一类随机动态规划的近似算法

（申请清华大学工学博士学位论文）

培 养 单 位：交 叉 信 息 研 究 院
学　　科：计 算 机 科 学 与 技 术
研　究　生：傅　昊
指 导 教 师：李　建　副　教　授

二〇一八年六月

Approximation Algorithms for a class of stochastic dynamic programs

Dissertation Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Doctor of Philosophy

in

Computer Science and Technology

by

Hao Fu

Dissertation Supervisor : Associate Professor Jian Li

June, 2018

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后应遵守此规定）

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

随机动态规划（SDP）是一种对不确定条件下的决策问题进行建模的技术。Richard Bellman 提出了 “最优化原则”。这个原则能推导出动态规划算法，它可以用来解决一系列随机优化问题。然而，由于" 维数灾难" 和很大的状态空间，对于许多现实的随机组合问题，Bellman 的原则并不能立即推导出一个有效的算法。

为了应对这些挑战，这篇毕业论文开发了一个基于 SDP 的框架。在这个框架下，我们设计了一个多项式时间的近似方案（PTAS）。当使用我们的框架，就可以为一类随机组合优化问题设计 PTAS。本文研究的问题如下：（a）随机探测问题，（b）随机背包问题，（c）随机二十一点背包问题和（d）随机目标问题。我们为所有上述问题设计多项式时间的近似算法。我们开发的这项技术很基本，相信会有更广泛的应用。

我们着重介绍一下随机探测问题的一个特例：探测最大值问题。在这个问题中，输入是一组已知分布的独立随机变量。我们可以依次从中挑选几个随机变量。每当我们挑选一个，就可以观察它的真实值。这个值是遵循其分配。挑选的变量中最大的那个值就是我们最后的受益。由于挑选的个数有限，那么怎样设计一个自适应策略，使得期望受益最大。据我们所知，现在最好的近似算法是 Asadpour 设计 $1-1/e$。我们可以设计一个 PTAS。

关键词：随机优化，动态规划，近似算法，马尔科夫过程，模块策略

Abstract

Stochastic Dynamic Programming (SDP) is a technique for modeling and solving problems of decision making under uncertainty. Richard Bellman introduced the "principle of optimality" which leads to dynamic programming algorithms for solving sequential stochastic optimization problems. However, Bellman's principle does not immediately lead to efficient algorithms for many realistic stochastic combinatorial problems, due to the "curse of dimensionality" and the large state space.

To address these challenges, we develop a framework based on SDP in this dissertation. For this framework, we obtain a polynomial time approximation scheme (PTAS). Using our framework, we obtain the first PTAS for a class of stochastic combinatorial optimization problems. The problems we study in this thesis are the following: (a) stochastic probing problem, (b) stochastic knapsack problem, (c) stochastic blackjack knapsack and (d) stochastic target problem. We present polynomial-time algorithms for all the above problems with near-optimal approximation guarantees. We believe that the techniques are fairly general and have wider applications.

We highlight the application on Probemax problem which is a special case of stochastic probing 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 *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 $1 - 1/e$. We also obtain PTAS for some generalizations and variants of the problem.

Key words: stochastic optimization; dynamic program; markov decision process; block policy; approximation algorithm

目 录

主要符号对照表

ProbeMax	探测最大值问题
committed	承诺模型
non-committed	未承诺模型
adaptively	自适应的
block adaptive policy	块自适应策略
adaptivity gap	自适应差距
commitment gap	承诺差距
canonical policy	规范策略
SKP	随机背包问题
SBK	随机二十一点背包问题
STP	随机目标问题
ProbeTop-k	探测前 k 大和问题
Pandora's Box	承诺模型潘多拉盒子问题
OPT	最优值
matroid	拟阵
stochastic probing problem	随机探测问题

第 1 章　Introduction

In recent years, uncertain data has been generated in many application scenarios, occurring in almost all fields of science and engineering from telecommunications, medicine, to finance. In these scenarios, uncertainty is inevitable. How to analyze and solve these problems in a rigorous manner has become a hot topic in recent years. This area originated from Dantzig's work[1], and has been studied extensively and found numerous applications in many areas. Richard Bellman[2] proposed the stochastic dynamic programming which is a technique for modeling and solving problems of decision making under uncertainty. It plays a very important role in the stochastic optimization model because it is a very general model that can be used to design optimization algorithms. For some specific problems, it is possible to obtain a closed form or exact solution in polynomial-time by solving the Bellman equation directly. On the other hand, due to "curse of dimensions" and the large state space, this method has several limitations. We refer to the books[3,4].

Stochastic optimization has established itself as a major method to handle uncertainty in various optimization problems, by modeling the uncertainty by a probability distribution over possible realizations. For some specific scenarios, researchers have proposed a set of models and given corresponding polynomial-time approximation algorithms, such as stochastic knapsack[5,6], stochastic matching[7,8], stochastic set cover[9,10], stochastic bin packing[11] and so on. We refer interested readers to the survey[12].

In this dissertation, we focus on a class of stochastic dynamic programs. For these problems, we propose a model based on the stochastic dynamic program and design an effective polynomial-time approximation algorithm. By our model, we can design polynomial algorithms for a bunch of stochastic combinatorial optimization problems which fit in the model.

Now, we introduce two classes of problems in this dissertation: the stochastic probing problem and stochastic knapsack. We also briefly state our contributions one by one as follows.

1.1　Stochastic Probing Problem

Suppose a company is going to hire a secretary. After putting out recruitment advertisements, a total of n resumes were received. Through a resume, the company can

assess whether the candidate matches the company's position. However the matching score is uncertain and follows a distribution. To know the true score, the company needs to arrange an interview. Due to limited time and energy, the company can only select $m(\leq n)$ candidates for interview. What strategy should be designed to allow companies to choose a secretary with the highest possible matching score? This problem can be expressed by the following model.

Problem 1.1 (ProbeMax)：　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.

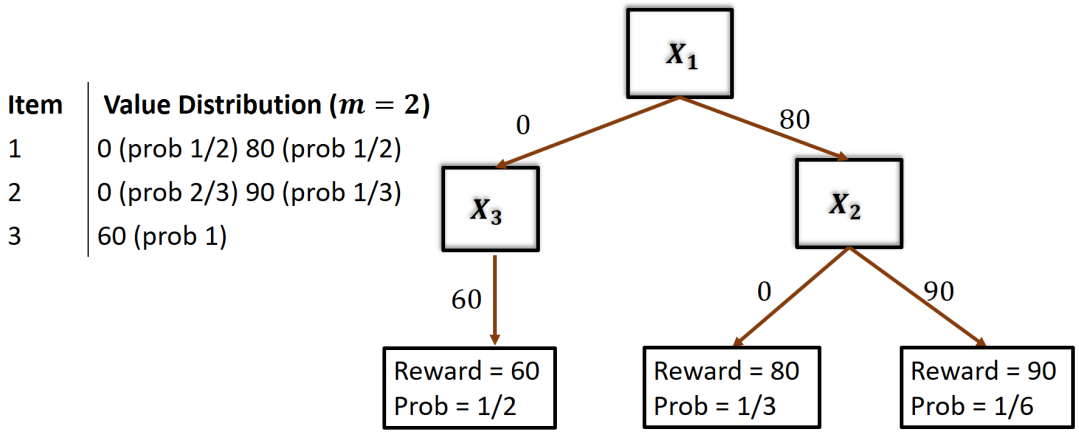

Figure 1.1　Instance of the ProbeMax.
Notes: An optimal adaptive policy for ProbeMax shown as a decision tree achieves an expected reward of 71.67. For a non-adaptive policy, the optimal is to select items 1 and 3 for which the expected reward is equal to $\mathbb{E}\max\{X_1, X_3\} = 70$.

The problem and its variants have been studied in a number of papers (see *e.g.*, [13–18]). 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.

A *non-adaptive* policy is simply a priori fixed set P of items to probe. 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.*[13]). Chen *et al.*[13] obtained the first PTAS, among all non-adaptive policies. The benefit of an adaptive solution can be shown by a simple example, see Figure 1.1. For the ProbeMax problem, the best-known approximation ratio is $1 - \frac{1}{e}$[17,19]. Indeed, this can be obtained using the algorithm for stochastic monotone submodular maximization in Asadpour *et al.*[17]. This is also a non-adaptive policy, which implies the *adaptivity gap* — the ratio between the values of optimal adaptive and optimal non-adaptive policies — is at most $\frac{e}{e-1}$. Golovin[19] introduced the notion of adaptive submodularity which is a generalization of Asadpour's setting. In this dissertation, we provide the first PTAS, among all adaptive policies. Note that our policy is indeed adaptive. We will discuss this problem and its generalizations in more details in Chapter 4.

Theorem 1.1: There exists a PTAS for the ProbeMax problem among all adaptive policies. 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 value at least $(1 - \varepsilon)\mathsf{OPT}$, where OPT is the expected value of the optimal adaptive policy.

The ProbeMax problem is a special case of the following general stochastic probing framework which was formulated by Gupta *et al.*[18].

Problem 1.2 (Stochastic Probing): An instance $\mathcal{J}$ of the stochastic probing problem is defined with tuple $\mathcal{J} = (U, \pi, \mathcal{I})$, where U is a set of n items, each having a value X_i following a known (discrete) distribution π_i, and $\pi = \{\pi_i\}_{[i \in U]}$. The constraint $\mathcal{I}$ consists of two independent set systems $\mathcal{I}_{out}$ and $\mathcal{I}_{in}$, which capture the outer (packing) constraint and the inner (packing) constraint respectively. We can adaptively probe a subset $P \subseteq U$ of items sequentially and choose another subset $C \subseteq P$ as the final set. When we probe the ith item, its value is realized, which is sampled from the distribution π_i. (The item values are independent of each other.) We require that the sequence P of items that we have probed must be in $\mathcal{I}_{out}$ and the set C of items that we output must be in $\mathcal{I}_{in}$. Our goal is to design a policy such that the expected value $\mathbb{E}[\sum_{i \in C} X_i]$ is maximized.

There are two kinds of constraint:

- *Outer constraint* $\mathcal{I}_{out}$[①]*:* this is the constraint on the sequence of items that we can probe. It requires that the sequence P of items that we have probed must be in $\mathcal{I}_{out}$ and each item can be probed at most once.

① $\mathcal{I}_{out}$ should be better understood as a collection of sequences. If $\mathcal{I}_{out}$ is a collection of subsets, then any sequence of a subset in $\mathcal{I}_{out}$ can be probed.

- *Inner constraint $\mathcal{I}_{in}$:* this is the constraint on the set of items that we can finally output. It requires that the set C of items that we output must be in $\mathcal{I}_{in}$.

Depending on how we choose the final set C, we distinguish two natural models: the *committed* and *non-committed* models [①].

- Committed: once we probe an item to observe its value realization, we are committed to making 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.
- Non-committed: we can choose the set C after observing the values of all probed items in P. In other words, the choice of C can be made after all value realizations of the items in P, *i.e.,* the probing stage.

In fact, every committed policy is a special case of non-committed policies. The stochastic probing problem naturally generalizes the *ProbeMax* problem which is on the non-committed model. Here, the outer constraint $\mathcal{I}_{out}$ is the simple m-uniform matroid, *i.e.,* $\mathcal{I}_{out} = \{P \mid |P| \leq m\}$. The inner constraint $\mathcal{I}_{in}$ is the 1-uniform matroid, *i.e.,* we can output one item. We can replace the outer constraint with some more general constraint, *e.g.,* a matroid. If we replace the inner constraint in ProbeMax with a k-uniform matroid (*i.e.,* $\mathcal{I}_{in} = \{C \mid |C| \leq k\}$), we obtain the ProbeTop-k problem. In this thesis, we assume k is a constant. Now, we list our main results in Table 1.1. All approximation ratios in Table 1.1 are compared to the best adaptive strategies.

Table 1.1　A summary of approximation ratios for the stochastic probing problem

Inner constraint	Outer constraint	Committed	Non-committed
1-Uniform	Uniform, Partition	PTAS [Thm 1.5 in [11]]	$1-1/e$ [17], $1/8$ [14], PTAS [Thm 1.1 in §4.4]
	General matroid	PTAS [Thm 4.2 in §4.5.2]	$1-1/e$ [17]
	Two matroids	PTAS [Thm 4.2 in §4.5.2]	$1/3$ [Thm 4.6], $1/2+\varepsilon$ [Thm 1.2 in §4.5.3]
k-Uniform	Uniform, Partition	PTAS [Thm 4.3 in §4.5.1]	$1-1/e$ [17], PTAS [Corollary 4.1 in §4.4.2]
General matroid	General matroid	$1/2$ [Thm 4.6 in §4.6]	$1-1/e$ [17]
k^{in} matroids	k^{out} matroids	$\frac{1}{k^{in}+k^{out}}$ [Thm 4.6 in §4.6]	

① In [18], they are called online and offline decision-making models.

For the stochastic probing problem, we denote *commitment gap* the upper bound on the ratio of the value of the optimal non-committed policy to the value of the optimal committed policy.

Theorem 1.2：　The commitment gap of any stochastic probing problem where the outer constraint is prefix-closed and the inner constraint is a uniform matroid is at most 2.

For the non-committed model, we denote *adaptivity gap* the upper bound on the ratio of the value of the optimal adaptive policy to the value of the optimal non-adaptive policy. Our techniques also allow us to derive the following useful result from the proof of Theorem 1.2, which improves the result of 3.51 in[18].

Corollary 1.1：　The adaptivity gap of any stochastic probing problem where the outer constraint is prefix-closed and the inner constraint is a uniform matroid is at most 2.

1.2　Stochastic Knapsack and Variants

Problem 1.3 (Stochastic Knapsack Problem (SKP))：　We are given a knapsack of capacity $\mathbb{C}$ and a set of n items, each item $i \in [n]$ having a random size X_i with a known distribution π_i and a profit p_i. Once we decide to insert an item i into a knapsack, we observe its real size s_i which follows the distribution π_i. We can adaptively insert the items to the knapsack, as long as the capacity constraint is not violated. Once one item violates the constraint, the selection process stops. The goal is to design a policy that maximizes the expected total profit of all items that are successfully inserted into the knapsack.

This problem is a classical problem in stochastic combinatorial optimization. One natural motivation is the stochastic scheduling problem as follows. Suppose we want to schedule a subset of n jobs on a single machine. Each job has a processing time and a profit. We have a fixed deadline and we want to gain as much profit as possible before the deadline. However, the processing time of each job is a random variable and follows an individual distribution.

We also consider *adaptive policies* which is very different from fixed set policies (see Figure 1.2). The problem is likely to be PSPACE-hard (a variant of the problem and its generalizations has been shown to be PASACE-hard[5]). Designing good approximation algorithms for this problem has received considerable attention in recent years[5,6,11,20–22], which we discuss in more detail in Chapter 5. Levin *et al.*[23] introduced another variant of stochastic knapsack where all the profits are lost if we violate the constraint.

Item	Size Distribution (C= 1)	Profit
1	0.2 (prob 0.5) 0.6 (prob 0.5)	1
2	0.8 (prob 1)	1
3	0.4 (prob 0.5) 0.9 (prob 0.5)	1

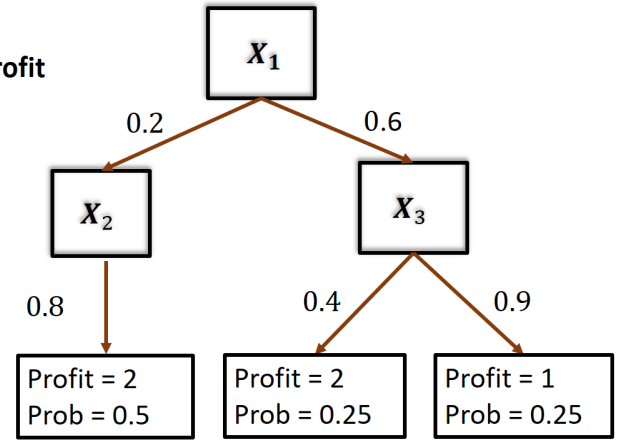

Figure 1.2　Instance of the stochastic knapsack problem.
Notes: An optimal adaptive policy for the stochastic knapsack problem showed as a decision tree achieves an expected profit of 1.75. An optimal nonadaptive policy inserts items in order 1, 2, 3, and achieves expected an profit of 1.5.

Problem 1.4 (Stochastic Blackjack Knapsack (SBK))：　In this problem, we are given an instance $\mathcal{J} = (\pi, \mathbb{C})$. Our goal is to design an adaptive policy such that the expected total profits of all items inserted is maximized. The difference from SKP is that we gain zero profit if the total size of all items inserted is larger than the capacity $\mathbb{C}$.

Theorem 1.3：　There exists a PTAS for stochastic blackjack knapsack if we relax the capacity to $(1+\varepsilon)\mathbb{C}$. In other words, for any given constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm that finds a policy with the expected profit at least $(1-\varepsilon)\mathsf{OPT}$ if we relax the capacity to $(1+\varepsilon)\mathbb{C}$, where OPT is the expected profit of an optimal adaptive policy for the stochastic blackjack knapsack without relaxing the capacity.

As a variant of stochastic knapsack, another problem is motivated by fishing ground selection. Assume that Jack has a fishing ship and is ready to go fishing. There are several fishing grounds to choose from. The yields of fishing grounds can be estimated by almost real time oceanographic analysis and historical figures in advance. But, the yield of a fishing ground is a random variable which will not be realized until the end of harvesting at that location. Due to the limited number of fishing trips, how to pick the right fishing ground to fill up the storage with the highest probability. This problem can be expressed by the following model.

Problem 1.5 (Stochastic Target Problem)：　We are given a predetermined target $\mathbb{T}$ and a set of n items. Each item $i \in [n]$ has a profit 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 profit 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.

Theorem 1.4: 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 $\mathsf{OPT} - \varepsilon$[①], where OPT is the resulting probability of an optimal adaptive policy.

Item	Value Distribution (target = 30)
1	10 (prob 1)
2	10 (prob 1/2) 20 (prob 1/2)
3	0 (prob 1/2) 20 (prob 1/2)

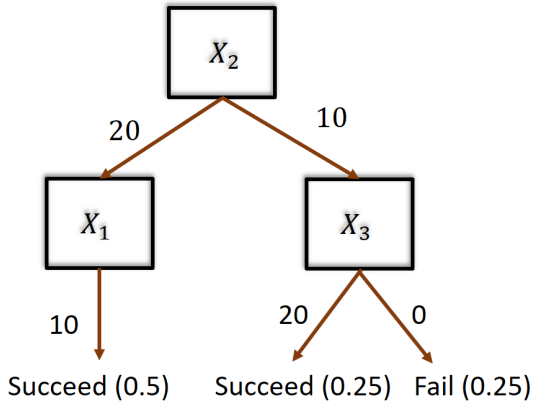

Figure 1.3　Instance of the stochastic target problem.
Notes: An optimal adaptive policy for stochastic target problem shown as a decision tree reaches the target with probability 0.75. The optimal non-adaptive policy is to select items 1 and 3 for which the probability of reaching the target level is $\Pr[X_1 + X_3 \geq 30] = 0.5$.

The benefit of an adaptive solution can be shown by a simple example. See Figure 1.3. As this example shows, it may significantly increase the optimal value of the objective function if one uses an adaptive policy. Now, we list our main results in Table 1.2.

If the sizes of items are bounded, we have following theorem, which can be found in Section 5.4.3.

Theorem 1.5: For any $\varepsilon \in (0,1)$, there is a polynomial algorithm that finds a $(1-4\varepsilon)$-approximate adaptive policy for stochastic knapsack problem when the sizes are bounded by ε, *i.e.,* for any $i \in [n]$, $\Pr[s_i \leq \varepsilon] = 1$.

