) + h(I_{T+1})\Big]$. Formally, we want to determine:

$$\mathrm{DP}^*(s_1) = \max_{\{a_1,\ldots,a_T\}\subseteq\mathcal{A}} \mathbb{E}\Big[\sum_{t=1}^{T} g(I_t, a_t) + h(I_{T+1})\Big] \qquad \text{(DP)}$$
$$\text{subject to: } I_{t+1} = f(I_t, a_t), \qquad t = 1, \ldots, T.$$

By Bellman's equation [3], for every initial state $s_1 = (I_1, \mathcal{A})$, the optimal value $\mathrm{DP}^*(s_1)$ is given by $\mathrm{DP}_1(I_1, \mathcal{A})$. Here DP_1 is the function defined by $\mathrm{DP}_{T+1}(I_{T+1}) = h(I_{T+1})$ together with the recursion:

$$\mathrm{DP}_t(I_t, \mathcal{A}_t) = \max_{a_t\in\mathcal{A}_t} \mathbb{E}\Big[\mathrm{DP}_{t+1}(f(I_t, a_t), \mathcal{A}_t \setminus a_t) + g(I_t, a_t)\Big], \quad t = 1, \ldots, T. \tag{2}$$

When the value and the item spaces are finite, and the expectations can be computed, this recursion yields an algorithm to compute the optimal value. However, since the state space $\mathcal{S} = \mathcal{V} \times 2^{\mathcal{A}}$ is exponentially large, this exact algorithm requires exponential time. Since this model can capture several stochastic optimization problems which are known (or believed) be #P-hard or even PSPACE-hard, we are interested in obtaining polynomial-time approximation algorithms with provable performance guarantees.

1.1 Our Results

In order to obtain a polynomial time approximation scheme (PTAS) for the stochastic dynamic program, we need the following assumptions.

▶ **Assumption 1.** *In this paper, we make the following assumptions.*

1. *The value space $\mathcal{V}$ is discrete and ordered, and its size $|\mathcal{V}|$ is a constant. W.l.o.g., we assume $\mathcal{V} = (0, 1, \ldots, |\mathcal{V}| - 1)$.*
2. *The function f satisfies that $f(I_t, a_t) \geq I_t$, which means the value is nondecreasing.*
3. *The function $h : \mathcal{V} \to R^{\geq 0}$ is a nonnegative function. The expected profit $\mathbb{E}[g(I_t, a_t)]$ is nonnegative (although the function $g(I_t, a_t)$ may be negative with nonzero probability).*

Assumption (1) seems to be quite restrictive. However, for several concrete problems where the value space is not of constant size (*e.g.,* Probemax in Section 1.2), we can discretize the value space and reduce its size to a constant, without losing much profit. Assumption (2) and (3) are quite natural for many problems. Now, we state our main result.

▶ **Theorem 1.** *For any fixed $\varepsilon > 0$, if Assumption 1 holds, we can find an adaptive policy in polynomial time $n^{2^{O(\varepsilon^{-3})}}$ with expected profit at least $\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX}$ where $\mathsf{MAX} = \max_{I\in\mathcal{V}} \mathrm{DP}_1(I, \mathcal{A})$ and OPT denotes the expected profit of the optimal adaptive policy.*

Our Approach: For the stochastic dynamic program, an optimal adaptive policy σ can be represented as a decision tree $\mathcal{T}$ (see Section 2 for more details). The decision tree corresponding to the optimal policy may be exponentially large and arbitrarily complicated. Hence, it is unlikely that one can even represent an optimal decision for the stochastic dynamic program in polynomial space. In order to reduce the space, we focus a special class of policies, called *block adaptive policy*. The idea of *block adptive policy* was first introduced by Bhalgat*et al.* [6] and further generalized in [17] to the context of the stochastic knapsack. To the best of our knowledge, the idea has not been extended to other applications. In this paper, we make use of the notion of block policy as well, but we target at the development

of a general framework. For this sake we provide a general model of block policy (see Section 3). Since we need to work with the more abstract dynamic program, our construction of block adaptive policy is somewhat different from that in [6, 17].

Roughly speaking, in a block adaptive policy, we take a batch of items simultaneously instead of a single one each time. This can significantly reduce the size of the decision tree. Moreover, we show that there exists a block-adaptive policy that approximates the optimal adaptive policy and has only a constant number of blocks on the decision tree (the constant depends on ε). Since the decision tree corresponding to a block adaptive policy has a constant number of nodes, the number of all topologies of the block decision tree is a constant. Fixing the topology of the decision tree corresponding to the block adaptive policy, we still need to decide the subset of items to place in each block. Again, there is exponential number of possible choices. For each block, we can define a *signature* for it, which allows us to represent a block using polynomially many possible signatures. The signatures are so defined such that two subsets with the same signature have approximately the same reward distribution. Finally, we show that we can enumerate the signatures of all blocks in polynomial time using dynamic programming and find a nearly optimal block-adaptive policy. The high level idea is somewhat similar to that in [17], but the details are again quite different.

1.2 Applications

Our framework can be used to obtain the first PTAS for the following problems.

1.2.1 The Probemax Problem

In the Probemax problem, we are given a set of n items. Each item $i \in [n]$ has a value X_i which is an independent random variable following a known (discrete) distribution π_i. We can *probe* a subset $P \subseteq [n]$ of items sequentially. Each time after *probing* an item i, we observe its value realization, which is an independent sample from the distribution π_i. We can *adaptively* probe at most m items and each item can be probed at most once. The reward is the maximum among the m realized values. Our goal is to design an adaptive probing policy such that the expected value of the reward is maximized.

Despite being a very basic stochastic optimization problem, we still do not have a complete understanding of the approximability of the Probemax problem. It is not even known whether it is intractable to obtain the optimal policy. For the non-adaptive Probemax problem (*i.e.,* the probed set P is just a priori fixed set), it is easy to obtain a $1 - 1/e$ approximation by noticing that $f(P) = \mathbb{E}[\max_{i \in P} X_i]$ is a submodular function (see e.g., Chen *et al.* [8]). Chen *et al.* [8] obtained the first PTAS. When considering the adaptive policies, Munagala [19] provided a $\frac{1}{8}$-approximation ratio algorithm by LP relaxation. His policy is essentially a non-adaptive policy (it is related to the contention resolution schemes [23, 10]). They also showed that the *adaptivity gap* (the gap between the optimal adaptive policy and optimal non-adaptive policy) is at most 3. For the Probemax problem, the best-known approximation ratio is $1 - \frac{1}{e}$. Indeed, this can be obtained using the algorithm for stochastic monotone submodular maximization in Asadpour *et al.* [2]. This is also a non-adaptive policy, which implies the adaptivity gap is at most $\frac{e}{e-1}$. In this paper, we provide the first PTAS, among all adaptive policies. Note that our policy is indeed adaptive.

▶ **Theorem 2.** *There exists a PTAS for the Probemax problem. In other words, for any fixed constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for the Probemax problem that finds a policy with the expected profit at least $(1-\varepsilon)\mathsf{OPT}$, where OPT denotes the expected profit of the optimal adaptive policy.*

Let the value I_t be the maximum among the realized values of the probed items at the time period t. Using our framework, we have the following system dynamics for Probemax:

$$I_{t+1} = f(I_t, i) = \max\{I_t, X_i\}, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = I_{T+1} \tag{3}$$

$t = 1, 2, \ldots, T$. Clearly, Assumption 1 (2) and (3) are satisfied. But Assumption 1 (1) is not satisfied because the value space $\mathcal{V}$ is not of constant size. We can discretize the value space and reduce its size to a constant. See full version for more details. If the reward is the summation of top-k values ($k = O(1)$) among the m realized values, we obtain the ProbeTop-k problem. Our techniques also allow us to derive the following result.

▶ **Theorem 3.** *For the ProbeTop-k problem where k is a constant, there is a polynomial time algorithm that finds an adaptive policy with the expected profit at least $(1-\varepsilon)\mathsf{OPT}$, where OPT denotes the expected profit of the optimal adaptive policy.*

1.2.2 Committed ProbeTop-k Problem

We are given a set of n items. Each item $i \in [n]$ has a value X_i which is an independent random variable with a known (discrete) distribution π_i. We can *adaptively* probe at most m items and choose k values in the committed model, where k is a constant. In the *committed* model, once we probe an item and observe its value realization, we must make an irrevocable decision whether to choose it or not, *i.e.,* we must either add it to the final chosen set C immediately or discard it forever. [3] If we add the item to the final chosen set C, the realized profit is collected. Otherwise, no profit is collected and we are going the probe the next item. Our goal is to design an adaptive probing policy such that the expected value $\mathbb{E}[\sum_{i\in C} X_i]$ is maximized, where C is the final chosen set.

▶ **Theorem 4.** *There is a polynomial time algorithm that finds a committed policy with the expected profit at least $(1-\varepsilon)\mathsf{OPT}$ for the committed ProbeTop-k problem, where OPT is the expected total profit obtained by the optimal policy.*

Let b_i^θ represent the action that we probe item i with the threshold θ (*i.e.,* we choose item i if X_i realizes to a value s such that $s \geq \theta$). Let I_t be the the number of items that have been chosen at the period time t. Using our framework, we have following transition dynamics for the ProbeTop-k problem.

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + 1 & \text{if } X_i \geq \theta, I_t < k, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} X_i & \text{if } X_i \geq \theta, I_t < k, \\ 0 & \text{otherwise;} \end{cases} \tag{4}$$

for $t = 1, 2, \ldots, T$, and $h(I_{T+1}) = 0$. Since k is a constant, Assumption 1 is immediately satisfied. There is one extra requirement for the problem: in any realization path, we can choose at most one action b_i^θ from the set $\mathcal{B}_i = \{b_i^\theta\}_\theta$.

1.2.3 Committed Pandora's Box Problem

For Weitzman's "Pandora's box" problem [24], we are given n boxes. For each box $i \in [n]$, the probing cost c_i is deterministic and the value X_i is an independent random variable with a known (discrete) distribution π_i. Opening a box i incurs a cost of c_i. When we open the box i, its value is realized, which is a sample from the distribution π_i. The goal is to adaptively open

[3] In [10, 11], it is called the online decision model.

a subset $P \subseteq [n]$ to maximize the expected profit: $\mathbb{E}\left[\max_{i\in P}\{X_i\} - \sum_{i\in P} c_i\right]$. Weitzman provided an elegant optimal adaptive strategy, which can be computed in polynomial time. Recently, Singla [22] generalized this model to other combinatorial optimization problems such as matching, set cover and so on.