① Since the upper bound of the probability for the stochastic target problem is 1, we get the an additive PTAS, not a multiplicative.

Table 1.2　A summary of approximation ratios for the stochastic knapsack and variants

Problem	single-criterion approx	bi-criterion approx ($\mathbb{C}$ relaxed to $(1+\varepsilon)\mathbb{C}$ or $\mathbb{T}$ relaxed to $(1-\varepsilon)\mathbb{T}$)
SKP	$\frac{1}{3}-\varepsilon$ [5], $\frac{3}{8}-\varepsilon$ [6], $\frac{1}{2}-\varepsilon$ [22]	$1-\varepsilon$ [6,11] [Theorem 5.1 in $\mathbb{S}$5.4]
SBK	$(\sqrt{2}-1)^2/2 \approx 1/11.66$ [23] $1/8-\varepsilon$ [Theorem 5.4 in $\mathbb{S}$5.5.3]	$1-\varepsilon$ [Theorem 1.3 in $\mathbb{S}$5.5]
STP	-	$\mathsf{OPT}-\varepsilon$ [Theorem 1.4 in $\mathbb{S}$5.6]

1.3　Stochastic Dynamic Programs

The above problems are all important stochastic optimization problems. From three examples (Figure 1.1,1.2,1.3), we can see the advantages of adaptive strategies. But it is quite difficult to design an adaptive strategy. This is because the decision tree corresponding to the optimal strategy may be exponentially large and arbitrarily complicated. Therefore, it would seem impossible to represent an optimal decision tree in a polynomial space. This is why researcher is keen to design non-adaptive strategies. In this dissertation, we provide a solution. To solve these problems, we develop a generic model that is based on the stochastic dynamic program. We also design an effective polynomial-time approximation algorithm for it to find adaptive strategies. There are a number of other stochastic optimization problems fit in this model, including those mentioned earlier.

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.

Problem 1.6 (Stochastic Dynamic Programs)：　Given a 6-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 $a_t \in \mathcal{A}$. 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 value $I_1 \in \mathcal{V}$. The goal is to find an adaptive policy that maximizes the expectation of the total profits $\mathbb{E}\Big[\sum_{t=1}^{T} g(I_t, a_t) + h(I_{T+1})\Big]$.

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

Assumption 1.1: In this thesis, we make the following assumptions.

1. The value space $\mathcal{V}$ is discrete and ordered, and 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, see Section 4.4), 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 1.6: For any fixed $\varepsilon > 0$, if Assumption 1.1 holds, we can find an adaptive policy in polynomial time $n^{2^{O(\varepsilon^{-3})}}$ with expected profit at least $\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX}$. Here $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}(I, \mathcal{A})$ is an upper bound for all initial value $I \in \mathcal{V}$ where $\mathrm{DP}(I, \mathcal{A})$ is the expected profit of the optimal adaptive policy for initial value I and $\mathsf{OPT} = \mathrm{DP}(I_1, \mathcal{A})$ is the expected profit for the given initial value I_1.

By Theorem 1.6 and different discretization techniques, we obtain the first PTAS for ProbeMax (Theorem 1.1), SBK (Theorem 1.3) and STP (Theorem 1.4). Theorem 1.6 also can be used to design PTASs for committed ProbeTop-k problem (see Section 4.5.1), committed Pandora's Box problem (see Section 4.5.4) and SKP (see Section 5.4). This work is published in the 45th International Colloquium on Automata, Languages, and Programming (ICALP2018)[24].

1.4 Outline

In Chapter 2, we list some of the basics of mathematics and computer science and the terms that will be used in the dissertation. In Chapter 3, we discuss the stochastic

dynamic program (Problem 1.6) and prove Theorem 1.6. In the next chapters, we discuss a few problems and techniques in details. In particular, we discuss the stochastic probing problem (Problem 1.2) and the ProbeMax (Problem 1.1) in Chapter 4, the stochastic knapsack (Problem 1.3), the stochastic target problem (Problem 1.5) and stochastic blackjack knapsack (Problem 1.4) in Chapter 5. Finally, we summarize the dissertation in Chapter 6.

第 2 章　Preliminaries and Notations

For a maximization problem, we say an algorithm achieves approximation ratio of $\alpha \leq 1$ if

$$\mathbb{E}[\mathsf{SOL}] \geq \alpha \mathsf{OPT},$$

here SOL denotes the value of the solution found by the algorithm, OPT denotes the optimal value and the expectation is over the randomness of the problem instance and the algorithm (if it is randomized). Similarly, for a minimization problem, we say an algorithm achieves approximation ratio of $\alpha \geq 1$ if $\mathbb{E}[\mathsf{SOL}] \leq \alpha \mathsf{OPT}$.

A *polynomial time approximation scheme (PTAS)* is an algorithm which takes an instance of a maximization problem and a parameter ε and produces a solution whose value is at least $(1 - \varepsilon)\mathsf{OPT}$, and the running time is polynomial in the size of input, for fixed ε. If ε appears as an additive factor in the above definition, namely the value of the solution is at least $\mathsf{OPT} - \varepsilon$, we say the algorithm is an *additive PTAS*. We say a PTAS is a *fully polynomial time approximation scheme (FPTAS)* is the running time is polynomial in the size of the input and $\frac{1}{\varepsilon}$.

2.1　Adaptive vs Non-adaptive

According to the permissions of the policy, we distinguish two classes of policies: *adaptive* and *non-adaptive*.

- Adaptive: At each step, we can adaptively make decision based on the observations so far. All the information regarding the previous actions of the policy is known. In other words, the policy has access to the actual realized value of all the elements it has picked so far.
- Non-adaptive: It does not have access to such information and should make their decisions before observing the outcome of any of them.

We denote the *adaptivity gap* by the ratio between the values of optimal adaptive and optimal non-adaptive policies. In this dissertation, if not specified, we are considering adaptive policies.

2.2 Prefix-close and Downward-closed

A pair $(E, \mathcal{I})$ where $\mathcal{I}$ is a collection of sequences of distinct items from E is *prefix-closed* if for every sequence $I \in \mathcal{I}$, any prefix of I is also in $\mathcal{I}$. This captures most reasonable outer constraints, *e.g.,* a tour of total length at most B in the metric space (see[18] for more examples). A pair $(E, \mathcal{I})$ where $\mathcal{I}$ is a collection of distinct (unordered) items from E is *downward-closed* if for every set $I \in \mathcal{I}$, any subset of I is also in $\mathcal{I}$. In the case when $\mathcal{I}$ is a collection of subsets of (unordered) items, $\mathcal{I}$ is prefix-closed iff $\mathcal{I}$ is downward-closed.

2.3 Matroid

Definition 2.1 : A pair $(E, \mathcal{I})$ is called a matroid if E is a finite set and $\mathcal{I}$ is a nonempty collection of subsets of E satisfying:

(i) if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$,

(ii) if $I, J \in \mathcal{I}$ and $|I| < |J|$, then $I + z \in \mathcal{I}$ for some $z \in J \setminus I$.

The *rank* of a matroid $M = (E, \mathcal{I})$ is defined by $r_M := \max\{|Z| \mid Z \in \mathcal{I}\}$. A *uniform matroid* is determined by a set E and a number k where the independent sets are the subsets I of E with $|I| \leq k$. *Partition matroids.* Let B_i be a collection of disjoint sets, and let d_i be integers with $0 \leq d_i \leq |B_i|$. The independent sets are the subsets I of E such that for every index i, $|I \cap B_i| \leq d_i$.

2.4 Submodular functions

Definition 2.2 : A function $f : 2^{[n]} \to \mathcal{R}$ is called submodular, if and only if $\forall S, T \subseteq [n]$,

$$f(S) + f(T) \geq f(S \cap T) + f(S \cup T).$$

It is called monotone if for any two subsets $T \subseteq S \subseteq [n] : f(T) \leq f(S)$.

Linear function $f(S) = \sum_{i \in S} w_i$ for some weights $\{w_i\}$ is a special monotone submodular function.

第 3 章　Stochastic Dynamic Programs

Stochastic dynamic program has been widely studied in computer science and operation research (see, for example,[3,4]) and has many applications in different fields. It is a natural model for decision making under uncertainty. In 1950s, Richard Bellman[2] 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.

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.

3.1　Model

The above problem can be modeled as a class of stochastic dynamic programs which was introduced by Bellman[2]. There are many problems in stochastic combinatorial optimization which fit in this model, *e.g.,* the stochastic knapsack problem[5], the ProbeMax problem[14]. Formally, the problem is specified by a 6-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

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

$a_t \in \mathcal{A}_t$, the evolution of the state is determined by the state transition function f:

$$s_{t+1} = (I_{t+1}, \mathcal{A}_{t+1}) = (f(I_t, a_t), \mathcal{A}_t \setminus a_t) \quad t = 1, \ldots, T. \tag{3-1}$$

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. [①] The goal is to find a policy that maximizes the 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[2], 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:

$$\mathrm{DP}_t(I_t, \mathcal{A}_t) = \max_{a_t \in \mathcal{A}_t} \mathbb{E}\Big[\mathrm{DP}_{t+1}(f(I_t, a_t), \mathcal{A}_t \setminus a_t) + g(I_t, a_t)\Big], \quad t = 1, \ldots, T. \tag{3-2}$$

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) to be #P-hard or even PSPACE-hard, we are interested in obtaining polynomial-time approximation algorithms with provable performance guarantees.

① 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 $I \in \mathcal{V}$.

3.2 Our Results

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

Assumption 3.1 : In this thesis, we make the following assumptions.

1. The value space $\mathcal{V}$ is discrete and ordered, and 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 4.4), we can discretize the value space and reduce its size to constant, without losing much profit. Assumption (2) and (3) are quite natural for many problems. Now, we state our main result.

Theorem 3.1 : For any fixed $\varepsilon > 0$, if Assumption 3.1 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.

3.3 Main Technique

For the stochastic dynamic program, an optimal adaptive policy σ can be represented as a decision tree $\mathcal{T}$ (see Section 3.5 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 on a special class of policies, called *block adaptive policies*. The idea of *block policy* was first introduced by Bhalgat *et al.*[6] and further generalized in[11] to the context of the stochastic knapsack. To the best of our knowledge, the idea has not been extended to other applications. In this thesis, 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 3.6). Since we need to work with the more abstract dynamic program, our construction of block adaptive policy is somewhat different from that in[6,11].

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 are an 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[11], but the details are again quite different.

3.4 Related Work

There are some constructive frameworks that provide approximation schemes for certain classes of stochastic dynamic programs. Shmoys *et al.*[25] dealt with stochastic linear programs. Halman *et al.*[26–28] studies 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. In contrast, 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.

3.5 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

$$\mathbb{P}(v) = \mathcal{G}_v + \sum_{e=(v,u)} \pi_e \cdot \mathbb{P}(u), \tag{3-3}$$

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 $\mathbb{P}$(the root of $\mathcal{T}_\sigma$). Then, according to Equation (3-2), we have

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

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

$$\mathbb{P}(\sigma) = \sum_{v \in \mathcal{T}_\sigma} \Phi(v) \cdot \mathcal{G}_v. \tag{3-4}$$

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 1.1 (2) that $f(I_v, a_v) \ge I_v$, 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 3.1 : Given a 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 $\mathcal{R}$, $\mu(\mathcal{R}) \le O(1/\varepsilon)$, where $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A})$.

Proof. Consider a random realization path $\mathcal{R} = (v_1, v_2, \ldots, v_{T+1})$ generated by σ. Recall in Assumption 1.1 (1), the value space is $\mathcal{V} = \{0, 1, \cdots, |\mathcal{V}| - 1\}$. For each node v on the

tree, we define $y_v := \mathbb{E}[f(I_v, a_v)] - I_v$, which is larger than

$$I_v \cdot \Pr[f(I_a, a_v) = I_v] + (I_v + 1) \cdot \Pr[f(I_v, a_v) > I_v] - I_v = \Pr[f(I_v, a_v) > I_v] = \mu_v.$$

We now define a sequence of random variables $\{Y_t\}_{t \in [T+1]}$:

$$Y_t = I_t - \sum_{i=1}^{t-1} y_{v_i}.$$

This sequence $\{Y_i\}$ is a martingale: conditioning on current value Y_t, we have

$$\begin{aligned}
\mathbb{E}[Y_{t+1} \mid Y_t] &= \mathbb{E}\left[I_{t+1} - \sum_{i=1}^{t} y_{v_i} \;\middle|\; Y_t \right] \\
&= \mathbb{E}\left[\left(I_t - \sum_{v=1}^{t-1} y_{v_i} \right) + I_{t+1} - I_t - y_{v_t} \;\middle|\; Y_t \right] \\
&= Y_t + \mathbb{E}[I_{t+1} \mid Y_t] - I_t - y_{v_t} = Y_t.
\end{aligned}$$

The last equation is due to the definition of y_{v_t}. By the martingale property, we have $\mathbb{E}[Y_{T+1}] = \mathbb{E}[Y_t] = Y_1 = 0$ for any $t \in [T]$. Thus, we have

$$|\mathcal{V}| \geq \mathbb{E}[I_{T+1}] = \mathbb{E}\left[\sum_{i=1}^{T} y_{v_i} \right] = \mathbb{E}\left[\sum_{v \in \mathcal{R}} y_v \right] \geq \mathbb{E}\left[\mu(\mathcal{R}) \right].$$

Let E be the set of realization paths r on the tree for which $\mu(r) \geq 1/\varepsilon$. Then, we have $\mathbb{E}[\mu(\mathcal{R})] \geq \sum_{r \in E} \left[\Phi(r) \cdot \frac{1}{\varepsilon} \right]$ which implies that $\sum_{r \in E} \Phi(r) \leq \varepsilon \cdot \mathbb{E}[\mu(\mathcal{R})] \leq O(\varepsilon)$, where $\Phi(r)$ is the probability of passing the path r. For each path $r \in E$, let v_r be the first node on the path such that $\mu(\mathcal{R}(v_r)) \geq 1/\varepsilon$, where $\mathcal{R}(v_r)$ is the path from the root to the node v_r. Let F be the set of such nodes. For the policy σ, we have a truncation on the node v_r when we reach the node v_r, *i.e.,* we do not select items (include v_r) any more in the new policy σ'. The total profit loss is at most

$$\sum_{v \in F} [\Phi(v) \cdot \mathbb{P}(v)] \leq \mathsf{MAX} \cdot \sum_{r \in E} \Phi(r) \leq O(\varepsilon) \cdot \mathsf{MAX},$$

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 $\mathcal{R}$, $\mu(\mathcal{R}) \leq O(1/\varepsilon)$.

3.6 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[6,11]. 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 of 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 1.1 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 (3-3), 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

$$\mathbb{P}(M) = \mathcal{G}_M^{\varphi} + \sum_{e=(M,N)} \pi_e^{\varphi} \cdot \mathbb{P}(N) \tag{3-5}$$

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:

$$\widetilde{\mathcal{G}}_M = \sum_{a\in M} \mathcal{G}_a(I_M) \quad \text{and} \quad \widetilde{\pi}_e = \sum_{a\in M}\left[\left(\prod_{b\in M\setminus a} \Phi_b(I_M, I_M)\right)\cdot \Phi_a(I_M, I_N)\right] \tag{3-6}$$

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

$$\widetilde{\mathbb{P}}(M) = \widetilde{\mathcal{G}}_M + \sum_{e=(M,N)} \widetilde{\pi}_e \cdot \widetilde{\mathbb{P}}(N), \tag{3-7}$$

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 3.2 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 3.2: For any block-adaptive policy $\hat{\sigma}$ satisfying Property (P1), we have

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

Proof. The right hand of this lemma can be proved by induction: for each block M on the decision tree, we have

$$\mathbb{P}(M) \ge (1 - \varepsilon^2)\cdot \widetilde{\mathbb{P}}(M). \tag{3-8}$$

If M is a leaf, we have $\mathbb{P}(M) = \widetilde{\mathbb{P}}(M)$ which implies that Equation (3-8) holds. For an internal block M, by Property (P1), we have

$$\mathcal{G}^{\varphi}_M \ge \left[\prod_{b\in M} \Phi_b(I_M, I_M)\right]\cdot \sum_{a\in M} \mathcal{G}_a(I_M) \ge \left[1 - \sum_{b\in M}\left(1 - \Phi_b(I_M, I_M)\right)\right]\cdot \widetilde{\mathcal{G}}_M \ge (1-\varepsilon^2)\cdot \widetilde{\mathcal{G}}_M$$

if M has more than one item and $\mathcal{G}^{\varphi}_M = \widetilde{\mathcal{G}}_M$ if M has only one item. For each edge

$e = (M, N)$, we have $\pi_e^{\varphi} \geq \widetilde{\pi}_e$. Then, by induction, we have

$$\begin{aligned}\mathbb{P}(M) &= \mathcal{G}_M^{\varphi} + \sum_{e=(M,N)} \pi_e^{\varphi} \cdot \mathbb{P}(N) \\ &\geq (1-\varepsilon^2) \cdot \widetilde{\mathcal{G}}_M + \sum_{e=(M,N)} \widetilde{\pi}_e \cdot \left[(1-\varepsilon^2) \cdot \widetilde{\mathbb{P}}(N)\right] \\ &= (1-\varepsilon^2) \cdot \widetilde{\mathbb{P}}(M). \end{aligned}$$

□

To prove the left hand of the lemma, we use Equation (3-4):

$$\mathbb{P}(\hat{\sigma}) = \sum_{M \in \mathcal{T}_{\hat{\sigma}}} \Phi(M) \cdot \mathcal{G}_M^{\varphi}$$

where $\Phi(M)$ is the probability of reaching the block M. For each edge $e = (M, N)$, if $I_M = I_N$ or M has only one item, we have $\widetilde{\pi}_e = \pi_e^{\varphi}$. Otherwise, we have

$$\widetilde{\pi}_e \geq \left[\prod_{b \in M} \Phi_b(I_M, I_M)\right] \cdot \sum_{a \in M} \Phi_a(I_M, I_N) \geq (1-\varepsilon^2) \cdot \sum_{a \in M} \Phi_a(I_M, I_N) \geq (1-\varepsilon^2) \cdot \pi_e^{\varphi}.$$

Then, for each block M and its realization path $\mathcal{R}(M) = (M_0, M_1, \ldots, M_m = M)$, we have

$$\frac{\widetilde{\Phi}(M)}{\Phi(M)} = \prod_{i=0}^{m-1} \frac{\widetilde{\pi}_{(M_i, M_{i+1})}}{\pi^{\varphi}_{(M_i, M_{i+1})}} = \prod_{i\,:\, I_{M_i} < I_{M_{i+1}}} \frac{\widetilde{\pi}_{(M_i, M_{i+1})}}{\pi^{\varphi}_{(M_i, M_{i+1})}} \geq (1-\varepsilon^2)^{|\mathcal{V}|} = 1 - O(\varepsilon^2),$$

where the last inequality holds because the value is nondecreasing and $|\mathcal{V}| = O(1)$. Thus we have

$$\widetilde{\mathbb{P}}(\hat{\sigma}) = \sum_{M \in \mathcal{T}_{\hat{\sigma}}} \widetilde{\Phi}_M \cdot \widetilde{\mathcal{G}}_M \geq \sum_{M \in \mathcal{T}_{\hat{\sigma}}} \left[(1 - O(\varepsilon^2)) \cdot \Phi(M)\right] \cdot \mathcal{G}_M \geq \left(1 - O(\varepsilon^2)\right) \cdot \mathbb{P}(\hat{\sigma}).$$

3.6.1 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[11].