In this paper, we focus on the committed model, which is mentioned in Section 1.2.2. Again, we can *adaptively* open the boxes and choose at most k values in the committed way, where k is a constant. Our goal is to design an adaptive policy such that the expected value $\mathbb{E}\left[\sum_{i\in C} X_i - \sum_{i\in P} c_i\right]$ is maximized, where $C \subseteq P$ is the final chosen set and P is the set of opened boxes. Although the problem looks like a slight variant of Weitzman's original problem, it is quite unlikely that we can adapt Weitzman's argument (or any argument at all) to obtain an optimal policy in polynomial time. When $k = O(1)$, we provide the first PTAS for this problem. Note that a PTAS is not known previously even for $k = 1$.

▶ **Theorem 5.** *When $k = O(1)$, there is a polynomial time algorithm that finds a committed policy with the expected value at least $(1-\varepsilon)\mathsf{OPT}$ for the committed Pandora's Box problem.*

Similar to the committed ProbeTop-k problem, let b_i^θ represent the action that we open the box i with threshold θ. Let I_t be the number of boxes that have been chosen at the time period t. Using our framework, we have following system dynamics for the committed Pandora's Box problem:

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + 1 & \text{if } X_i \geq \theta, I_t < k, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} X_i - c_i & \text{if } X_i \geq \theta, I_t < k, \\ -c_i & \text{otherwise;} \end{cases} \tag{5}$$

for $t = 1, 2, \cdots, T$, and $h(I_{T+1}) = 0$. Notice that we never take an action b_i^θ for a value $I_t < k$ if $\mathbb{E}[g(I_t, b_i^\theta)] = \Pr[X_t \geq \theta] \cdot \mathbb{E}[X_i \mid X_i \geq \theta] - c_i < 0$. Then Assumption 1 is immediately satisfied.

1.2.4 Stochastic Target Problem

İlhan *et al.* [15] introduced the following stochastic target problem. [4] In this problem, we are given a predetermined target $\mathbb{T}$ and a set of n items. Each item $i \in [n]$ has a value X_i which is an independent random variable with a known (discrete) distribution π_i. Once we decide to insert an item i into a knapsack, we observe a reward realization X_i which follows the distribution π_i. We can insert at most m items into the knapsack and our goal is to design an adaptive policy such that $\Pr[\sum_{i\in P} X_i \geq \mathbb{T}]$ is maximized, where $P \subseteq [n]$ is the set of inserted items. For the stochastic target problem, İlhan *et al.* [15] provided some heuristic based on dynamic programming for the special case where the random profit of each item follows a known normal distribution. In this paper, we provide an additive PTAS for the stochastic target problem when the target is relaxed to $(1-\varepsilon)\mathbb{T}$.

▶ **Theorem 6.** *There exists an additive PTAS for stochastic target problem if we relax the target to $(1-\varepsilon)\mathbb{T}$. In other words, for any given constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm that finds a policy such that the probability of the total rewards exceeding $(1-\varepsilon)\mathbb{T}$ is at least $\mathsf{OPT} - \varepsilon$, where OPT is the resulting probability of an optimal adaptive policy.*

[4] [15] called the problem the adaptive stochastic knapsack instead. However, their problem is quite different from the stochastic knapsack problem studied in the theoretical computer science literature. So we use a different name.

Let the value I_t be the total profits of the items in the knapsack at time period t. Using our framework, we have following system dynamics for the stochastic target problem:

$$I_{t+1} = f(I_t, i) = I_t + X_i, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} 1 & \text{if } I_{T+1} \geq \mathbb{T}, \\ 0 & \text{otherwise;} \end{cases} \tag{6}$$

for $t = 1, 2, \cdots, T$. Then Assumption 1 (2,3) is immediately satisfied. But Assumption 1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. We can discretize the value space and reduce its size to a constant.

1.2.5 Stochastic Blackjack Knapsack

Levin *et al.* [16] introduced the *stochastic blackjack knapsack*. In this problem, we are given a capacity $\mathbb{C}$ and a set of n items, each item $i \in [n]$ has a size X_i which is an independent random variable with a known distribution π_i and a profit p_i. We can adaptively insert the items into a knapsack, as long as the capacity constraint is not violated. Our goal is to design an adaptive policy such that the expected total profits of all items inserted is maximized. The key feature here different from classic stochastic knapsack is that we gain zero if overflow, *i.e.,* we will lose the profits of all items inserted already if the total size is larger than the capacity. This extra restriction might induce us to take more conservative policies. Levin *et al.* [16] presented a non-adaptive policy with expected value that is at least $(\sqrt{2}-1)^2/2 \approx 1/11.66$ times the expected value of the optimal adaptive policy. Chen *et al.* [7] assumed each size X_i follows a known exponential distribution and gave an optimal policy for $n = 2$ based on dynamic programming. In this paper, we provide the first bi-criteria PTAS for the problem.

▶ **Theorem 7.** *For any fixed constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for stochastic blackjack knapsack that finds a policy with the expected profit at least $(1-\varepsilon)\mathsf{OPT}$, when the capacity is relaxed to $(1+\varepsilon)\mathbb{C}$, where OPT is the expected profit of the optimal adaptive policy.*

Denote $I_t = (I_{t,1}, I_{t,2})$ and let $I_{t,1}, I_{t,2}$ be the total sizes and total profits of the items in the knapsack at the time period t respectively. When we insert an item i into the knapsack and observe its size realization, say s_i, we define the system dynamics function to be

$$I_{t+1} = f(I_t, i) = (I_{t,1} + s_i, I_{t,2} + p_i), \quad h(I_{T+1}) = \begin{cases} I_{T+1,2} & \text{if } I_{T+1,1} \leq \mathbb{C}, \\ 0 & \text{otherwise;} \end{cases} \tag{7}$$

and $g(I_t, i) = 0$ for $t = 1, 2, \cdots, T$. Then Assumption 1 (2,3) is immediately satisfied. But Assumption 1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. We can discretize the value space and reduce its size to a constant.

1.3 Related Work

Stochastic dynamic program has been widely studied in computer science and operation research (see, for example, [4, 20]) and has many applications in different fields. It is a natural model for decision making under uncertainty. In 1950s, Richard Bellman [3] introduced the "principle of optimality" which leads to dynamic programming algorithms for solving sequential stochastic optimization problems. However, Bellman's principle does not immediate lead to efficient algorithms for many problems due to "curse of dimensionality" and the large state space.

There are some constructive frameworks that provide approximation schemes for certain classes of stochastic dynamic programs. Shmoys *et al.* [21] dealt with stochastic linear programs. Halman *et al.* [12, 13, 14] studied stochastic discrete DPs with scalar state and action spaces and designed an FPTAS for their framework. As one of the applications, they used it to solve the stochastic ordered adaptive knapsack problem. As a comparison, in our model, the state space $\mathcal{S} = \mathcal{V} \times 2^{\mathcal{A}}$ is exponentially large and hence cannot be solved by previous framework.

Stochastic knapsack problem (SKP) is one of the most well-studied stochastic combinatorial optimization problem. We are given a knapsack of capacity $\mathbb{C}$. Each item $i \in [n]$ has a random value X_i with a known distribution π_i and a profit p_i. We can adaptively insert the items to the knapsack, as long as the capacity constraint is not violated. The goal is to maximize the expected total profit of all items inserted. For SKP, Dean *et al.* [9] first provide a constant factor approximation algorithm. Later, Bhalgat *et al.* [6] improved that ratio to $\frac{3}{8} - \varepsilon$ and gave an algorithm with ratio of $(1 - \varepsilon)$ by using ε extra budget for any given constant $\varepsilon \geq 0$. In that paper, the authors first introduced the notion of block adaptive policies, which is crucial for this paper. The best known single-criterion approximation factor is 2 [5, 17, 18].

The Probemax problem and ProbeTop-k problem are special cases of the general stochastic probing framework formulated by Gupta *et al.* [11]. They showed that the adaptivity gap of any stochastic probing problem where the outer constraint is prefix-closed and the inner constraint is an intersection of p matroids is at most $O(p^3 \log(np))$, where n is the number of items. The Bernoulli version of stochastic probing was introduced in [10], where each item $i \in U$ has a fixed value w_i and is "active" with an independent probability p_i. Gupta *et al.* [10] presented a framework which yields a $\frac{1}{4(k^{in}+k^{out})}$-approximation algorithm for the case when $\mathcal{I}_{in}$ and $\mathcal{I}_{out}$ are respectively an intersection of k^{in} and k^{out} matroids. This ratio was improved to $\frac{1}{(k^{in}+k^{out})}$ by Adamczyk *et al.* [1] using the iterative randomized rounding approach. Weitzman's Pandora's Box is a classical example in which the goal is to find out a single random variable to maximize the utility minus the probing cost. Singla [22] generalized this model to other combinatorial optimization problems such as matching, set cover, facility location, and obtained approximation algorithms.

2 Policies and Decision Trees

An instance of stochastic dynamic program is given by $\mathcal{J} = (\mathcal{V}, \mathcal{A}, f, g, h, T)$. For each item $a \in \mathcal{A}$ and values $I, J \in \mathcal{V}$, we denote $\Phi_a(I, J) := \Pr[f(I, a) = J]$ and $\mathcal{G}_a(I) := \mathbb{E}[g(I, a)]$. The process of applying a feasible adaptive *policy* σ can be represented as a decision tree $\mathcal{T}_\sigma$. Each node v on $\mathcal{T}_\sigma$ is labeled by a unique item $a_v \in \mathcal{A}$. Before selecting the item a_v, we denote the corresponding time index, the current value and the set of the remaining available items by t_v, I_v and $\mathcal{A}(v)$ respectively. Each node has several children, each corresponding to a different value realization (one possible $f(I_v, a_v)$). Let $e = (v, u)$ be the s-th edge emanating from $s \in \mathcal{V}$ where s is the realized value. We call u the s-child of v. Thus e has probability $\pi_e := \pi_{v,s} = \Phi_{a_v}(I_v, s)$ and weight $w_e := s$.