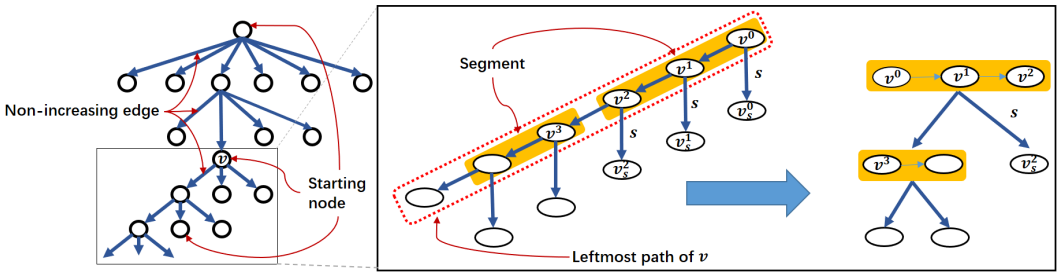

Figure 3.1　Decision tree and block policy

Lemma 3.3 : 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. 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. 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

$$|\mathbb{P}(u_s) - \mathbb{P}(w_s)| \le \varepsilon^2 \cdot \mathsf{MAX} \text{ and } \mu(M) \le \varepsilon^2. \tag{3-9}$$

Since $\mu(\mathcal{R})$ is at most $O(1/\varepsilon)$ for each root-to-leaf path $\mathcal{R}$ by Lemma 3.1, the second inequality in (3-9) can yield at most $O(\varepsilon^{-3})$ blocks. Now focus on the first inequality in (3-9). Fix a particular leftmost path $\mathcal{R}^v = (v^0, v^1, \ldots, v^m)$ from a starting node $v (v = v^0)$ on $\mathcal{T}_\sigma$. For each value $s \in \mathcal{V}$, we have

$$\mathsf{MAX} \ge \mathrm{DP}_1(s, \mathcal{A}) \ge \mathbb{P}(v_s^0) \ge \mathbb{P}(v_s^1) \ge \cdots \ge \mathbb{P}(v_s^m) \ge 0.$$

Otherwise, replacing the subtree $T_{v_s^i}$ with $T_{v_s^j}$ increases the profit of the policy σ for some $i < j \le m$ if $\mathbb{P}(v_s^i) < \mathbb{P}(v_s^j)$. Thus, for each particular size $s \in \mathcal{V}$, we could cut the path $\mathcal{R}^v$ at most ε^{-2} times. Since $|\mathcal{V}| = O(1)$, we have at most $O(\varepsilon^{-2})$ segments on the

leftmost path $\mathcal{R}^v$. Now, fix a particular root-to-leaf path. Since the value is nondecreasing by Assumption 1.1 (2), there are at most $|\mathcal{V}| = O(1)$ starting nodes on the path. Thus the first inequality in (3-9) can yield at most $O(\varepsilon^{-2})$ segments on the root-to-leaf path. In total, there are at most $O(\varepsilon^{-3})$ segments on any root-to-leaf path on the decision tree.

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 be taken and the value does not change, visit $l(v)_{I_v}$.
6: **until** A leaf on $\mathcal{T}_\sigma$ is reached. □

Now, we are ready to describe the algorithm, which takes a policy σ as input and outputs a block adaptive 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. Now we show that the expected profit $\mathbb{P}(\hat{\sigma})$ that the new policy $\hat{\sigma}$ can obtain is at least $\mathsf{OPT} - O(\varepsilon^2) \cdot \mathsf{MAX}$.

Our algorithm deviates the policy σ when the first time a node u in the segment $\mathsf{seg}(v)$ makes a transition to an increasing value, say s. In this case, σ would visit u_s, the s-child of u and follows $\mathcal{T}_{u_s}$ from then on. But our algorithm visits $l(v)_s$, the s-child of $l(v)$ (*i.e.,* the last node of $\mathsf{seg}(v)$), and follows $\mathcal{T}_{l(v)_s}$. The expected profit gap in each such event can be bounded by

$$\mathbb{P}(u_s) - \mathbb{P}(l(v)_s) \le \varepsilon^2 \cdot \mathsf{MAX},$$

due to the first inequality in Equation (3-9). Suppose σ pays such a profit loss, and switches to visit $l(v)_s$. Then, σ and our algorithm always stay at the same node. Note that there are at most $|\mathcal{V}| = O(1)$ starting nodes on any root-to-leaf path. Thus σ pays at

most $O(1)$ times in any realization. Therefore, the total profit loss is at most $O(\varepsilon^2) \cdot \mathsf{MAX}$. By Lemma 3.2, we have

$$\widetilde{\mathbb{P}}(\hat{\sigma}) \geq \left(1 - O(\varepsilon^2)\right) \cdot \mathbb{P}(\hat{\sigma}) \geq \left(1 - O(\varepsilon^2)\right) \cdot \left(\mathsf{OPT} - O(\varepsilon^2) \cdot \mathsf{MAX}\right) \geq \mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX}.$$

3.6.2 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

$$\bar{\Phi}_a(I, J) = \left\lfloor \Phi_a(I, J) \cdot \frac{n}{\varepsilon^4} \right\rfloor \cdot \frac{\varepsilon^4}{n} \quad \text{and} \quad \bar{\mathcal{G}}_a(I) = \left\lfloor \mathcal{G}_a(I) \cdot \frac{n}{\varepsilon^4 \mathsf{MAX}} \right\rfloor \cdot \frac{\varepsilon^4 \mathsf{MAX}}{n}$$

for any $J \in \mathcal{V}$. [①] 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 3.4 : 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 satisfy Property (P1) and (P2). If for each block M_1 on $\mathcal{T}_1$, the block M_2 at the

① 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.

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 $|\widetilde{\mathbb{P}}(\mathcal{T}_1) - \widetilde{\mathbb{P}}(\mathcal{T}_2)| \leq O(\varepsilon) \cdot \mathsf{MAX}$.

Proof. We focus on the case when M has more than one item. Recall that for each $e = (M, N)$, we have

$$\widetilde{\pi}_e = \sum_{a \in M} \left[\left(\prod_{b \in M \setminus a} \Phi_b(I_M, I_M) \right) \cdot \Phi_a(I_M, I_N) \right]$$

if $I_N > I_M$ and $\widetilde{\pi}_e = \prod_{a \in M} \Phi_a(I_M, I_M)$ if $I_N = I_M$. For simplicity, we use (I, J) to replace (I_M, I_N) if the context is clear, and write $\widetilde{\pi}_e$ as π_M^I if $J = I$ and π_M^J if $J > I$.

Fixing a block M, for each item $a \in M$, we define $\mu_a := \Pr[f(I, a) > I]$. By Property (P1) that $\mu(M) = \sum_{a \in M}[1 - \Phi_a(I, I)] = \sum_{a \in M} \mu_a \leq \varepsilon^2$, we have

$$\pi_M^I = \prod_{a \in M} (1 - \mu_a) \leq \left(1 - \frac{\sum_{a \in M} \mu_a}{|M|} \right)^{|M|} \leq \exp\left(-\mu(M)\right) \leq 1 - \mu(M) + \mu(M)^2 \leq 1 - \mu(M) + \varepsilon^4$$

and $\pi_M^I = \prod_{a \in M}(1 - \mu_a) \geq 1 - \sum_{a \in M} \mu_a = 1 - \mu(M)$. Since $\sum_{a \in M} \Phi_a(I, J) \leq \sum_{a \in M} \mu_a$ for any $J > I$, we have

$$\pi_M^J = \left[\prod_{b \in M} \Phi_b(I, I) \right] \cdot \left[\sum_{a \in M} \frac{\Phi_a(I, J)}{\Phi_a(I, I)} \right] \geq (1 - \varepsilon^2) \cdot \sum_{a \in M} \Phi_a(I, J) \geq \sum_{a \in M} \Phi_a(I, J) - \varepsilon^4.$$

It is straightforward to verify the following property when M has only one item:

$$\pi_M^I = 1 - \mu(M) \text{ and } \pi_M^J = \sum_{a \in M} \Phi_a(I, J) \text{ for any } J > I.$$

Let M_1, M_2 be the root blocks of $\mathcal{T}_1, \mathcal{T}_2$ respectively. Since $\mathsf{Sg}_I(M_1) = \mathsf{Sg}_I(M_2)$, we have that

$$\left| \sum_{a \in M_1} \Phi_a(I, J) - \sum_{a \in M_2} \Phi_a(I, J) \right| \leq \varepsilon^4,$$

for any $J \in \mathcal{V}$. Then, we have

$$\pi_{M_1}^I - \pi_{M_2}^I \leq 1 - \mu(M_1) + \varepsilon^4 - (1 - \mu(M_2)) = (\mu(M_2) - \mu(M_1)) + \varepsilon^4 = O(\varepsilon^4).$$

and for any $J > I$

$$\pi^J_{M_1} - \pi^J_{M_2} \le \sum_{a \in M_1} \Phi_a(I, J) - \sum_{a \in M_2} \Phi_a(I, J) + \varepsilon^4 = O(\varepsilon^4).$$

On the tree $\mathcal{T}_1$, we replace M_1 with M_2. For each $s \in \mathcal{V}$, we use M_s to denote the s-child of block M_1 on $\mathcal{T}_1$. Then we have

$$\begin{aligned}\widetilde{\mathbb{P}}(M_1) - \widetilde{\mathbb{P}}(M_2) &= \left(\widetilde{\mathcal{G}}_{M_1} - \widetilde{\mathcal{G}}_{M_2}\right) + \sum_{s \in \mathcal{V}} \widetilde{\mathbb{P}}(M_s) \cdot \left(\pi^s_{M_1} - \pi^s_{M_2}\right) \\ &\le \varepsilon^4 \cdot \mathsf{MAX} + O(\varepsilon^4) \cdot \mathsf{MAX} \\ &= O(\varepsilon^4) \cdot \mathsf{MAX}.\end{aligned}$$

We replace all the blocks on $\mathcal{T}_1$ by the corresponding blocks on $\mathcal{T}_2$ one by one from the root to leaf. The total profit loss is at most $\sum_{M \in \mathcal{T}_2} \left[\Phi(M) \cdot O(\varepsilon^4) \cdot \mathsf{MAX}\right] \le O(\varepsilon)\mathsf{MAX}$, where $\Phi(M)$ is the probability of reaching M. The inequality holds because the depth of $\mathcal{T}_2$ is at most $O(\varepsilon^{-3})$ by Property (P2), which implies that $\sum_{M \in \mathcal{T}_2} \Phi(M) \le O(\varepsilon^{-3})$. □

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 3.3, 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.

3.7 Finding a Nearly Optimal Block-adaptive Policy

In this section, we find a nearly optimal block-adaptive policy and prove Theorem 1.6. 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 $i \in [n]$, 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:

$$\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = \max\left\{\mathcal{M}(i-1, \mathsf{C}, \mathsf{CA}), \mathcal{M}(i-1, \mathsf{C}', \mathsf{CA}')\right\} \tag{3-10}$$

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
.

Proof. (The proof of Theorem 1.6) 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 3.3, 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 3.4, 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 3.2, 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 1.6 is completed. □

3.8 Partition Matroid Constraint

Our model proceeds for at most T rounds and each items can be chosen at most once, which forms a T-uniform matroid constraint. We can extend Theorem 1.6 to partition matroid, *i.e.,* the set of items satisfies partition matroid.

This proof is the same argument structure to of Theorem 1.6, except the last step, *i.e.,* finding a nearly optimal block-adaptive policy which satisfies the partition matroid $(U, \mathcal{I})$. Let $\{B_i\}_{i\in[\ell]}$ be a collection of disjoint sets, and let d_i be integers with $0 \le d_i \le |B_i|$. The independent sets $\mathcal{I}$ are the subsets I of U such that for every index $i \in [\ell]$, $|I \cap B_i| \le d_i$.

Let t_1 be the number of paths on $\mathcal{T}$. In dynamic programming, let $\mathsf{CA} = \{u_{i,j}\}_{(i,j)\in[t_1]\times[\ell]}$ be a capacity vector where $u_{i,j}$ is the upper bound of the number of items on $\mathcal{R}_i \cap B_j$ ($\mathcal{R}_i$ is the ith path). 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:

$$\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = \max\left\{\mathcal{M}(i-1, \mathsf{C}, \mathsf{CA}), \mathcal{M}(i-1, \mathsf{C}', \mathsf{CA}')\right\}. \tag{3-11}$$

Note the order of items is arranged by the $\{B_i\}_{i\in[\ell]}$. For any $i \in [t_1]$, the total number of capacity vectors $\mathsf{CA}_i = \{u_{i,j}\}_{j\in[\ell]}$ is $\sum_{i\in[\ell]} d_i$. Then total number of vectors CA is $(\sum_{i\in[\ell]} d_i)^{t_1}$ which is polynomial.

3.9 Summary

In this chapter, we introduce a general framework for stochastic dynamic programs. The main technical ingredient for obtaining the PTAS for the stochastic dynamic program is the idea of block adaptive policy, which was previously used in the context of stochastic knapsack. We believe the framework can be used in designing better approximation algorithms for other stochastic optimization problems.

第 4 章　Stochastic Probing Problem

In this chapter, we focus on the *stochastic probing* problem. Recall that on the special case *ProbeMax*, we are given n items, each associated with a known (discrete) distribution $\{\pi_i\}_{i\in[n]}$. When we *probe* the ith item, its value is realized, 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.

4.1 Model

The ProbeMax problem is a special case of the following general stochastic probing framework which was formulated by Gupta *et al.*[18]: An instance $\mathcal{J}$ of the stochastic probing problem is defined with tuple $\mathcal{J} = (U, \pi, \mathcal{I})$, where U is a set of n items, each having a value X_i following a known (discrete) distribution π_i, and $\pi = \{\pi_i\}_{[i\in U]}$. We can probe a subset $P \subseteq U$ of items sequentially and choose another subset $C \subseteq P$ as the final set. When we probe the ith item, its value is realized, which is sampled from the distribution π_i. (The item values are independent of each other.) Our goal is to design a policy such that the expected value $\mathbb{E}[\sum_{i\in C} X_i]$ is maximized. The constraint $\mathcal{I}$ consists of two *independent set* systems $\mathcal{I}_{out}$ and $\mathcal{I}_{in}$, which capture the *outer (packing) constraint* and the *inner (packing) constraint* respectively.

- *Outer constraint $\mathcal{I}_{out}$*[①]*:* this is the constraint on the sequence of items that we can probe. It requires that the sequence P of items that we have probed must be in $\mathcal{I}_{out}$ and each item can be probed at most once.
- *Inner constraint $\mathcal{I}_{in}$:* this is the constraint on the set of items that we can finally output. It requires that the set C of items that we output must be in $\mathcal{I}_{in}$.

Depending on how we choose the probing sequence P, we distinguish two classes of policies: *adaptive* and *non-adaptive*.

- Adaptive: we can adaptively choose the next item to probe based on the observations so far. In other words, at each time point we have access to the value realizations of

① $\mathcal{I}_{out}$ should be better understood as a collection of sequences. If $\mathcal{I}_{out}$ is a collection of subsets, then any sequence of a subset in $\mathcal{I}_{out}$ can be probed.

all the items we have probed so far.

- Non-adaptive: We should make decisions before observing the value realization of any of them. In other words, the probing sequence P is just a priori fixed probing order (probably over a subset of items) which satisfies the outer constraint $\mathcal{I}_{out}$.

In fact, every non-adaptive policy is a special case of adaptive policies. Depending on how we choose the final set C, we distinguish two natural models: the *committed* and *non-committed* models [①].

- Committed: once we probe an item to observe its value realization, we are committed to making 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.
- Non-committed: we can choose the set C after observing the values of all probed items in P. In other words, the choice of C can be made after all value realizations of the items in P, *i.e.,* the probing stage.

In fact, every committed policy is a special case of non-committed policies. The stochastic probing problem naturally generalizes the *ProbeMax* problem which is on the non-committed model. Here, the outer constraint $\mathcal{I}_{out}$ is the simple m-uniform matroid, *i.e.,* $\mathcal{I}_{out} = \{P \mid |P| \leq m\}$. The inner constraint $\mathcal{I}_{in}$ is the 1-uniform matroid, *i.e.,* we can output one item. We can replace the outer constraint with some more general constraint, *e.g.,* a matroid. If replace the inner constraint in ProbeMax with a k-uniform matroid (*i.e.,* $\mathcal{I}_{in} = \{C \mid |C| \leq k\}$), we obtain the ProbeTop-k problem. In this thesis, we assume k is a constant.

There are four different types (AN, NN, AC, NC) of policies for the stochastic probing problem as showed in Figure 4.1, where the first symbol indicates if a policy is adaptive (A) or not (N) while the second indicates if a policy is committed (C) or not (N). For a particular stochastic probing instance $\mathcal{J}$, we use $\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})$ to denote the expected value of an optimal policy restricted to the type AN. We also use the shorthand notation $\mathsf{OPT}_{\mathsf{AN}}$ if the context is clear. Similarly we have $\mathsf{OPT}_{\mathsf{AC}}(\mathcal{J})$, $\mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})$ and $\mathsf{OPT}_{\mathsf{NC}}(\mathcal{J})$.

Notice that $\mathsf{OPT}_{\mathsf{AN}}$ is no less than $\mathsf{OPT}_{\mathsf{NN}}$ and $\mathsf{OPT}_{\mathsf{AC}}$ for each given instance $\mathcal{J}$. Hence, it would be interesting to study the following quantities, called *adaptivity gap* and *commitment gap* respectively, which measures how much the former can be larger than the later.

Definition 4.1 : *(Adaptivity Gap[②])* For a given stochastic probing instance $\mathcal{J}$, we define

① In [18], they are called online and offline decision-making models.

② In [18], they considered the committed model and referred to the adaptivity gap as $\mathsf{Gap}_{\mathcal{J}}(\mathsf{AC}, \mathsf{NC}) = \frac{\mathsf{OPT}_{\mathsf{AC}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{NC}}(\mathcal{J})}$,

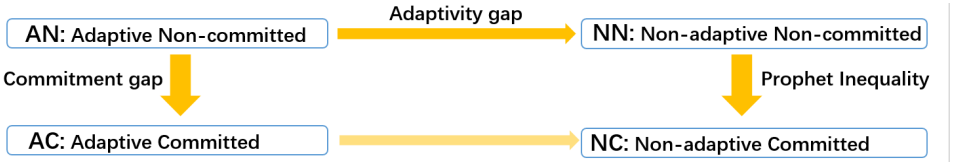

Figure 4.1　Adaptivity gap and Commitment gap

the adaptivity gap for instance $\mathcal{J}$ on the non-commited model as

$$\mathsf{AdaptiveGap}(\mathcal{J}) = \frac{\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})} \tag{4-1}$$

Definition 4.2:　*(Commitment Gap*[①] *)* For a particular stochastic probing instance $\mathcal{J}$, we define the commitment gap for instance $\mathcal{J}$ as

$$\mathsf{CommitGap}(\mathcal{J}) = \frac{\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{AC}}(\mathcal{J})} \tag{4-2}$$

Consider the stochastic probing framework where the order of items we can probe is fixed, instead of being chosen by the policy. This can be captured by the outer constraint contains all prefix orders. For the special case in which we can only choose one item was studied by Krengel *et al.*[29] back in the 70s. They designed a committed policy which returns a single item of expected value at least half of $\mathbb{E}[\max_{i\in[n]} X_i]$, *i.e.*, there exists a stopping rule τ such that

$$2 \cdot \mathbb{E}[X_\tau] \geq \mathbb{E}[\max_{i\in[n]} X_i]. \tag{4-3}$$

This is the well known *Prophet Inequality*, which implies that the commitment gap of any ProbeMax problem is at most 2 when the outer constraint is induced by a fixed order. Recently, Kleinberg and Weinberg[30] generalized the result when we are allowed to make more than one selection, subject to a matroid inner constraint and showed that the factor 2 still can be achieved. They also showed that under an intersection of p matroids inner constraint, a factor $O(p)$ can be achieved. We call the factor *prophet inequality factor* and denote as $\mathsf{ProphetFactor}(\mathcal{I}_{in})$ for an inner constraint $\mathcal{I}_{in}$. Then, for an optimal type NN policy for instance $\mathcal{J}$, there exists a corresponding type NC policy with the same probing order which achieves the expected value at least $(1/\mathsf{ProphetFactor}(\mathcal{I}_{in})) \cdot \mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})$,

which can be bounded by $\mathsf{Gap}_{\mathcal{J}}(\mathsf{AN}, \mathsf{NN}) \times \mathsf{Gap}_{\mathcal{J}}(\mathsf{NN}, \mathsf{NC})$.