We use $\mathbb{P}(\sigma)$ to denote the expected profit that the policy σ can obtain. For each node v on $\mathcal{T}_\sigma$, we define $\mathcal{G}_v := \mathcal{G}_{a_v}(I_v)$. In order to clearly illustrate the tree structure, we add a dummy node at the end of each root-to-leaf path and set $\mathcal{G}_v = h(I_v)$ if v is a dummy node. Then, we recursively define the expected profit of the subtree $\mathcal{T}_v$ rooted at v to be

$$\mathbb{P}(v) = \mathcal{G}_v + \sum_{e=(v,u)} \pi_e \cdot \mathbb{P}(u), \tag{8}$$

if v is an internal node and $\mathbb{P}(v) = \mathcal{G}_v = h(I_v)$ if v is a leaf (*i.e.,* the dummy node). The expected profit $\mathbb{P}(\sigma)$ of the policy σ is simply $\mathbb{P}(\text{the root of } \mathcal{T}_\sigma)$. Then, according to Equation (2), we have

$$\mathbb{P}(v) \leq \mathrm{DP}_{t_v}(I_v, \mathcal{A}(v)) \leq \mathrm{DP}_1(I_v, \mathcal{A}) \leq \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathsf{MAX}$$

for each node v. For a node v, we say the path from the root to it on $\mathcal{T}_\sigma$ as the *realization path* of v, and denote it by $\mathcal{R}(v)$. We denote the probability of reaching v as $\Phi(v) = \Phi(\mathcal{R}(v)) = \prod_{e \in \mathcal{R}(v)} \pi_e$. Then, we have

$$\mathbb{P}(\sigma) = \sum_{v \in \mathcal{T}_\sigma} \Phi(v) \cdot \mathcal{G}_v. \tag{9}$$

We use OPT to denote the expected profit of the optimal adaptive policy. For each node v on the tree $\mathcal{T}_\sigma$, by Assumption 1 (2) that $f(I_v, a_v) \geq I_v$, we define $\mu_v := \Pr[f(I_v, a_v) > I_v] = 1 - \Phi_{a_v}(I_v, I_v)$. For a set of nodes P, we define $\mu(P) := \sum_{v \in P} \mu_v$.

▶ **Lemma 8.** *Given an policy σ, there is a policy σ' with profit at least $\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX}$ which satisfies that for any realization path $\mathcal{R}$, $\mu(\mathcal{R}) \leq O(1/\varepsilon)$, where $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A})$.*

W.l.o.g, we assume that all (optimal or near optimal) policies σ considered in this paper satisfy that for any realization $\mathcal{R}$, $\mu(\mathcal{R}) \leq O(1/\varepsilon)$.

3 Block Adaptive Policies

The decision tree corresponding to the optimal policy may be exponentially large and arbitrarily complicated. Now we consider a restrict class of policies, called block-adaptive policy. The concept was first introduced by Bhalgat *et al.* [6] in the context of stochastic knapsack. Our construction is somewhat different from that in [6, 17]. Here, we need to define an order for each block and introduce the notion of approximate block policy.

Formally, a block-adaptive policy $\hat{\sigma}$ can be thought as a decision tree $\mathcal{T}_{\hat{\sigma}}$. Each node on the tree is labeled by a *block* which is a set of items. For a block M, we choose an arbitrary order φ for the items in the block. According to the order φ, we take the items one by one, until we get a bigger value or all items in the block are taken but the value does not change (recall from Assumption 1 that the value is nondecreasing). Then we visit the child block which corresponds to the realized value. We use I_M to denote the current value right before taking the items in the block M. Then for each edge $e = (M, N)$, it has probability

$$\pi_e^\varphi = \sum_{a \in M} \left[\left(\prod_{\varphi_b < \varphi_a} \Phi_b(I_M, I_M) \right) \cdot \Phi_a(I_M, I_N) \right]$$

if $I_N > I_M$ and $\pi_e^\varphi = \prod_{a \in M} \Phi_a(I_M, I_M)$ if $I_N = I_M$.

Similar to Equation (8), for each block M and an arbitrary order φ for M, we recursively define the expected profit of the subtree $\mathcal{T}_M$ rooted at M to be

$$\mathbb{P}(M) = \mathcal{G}_M^\varphi + \sum_{e=(M,N)} \pi_e^\varphi \cdot \mathbb{P}(N) \tag{10}$$

if M is an internal block and $\mathbb{P}(M) = h(I_M)$ if M is a leaf (*i.e.,* the dummy node). Here $\mathcal{G}_M^\varphi$ is the expected profit we can get from the block which is equal to

$$\mathcal{G}_M^\varphi = \sum_{a \in M} \left[\left(\prod_{\varphi_b < \varphi_a} \Phi_b(I_M, I_M) \right) \cdot \mathcal{G}_a(I_M) \right].$$

Since the profit $\mathcal{G}_M^{\varphi}$ and the probability π_e^{φ} are dependent on the order φ and thus difficult to deal with, we define the approximate block profit and the approximate probability which do not depend on the choice of the specific order φ:

$$\widetilde{\mathcal{G}}_M = \sum_{a\in M} \mathcal{G}_a(I_M) \quad \text{and} \quad \widetilde{\pi}_e = \sum_{a\in M} \left[\left(\prod_{b\in M\setminus a} \Phi_b(I_M, I_M) \right) \cdot \Phi_a(I_M, I_N) \right] \tag{11}$$

if $I_N > I_M$ and $\widetilde{\pi}_e = \prod_{a\in M} \Phi_a(I_M, I_M)$ if $I_N = I_M$. Then we recursively define the approximate profit

$$\widetilde{\mathbb{P}}(M) = \widetilde{\mathcal{G}}_M + \sum_{e=(M,N)} \widetilde{\pi}_e \cdot \widetilde{\mathbb{P}}(N), \tag{12}$$

if M is an internal block and $\widetilde{\mathbb{P}}(M) = \mathbb{P}(M) = h(I_M)$ if M is a leaf. For each block M, we define $\mu(M) := \sum_{a\in M} [1 - \Phi_a(I_M, I_M)]$. Lemma 9 below can be used to bound the gap between the approximate profit and the original profit if the policy satisfies the following property.Then it suffices to consider the approximate profit for a block adaptive policy $\hat{\sigma}$ in this paper.

(P1) Each block M with more than one item satisfies that $\mu(M) \le \varepsilon^2$.

▶ **Lemma 9.** *For any block-adaptive policy $\hat{\sigma}$ satisfying Property (P1), we have*

$$\left(1 + O(\varepsilon^2)\right) \cdot \widetilde{\mathbb{P}}(\hat{\sigma}) \ge \mathbb{P}(\hat{\sigma}) \ge \left(1 - \varepsilon^2\right) \cdot \widetilde{\mathbb{P}}(\hat{\sigma}).$$

3.1 Constructing a Block Adaptive Policy

In this section, we show that there exists a block-adaptive policy that approximates the optimal adaptive policy. In order to prove this, from an optimal (or nearly optimal) adaptive policy σ, we construct a block adaptive policy $\hat{\sigma}$ which satisfies certain nice properties and can obtain almost as much profit as σ does. Thus it is sufficient to restrict our search to the block-adaptive policies. The construction is similar to that in [17].

▶ **Lemma 10.** *An optimal policy σ can be transformed into a block adaptive policy $\hat{\sigma}$ with approximate expected profit $\widetilde{\mathbb{P}}(\hat{\sigma})$ at least $\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX}$. Moreover, the block-adaptive policy $\hat{\sigma}$ satisfies Property (P1) and (P2):*

(P1) *Each block M with more than one item satisfies that $\mu(M) \le \varepsilon^2$.*

(P2) *There are at most $O(\varepsilon^{-3})$ blocks on any root-to-leaf path on the decision tree.*

Proof (sketch). For a node v on the decision tree $\mathcal{T}_\sigma$ and a value $s \in \mathcal{V}$, we use v_s to denote the s-child of v, which is the child of v corresponding to the realized value s (see Figure 1). We say an edge $e_{v,u}$ is *non-increasing* if $I_v = I_u$ and define the *leftmost path* of v to be the realization path which starts at v, ends at a leaf, and consists of only the non-increasing edges.

We say a node v is a *starting node* if v is the root or v corresponds to an increasing value of its parent v' (*i.e.,* $I_v > I_{v'}$). For each staring node v, we greedily partition the leftmost path of v into several segments such that for any two nodes u, w in the same segment M and for any value $s \in \mathcal{V}$, we have

$$|\mathbb{P}(u_s) - \mathbb{P}(w_s)| \le \varepsilon^2 \cdot \mathsf{MAX} \text{ and } \mu(M) \le \varepsilon^2. \tag{13}$$

For each root-to-leaf path $\mathcal{R}$, Equation (13) can yield at most $O(\varepsilon^{-3})$ blocks. Now, we are ready to describe the algorithm, which takes a policy σ as input and outputs a block adaptive

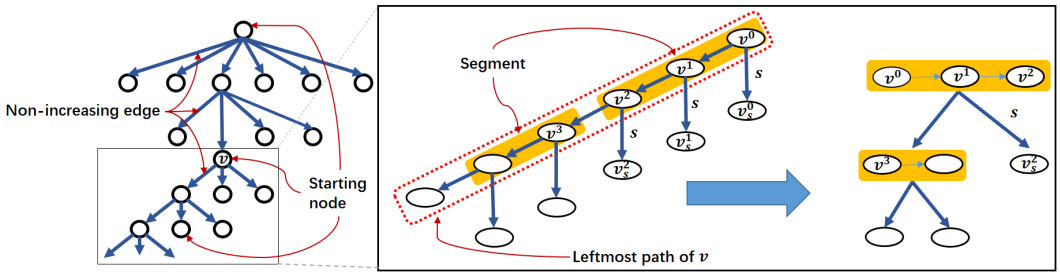

Figure 1 Decision tree and block policy.

Algorithm 1 A policy $\hat{\sigma}$.

Input: A policy σ.