① For non-adaptive policies, $\mathsf{Gap}_{\mathcal{J}}(\mathsf{NN}, \mathsf{NC}) = \frac{\mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{NC}}(\mathcal{J})}$ is bounded by a prophet inequality factor which we will discuss later.

where $\mathcal{I}_{in}$ is the inner constraint of $\mathcal{J}$. Thus we have

$$\mathsf{Gap}_{\mathcal{J}}(\mathsf{NN},\mathsf{NC}) = \frac{\mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{NC}}(\mathcal{J})} \leq \mathsf{ProphetFactor}(\mathcal{I}_{in}), \tag{4-4}$$

where $\mathcal{I}_{in}$ is the inner constraint of $\mathcal{J}$ and the following proposition.

Proposition 4.1 : For any stochastic probing instance $\mathcal{J}$, we have

$$\mathsf{CommitGap}(\mathcal{J}) \leq \mathsf{AdaptiveGap}(\mathcal{J}) \times \mathsf{ProphetFactor}(\mathcal{I}_{in}), \tag{4-5}$$

where $\mathcal{I}_{in}$ is the inner constraint of $\mathcal{J}$.

4.2 Our Results

ProbeMax problem: 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.*[13]). Chen *et al.*[13] obtained the first PTAS. When considering the adaptive policies, Munagala[14] 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[15,16]). They also showed that the *adaptivity gap* 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.*[17]. This is also a non-adaptive policy, which implies the adaptivity gap is at most $\frac{e}{e-1}$. In this thesis, we provide the first PTAS, among all adaptive policies. Note that our policy is indeed adaptive.

Theorem 4.1 : There exists a PTAS for the ProbeMax problem among all adaptive policies. In other words, for any fixed constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for the ProbeMax problem that finds a policy (type AN) with the expected value at least $(1 - \varepsilon)\mathsf{OPT}_{\mathsf{AN}}$.

Our techniques also allow us to derive the following result.

Corollary 4.1：　For the ProbeTop-k problem,

1. There is a polynomial time algorithm that finds a policy (type AN) with the expected value at least $(1-\varepsilon)\mathsf{OPT}_{\mathsf{AN}}$.
2. There is a polynomial time algorithm that finds a non-adaptive policy (type NN) with the expected value at least $(1-\varepsilon)\mathsf{OPT}_{\mathsf{NN}}$.

Our techniques: We use the framework formulated in Chapter 3. Let the value I_t be the maximum among the realized values of the probed items at the time period t. Then, we have the following system dynamics for ProbeMax:

$$I_{t+1} = f(I_t, i) = \max\{I_t, X_i\}, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = I_{T+1} \tag{4-6}$$

$t = 1, 2, \ldots, T$. Clearly, Assumption 1.1 (2) and (3) are satisfied. But Assumption 1.1 (1) is not satisfied because the value space $\mathcal{V}$ is not of constant size. Hence, we need to discretize the value space and reduce its size to constant. See Section 4.4 for more details.

Table 4.1　A summary of approximation ratios for the stochastic probing problem

Inner constraint	Outer constraint	Committed $\mathsf{OPT}_{\mathsf{AC}}$	Non-committed $\mathsf{OPT}_{\mathsf{AN}}$
1-Uniform	Uniform, Partition	PTAS [Theorem 1.5 in[11]]	$1-1/e$[17], $1/8$[14] PTAS [Theorem 4.1]
	General matroid	PTAS [Theorem 4.2]	$1-1/e$[17]
	Two matroids	PTAS [Theorem 4.2]	$1/3$ [Theorem 4.6] $1/2+\varepsilon$ [Theorem 4.2, 4.4]①
k-Uniform	Uniform, Partition	PTAS [Theorem 4.3]	$1-1/e$[17], PTAS [Corollary 4.1]
General matroid	General matroid	$1/2$ [Theorem 4.6]	$1-1/e$[17]
k^{in} matroids	k^{out} matroids	$\frac{1}{k^{in}+k^{out}}$ [Theorem 4.6]	

Committed ProbeMax Problem: We call the committed ProbeMax problem when the item is chosen in the committed way. A PTAS for the committed ProbeMax problem can be obtained by a reduction from the committed stochastic knapsack (see Chapter 5) while

① The PTAS for the committed ProbeMax problem provides a $(0.5+\varepsilon)$-approximation ratio for the non-committed ProbeMax problem by Theorem 4.4.

the capacity is 1 and each item has a fixed size 1. Based on the similar technique and Multi-budget optimization [31], we can obtain the first PTAS for the committed ProbeMax problem with more general outer constraint. For the committed ProbeTop-k problem, we use the framework of stochastic dynamic program formulated in Chapter 3 to obtain a PTAS.

Theorem 4.2 : There is a polynomial time algorithm that finds a committed policy (type NC) with the expected value at least $(1-\varepsilon)\mathsf{OPT}_{\mathsf{AC}}$ for the committed ProbeMax problem when the outer constraint is a matroid or the intersection of two matroids.

Theorem 4.3 : There is a polynomial time algorithm that finds a committed policy with the expected profit at least $(1-\varepsilon)\mathsf{OPT}_{\mathsf{AC}}$ for the committed ProbeTop-k problem, where $\mathsf{OPT}_{\mathsf{AC}}$ is the expected total profit obtained by the optimal policy.

Our techniques: On the committed ProbeMax problem, we are committed to making a decision immediately whenever we probe an item. If we accept the item, we stop, otherwise reject the item and go on. Thus a committed policy for the committed ProbeMax problem can be represented by a path, rather than a decision tree. Then we can approximate the signature of the path by multi-budget optimization to get a nearly optimal policy.

For the committed ProbeTop-k problem, we use the framework of stochastic dynamic program formulated in Chapter 3 to obtain a PTAS. More precisely, 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.

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + 1 & \text{if } X_i \geq \theta, I_t < k, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} X_i & \text{if } X_i \geq \theta, I_t < k, \\ 0 & \text{otherwise;} \end{cases} \tag{4-7}$$

for $t = 1, 2, \ldots, T$, and $h(I_{T+1}) = 0$. Since k is a constant, Assumption 1.1 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 $\mathcal{B}_i = \{b_i^\theta\}_\theta$. See Section 4.5.1 for more details.

Commitment Gap: Gupta *et al.* [18] 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. By Proposition 4.1, we can immediately obtain that the commitment gap of any stochastic

probing problem is at most $O(p \cdot p^3 \log(np))$ for the same inner outer constraints. Now, we bound the commitment gap for the uniform matroid inner constraint and an arbitrary prefix-closed outer constraint.

Theorem 4.4：The commitment gap of any stochastic probing problem where the outer constraint is prefix-closed and the inner constraint is a uniform matroid is at most 2.

Note that Theorem 4.4 is a generalization of Inequality (4-3) which considered a special case when the inner constraint is a 1-uniform matroid while the outer constraint is induced by a fixed order. Notice that the factor 2 in Inequality (4-3) is tight, which implies the result in Theorem 4.4 is also tight. Our techniques also allow us to derives the following useful result from the proof of Theorem 4.4, which improves the result of 3.51 in[18].

Corollary 4.2：The adaptivity gap of any stochastic probing problem where the outer constraint is prefix-closed and the inner constraint is a uniform matroid is at most 2.

Our techniques: Before talking about our techniques, we present the previous approaches to bounding the adaptivity gap for arbitrary outer constraints. Gupta *et al.*[18,32] worked on the optimal decision tree directly. In order to upper bound the adaptive gap, they took a random path (following the probability of the path in the tree) down the tree and showed that the expected value of this path, regarded as a non-adaptive strategy, was good. A natural way is to consider the best path of the tree which with the best the expected value, instead of the random path following a distribution. Our approach derives from this and shows that there exists a path with the expected value of $\frac{\mathsf{OPT}_{\mathsf{AN}}}{2}$. The algorithm is quite simple: compute an identical threshold and accept the items whose values exceed the threshold on the path. This is a non-adaptive committed (type NC) policy.

Committed Pandora's Box Problem: For Weitzman's "Pandora's box" problem[33], 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 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[34] generalized this model to other combinatorial optimization problems such as matching, set cover and so on. In this section, we focus on the committed model. 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 4.5: 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.

Our techniques: 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:

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + 1 & \text{if } X_i \geq \theta, I_t < k, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} X_i - c_i & \text{if } X_i \geq \theta, I_t < k, \\ -c_i & \text{otherwise;} \end{cases} \tag{4-8}$$

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 1.1 is immediately satisfied. See Section 4.5.4 for more details.

Stochastic Probing: The Bernoulli version of stochastic probing was introduced in[16], where each item $i \in U$ has a fixed value w_i and is "active" with an independent probability p_i. If an item i is probed and found to be "active", then it must be added to the final set C. It can be equivalently viewed as a special case of stochastic probing when each item has a Bernoulli random value X_i which is equal to w_i with p_i and 0 otherwise. They

essentially considered the *committed* setting since we would never probe an item which we prepare to discard when it is "active". Gupta *et al.*[16] 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.*[35] using the iterative randomized rounding approach. We generalize this result to the case when the value of each item is no longer restricted as a Bernoulli random variable. Our approach is a slight generalization of[16,35].

Theorem 4.6: There is a polynomial time algorithm that finds an adaptive committed policy (type AC) for the stochastic probing problem with the expected value at least $\frac{\mathsf{OPT}_{\mathsf{AN}}}{k^{in}+k^{out}}$, when $\mathcal{I}_{out}$ is an intersection of k^{out} matroids and $\mathcal{I}_{in}$ is an intersection of k^{in} matroids.

4.3 Related work

If there is no outer constraint for the stochastic probing on the committed model, *i.e.,* we can probe any sequence of items we need, this is called *Bayesian online selection problem* (BOSP). Chawla *et al.*[36] proposed a simple mechanism, called *sequential posted pricing* mechanism (SPM), for *Bayesian single-parameter auctions*. They gave a committed algorithm with a ratio of $\frac{1}{2}$ (compares to the best non-committed policy) for BOSP where the inner constraint is a matroid and Yan[37] improved the ratio to $1-\frac{1}{e}$ which is tight. This implies that the commitment gap for the matroid inner constraint is at most $1-\frac{1}{e}$ when there is no outer constraint.

In another setting, the order is random instead of being chosen by the policy, which was introduced by Esfandiari *et al.*[38]. The approximation ratio for the single-item case can be improved from $\frac{1}{2}$ to $1-\frac{1}{e}$ which compares to the prophet inequality where the order is fixed. They called this *prophet secretary* problem and showed that the upper bound of the ratio is 0.75. When the distributions of items are unknown and the order is random, this is the traditional *secretary problem*, introduced by Dynkin in 1960s[39]. They designed a simple committed strategy with a ratio of $\frac{1}{e}$ which is tight for large n. Kleinberg[40] provided an algorithm with ratio of $1-O(\sqrt{1/k})$ when inner constraint is a k-uniform matroid. However, for the general matroid inner constraint, the current best ratio is $O(1/\log\log r)$[41,42], where r is the rank of the given matroid.

4.4 ProbeMax Problem

In this section, we demonstrate the application of our framework to the ProbeMax problem. Define the value set $S = \bigcup_{i\in[n]} S_i$ where S_i is the support of the random variable X_i and the item set $\mathcal{A} = \{1, 2, \ldots, n\}$. Let the value I_t be the maximum among the realized values of the probed items at the time period t. Thus, we begin with the initial value $I_1 = 0$. Since we can probe at most m items, we set the number of rounds to be $T = m$. When we probe an item i and observe its value realization, say X_i, we have the system dynamic functions

$$I_{t+1} = f(I_t, i) = \max\{I_t, X_i\}, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = I_{T+1} \tag{4-9}$$

for $I_t \in S$ and $t = 1, 2, \ldots, T$. Assumption 1.1 (2,3) is immediately satisfied. But Assumption 1.1 (1) is not satisfied because the value space S is not of constant size. Hence, we need discretization.

4.4.1 Discretization

Now, we need to discretize the value space, using parameter ε. We start with a constant factor approximate solution $\widetilde{\mathsf{OPT}}$ for the ProbeMax problem with $\mathsf{OPT} \geq \widetilde{\mathsf{OPT}} \geq (1 - 1/e)^2\mathsf{OPT}$ (this can be obtained by a simple greedy algorithm See e.g., Appendix C of[13]). Let X be a discrete random variable with a support $S = (s_1, s_2, \ldots, s_l)$ and $p_{s_i} = \Pr[X = s_i]$. Let $\theta = \frac{\widetilde{\mathsf{OPT}}}{\varepsilon}$ be a threshold. For "large" size s_i, *i.e.,* $s_i \geq \theta$, set $D_X(s_i) = \theta$. For "small" size s_i, *i.e.,* $s_i < \theta$, set $D_X(s_i) = \left\lfloor \frac{s_i}{\varepsilon\widetilde{\mathsf{OPT}}} \right\rfloor \varepsilon\widetilde{\mathsf{OPT}}$. We use $\mathcal{V} = \{0, \varepsilon\widetilde{\mathsf{OPT}}, \ldots, \widetilde{\mathsf{OPT}}/\varepsilon\}$ to denote the discretized support. Now, we describe the discretized random variable $\widetilde{X}$ with the support $\mathcal{V}$. For "large" size, we set

$$\widetilde{p}_\theta = \Pr[\widetilde{X} = \theta] = \Pr[X \geq \theta] \cdot \frac{\mathbb{E}[X \mid X \geq \theta]}{\theta}. \tag{4-10}$$

Under the constraint that the sum of probabilities remains 1, for "small" size $d \in \mathcal{V} \setminus \{\theta\}$, we scale down the probability by setting

$$\widetilde{p}_d = \Pr[\widetilde{X} = d] = \frac{1 - \Pr[\widetilde{X} = \theta]}{\Pr[X < \theta]} \cdot \left(\sum_{s\in S, D_X(s)=d} \Pr[X = s] \right) \leq \sum_{s\in S, D_X(s)=d} \Pr[X = s]. \tag{4-11}$$

Although the above discretization is quite natural, there are some technical details. We know how to solve the problem for the discretized random variables supported on $\mathcal{V}$ but the realized values are in S. Hence, we need to introduce the notion of *canonical policys* (the notion was introduced in Bhalgat et al.[6] for stochastic knapsack). The policy makes decisions based on the discretized sizes of variables, not their true size. More precisely, when the canonical policy $\widetilde{\sigma}$ probes an item X which realizes to $s \in S$, the policy makes decisions based on discretized size $D_X(s)$. In this following lemma, we show it suffices to only consider canonical policies. We use $\mathbb{P}(\sigma, \pi)$ to denote the expected profit that the policy σ can obtain with the given distribution π.

Lemma 4.1 : Let $\pi = \{\pi_i\}$ be the set of distributions of random variables and $\widetilde{\pi}$ be the discretized version of π. Then, we have:

1. For any policy σ, there exists a (canonical) policy $\widetilde{\sigma}$ such that

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi) - O(\varepsilon)\mathsf{OPT};$$

2. For any canonical policy $\widetilde{\sigma}$,

$$\mathbb{P}(\widetilde{\sigma}, \pi) \geq \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}).$$

Proof. (The proof of Theorem 4.1) Suppose σ^* is the optimal policy with expected profit $\mathbb{P}(\sigma^*, \pi) = \mathsf{OPT}$. Given an instance π, we compute the discretized distribution $\widetilde{\pi}$. By Lemma 4.1 (1), there exists a canonical policy $\widetilde{\sigma}^*$ such that

$$\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}) \geq (1 - O(\varepsilon)) \cdot \mathbb{P}(\sigma^*, \pi) - O(\varepsilon)\mathsf{OPT} = (1 - O(\varepsilon))\mathsf{OPT}.$$

Now, we present a stochastic dynamic program for the ProbeMax problem with the discretized distribution $\widetilde{\pi}$. Define the value set $\mathcal{V} = \{0, \varepsilon\widetilde{\mathsf{OPT}}, \ldots, \widetilde{\mathsf{OPT}}/\varepsilon\}$ and the item set $\mathcal{A} = \{1, 2, \ldots, n\}$, and set $T = m$ and $I_1 = 0$. When we probe an item i to observe its value realization, say X_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, i) = \max\{I_t, X_i\}, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = I_{T+1} \tag{4-12}$$

for $I_t \in \mathcal{V}$ and $t = 1, 2, \ldots, T$. Then Assumption 1.1 is immediately satisfied. By

Theorem 1.6, we can find a policy σ with profit at least

$$\mathsf{OPT}_d - O(\varepsilon^2) \cdot \mathsf{MAX},$$

where OPT_d denotes the expected profit of the optimal policy for the discretized version $\widetilde{\pi}$ and $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(\widetilde{\mathsf{OPT}}/\varepsilon, \mathcal{A}) = \widetilde{\mathsf{OPT}}/\varepsilon$. We can see that $\mathsf{OPT}_d \geq \mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}) \geq (1 - O(\varepsilon))\mathsf{OPT}$. Thus, by Lemma 4.1 (2), we have

$$\mathbb{P}(\sigma, \pi) \geq \mathbb{P}(\sigma, \widetilde{\pi}) \geq \mathsf{OPT}_d - O(\varepsilon^2)\mathsf{MAX} \geq (1 - O(\varepsilon))\mathsf{OPT} - O(\varepsilon)\mathsf{OPT} = (1 - O(\varepsilon))\mathsf{OPT},$$

which completes the proof. □

Proof. (Proof of Lemma 4.1) Recall that for each node v on the decision tree $\mathcal{T}_\sigma$, the value I_v is the maximum among the realized value of the probed items right before probing the item a_v. For a path $\mathcal{R}$, we use $W(\mathcal{R})$ to the denote the value of the last node on the path. Let E be the set of all root-to-leaf paths in T_σ. Then we have

$$\mathbb{P}(\sigma, \pi) = \sum_{r \in E} \Phi(r) \cdot W(r). \tag{4-13}$$

For the first result of Lemma 4.1, we prove that there is a randomized canonical policy σ_r such that $\mathbb{P}(\sigma_r, \widetilde{\pi}) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi) - O(\varepsilon)\mathsf{OPT}$. Thus such a deterministic policy $\widetilde{\sigma}$ exists. Let $\theta = \frac{\widetilde{\mathsf{OPT}}}{\varepsilon}$ be a threshold. We interrupt the process of the policy σ on a node v when the first time we probe an item whose weight exceeds this threshold to get a new policy σ' *i.e.,* we have a truncation on the node v and do not probe items (include v) any more in the new policy σ'. The total profit loss is equal to

$$\sum_{v \in LF} \Phi(v) \cdot [\mathbb{P}(v) - I_v] \leq \sum_{v \in LF} \Phi(v) \cdot \mathsf{OPT} = \mathsf{OPT} \times \sum_{v \in LF} \Phi(v) \leq O(\varepsilon) \cdot \mathsf{OPT},$$

where LF is the set of the nodes on which we have a truncation. The last inequality holds because $\mathsf{OPT} \geq \sum_{v \in LF} \Phi(v) \cdot \mathbb{P}(v) \geq \theta \cdot \sum_{v \in LF} \Phi(v)$.

The randomized policy σ_r is derived from σ' as follows. $\mathcal{T}(\sigma_r, \widetilde{\pi})$ has the same tree structure as $\mathcal{T}(\sigma', \pi)$. If σ_r probes an item $\widetilde{X}$ and observes a discretized size $d \in \mathcal{V}$,

it chooses a random branch in $\mathcal{T}(\sigma_r, \widetilde{\pi})$ among those sizes that are mapped to d, i.e., $\{w_e \mid D_X(w_e) = d\}$ according to the probability distribution

$$\Pr[\text{branch } e \text{ is chosen}] = \frac{\Pr[X = w_e]}{\sum_{s \mid D_X(s)=d} \Pr[X = s]}.$$

Then by Equation (A-8), if $w_e < \theta$ we have

$$\widetilde{p}_e = \Pr[\widetilde{X} = d] \cdot \Pr[\text{branch } e \text{ is chosen}] = p_e \cdot \frac{1 - \Pr[\widetilde{X} = \theta]}{\Pr[X < \theta]}$$

and

$$\widetilde{w}_e = \left\lfloor \frac{w_e}{\varepsilon\widetilde{\mathsf{OPT}}} \right\rfloor \cdot \varepsilon\widetilde{\mathsf{OPT}} \geq w_e - \varepsilon\widetilde{\mathsf{OPT}}.$$

Fact: For any node v in the tree $\mathcal{T}(\sigma', \pi)$ such that $I_v < \theta$, we have

$$\widetilde{\Phi}(v) \geq (1 - O(\varepsilon))\Phi(v). \tag{4-14}$$

When we regard the path $\mathcal{R}(v)$ as a policy, the expected profit of the path $\mathcal{R}(v)$ can obtain is at least

$$\theta \cdot \left[1 - \prod_{i \in \mathcal{R}(v)} \left(1 - \Pr[\widetilde{X}_i = \theta]\right)\right],$$

which is less than OPT, where $\mathcal{R}(v)$ is the path from the root to the node v. Then we have $\prod_{i \in \mathcal{R}(v)} \left(1 - \Pr[\widetilde{X}_i = \theta]\right) \geq 1 - O(\varepsilon)$, which implies that

$$\widetilde{\Phi}(v) = \Phi(v) \cdot \prod_{i \in \mathcal{R}(v)} \frac{1 - \Pr[\widetilde{X}_i = \theta]}{\Pr[X_i \leq \theta]} \geq (1 - O(\varepsilon))\Phi(v).$$

Now we bound the profit that we can obtain from $\mathcal{T}(\sigma_r, \widetilde{\pi})$. Let E be the set of all root-to-leaf paths in $\mathcal{T}(\sigma', \pi)$. We split it into two parts $E_1 = \{r \in E : W(r) < \theta\}$ and $E_2 = \{r \in E : W(r) \geq \theta\}$. For the first part, we have

$$\sum_{r \in E_1} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) \geq \sum_{r \in E_1} \widetilde{\Phi}(r) \cdot \left[W(r) - O(\varepsilon\widetilde{\mathsf{OPT}})\right]$$

$$\geq (1 - O(\varepsilon)) \left[\sum_{r \in E_1} \Phi(r) \cdot W(r) \right] - O(\varepsilon \widetilde{\mathsf{OPT}}).$$

As mentioned before, for any path $r \in E_2$, we interrupt the process of the policy σ when the first time we probe an item whose weight exceeds this threshold θ. We use ℓ_r to denote the item for path r. By Equation (A-7), we have $\Pr[\widetilde{X} = \theta] \cdot \theta = \Pr[X \geq \theta] \cdot \mathbb{E}[X \mid X \geq \theta]$. Then, we have

$$\begin{aligned}
\sum_{r \in E_2} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) &= \sum_{r \in E_2} \widetilde{\Phi}(\ell_r) \cdot \Pr[\widetilde{X}_{\ell_r} = \theta] \cdot \theta \\
&= \sum_{r \in E_2} \widetilde{\Phi}(\ell_r) \cdot \Pr[X_{\ell_r} \geq \theta] \cdot \mathbb{E}[X_{\ell_r} \mid X_{\ell_r} \geq \theta] \\
&\geq \sum_{r \in E_2} (1 - O(\varepsilon)) \Phi(\ell_r) \cdot \Pr[X_{\ell_r} \geq \theta] \cdot \mathbb{E}[X_{\ell_r} \mid X_{\ell_r} \geq \theta] \\
&= (1 - O(\varepsilon)) \sum_{r \in E_2} \Phi(r) \cdot W(r).
\end{aligned}$$

In summation, the expected profit $\mathbb{P}(\sigma_r, \widetilde{\pi})$ is equal to

$$\begin{aligned}
\sum_{r \in E} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) &\geq (1 - O(\varepsilon)) \sum_{r \in E} \Phi(r) \cdot W(r) - O(\varepsilon \widetilde{\mathsf{OPT}}) \\
&= (1 - O(\varepsilon)) \mathbb{P}(\sigma', \pi) - O(\varepsilon) \mathsf{OPT} \\
&= (1 - O(\varepsilon)) \mathbb{P}(\sigma, \pi) - O(\varepsilon) \mathsf{OPT}.
\end{aligned}$$

Next, we prove the second result of Lemma 4.1. Recall that a canonical policy makes decisions based on the discretized sizes. Then $\mathcal{T}(\widetilde{\sigma}, \pi)$ has the same tree structure as $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi})$, except that it obtains the true profit rather than the discretized profit. By Equation (A-8), for an edge e with a weight $\widetilde{w}_e < \theta$ on $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi})$, we have

$$\pi_e = \sum_{s \in S : D_X(s) = \widetilde{w}_e} \Pr[X = s] \geq \widetilde{\pi}_e.$$

Fact: For any node v in the tree $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi})$ with $I_v < \theta$, we have

$$\widetilde{\Phi}(v) \leq \Phi(v). \tag{4-15}$$

Similarly, we split the root-to-leaf paths set E into two parts $E_1 = \{r \in E : \max_{e \in r} \widetilde{w}_e < \theta\}$ and $E_2 = \{r \in E : \max_{e \in r} \widetilde{w}_e = \theta\}$. Then, we have

$$\begin{aligned}
\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}) &= \sum_{r \in E_1} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) + \sum_{r \in E_2} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) \\
&= \sum_{r \in E_1} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) + \sum_{r \in E_2} \widetilde{\Phi}(\ell_r) \cdot \Pr[\widetilde{X}_{\ell_r} = \theta] \cdot \theta \\
&\leq \sum_{r \in E_1} \Phi(r) \cdot W(r) + \sum_{r \in E_2} \Phi(\ell_r) \cdot \Pr[X_{\ell_r} \geq \theta] \cdot \mathbb{E}[X_{\ell_r} \mid X_{\ell_r} \geq \theta] \\
&= \sum_{r \in E_1} \Phi(r) \cdot W(r) + \sum_{r \in E_2} \Phi(r) \cdot W(r) \\
&= \mathbb{P}(\widetilde{\sigma}, \pi)
\end{aligned}$$

□

4.4.2 ProbeTop-k Problem

In this section, we consider the ProbeTop-k problem where the reward is the summation of top-k values and k is a constant.

Theorem 4.7 : There exists a PTAS for the ProbeTop-k problem. In other words, for any fixed constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for the ProbeTop-k problem that finds a policy with the expected value at least $(1 - \varepsilon)\mathsf{OPT}$.

In this case, I_t is a vector of the top-k values among the realizes values of the probed items at the time period t. Thus, we begin with the initial vector $I_1 = \{0\}^k$. When we probe an item i and observe its value realization X_i, we update the vector by

$$I_{t+1} = \{I_t + X_i\} \setminus \min\{I_t, X_i\}.$$

We set $g(I_t, i) = 0$ and $h(I_{T+1}) = \mathrm{sum}(I_{T+1})$. Assumption 1.1 (2,3) is immediately satisfied. Then We also need the discretization to satisfy the Assumption 1.1 (1). For Lemma 4.1, we make a small change as shown in Lemma 4.2.Thus, we can prove Theorem 4.7 which is essentially the same as the proof of Theorem 4.1 and we omit it here.

Lemma 4.2 : Let $\pi = \{\pi_i\}$ be the set of distributions of random variables and $\widetilde{\pi}$ be the discretized version of π. Then, we have:

1. For any policy σ, there exists a canonical policy $\widetilde{\sigma}$ such that

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi) - O(\varepsilon)\mathsf{OPT};$$

2. For any canonical policy $\widetilde{\sigma}$,

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

Proof. This can be proved by an analogous argument as Lemma 4.1. For the first result, we design a randomized canonical policy σ_r as before. Here, $W(r)$ is the summation of the top-k weights on the path r. For a root-to-leaf path r, the profit we get is equal to

$$W(r) = \max_{C \subseteq r, |C| \leq k} \left[\sum_{i \in C} X_i \right]. \tag{4-16}$$

Now we bound the profit we can obtain from $\mathcal{T}(\sigma_r, \widetilde{\pi})$. recall that $E_1 = \{r \in E : \max_{e \in r} w_e < \theta\}$ and $E_2 = \{r \in E : \max_{e \in r} w_e \geq \theta\}$ where E is the set of all root-to-leaf paths. Then for any $r \in E_1$, we have

$$\widetilde{W}(r) \leq W(r) - k \cdot \varepsilon\widetilde{\mathsf{OPT}} = W(r) - O(\varepsilon\widetilde{\mathsf{OPT}}).$$

For the first part, we have

$$\sum_{r \in E_1} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) \geq (1 - O(\varepsilon)) \sum_{r \in E_1} [\Phi(r) \cdot W(r)] - O(\varepsilon\widetilde{\mathsf{OPT}}).$$

For the second part, we have

$$\begin{aligned}
\sum_{r \in E_2} \widetilde{\Phi}(r) \cdot \widetilde{W}(r) &= \sum_{r \in E_2} \widetilde{\Phi}(\ell_r) \cdot \Pr[\widetilde{X}_{\ell_r} = \theta] \cdot (\theta + \widetilde{W}'(\ell_r)) \\
&\geq \sum_{r \in E_2} \widetilde{\Phi}(\ell_r) \cdot \Pr[X_{\ell_r} \geq \theta] \cdot \left(\mathbb{E}[X_{\ell_r} | X_{\ell_r} \geq \theta] + \widetilde{W}'(\ell_r) \right) \\
&\geq \sum_{r \in E_2} \widetilde{\Phi}(\ell_r) \cdot \Pr[X_{\ell_r} \geq \theta] \cdot \left(\mathbb{E}[X_{\ell_r} | X_{\ell_r} \geq \theta] + W'(\ell_r) - O(\varepsilon\widetilde{\mathsf{OPT}}) \right) \\
&\geq (1 - O(\varepsilon)) \sum_{r \in E_2} \Phi(r) \cdot W(r) - O(\varepsilon\widetilde{\mathsf{OPT}})
\end{aligned}$$

where $W'(r)$ is the summation of top $k-1$ weights on the path r. In summation, the expected profit $\mathbb{P}(\sigma_r, \widetilde{\pi})$ is equal to

$$\sum_{r\in E} \widetilde{\Phi}(r)\cdot\widetilde{W}(r) \geq (1-O(\varepsilon))\mathbb{P}(\sigma,\pi) - O(\varepsilon)\mathsf{OPT}.$$

Now, we prove the second result. Similarly, we have

$$\sum_{r\in E_1} \widetilde{\Phi}(r)\cdot\widetilde{W}(r) \leq \sum_{r\in E_1} \Phi(r)\cdot W(r)$$

and

$$\begin{aligned}\sum_{r\in E_2} \widetilde{\Phi}(r)\cdot\widetilde{W}(r) &= \sum_{r\in E_2} \widetilde{\Phi}(\ell_r)\cdot \Pr[\widetilde{X}_{\ell_r}=\theta]\cdot(\theta+\widetilde{W}'(\ell_r))\\ &\leq \sum_{r\in E_2} \Phi(\ell_r)\cdot\Pr[X_{\ell_r}\geq\theta]\cdot\mathbb{E}[X_{\ell_r}\mid X_{\ell_r}\geq\theta] + O(\epsilon)\mathsf{OPT}\\ &\leq \sum_{r\in E_2}\Phi(r)\cdot W(r) + O(\epsilon)\mathsf{OPT}\end{aligned}$$

where the first inequality holds since $\Pr[\widetilde{X}=\theta]\leq\varepsilon$. Hence, the proof of the lemma is completed. □

4.4.3 Non-adaptive ProbeTop-k Problem

Now, we consider the non-adaptive version of the ProbeTop-k problem. Define

$$f(P) = \mathbb{E}\left[\max_{\{i_1,i_2,\ldots,i_k\}\subseteq P}\sum_{j\in[k]} X_{i_j}\right] \tag{4-17}$$

Our goal is to find a subset $P\subseteq[n]$ of cardinality m such that the expected reward $f(P)$ is maximized.

Theorem 4.8: There exists a PTAS for the non-adaptive ProbeTop-k problem.

Proof. Suppose $P^* = \{X_1, X_2, \ldots, X_m\}$ is the optimal solution with expected reward OPT. Then there is a pseudo-adaptive policy σ^* only using the items in the set $P^* = \{X_1, X_2, \ldots, X_m\}$. The pseudo-adaptive policy is adaptive policy which can be represented as a decision tree. In fact, for any root-to-leaf path, the set of items on the path is identical.

Lemma 4.2 still holds. Now, we can find a nearly optimal pseudo-block-adaptive policy only using m items in total. This can be done by modifying a little bit about the dynamic program (3-10):

$$\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = \max\Big(\mathcal{M}(i-1, \mathsf{C}, \mathsf{CA}), \mathcal{M}(i-1, \mathsf{C}', \mathsf{CA}-1)\Big), \tag{4-18}$$

where the capacity CA is the number of all items used in the tree, rather than a vector capacity for each root-to-leaf path. We can use the dynamic program to find a nearly optimal pseudo-block-adaptive policy σ (*i.e.,* a non-adaptive policy) with expect profit $(1 - O(\varepsilon))\mathsf{OPT}$. □

4.5 Committed Model

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. [①] 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 to probe the next item.

4.5.1 Committed ProbeTop-k Problem

In this section, we prove Theorem 4.3, *i.e.,* obtaining a PTAS for the committed ProbeTop-k. In the committed model, once we probe an item and observe its value realization, we are committed to making an irrevocable decision immediately whether to choose it or not. 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 to probe the next item.

Let σ^* be the optimal committed policy. Suppose σ^* is going to probe the item i and choose the item i if X_i realizes to a value $\theta \in S_i$, where S_i is the support of the random variable X_i. Then σ^* would choose the item i if X_i realizes to a larger value $s \geq \theta$. We call θ threshold for the item i. Thus the committed policy σ^* for the committed ProbeTop-k problem can be represented as a decision tree $\mathcal{T}_{\sigma^*}$. Every node v is labeled by an unique item a_v and a threshold $\theta(v)$, which means the policy chooses the item a_v if X_v realizes to a size $s \geq \theta(v)$, and otherwise rejects it.

① In [16,18], it is called the online decision model.

Now, we present a stochastic dynamic program for this problem. For each item i, we create a set of actions $\mathcal{B}_i = \{b_i^\theta\}_\theta$, where b_i^θ represents the action that we probe item i with the threshold θ. Since we assume discrete distribution (given explicitly as the input), there are at most a polynomial number of thresholds. Hence the set of action $\mathcal{A} = \cup_{i\in[n]}\mathcal{B}_i$ is bounded by a polynomial. The only requirement is that at most one action from $\mathcal{B}_i$ can be selected.

Let I_t be the number of items that have been chosen at the period time t. Then we set $\mathcal{V} = \{0, 1, \ldots, k\}$, $I_1 = 0$. Since we can probe at most m items, we set $T = m$. When we select an action b_i^θ to probe the item i and observe its value realization, say X_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + 1 & \text{if } X_i \geq \theta, I_t < k, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} X_i & \text{if } X_i \geq \theta, I_t < k, \\ 0 & \text{otherwise;} \end{cases} \tag{4-19}$$

for $I_t \in \mathcal{V}$ and $t = 1, 2, \ldots, T$, and $h(I_{T+1}) = 0$. Since k is a constant, Assumption 1.1 is immediately satisfied. However, in this case, we cannot directly use Theorem 4.1, due to the extra requirement that at most one action from each $\mathcal{B}_i$ can be selected. In this case, we need to slightly modify the dynamic program in Section A.2.4 to satisfy the requirement. To compute $\mathcal{M}(i, \mathsf{C}, \mathsf{CA})$, once we decide the position of the item i, we need to choose a threshold for the item. Since there are at most a polynomial number of thresholds, it can be computed at polynomial time. Hence, again, we can find a policy σ with profit at least

$$\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX} = (1 - O(\varepsilon))\,\mathsf{OPT},$$

where OPT denotes the expected profit of the optimal policy and $\mathsf{MAX} = \max_{I\in\mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(0, \mathcal{V}) = \mathsf{OPT}$.

4.5.2 Matroid Constraint

In this section, we prove Theorem 4.2, *i.e.,* obtaining a PTAS for the committed ProbeMax problem. On the committed model, we are committed to making an irrevocable decision immediately whether to pick it or not whenever we probe an item. If we add the item to the output set C, the profit is collected. Otherwise, no profit is collected and we are going to probe the next item. Based on Multi-budget optimization[31], we can obtain the PTAS for the committed ProbeMax problem with more general outer constraints.

Definition 4.3 (Multi-object/Multi-budget optimization)：Given l_1 linear function $f_1, f_2, \ldots, f_{\ell_1} : 2^U \to \mathbb{R}_+$, ℓ_2 linear function $g_1, g_2, \ldots, g_{\ell_2} : 2^U \to \mathbb{R}_+$ and a outer constraint $\mathfrak{A}$, is there a feasible solution C of $\mathfrak{A}$ satisfying $f_i(C) \geq D_i$ for all $i \in [\ell_1]$ and $g_i(I) \leq B_i$ for all $i \in [\ell_2]$?

We denote the above problem as Multi-$\mathfrak{A}$. A *multi-criteria* PTASs is an algorithm which produces a feasible solution C of $\mathfrak{A}$ such that $f_i(C) \geq (1-\varepsilon)D_i$ for all $i \in [\ell_1]$ and $g_i(I) \leq B_i$ for all $i \in [\ell_2]$ and the running time is polynomial in the size of the input for any fixed ε.

Theorem 4.9：Assume there is a multi-criteria PTASs for Multi-$\mathfrak{A}$. For any ε, there is a polynomial time algorithm that finds a committed policy (type NC) with the expected value at least $(1-\varepsilon)\mathsf{OPT}_{\mathsf{AC}}$ for the committed ProbeMax problem when the outer constraint is $\mathfrak{A}$.

Proof. Let σ^* be the optimal committed policy. Suppose σ^* is going to probe the next item i and choose i if X_i realizes to a value $\theta \in S$. Then σ^* would choose i if X_i realizes to a larger value $s \geq \theta$. We call θ threshold for the item i. Thus the committed policy σ^* for the committed ProbeMax problem can be represented a path R^*. Every node v is labeled by an unique item $i(v)$ and a threshold $\theta(v)$, which means the policy accepts the item $i(v)$ and stops if X_v realizes to a size $s \geq \theta(v)$, and otherwise rejects it and goes on. It can be equivalently viewed as a Bernoulli case where each item i has weight $w_i = \mathbb{E}[X_i \mid X_i \geq \theta]$ and is "active" with probability $p_i = \Pr[X_i \geq \theta]$.