1: We start at the root of $\mathcal{T}_\sigma$.
2: **repeat**
3: Suppose we are at node v on $\mathcal{T}_\sigma$. Take the items in $\mathsf{seg}(v)$ one by one in the original order (the order of items in policy σ) until some node u makes a transition to an increasing value, say s.
4: Visit the node $l(v)_s$, the s-child of $l(v)$ (*i.e.,* the last node of $\mathsf{seg}(v)$).
5: If all items in $\mathsf{seg}(v)$ have been taken and the value does not change, visit $l(v)_{I_v}$.
6: **until** A leaf on $\mathcal{T}_\sigma$ is reached.

policy $\hat{\sigma}$. For each node v, we denote its segment $\mathsf{seg}(v)$ and use $l(v)$ to denote the last node in $\mathsf{seg}(v)$. In Algorithm 1, we can see that the set of items which the policy $\hat{\sigma}$ attempts to take always corresponds to some realization path in the original policy σ. Property (P1) and (P2) hold immediately following from the partition argument. Then we can show that the expected profit $\mathbb{P}(\hat{\sigma})$ that the new policy $\hat{\sigma}$ can obtain is at least $\mathsf{OPT} - O(\varepsilon^2) \cdot \mathsf{MAX}$. ◀

3.2 Enumerating Signatures

To search for the (nearly) optimal block-adaptive policy, we want to enumerate all possible structures of the block decision tree. Fixing the topology of the decision tree, we need to decide the subset of items to place in each block. To do this, we define the *signature* such that two subsets with the same signature have approximately the same profit distribution. Then, we can enumerate the signatures of all blocks in polynomial time and find a nearly optimal block-adaptive policy. Formally, for an item $a \in \mathcal{A}$ and a value $I \in \mathcal{V} = (0, 1, \ldots, |\mathcal{V}| - 1)$, we define the *signature* of a on I to be the following vector

$$\mathsf{Sg}_I(a) = \left(\bar{\Phi}_a(I,0), \bar{\Phi}_a(I,1), \ldots, \bar{\Phi}_a(I,|\mathcal{V}|-1), \bar{\mathcal{G}}_a(I)\right),$$

where $\bar{\Phi}_a(I,J) = \left\lfloor \Phi_a(I,J) \cdot \frac{n}{\varepsilon^4} \right\rfloor \cdot \frac{\varepsilon^4}{n}$ and $\bar{\mathcal{G}}_a(I) = \left\lfloor \mathcal{G}_a(I) \cdot \frac{n}{\varepsilon^4 \mathsf{MAX}} \right\rfloor \cdot \frac{\varepsilon^4 \mathsf{MAX}}{n}$ for any $J \in \mathcal{V}$. [5] For a block M of items, we define the *signature* of M on I to be $\mathsf{Sg}_I(M) = \sum_{a \in M} \mathsf{Sg}_I(a)$.

▶ **Lemma 11.** *Consider two decision trees $\mathcal{T}_1, \mathcal{T}_2$ corresponding to block-adaptive policies with the same topology* (i.e., *$\mathcal{T}_1$ and $\mathcal{T}_2$ are isomorphic) and the two block adaptive policies*

[5] If $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A})$ is unknown, for some several concrete problems (*e.g.,* Probemax), we can get a constant approximation result for MAX, which is sufficient for our purpose. In general, we can guess a constant approximation result for MAX using binary search.

satisfy Property (P1) and (P2). If for each block M_1 on $\mathcal{T}_1$, the block M_2 at the corresponding position on $\mathcal{T}_2$ satisfies that $\mathsf{Sg}_I(M_1) = \mathsf{Sg}_I(M_2)$ where $I = I_{M_1} = I_{M_2}$, then $|\widetilde{\mathbb{P}}(\mathcal{T}_1) - \widetilde{\mathbb{P}}(\mathcal{T}_2)| \leq O(\varepsilon) \cdot \mathsf{MAX}$.

Since $|V| = O(1)$, the number of possible signatures for a block is $O\left((n/\varepsilon^4)^{|\mathcal{V}|}\right) = n^{O(1)}$, which is a polynomial of n. By Lemma 10, for any block decision tree $\mathcal{T}$, there are at most $(|\mathcal{V}|)^{O(\varepsilon^{-3})} = 2^{O(\varepsilon^{-3})}$ blocks on the tree which is a constant.

3.3 Finding a Nearly Optimal Block-adaptive Policy

In this section, we find a nearly optimal block-adaptive policy and prove Theorem 1. To do this, we enumerate over all topologies of the decision trees along with all possible signatures for each block. This can be done by a standard dynamic programming.

Consider a given tree topology $\mathcal{T}$. A configuration C is a set of signatures each corresponding to a block. Let t_1 and t_2 be the number of paths and blocks on $\mathcal{T}$ respectively. We define a vector $\mathsf{CA} = (u_1, u_2, \ldots, u_{t_1})$ where u_j is the upper bound of the number of items on the jth path. For each given $i \in [n], \mathsf{C}$ and CA, let $\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = 1$ indicate that we can reach the configuration C using a subset of items $\{a_1, \ldots, a_i\}$ such that the total number of items on each path j is no more than u_j and 0 otherwise. Set $\mathcal{M}(0, \mathbf{0}, \mathbf{0}) = 1$ and we compute $\mathcal{M}(i, \mathsf{C}, \mathsf{CA})$ in an lexicographically increasing order of $(i, \mathsf{C}, \mathsf{CA})$ as follows:

$$\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = \max\left\{\mathcal{M}(i-1, \mathsf{C}, \mathsf{CA}), \mathcal{M}(i-1, \mathsf{C}', \mathsf{CA}')\right\} \tag{14}$$

Now, we explain the above recursion as follows. In each step, we should decide how to place the item a_i on the tree $\mathcal{T}$. Notice that there are at most $t_2 = (|\mathcal{V}|)^{O(\varepsilon^{-3})} = 2^{O(\varepsilon^{-3})}$ blocks and therefore at most 2^{t_2} possible placements of item a_i and each placement is called *feasible* if there are no two blocks on which we place the item a_i have an ancestor-descendant relation. For a feasible placement of a_i, we subtract $\mathsf{Sg}(a_i)$ from each entry in C corresponding to the block we place a_i and subtract 1 from CA on each entry corresponding to a path including a_i, and in this way we get the resultant configuration C' and CA' respectively. Hence, the max is over all possible such $\mathsf{C}', \mathsf{CA}'$.