WLOG we assume $\sum_{i \in R^*} p_i \leq 1/\varepsilon$ similar in Lemma 4.1 in[11] (otherwise we can have a truncation on the path R^* and the profit loss is at most $\varepsilon\mathsf{OPT}$). We call an item i a *heavy item* if $p_i \geq \varepsilon^2$. Otherwise we call it *light*. Then the number of heavy items in the path R^* is at most ε^{-3}. For an item i, we define the *signature* of i to be

$$\mathsf{Sg}(i) = \left(\left\lfloor p_i \cdot \frac{n}{\varepsilon^4} \right\rfloor \cdot \frac{\varepsilon^4}{n}, \left\lfloor p_i w_i \cdot \frac{n}{\varepsilon^4 \mathsf{OPT}} \right\rfloor \cdot \frac{\varepsilon^4 \mathsf{OPT}}{n} \right)$$

For a block M, define its signature to be $\mathsf{Sg}(M) = (\mathsf{Sg}^1(M), \mathsf{Sg}^2(M)) = \sum_{i \in M} \mathsf{Sg}(i)$.

Now, we greedily partition the path R^* into several segments, such that: each segment contains one heavy item or several light items such that $\sum_{i \in \mathsf{seg}} p_i \leq \varepsilon^2$. Then the number of segments is at most ε^{-3}. We regard each segment as a block to get a block policy R^b with the expected value at least $(1-\varepsilon)\mathsf{OPT}$ (we lose the profit when there are two items in the same block which are active simultaneously).

Algorithm 2 Approximation Algorithm of the committed ProbeMax problem

1: Enumerate all possible heavy items set H;
2: **for** each such H **do**
3: Enumerate all possible signatures Sg:
4: **for** each such Sg **do**
5: Try to find a set of blocks $\mathcal{B}$ such that $H \cup \mathcal{B} \in \mathfrak{A}$ (feasible) and
6: $\mathsf{Sg}^1(B_i) \le \mathsf{Sg}_i^1$, $\mathsf{Sg}^2(B_i) \ge \mathsf{Sg}_i^2$ for all $B_i \in \mathcal{B}$;
7: **end for**
8: **end for**
9: Pick the feasible $H \cup \mathcal{B}$ with the largest profit.

Denote $R^b = H^* \cup \mathcal{B}^*$, where H^* is the set of heavy items and $\mathcal{B}$ is a set of blocks which contain light items. Assume our algorithm 2 has guessed H^* correctly. By the assumption, we can obtain a set of blocks $\mathcal{B}$ of $\mathfrak{A}$ such that

$$\mathsf{Sg}^1(B_i) \le \mathsf{Sg}^1(B_i^*) \text{ and } \mathsf{Sg}^2(B_i) \ge (1-\varepsilon^2)\mathsf{Sg}^2(B_i^*) \text{ for all } B_i \in \mathcal{B}. \tag{4-20}$$

Fact: The expected value of $H^* \cup \mathcal{B}$ is at least $(1 - O(\varepsilon))\mathsf{OPT}$.

Let π_B^0 be the probability of all items in the block B are non-active. Then $\pi^0(B) = \prod_{i\in B}(1-p_i) \ge 1 - \sum_{i\in B} p_i \ge 1 - \varepsilon^2$. On the other hand

$$\pi^0(B) = \prod_{i\in B}(1-p_i) \le \left(1 - \frac{\sum_{i\in B} p_i}{|B|}\right)^{|B|} \le 1 - \sum_{i\in B} p_i + (\sum_{i\in B} p_i)^2.$$

Thus, by the definition of the signature and Inequality (4-20), we have

$$\begin{aligned}
\pi^0(B^*) - \pi^0(B) &= \prod_{i\in B^*}(1-p_i) - \prod_{i\in B}(1-p_i) \\
&\le \left(1 - \sum_{i\in B^*} p_i + \varepsilon^4\right) - \left(1 - \sum_{i\in B} p_i\right) \\
&\le -\mathsf{Sg}^1(B^*) + \left(\mathsf{Sg}^1(B) + \varepsilon^4\right) + \varepsilon^4 \\
&\le O(\varepsilon^4).
\end{aligned}$$

Let $\mathbb{P}(B)$ be the expected profit we can get from B. Suppose we insert the items in B one by one. For any item $i \in B$, with probability at least $\pi^0(B)$, all the previous items are not

”active”. Thus the expected profit we can get from i is at least $\pi^0(B) \cdot p_i w_i$. Thus, we have

$$\mathbb{P}(B) \geq \pi^0(B) \times \sum_{i \in B} p_i w_i \geq (1 - \varepsilon^2)\mathsf{Sg}^2(B).$$

We also have that

$$\mathbb{P}(B) \leq \sum_{i \in B} p_i w_i \leq \mathsf{Sg}^2(B) + \varepsilon^4 \mathsf{OPT}.$$

Let B_1^* be the root block of the optimal policy. Then, replacing B_1^* with B_1 in the path R^*, the expected profit loss is at most

$$\begin{aligned}
&\mathbb{P}(B_1^*) - \mathbb{P}(B_1) + \mathsf{OPT}(\pi^0_{B_1^*} - \pi^0_{B_1}) \\
\leq &\left(\mathsf{Sg}^2(B_1^*) + \varepsilon^4 \mathsf{OPT}\right) - (1 - \varepsilon^2)\mathsf{Sg}^2(B_1) + O(\varepsilon^4)\mathsf{OPT} \\
\leq &(1 + \varepsilon^2)\mathsf{Sg}^2(B_1) - (1 - \varepsilon^2)\mathsf{Sg}^2(B_1) + O(\varepsilon^4)\mathsf{OPT} \\
\leq &O(\varepsilon) \cdot \mathbb{P}(B_1) + O(\varepsilon^4)\mathsf{OPT}.
\end{aligned}$$

When we replace all the blocks in $\mathcal{B}^*$ by the corresponding ones in $\mathcal{B}$, the total profit loss is at most

$$\begin{aligned}
&\sum_B \Phi(B)[O(\varepsilon) \cdot \mathbb{P}(B) + O(\varepsilon^4)\mathsf{OPT}] \\
= &O(\varepsilon) \cdot \sum_B \Phi(B) \cdot \mathbb{P}(B) + O(\varepsilon^4)\mathsf{OPT} \sum_B \Phi(M) \\
\leq &O(\varepsilon) \cdot \mathsf{OPT} + O(\varepsilon^4)\mathsf{OPT} \cdot \varepsilon^{-3} \\
= &O(\varepsilon)\mathsf{OPT},
\end{aligned}$$

where $\Phi(B)$ is the probability of reaching the block B. □

Theorem 4.10 (Theorem 5 in[31]): For any $\varepsilon \geq 0$ and any constant number of $\ell_1 + \ell_3$, there is a polynomial-time randomized algorithm which is multi-criteria PTASs with high probability for matching or matroid intersection.

By combining Theorem 4.9 and Theorem 4.10, we get Theorem 4.2.

4.5.3 Commitment Gap

In this section, we prove Theorem 4.4, *i.e.,* bounding the commitment gap for the uniform matroid inner constraint and arbitrary prefix-closed outer constraint. Recall that for a particular stochastic probing instance $\mathcal{J}$, we have

$$\mathsf{CommitGap}(\mathcal{J}) = \frac{\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{AC}}(\mathcal{J})} \tag{4-21}$$

For the prophet inequality where we can only choose one item from a fixed order, a simple policy which uses $\theta = \mathsf{OPT}/2 = \mathbb{E}[\max_{i\in[n]} X_i]/2$ as a threshold and accepts the first item whose weight exceeds this threshold can achieve the 2-approximation ratio in Inequality (4-3).

For the m-uniform matroid inner constraint and arbitrary prefix-closed outer constraint, a best adaptive non-committed policy σ can be represented as a decision tree $\mathcal{T}(\sigma, \pi)$. We let $\theta = \frac{\mathsf{OPT}_{\mathsf{AN}}}{2m}$ be the threshold where $\mathsf{OPT}_{\mathsf{AN}}$ is the expected value of the optimal adaptive non-committed policy σ. Set $X^+ = \max\{X, 0\}$.

Lemma 4.3 : Then, we have

$$\mathsf{OPT}_{\mathsf{AN}} \le m \cdot \theta + \sum_{R\in\mathcal{R}} \Phi(R) \cdot \left(\sum_{v\in R} \mathbb{E}[(X_v - \theta)^+]\right). \tag{4-22}$$

Proof. By the definition, we have $\mathsf{OPT}_{\mathsf{AN}} = \sum_{R\in\mathcal{R}} \Phi(R) \cdot W(R)$, where $W(R)$ is the summation of the top-m weights on the path R. Then, we have

$$\begin{aligned}
\mathsf{OPT}_{\mathsf{AN}} - m \cdot \theta &= \sum_{R\in\mathcal{R}} \Phi(R) \cdot (W(R) - m \cdot \theta) \\
&\le \sum_{R\in\mathcal{R}} \Phi(R) \cdot \sum_{e\in R} (w_e - \theta)^+ \\
&= \sum_{v\in\mathcal{T}_\sigma} \Phi(v) \cdot \mathbb{E}[(X_v - \theta)^+] \\
&= \sum_{R\in\mathcal{R}} \Phi(R) \cdot \left(\sum_{v\in R} \mathbb{E}[(X_v - \theta)^+]\right).
\end{aligned}$$

Note that the leaf nodes are dummy nodes which can be ignored. □

Thus, there is at least a root-to-leaf path, say $R^* \in \mathcal{R}$, such that $\sum_{v\in R^*} \mathbb{E}[(X_v - \theta)^+] \ge \mathsf{OPT}_{\mathsf{AN}}/2$. Consider the following simple committed strategy:

Algorithm 3 A simple non-adaptive committed strategy ALG(R^*)

1: Follow the path R^* and accept items whose value exceeds the threshold $\theta = \frac{\mathsf{OPT}_{\mathsf{AN}}}{2m}$, until we accept m items.

Claim : The the expected value of ALG(R^*) shown in Algorithm 3 is at least $\mathsf{OPT}_{\mathsf{AN}}/2$.

Proof. Let p be the probability that there are at least m items whose value exceed θ on the path R^*. For a node v on the path R^*, let $I(v)$ be the indicator function which is equal to 1 if there are at most $m-1$ items whose values exceed θ before probing v and 0 otherwise. Thus for any node v, $\mathbb{E}[I(v)] \geq 1-p$. And we let $I(X_v \geq \theta)$ be the indicator function which is equal to 1 if $X_v \geq \theta$ and 0 otherwise. Then the expected profit we obtain from the path R^* is equal to

$$\begin{aligned}
\mathbb{E}[\mathrm{ALG}(R^*)] &= \sum_{v \in R^*} \mathbb{E}[X_v \cdot I(X_v \geq \theta) \cdot I(v)] \\
&\geq p \cdot m \cdot \theta + \sum_{v \in R^*} \mathbb{E}[(X_v - \theta) \cdot I(X_v \geq \theta) \cdot I(v)] \\
&= p \cdot \frac{\mathsf{OPT}_{\mathsf{AN}}}{2} + \sum_{v \in R^*} \mathbb{E}[(X_v - \theta)^+] \cdot \mathbb{E}[I(v)] \\
&\geq p \cdot \frac{\mathsf{OPT}_{\mathsf{AN}}}{2} + (1-p) \sum_{v \in R^*} \mathbb{E}[(X_v - \theta)^+] \\
&\geq \frac{\mathsf{OPT}_{\mathsf{AN}}}{2}.
\end{aligned}$$

□

Thus, there is a committed policy which can obtain at least $\frac{\mathsf{OPT}_{\mathsf{AN}}}{2}$ profit. Hence, Theorem 4.4 is proved. The fact that the policy is non-adaptive implies that:

Corollary 4.3 : The adaptivity gap of any stochastic probing problem where the outer constraint is prefix-closed and the inner constraint is an uniform matroid is at most 2.

4.5.4 Committed Pandora's Box Problem

In this section, we obtain a PTAS for the committed Pandora's Box problem. This can be proved by an analogous argument to Theorem 4.3 in Section 4.5.1. Similarly, for each box i, we create a set of actions $\mathcal{B}_i = \{b_i^\theta\}$, where b_i^θ represents 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. Then we set $\mathcal{A} = \cup_{i \in [n]} \mathcal{B}_i$, $\mathcal{V} = \{0, 1, \ldots, k\}$, $T = n$ and $I_1 = 0$. When

we select an action b_i^θ to open the box i and observe its value realization, say X_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + 1 & \text{if } X_i \geq \theta, I_t < k, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} X_i - c_i & \text{if } X_i \geq \theta, I_t < k, \\ -c_i & \text{otherwise;} \end{cases} \tag{4-23}$$

for $I_t \in \mathcal{V}$ and $t = 1, 2, \cdots, T$, 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 1.1 is immediately satisfied. Similar to the Committed ProbeTop-k Problem, we can choose at most one action from each $\mathcal{B}_i$. This can be handled in the same way. So again we can find a policy σ with profit at least $\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX} = (1 - O(\varepsilon))\,\mathsf{OPT}$, where OPT denotes the expected profit of the optimal policy and $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(0, \mathcal{A}) = \mathsf{OPT}$.

4.6 Stochastic Probing Problem

Consider the stochastic probing problem (Stoch-Prob) with general constraints. We use the same LP as introduced in Munagala *et al.*[14] and develop several policies based on the fractional optimal solution. Consider an instance of the stochastic probing problem with inner constraints $(U, \mathcal{I}_{in})$ and outer constraints $(U, \mathcal{I}_{out})$. Let S be the union of supports of all $\{X_i\}$. For each $s \in S$, let $p_{i,s} = \Pr[X_i = s]$. Suppose in an optimal adaptive non-committed policy (type AN) for Stoch-Prob, (x_i, y_i) is the probability that item i is chosen (to output) and probed respectively while $x_{i,s}$ is the probability that X_i is added into the final set and the value realization of X_i is s. Thus $x_i = \sum_{s \in S} x_{i,s}$. Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. We have the following LP relaxation:

$$\max \quad \sum_{i,s} s \cdot x_{i,s} \tag{4-24}$$

$$s.t. \quad x_{i,s} \leq y_i \cdot p_{i,s} \quad \forall i \in U, s \in S \tag{4-25}$$

$$x \in \mathcal{P}(\mathcal{I}_{in}) \tag{4-26}$$

$$y \in \mathcal{P}(\mathcal{I}_{out}) \tag{4-27}$$

Claim: The optimal value of LP (4-24) is an upper bound of $\mathsf{OPT}_{\mathsf{AN}}$.

4.6.1 Contention Resolution Schemes

We need to design a policy from a solution (x, y) for the LP relaxation. We use the rounding scheme called *contention resolution schemes (CR schemes)* to obtain constant approximations under several combinatorial constraints. Here is the formal definition.

Definition 4.4: (CR Scheme) A (b, c)-balanced CR scheme π for a downwards-closed set system $\mathcal{I}$ is a scheme such that for any $x \in \mathcal{P}(\mathcal{I})$, the scheme returns a set $\pi(I) \subseteq I = R(bx)$ with the following property, where $R(bx) \subseteq U$ is the random set obtained by independently choosing each item $i \in U$ with probability bx_i.

1. $\pi(I) \in \mathcal{I}$.
2. $\Pr[i \in \pi(I) \mid i \in I] \geq c$ for every item i.

A scheme is said to be monotone if $\Pr[i \in \pi(R_1)] \geq \Pr[i \in \pi(R_2)]$ for any $i \in R_1 \subseteq R_2$. A scheme is said to be ordered if there is a (possibly random) permutation σ on U so that for any $I = R(bx) \subseteq U$, the output of the scheme $\pi(I)$ is the maximal independent subset of I obtained by considering items in the order of σ.

Theorem 4.11 ([15,16]): There are monotone CR-schemes for the following downwards closed ($0 < b \leq 1$ is any value)

- $(b, (1 - e^{-b})/b)$-balanced CR-scheme for matroid.
- $(b, 1 - kb)$-balanced order CR-scheme for k-matroids.
- $(b, 1 - 2kb)$ CR-scheme for k-column sparse packing integer programs set system.

Gupta *et al.*[16] considers the Bernoulli version of Stoch-Prob with commitment satisfying the following properties:

(a). There is a (b, c_{out})-CR scheme π_{out} for $\mathcal{P}(\mathcal{I}_{\text{out}})$.

(b). There is a monotone (b, c_{in}) order CR scheme π_{in} for $\mathcal{P}(\mathcal{I}_{\text{in}})$.

Gupta *et al.*[16] gave an algorithm achieving a ratio of $\frac{1}{b(c_{\text{out}}+c_{in}-1)}$. Now we assume the same properties while we consider the Stoch-Prob: the value of each item can be an arbitrary discrete random variable. We extend the same result here.

Theorem 4.12: There is an algorithm (see ALG 4) that finds a non-adaptive committed policy (type NC) with the expected value at least $\frac{\mathsf{OPT}_{\mathsf{AN}}}{b(c_{\text{out}}+c_{in}-1)}$ for the Stoch-Prob satisfying Property (a) and (b).

By combining Theorems 4.11 and 4.12, we have

Corollary 4.4: There is a $\frac{1}{4(k+\ell)}$-approximation algorithm when the inner and outer constraints are intersections of k and ℓ matroids respectively.

Now, we present a formal statement of Algorithm 4.

Algorithm 4 Rounding Algorithm for the Stochastic Probing Problem

1: Solve the LP relaxation and obtain the optimal LP solution (x, y);
2: Pick $I \subseteq 2^U$ by choosing each $i \in U$ independently with probability by_i;
3: Let $P = \pi_{\text{out}}(I) \in \mathcal{I}_{\text{out}}$;
4: Order items in P according to the permutation given by the ordered CR scheme π_{in};
5: $C \leftarrow \emptyset$;
6: **for** $i = 1, 2, \ldots, |P|$ **do**
7: **if** $C \cup i \in \mathcal{I}_{\text{in}}$ **then**
8: Probe the item i;
9: If $X_i = s$, let $C \leftarrow C \cup \{i\}$ with probability $x_{i,s}/(y_i \cdot p_{i,s})$;

Now, we analyze the expected value of the above algorithm and prove Theorem 4.12.

Proof. Notice that the probability we choose i conditioning on its probing is

$$p_i = \sum_{s \in S} \Pr[X_i = s] \cdot \frac{x_{i,s}}{y_i \cdot p_{i,s}} = \sum_{s \in S} \frac{x_{i,s}}{y_i} \leq 1.$$

Let $J \subseteq U$ be the random set by choosing each $i \in U$ independently with probability p_i. From the result of Lemma 3.5 in[16], we see for each $i \in U$,

$$\Pr_{I, \pi_{out}, J, \pi_{in}} [i \in \pi_{in}(\pi_{out}(I) \cap J)] \;\geq\; b \cdot (c_{out} + c_{in} - 1) \cdot p_i y_i.$$

Now, the expected value of the chosen set C is

$$\begin{aligned} \mathbb{E}\left[\sum_{i \in C} X_i\right] &= \sum_{i \in U} \Pr[i \in \pi_{in}(P \cap J)] \cdot \mathbb{E}[X_i \mid i \in C] \\ &\geq b \cdot (c_{out} + c_{in} - 1) \cdot \sum_{i \in U} p_i y_i \cdot \mathbb{E}[X_i \mid i \in C]. \end{aligned}$$

Notice that

$$\begin{aligned}\mathbb{E}[X_i \mid i \in C] = \mathbb{E}[X_i \mid i \in J] &= \sum_{s \in S} s \cdot \frac{\Pr[X_i = s \wedge i \in J]}{\Pr[i \in J]} \\ &= \sum_{s \in S} s \cdot \frac{\Pr[X_i = s] \cdot \Pr[i \in J \mid X_i = s]}{\Pr[i \in J]} \\ &= \sum_{s \in S} \frac{s \cdot p_{i,s} \cdot x_{i,s}/(y_i \cdot p_{i,s})}{p_i} = \frac{1}{p_i y_i} \sum_{s \in S} s \cdot x_{i,s}. \end{aligned}$$ □

Indeed, we can see that

$$\mathbb{E}\left[\sum_{i \in C} X_i\right] \geq b \cdot (c_{out} + c_{in} - 1) \sum_{i \in U, s \in S} s \cdot x_{i,s} \geq b \cdot (c_{out} + c_{in} - 1)\mathsf{OPT}_{\mathsf{AN}}.$$

4.6.2 The Iterative Randomized Rounding Approach

Next, we give a description of another rounding approach called Iterative Randomized Rounding approach which derives from[35]. We only focus on matroid intersections constraints, i.e. on the special case in which $\mathcal{I}_{out}$ is an intersection of k^{out} matroids $\mathcal{M}_1^{out}, \ldots, \mathcal{M}_{k^{out}}^{out}$, and $\mathcal{I}_{in}$ is an intersection of k^{in} matroids $\mathcal{M}_1^{in}, \ldots, \mathcal{M}_{k^{in}}^{in}$. Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid. For an item $i \in E$, we denote the the matroid $\mathcal{M}$ with i contracted by M/i, i.e. $\mathcal{M}/i = (E - e, \{S \subset E - e \mid S + e \in \mathcal{I}\})$. Then $\mathcal{I}_{in}/i$ is an intersection of matroids $\mathcal{M}_1^{in}/i, \ldots, \mathcal{M}_{k^{in}}^{in}/i$ and $\mathcal{I}_{out}/i$ is an intersection of matroids $\mathcal{M}_1^{out}/i, \ldots, \mathcal{M}_{k^{out}}^{out}/i$.

Initially, $x = x^0, y = y^0, C = \emptyset$. At each step, we select single item i to probe, then permanently set $y_i = 0, x_i = 0$. Then update the outer constraints by replacing $\mathcal{I}_{out}$ with $\mathcal{I}_{out}/i$. If item i is added into C, we update the inner constraints by replacing $\mathcal{I}_{in}$ with $\mathcal{I}_{in}/i$. Finally, we modify our fractional solution (x, y) such that it is feasible for the updated constraints. The algorithm terminates when $(x, y) = 0$.

Let x^t, y^t, and C^t be the current value of x, y, and C at the beginning of step $t + 1$. The the expected value of our algorithm is guaranteed by following lemma based on Doob's optional stopping theorem for martingales.

Lemma 4.4 (Lemma 8 in[35]): Suppose the algorithm runs for τ steps and that $g(x^0) \geq \beta \cdot \mathbb{E}[\mathsf{OPT}], x^\tau = 0$. Let $\mathcal{F}_i$ represent all information available after the ith iteration. Suppose that in each step, $\mathbb{E}[f(C^{t+1}) - f(C^t) \mid \mathcal{F}_t] \geq \alpha \cdot \mathbb{E}[g(x^t) - g(x^{t+1}) \mid \mathcal{F}_t]$. Then, the final output C^τ satisfies $\mathbb{E}[f(C^\tau)] \geq \alpha\beta \cdot \mathbb{E}[\mathsf{OPT}]$.

Algorithm 5 The Iterative Randomized Rounding for Stochastic Probing

Solve the LP relaxation and obtain the optimal LP solution (x^0, y^0);
Set $C = \emptyset, x = x^0, y = y^0$.
repeat
    Randomly select an item i with probability $y_i / \sum_j y_j$ and then probe it;
    If $X_i = s$, let $C \leftarrow C \cup \{i\}$ with probability $x_{i,s}/(y_i \cdot p_{i,s})$;
    Update y such that $y \in \mathcal{P}(\mathcal{I}_{out}/i)$;
    Update x such that the set of Inequalities (4-25) holds;
    If i is added into C, **then**
        Update x such that $x \in \mathcal{P}(\mathcal{I}_{in}/i)$;
until $x = 0$;
Output C.

In this setting, $g(x) = \sum_{i \in U, s \in S} s \cdot x_{i,s}$ and $f(C) = \sum_{i \in C} w_i$. The x^0 is the optimal LP solution, then $z^0 = g(x^0) \geq \mathsf{OPT}_{\mathsf{AN}}$. Thus $\beta = 1$. We discuss arbitrary step $t+1$, then denote (x^t, y^t) by (x, y) and (x^{t+1}, y^{t+1}) by (x', y'). We denote $\Lambda = \sum_{i \in U} y_i$, then

$$\mathbb{E}[f(C^{t+1} - f(C^t)) \mid \mathcal{F}_t] = \sum_{i \in U, s \in S} \frac{y_i}{\Lambda} \times p_{i,s} \times \frac{x_{i,s}}{y_i \cdot p_{i,s}} \times s = \frac{1}{\Lambda} \sum_{i \in U, s \in S} s \cdot x_{i,s} = \frac{g(x^t)}{\Lambda}. \quad (4\text{-}28)$$

We now describe how to update the current solution (x, y) to ensure feasibility in each of the update matroid constraints. We use the same technique as used in[35]. Let i be the item that we probed and $\mathcal{M}_j^{out}$ be some outer matroid. Currently we have $y \in \mathcal{P}(\mathcal{M}_j^{out})$ and we want to get a solution y' such that $y' \in \mathcal{P}(\mathcal{M}_j^{out}/i)$. We represent y as a convex combination $y = \sum_{\ell=1}^{m} \lambda_i \mathbf{1}_{\mathbf{B}_\ell}$, where $B_1, B_2, \ldots, B_m$ is the support of y. We random pick one set B_a which contains item i with probability λ_a / y_i (note that for any item i, $\sum_{a: i \in B_a} \lambda_a = y_i$). Then for any set $B_b : i \notin B_b$, there is item $\phi_{a,b}$ such that $B_b + e - \phi_{a,b} \in \mathcal{M}_j^{out}$. By replacing set B_b with $B_b + e - \phi_{a,b}$ and setting $y_i = 0$, we obtain $y' \in \mathcal{P}(\mathcal{M}_j^{out}/i)$.

Lemma 4.5 (Lemma 9 in[35]): Let y and y' be the current fractional solution before and after one update for a given outer matroid $\mathcal{M}_j^{out}$. Then, for each $i \in U$, we have $\mathbb{E}[(y_i - y_i')] \leq \frac{1}{\Lambda}(1 - y_i) y_i$.

In order to satisfy the set of Inequalities (4-25), we set $x'_{i,s} = \min\{x_{i,s}, y'_i \cdot p_{i,s}\}$ for any $i \in U$ and $s \in S$.

Lemma 4.6：　Then, for each $i \in U, s \in S$, we have

$$\mathbb{E}[\delta_{i,s}] = \mathbb{E}[(x_{i,s} - x'_{i,s})] \leq \frac{1}{\Lambda}(1 - y_i)\, x_{i,s}. \tag{4-29}$$

Proof. Since $x_{i,s} \leq p_{i,s} \cdot y_i$ and $y_i \geq y'_i$, thus

$$\begin{aligned}
\frac{\mathbb{E}x'_{i,s}}{x_{i,s}} &= \frac{\mathbb{E}[\min\{x_{i,s}, p_{i,s} \cdot y'_i\}]}{x_{i,s}} \\
&= \mathbb{E}\left[\min\left\{1, \frac{p_{i,s} \cdot y'_i}{x_{i,s}}\right\}\right] \\
&\geq \mathbb{E}\left[\min\left\{1, \frac{p_{i,s} \cdot y'_i}{p_{i,s} \cdot y_i}\right\}\right] \\
&= \frac{\mathbb{E}y'_i}{y_i} \\
&\geq 1 - \frac{1}{\Lambda}(1 - y_i),
\end{aligned}$$

the last equality holds by Lemma 4.5. Then, Equation (4-29) holds. □

Next, if the item i is added to C, we similarly update x for each inner matroid $\mathcal{M}_j^{in}$ and get x' such that $x' \in \mathcal{P}(\mathcal{M}_j^{in}/i)$.

Lemma 4.7 (Lemma 10 in[35])：　Let x and x' be the current fractional solution before and after one update for a given inner matroid $\mathcal{M}_i^{in}$. Then, for each $i \in U$, we have $\mathbb{E}[(x_i - x'_i)] \leq \frac{1}{\Lambda}(1 - x_i)x_i$

For any $i \in U, s \in S$, we set $x'_{i,s} = x_{i,s} \cdot \frac{x'_i}{x_i}$. Then $\sum_{s \in S} x'_{i,s} = \sum_{s \in S} x_{i,s} \cdot \frac{x'_i}{x_i} = x'_i$.

Lemma 4.8：　Then, for each $i \in U, s \in S$, we have

$$\mathbb{E}[\delta_{i,s}^{in}] = \mathbb{E}[(x_{i,s} - x'_{i,s})] \leq \frac{1}{\Lambda}(1 - x_i)x_{i,s}. \tag{4-30}$$

Proof. Because

$$x_{i,s} - \mathbb{E}[x'_{i,s}] = x_{i,s}\left(1 - \frac{\mathbb{E}x'_i}{x_i}\right) \leq \frac{1}{\Lambda}(1 - x_i)x_{i,s}.$$

We perform the matroid updates sequentially for each of the k^{out} and k^{in} matroids. Then we have

$$\begin{aligned}
&\mathbb{E}[\sum_{s\in S} s\cdot(x_{i,s}-x'_{i,s})]\\
&\le \sum_{s} s\cdot x_{i,s}\cdot \Pr[i \text{ is probed}] + k^{out}\sum_{s\in S} s\cdot\mathbb{E}[\delta^{out}_{i,s}] + k^{in}\sum_{s\in S} s\cdot\mathbb{E}[\delta^{in}_{i,s}]\\
&\le \frac{1}{\Lambda}\sum_{s\in S} s\cdot x_{i,s}\cdot y_i + \frac{k^{out}}{\Lambda}\sum_{s\in S} s\cdot(1-y_i)x_{i,s} + \frac{k^{in}}{\Lambda}\sum_{s\in S} s\cdot(1-x_i)x_{i,s}\\
&= \frac{k^{out}+k^{in}}{\Lambda}\sum_{s\in S} s\cdot x_{i,s} - \frac{k^{out}-1}{\Lambda}\sum_{s\in S} s\cdot x_{i,s}\cdot y_i - \frac{k^{in}}{\Lambda}\sum_{s\in S} s\cdot x_{i,s}\cdot x_i\\
&\le \frac{k^{out}+k^{in}}{\Lambda}\sum_{s\in S} s\cdot x_{i,s}.
\end{aligned}$$

Then

$$\mathbb{E}[g(x^t)-g(x^{t+1})] = \mathbb{E}[\sum_{i\in U, s\in S} s\cdot(x_{i,s}-x'_{i,s})] \le \frac{k^{out}+k^{in}}{\Lambda}\sum_{i\in U, s\in S} s\cdot x_{i,s}. \tag{4-31}$$

By combining Equation (4-28) and (4-31), we have

$$\mathbb{E}[f(C^{t+1})-f(C^t)\mid\mathcal{F}_t] \ge \frac{1}{k^{out}+k^{in}}\cdot\mathbb{E}[g(x^t)-g(x^{t+1})\mid\mathcal{F}_t].$$

By Lemma 4.4, we have $\mathbb{E}[f(C^\tau)] \ge \frac{\mathsf{OPT}_{\mathsf{AN}}}{k^{out}+k^{in}}$. This prove Theorem 4.6.

4.7 Summary

In this chapter, we propose an approximation algorithm for the general stochastic probing problem. The main technical ingredient for obtaining the PTAS for the ProbeMax problem is the framework of stochastic dynamic programs which is formulated in Chapter 3. We believe the framework can be used in designing better approximation algorithms for other stochastic optimization problems.

For the ProbeMax Problem with a general matroid outer constraint, the best known approximation ratio is $1-1/e$, based on submodular maximization[17] (which is in fact a non-adaptive policy). To design an adaptive policy and improve the approximation ratio is still a open problem. We note that in the corresponding committed model, we have obtained a PTAS (due to the special structure of the policy).

第 5 章　Stochastic Knapsack and Variants

The knapsack problem is one of the most well-studied combinatorial optimization problem. In this problem, we are given as input a set of n items. Each item $i \in [n]$ has a size s_i and a profit p_i. The goal is to find a maximum-profit subset of these items whose total size is at most a given capacity $\mathbb{C}$. The knapsack problem is NP-hard and has a fully polynomial time approximation scheme (FPTAS)[43].

5.1 Model

In many applications, the size and/or the profit of an item many not be fixed values and only their probability distributions are known to us in advance. The real size and profit of an item are revealed to us so soon as it inserted into the knapsack. For the stochastic variant of this problem, we have following definition.

Definition 5.1 : **Stochastic Knapsack** (SKP) An instance $\mathcal{J}$ of the stochastic knapsack problem is defined with tuple $\mathcal{J} = (\pi, \mathbb{C})$, where $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$. π_i is the joint distribution of size and profit for item i. The distributions for different items are mutually independent. In general, for each item i, its size s_i is a random variable and its profit p_i is deterministic. $\mathbb{C}$ is the capacity of the knapsack. Once we decide to insert an item i into the knapsack, we observe its real size s_i which follows the distribution π_i. We can adaptively insert the items to the knapsack, as long as the capacity constraint is not violated. The goal is to design a policy that maximizes the expected total profit of all items that are successfully inserted into the knapsack.

Gupta *et al.*[20] introduced a variant of stochastic knapsack which is referred as *Stochastic Knapsack with Correlated Rewards and Cancelations* (SK-CC) where (1) each item can have a random profit which can be correlated with its size and (2) we can cancel a job during its execution in the policy. For the committed model, we refer as *Stochastic Knapsack with Commitment* (SK-Commit [①]), where once we select an item and observe its size realization, we can make an irrevocable decision whether to insert it or not, *i.e.,* we must either insert it into the knapsack or discard it forever. Next, Levin *et al.*[23]

① Li *et al.*[11] called the *Bayesian online selection problem subject to a knapsack constraint*.

introduced another variant of stochastic knapsack where all the profits are lost if we violate the capacity constraint.

Definition 5.2：　**Stochastic Blackjack Knapsack** (SBK) ① In this problem, we are given a instance $\mathcal{J} = (\pi, \mathbb{C})$. Our goal is to design an adaptive policy such that the expected total profits of all items inserted is maximized. The different from SKP is that we gain zero if the total size of all items inserted is larger than the capacity $\mathbb{C}$.

For a particular stochastic knapsack instance $\mathcal{J}$, we use $\mathsf{OPT}_{\mathsf{SKP}}(\mathcal{J})$ to denote the expected profit of an optimal policy for stochastic knapsack. Similarly, we denote $\mathsf{OPT}_{\mathsf{SKC}}(\mathcal{J})$ for stochastic knapsack with commitment and $\mathsf{OPT}_{\mathsf{SBK}}(\mathcal{J})$ for stochastic blackjack knapsack. Since a policy for SBK is also a policy for SKP and a policy for SKP is also a policy for SK-Commit, we have following proposition.

Proposition 5.1：　For any stochastic knapsack instance $\mathcal{J}$, we have

$$\mathsf{OPT}_{\mathsf{SKC}}(\mathcal{J}) \geq \mathsf{OPT}_{\mathsf{SKP}}(\mathcal{J}) \geq \mathsf{OPT}_{\mathsf{SBK}}(\mathcal{J}). \tag{5-1}$$

The stochastic knapsack problem with deterministic sizes and random profits has been studied in[45]. In the model, a threshold profit is specified, and the goal is to design an adaptive policy to insert a set of items with maximum probability of achieving that threshold profit. We denote *Stochastic Target Problem* (STP) where all items are the same size. ②

Definition 5.3：　**Stochastic Target** (STP) We are given a predetermined target $\mathbb{T}$ and a set of n items. Each item $i \in [n]$ has a profit 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 profit 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 the inserted items.

① Chen *et al.*[44] call it "zero return if broken".

② [45] 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.

Table 5.1　A summary of approximation ratios for the stochastic knapsack and variants

Problem	single-criterion approx	bi-criterion approx ($\mathbb{C}$ relaxed to $(1+\varepsilon)\mathbb{C}$ or $\mathbb{T}$ relaxed to $(1-\varepsilon)\mathbb{T}$)
SKP	$\frac{1}{3}-\varepsilon$ [5], $\frac{3}{8}-\varepsilon$ [6], $\frac{1}{2}-\varepsilon$ [22]	$1-\varepsilon$ [6,11] [Theorem 5.1]
SK-CC	$\frac{1}{16}$ [20], $\frac{1}{2}-\varepsilon$ [21]	$1-\varepsilon$ [11]
SK-Commit	$\frac{1}{2}-\varepsilon$ [22]	$1-\varepsilon$ [11]
SBK	$(\sqrt{2}-1)^2/2 \approx 1/11.66$ [23] $1/8-\varepsilon$ [Theorem 5.4]	$1-\varepsilon$ [Theorem 5.3]
STP	-	OPT $-\varepsilon$ [Theorem 5.5]

5.2 Our Result

Stochastic Knapsack: For stochastic knapsack (SKP), Dean *et al.* [5] gave an algorithm with an approximation ratio of $\frac{1}{3}-\varepsilon$. 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 $\varepsilon \geq 0$. The best known single-criterion approximation factor is 2 [11,21,22]. It is still an open problem to design a PTAS for stochastic knapsack.

For the stochastic knapsack with correlated rewards and cancelations (SK-CC), Gupta *et al.* [20] gave a randomized algorithm achieving a ratio of $\frac{1}{16}$ and Ma [21] improved that ratio to $(\frac{1}{2}-\varepsilon)$ for any given $\varepsilon > 0$. Li *et al.* [11] gave an algorithm with ratio $(1-\varepsilon)$ by using ε extra budget for any given $\varepsilon > 0$.

In this dissertation, we use the framework formulated in Chapter 3 to simplify the proof for stochastic knapsack in [11]. For stochastic knapsack, let the value I_t be the total sizes of the items in the knapsack at time period t. Using our framework, we have following system dynamics for the stochastic knapsack:

$$I_{t+1} = f(I_t, i) = I_t + X_i, \quad g(I_t, i) = \begin{cases} p_i & \text{if } I_t + X_i \leq \mathbb{C}, \\ 0 & \text{otherwise;} \end{cases} \tag{5-2}$$

for $t = 1, 2, \cdots, T$ and $h(I_{T+1}) = 0$. See Section 5.4 for more details.

Theorem 5.1: For any $\varepsilon > 0$, there is a polynomial algorithm that finds a $(1-\varepsilon)$-approximate adaptive policy for stochastic knapsack problem when the capacity is relaxed to $(1+\varepsilon)\mathbb{C}$.

The above theorem is also holds for SK-CC and SK-Commit. If the sizes of items are bounded, we have following theorem.

Theorem 5.2: For any $\varepsilon \in (0,1)$, there is a polynomial algorithm that finds a $(1-4\varepsilon)$-approximate adaptive policy for stochastic knapsack problem when the sizes are bounded by ε, *i.e.,* for any $i \in [n]$, $\Pr[s_i \leq \varepsilon] = 1$.

Stochastic Blackjack Knapsack: This extra restriction that all profits are lost if we violate the capacity constraint might induce us to take more conservative policies. Levin *et al.* [23] 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.* [44] assumed each size X_i follows a known exponential distribution and gave an optimal policy for $n = 2$ based on dynamic programming. In this dissertation, we consider the general case where the distribution of each X_i is allowed to follow an arbitrary discrete distribution.

Theorem 5.3: For any fixed constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for the stochastic blackjack knapsack that finds a policy with the expected profit at least $(1 - O(\varepsilon))\mathsf{OPT}$, when the capacity is relaxed to $(1+\varepsilon)\mathbb{C}$, where OPT is the expected profit of the optimal adaptive policy.

For the case without relaxing the capacity, we can improve the result of 11.66 in [23] below.