We have shown that the total number of all possible configurations on $\mathcal{T}$ is n^{t_2}. The total number of vectors CA is $T^{t_1} \leq n^{t_1} \leq n^{t_2} = n^{t_2}$ where T is the number of rounds. For each given $(i, \mathsf{C}, \mathsf{CA})$, the computation takes a constant time $O(2^{t_2})$. Thus we claim for a given tree topology, finding the optimal configuration can be done within $O(n^{2^{O(\varepsilon^{-3})}})$ time .

The proof of Theorem 1. Suppose σ^* is the optimal policy with expected profit $\mathbb{P}(\sigma^*) = \mathsf{OPT}$. We use the above dynamic programming to find a nearly optimal block adaptive policy σ. By Lemma 10, there exists a block adaptive policy $\hat{\sigma}$ such that

$$\widetilde{\mathbb{P}}(\hat{\sigma}) \geq \mathsf{OPT} - O(\varepsilon)\mathsf{MAX}.$$

Since the configuration of $\hat{\sigma}$ is enumerated at some step of the algorithm, our dynamic programming is able to find a block adaptive policy σ with the same configuration (the same tree topology and the same signatures for corresponding blocks). By Lemma 11, we have

$$\widetilde{\mathbb{P}}(\sigma) \geq \widetilde{\mathbb{P}}(\hat{\sigma}) - O(\varepsilon)\mathsf{MAX} \geq \mathsf{OPT} - O(\varepsilon)\mathsf{MAX}.$$

By Lemma 9, we have $\mathbb{P}(\sigma) \geq \left(1 - \varepsilon^2\right) \cdot \widetilde{\mathbb{P}}(\sigma) \geq \mathsf{OPT} - O(\varepsilon)\mathsf{MAX}$. Hence, the proof of Theorem 1 is completed. ◀

References

1 Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. *Mathematics of Operations Research*, 41(3):1022–1038, 2016.

2 Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Management Science*, 62(8):2374–2391, 2015.

3 Richard Bellman. Dynamic programming. In *Princeton University Press*, 1957.

4 Dimitri P. Bertsekas. *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995.

5 Anand Bhalgat. A $(2 + \varepsilon)$-approximation algorithm for the stochastic knapsack problem. *Unpublished Manuscript*, 2011.

6 Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1647–1665, 2011.

7 Kai Chen and Sheldon M. Ross. An adaptive stochastic knapsack problem. *European Journal of Operational Research*, 239(3):625–635, 2014. doi:10.1016/j.ejor.2014.06.027.

8 Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. Combinatorial multi-armed bandit with general reward functions. *Advances in Neural Information Processing Systems*, 2016.

9 Brian C Dean, Michel X Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 395–404. Society for Industrial and Applied Mathematics, 2005.

10 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 205–216. Springer, 2013.

11 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1731–1747. Society for Industrial and Applied Mathematics, 2016.

12 Nir Halman, Diego Klabjan, Chung Lun Li, James Orlin, and David Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. In *Nineteenth Acm-Siam Symposium on Discrete Algorithms*, pages 700–709, 2008.

13 Nir Halman, Diego Klabjan, Chung-Lun Li, James Orlin, and David Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *SIAM Journal on Discrete Mathematics*, 28(4):1725–1796, 2014.

14 Nir Halman, Giacomo Nannicini, and James Orlin. A computationally efficient fptas for convex stochastic dynamic programs. *SIAM Journal on Optimization*, 25(1):317–350, 2015.

15 Taylan İlhan, Seyed MR Iravani, and Mark S Daskin. The adaptive knapsack problem with stochastic rewards. *Operations Research*, 59(1):242–248, 2011.

16 Asaf Levin and Aleksander Vainer. Adaptivity in the stochastic blackjack knapsack problem. *Theoretical Computer Science*, 516:121–126, 2014.

17 Jian Li and Wen Yuan. Stochastic combinatorial optimization via poisson approximation. *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 971–980, 2013.

18 Will Ma. Improvements and generalizations of stochastic knapsack and markovian bandits approximation algorithms. *Mathematics of Operations Research*, 2017. doi:10.1287/moor.2017.0884.

19 Kamesh Munagala. Approximation algorithms for stochastic optimization. `https://simons.berkeley.edu/talks/kamesh-munagala-08-22-2016-1`, Simons Institue for the Theory of Computing, 2016.

20 Warren B Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, volume 842. John Wiley & Sons, 2011.

21 David B Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM (JACM)*, 53(6):978–1012, 2006.

22 Sahil Singla. The price of information in combinatorial optimization. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2523–2532. SIAM, 2018.

23 Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 783–792. ACM, 2011.

24 Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.