Theorem 5.4: For any $\varepsilon \geq 0$, there is a polynomial time algorithm that finds a $(\frac{1}{8}-\varepsilon)$-approximate adaptive policy for SBK.

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

$$I_{t+1} = f(I_t, i) = (I_{t,1} + s_i, I_{t,2} + p_i), \quad h(I_{T+1}) = \begin{cases} I_{T+1,2} & \text{if } I_{T+1,1} \leq \mathbb{C}, \\ 0 & \text{otherwise;} \end{cases} \tag{5-3}$$

and $g(I_t, i) = 0$ for $t = 1, 2, \cdots, T$. Then Assumption 1.1 (2,3) is immediately satisfied. But Assumption 1.1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. Hence, we need discretization. Unlike the stochastic knapsack, we need to discretize the sizes and profits at the same time. See Section 5.5 for more details.

Stochastic Target Problem:

For the stochastic target problem, İlhan *et al.*[45] provided some heuristic based on dynamic programming for the special case where the random profit of each item follows a known normal distribution. Recall that the target is the goal to be achieved. If allowed to relax the target to $(1-\varepsilon)\mathbb{T}$, we provide an additive PTAS for it.

Theorem 5.5: 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.

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:

$$I_{t+1} = f(I_t, i) = I_t + X_i, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} 1 & \text{if } I_{T+1} \geq \mathbb{T}, \\ 0 & \text{otherwise;} \end{cases} \tag{5-4}$$

for $t = 1, 2, \cdots, T$. Then Assumption 1.1 (2,3) is immediately satisfied. But Assumption 1.1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. Hence, we need discretize the value space and reduce its size to a constant. See Section 5.6 for more details.

5.3 Policies and Decision Trees

The process of applying a policy σ on an instance $(\pi, \mathbb{C})$ can be represented as a decision tree $\mathcal{T}(\sigma, \pi, \mathbb{C})$. Each node v in $\mathcal{T}_\sigma$ is labeled by a unique item $i(v) \in U$ with a size s_v following a known (discrete) distribution π_v. It has several children, and let $e = (v, u)$ (u is the s-child) be the s-th edge emanating from $s \in S$ where S is the support of the size distribution. Thus e has probability $\pi_e := \Pr[s_v = s]$ and weight $w_e := s$. The edge e has profit $p_e := p_v$ if the item is successfully inserted into the knapsack and $p_e := 0$ otherwise. In order to clearly illustrate the tree structure, we add a dummy node at the end of each root-to-leaf path.

For a node v, we say the path from the root to it in $\mathcal{T}_\sigma$ as the *realization path* of v, and denote it by $\mathcal{R}(v)$. We denote the total weight $W(v) = W(\mathcal{R}(v)) = \sum_{e \in \mathcal{R}(v)} w_e$, total profit

$P(v) = P(\mathcal{R}(v)) = \sum_{e\in\mathcal{R}(v)} p_e$, and the probability of reaching v as $\Phi(v) = \Phi(R(v)) = \prod_{e\in R(v)} \pi_e$.

Let E be the set of all root-to-leaf paths in T_σ. We define $E_1 = \{r \in E : W(r) \le \mathbb{C}\}$. We use $\mathbb{P}(\sigma, \pi, \mathbb{C})$ to denote the expected profit that the policy σ can obtain with the given distribution π. For SKP, we have

$$\mathbb{P}(\sigma, \pi, \mathbb{C}) = \sum_{r\in E} \Phi(r) \cdot P(r). \tag{5-5}$$

For SBK, we have

$$\mathbb{P}(\sigma, \pi, \mathbb{C}) = \sum_{r\in E_1} \Phi(r) \cdot P(r). \tag{5-6}$$

5.4 Stochastic Knapsack

In this section, we use the framework formulated in Chapter 3 to simplify the proof for stochastic knapsack in[11]. Also, we show the same result can be obtained for stochastic knapsack with commitment.

For stochastic knapsack, define the item set $\mathcal{A} = \{1, 2, \ldots, n\}$. Let the value I_t be the total sizes of the items in the knapsack at time period t. Then we set $T = n$ and $I_1 = 0$. When we insert an item i into the knapsack and observe its size realization, say s_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, i) = I_t + s_i, \quad g(I_t, i) = \begin{cases} p_i & \text{if } I_t + s_i \le \mathbb{C}, \\ 0 & \text{otherwise;} \end{cases} \tag{5-7}$$

for $t = 1, 2, \cdots, T$ and $h(I_{T+1}) = 0$. Then Assumption 1.1 (2,3) is immediately satisfied. But Assumption 1.1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. Hence, we need to discretize the value space and reduce its size to a constant.

5.4.1 Discretization

We use the same discretization technique as in[11] for the Expected Utility Maximization. The main idea is as follows. Without loss of generality, we set $\mathbb{C} = 1$. Now, we discuss how to discretize the size distributions for items, using parameter ε. For an item b, we say X_b is a big realization if $X_b > \varepsilon^4$ and small otherwise. For a big realization of

X_b, we simple define the discretized version of $\widetilde{X}_b$ as $\lfloor \frac{X_b}{\varepsilon^5} \rfloor \varepsilon^5$. For a small realization of X_b, we define $\widetilde{X}_b = 0$ if $X_b < d$ and $\widetilde{X}_b = \varepsilon^4$ if $d \leq X_b \leq \varepsilon^4$, where d is a threshold such that $\Pr[X_b \geq d \mid X_b \leq \varepsilon^4]\varepsilon^4 = \mathbb{E}[X_b \mid X_b \leq \varepsilon^4]$. For more details, please refer to[11].

Consider a given adaptive policy σ. For each node $v \in \mathcal{T}_\sigma$, let $W(v)$ and $\widetilde{W}(v)$ be the sum of sizes on the path $\mathcal{R}(v)$ before and after discretization respectively. Recall that $\Phi(v)$ is the probability of reaching v. In the proof of Lemma 4.2 of[11], it shows that for any given set F of nodes in $\mathcal{T}_\sigma$ which contains at most one node from each root-leaf path, our discretization has the below property:

$$\sum_{v \in F : |W(v) - \widetilde{W}(v)| \geq 2\epsilon} \Phi(v) = O(\varepsilon). \tag{5-8}$$

We use the notion of canonical policies introduced in[6] (see Section 4.4.1).

Lemma 5.1 (Lemma 4.2 in[11]): Let π be the distribution of size for items and $\widetilde{\pi}$ be the discretized version π. Then, the following statement hold:

1. For any policy σ, there exists a canonical policy $\widetilde{\sigma}$ such that

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1 + 4\varepsilon)\mathbb{C}) = (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi, \mathbb{C});$$

2. For any canonical policy $\widetilde{\sigma}$,

$$\mathbb{P}(\widetilde{\sigma}, \pi, (1 + 4\varepsilon)\mathbb{C}) = (1 - O(\varepsilon))\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, \mathbb{C}).$$

Proof. (The proof of Theorem 5.1) Suppose σ^* is the optimal policy with expected value $\mathsf{OPT} = \mathbb{P}(\sigma^*, \pi, 1 + 5\varepsilon)$. Given an instance π, we compute the discretized distribution $\widetilde{\pi}$. By Lemma 5.1 (1), there exists a policy $\widetilde{\sigma}^*$ such that

$$\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1 + 4\varepsilon)) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma^*, \pi,) = (1 - O(\varepsilon))\mathsf{OPT}.$$

Now, we present a stochastic dynamic program for the instance $(\widetilde{\pi}, (1 + 4\varepsilon))$. Define the value set $\mathcal{V} = \{0, \varepsilon^5, 2\varepsilon^5, \ldots, 1 + 5\varepsilon\}$, the item set $\mathcal{A} = \{1, 2, \ldots, n\}$, $T = n$ and $I_1 = 0$. When we insert an item i into the knapsack and observe its profit realization, say s_i, we

define the system dynamic functions to be

$$I_{t+1} = f(I_t, i) = \min\{I_t + s_i, 1 + 5\varepsilon\}, \quad g(I_t, i) = \begin{cases} p_i & \text{if } I_t + s_i \le 1 + 4\varepsilon, \\ 0 & \text{otherwise;} \end{cases} \tag{5-9}$$

for $t = 1, 2, \cdots, T$ and $h(I_{T+1}) = 0$. Then Assumption 1.1 is immediately satisfied. By Theorem 1.6, we can find a policy σ with value $\mathbb{P}(\sigma, \widetilde{\pi}, 1 + 4\varepsilon)$ at least

$$\mathsf{OPT}_d - O(\varepsilon) \cdot \mathsf{MAX} \ge (1 - O(\varepsilon))\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1 + 4\varepsilon)) \ge (1 - O(\varepsilon))\mathsf{OPT},$$

where OPT_d denotes the expected value of the optimal policy for instance $(\widetilde{\pi}, 1 + 4\varepsilon)$ and $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(0, \mathcal{A}) = \mathsf{OPT}_d$. By Lemma 5.1 (2), we have

$$\mathbb{P}(\sigma, \pi, (1 + 4\varepsilon)(1 + 4\varepsilon)) \ge \mathbb{P}(\sigma, \widetilde{\pi}, (1 + 4\varepsilon)) \ge (1 - O(\varepsilon))\mathsf{OPT},$$

which completes the proof. □

Since Lemma 5.1 holds for correlated size and profit, Theorem 5.1 can be easily extended to SK-CC (see[11]).

5.4.2 Stochastic Knapsack with Commitment

Li *et al.*[11] gave a reduction from SK-Commit to SKP. The main idea is following. For the stochastic knapsack with commitment, we create a set of action $\mathcal{B}_i = \{b_i^\theta\}$ for each item, where b_i^θ represents the action that we choose the item i with threshold θ, *i.e.,* we insert the item into the knapsack is its size is less than or equal to θ. Then we set set $\mathcal{A} = \cup_{i \in [n]} \mathcal{B}_i$. Let the value I_t be the total sizes of the items in the knapsack at time period t. Then we set $T = n$ and $I_1 = 0$. When we insert an item i into the knapsack and observe its size realization, say X_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, b_i^\theta) = \begin{cases} I_t + X_i & \text{if } X_i \le \theta, \\ I_t & \text{otherwise;} \end{cases} \quad g(I_t, b_i^\theta) = \begin{cases} p_i & \text{if } (X_i \le \theta) \wedge (I_t + X_i \le \mathbb{C}), \\ 0 & \text{otherwise;} \end{cases} \tag{5-10}$$

for $t = 1, 2, \cdots, T$ and $h(I_{T+1}) = 0$.

Theorem 5.6： For any fixed constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for the stochastic knapsack with commitment that finds a policy with the

expected profit at least $(1 - O(\varepsilon))\mathsf{OPT}$, when the capacity is relaxed to $(1 + \varepsilon)\mathbb{C}$, where OPT is the expected profit of the optimal adaptive policy.

The action b_i^θ is equivalent to an item with the size s_i^θ and profit p_i^θ, jointly distributed as follows:

$$(s_i^\theta, p_i^\theta) = \begin{cases} (s_i, p_i) & \text{if } s_i \le \theta; \\ (0, 0) & \text{if } s_i > \theta, \end{cases}$$

This allows us to reduce SK-Commit to SKP. The only requirement is that at most one item from $\mathbb{B}_i$ can be packed in the knapsack. Since we assume discrete distributions, there are at most a polynomial number threshold. Theorem 5.6 directly follows from Theorem 5.1.

The the algorithm developed in[22] can be easily extended to SK-Commit and gives a $(\frac{1}{2} - \varepsilon)$-approximation ratio for SK-Commit without relaxing the capacity.

5.4.3 Stochastic Knapsack with Bound Size

Theorem 5.7: For any $\varepsilon \in (0, 1)$, there is a polynomial algorithm that finds a $(1 - 4\varepsilon)$-approximate adaptive policy for stochastic knapsack problem when the sizes are bounded by ε, *i.e.,* for any $i \in [n]$, $\Pr[s_i \le \varepsilon] = 1$.

Proof. We use OPT_λ to denote the expected profit of the optimal adaptive policy when the capacity is λ. Let $k = \lfloor \frac{1}{\varepsilon} \rfloor$.

We perform a *simulation*. There are k knapsacks. Each knapsack has a capacity $1 - \frac{1}{k}$. We simulate realization of σ^* for a knapsack with capacity 1. We follow the decision tree associated with σ^*, dynamically assigning each item chosen by σ^* to the knapsacks. We notice that decision is always before the item size is realized. The assignment strategy is very simple: for each item chosen by σ^*, we simultaneously assign it to $k - 3$ knapsack with largest remaining capacity. We define a vector $(W_1, W_2, \ldots, W_k)$ where W_i is total weight of items inserted in knapsack i. Then at any time, we have

$$|W_i - W_j| \le \frac{1}{k} \quad \forall i, j \in [k] \tag{5-11}$$

This is because that we always assign the item to the largest remaining capacity and the size is bounded by $\frac{1}{k}$. We claim that the k knapsack are never violated. Otherwise we

assume knapsack 1 is violated. Then we have $W_1 > 1 - \frac{2}{k}$ before insert the last item. We have

$$\sum_{i=1}^{k} W_i > 1 - \frac{2}{k} + (k-1) \cdot \left(1 - \frac{3}{k}\right) > k - 3,$$

which is a contradiction. Thus the experiment creates k policies, one for each knapsack. Then we have

$$\mathsf{OPT}_{1-1/k} \geq \left(1 - \frac{3}{k}\right) \mathsf{OPT}. \tag{5-12}$$

Then, we can use Theorem 5.1, to compute a policy with expected profit $(1 - \frac{1}{k})\mathsf{OPT}_{1-\frac{1}{k}}$, using total knapsack capacity of $(1 - \frac{1}{k}) + \frac{1}{k} = 1$. □

5.5 Stochastic Blackjack Knapsack

In this section, we consider the stochastic blackjack knapsack and prove Theorem 5.3. Define the item set $\mathcal{A} = \{1, 2, \ldots, n\}$. 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. We set $T = n$ and $I_1 = (0, 0)$. 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

$$I_{t+1} = f(I_t, i) = (I_{t,1} + s_i, I_{t,2} + p_i), g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} I_{T+1,2} & \text{if } I_{T+1,1} \leq \mathbb{C}, \\ 0 & \text{otherwise;} \end{cases} \tag{5-13}$$

for $t = 1, 2, \cdots, T$. Then Assumption 1.1 (2,3) is immediately satisfied. But Assumption 1.1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. Hence, we need discretization. Unlike the stochastic knapsack, we need to discretize the sizes and profits at the same time.

Consider a given adaptive policy σ. For each node $v \in \mathcal{T}_\sigma$, we have $P(v) = \sum_{i \in \mathcal{R}(v)} p_i$ where $\mathcal{R}(v)$ is the realization path from root to v. Define $\mathcal{D} = \{v \in \text{LF} : W(v) \leq \mathbb{C}\}$ where LF is the set of leaves on $\mathcal{T}_\sigma$. Then we have

$$\mathbb{P}(\sigma) = \sum_{v \in \mathcal{D}} \Phi(v) \cdot P(v). \tag{5-14}$$

Without loss of generality, we assume $\mathbb{C} = 1$ and $X_i \in [0, 1]$ for any $i \in [n]$. Let $\mathbb{P}(\sigma, \pi, 1)$ be the expected profit of the policy σ for the instance $(\pi, 1)$, where $\pi = \{\pi_i\}$ denotes the set of size distributions and 1 denotes the capacity.

5.5.1 Discretization

Next, we show that item profits can be assumed to be bounded $\theta_2 = \mathsf{OPT}/\varepsilon^2$. We set $\theta_1 = \mathsf{OPT}/\varepsilon$ and $\theta_3 = \mathsf{OPT}/\varepsilon^3$. Now, we define an item to be a *huge profit* item if it has profit greater that or equal to θ_2. We use the same discretization technique as in[6] for the stochastic knapsack. For a huge item b_i with size X_i and profit p_i, we define a new size $\hat{X}_i$ and profit $\hat{p}_i$ as follows: for $\forall s \leq 1$

$$\Pr[\hat{X}_i = s] = \Pr[\hat{X}_i = s] \cdot \frac{p_i}{\theta_2}, \quad \Pr[\hat{X}_i = 1 + 4\varepsilon] = 1 - \sum_{s \leq 1} \Pr[\hat{X}_i = s] \tag{5-15}$$

and $\hat{p}_i = \theta_2$. In Lemma 5.3, we show that this transformation can be performed with only an $O(\varepsilon)$ loss in the optimal profit. Before to prove the lemma, we need following useful lemma.

Lemma 5.2 : For any policy σ on instance (π, C), there exists a policy σ' such that $\mathbb{P}(\sigma', \pi, \mathbb{C}) = (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi, \mathbb{C})$ and in any realization path, the sum of profit of items except the last item that σ' inserts is less than θ_1.

Proof. We interrupt the process of the policy σ on a node v when the first time that $P(v) \geq \theta_1$ to get a new policy σ', *i.e.,* , we have a truncation on the node v and do not add items (include v) any more in the new policy σ'. Let F be the set of the nodes on which we have truncation. Then we have $\sum_{v \in F} \Phi(v) \leq \varepsilon$. Thus, the total profit loss is equal to $\sum_{v \in F} \Phi(v)\mathsf{OPT} \leq \varepsilon\mathsf{OPT}$. □

W.l.o.g, we assume that all (optimal or near optimal) policies σ considered in this section satisfy the following property.

(P1) In any realization path, the sum of profit of items except the last item that σ inserts is less than θ_1.

Lemma 5.3 : Let π be the distribution of size and profit for items and $\hat{\pi}$ be the scaled version of π by Equation (5-15). Then, the following statement holds:

1. For any policy σ, there exists a policy $\hat{\sigma}$ such that

$$\mathbb{P}(\hat{\sigma}, \hat{\pi}, \mathbb{C}) = (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi, \mathbb{C}).$$

2. For any policy σ,

$$\mathbb{P}(\sigma, \pi, \mathbb{C}) = (1 - O(\varepsilon))\mathbb{P}(\sigma, \hat{\pi}, \mathbb{C}).$$

Proof. (Proof of Lemma 5.3) For the first result, by Lemma 5.2, there exists a policy $\hat{\sigma}$ such that $\mathbb{P}(\hat{\sigma}, \pi, \mathbb{C}) = (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi, \mathbb{C})$ and in any realization path, there are at most one huge profit item and always at the end of the policy. For huge profit item v, the expected profit contributed by the realization path from root to v to $\mathbb{P}(\hat{\sigma}, \pi, \mathbb{C})$ is

$$\Phi(v) \cdot \Pr[X_v \leq \mathbb{C} - W(v)] \cdot (P(v) + p_v).$$

In $\mathbb{P}(\hat{\sigma}, \hat{\pi}, \mathbb{C})$ with scaled distributions on huge profit items, the expected profit contributed by the realization path from the root to v is

$$\begin{aligned} &\Phi(v) \cdot \Pr[\hat{X}_v \leq \mathbb{C} - W(v)] \cdot (P(v) + \theta_2) \\ = &\Phi(v) \cdot \left(\Pr[X_v \leq \mathbb{C} - W(v)] \cdot \frac{p_v}{\theta_2}\right) \cdot (P(v) + \theta_2). \end{aligned}$$

Since v is a huge profit item, we have $p_v \geq \theta_2$, which implies $\frac{p_v}{\theta_2} \cdot (P(v) + \theta_2) \geq P(v) + p_v$. This completes the proof of the first part.

Now, we prove the second part. By Property (P1), for a huge item v, we have $P(v) \leq \mathsf{OPT}/\varepsilon$. Then we have

$$\frac{p_v}{\theta_2} \cdot (P(v) + \theta_2) = p_v \cdot \left(1 + \frac{P(v)}{\theta_2}\right) \leq p_v \cdot (1 + \varepsilon) \leq (1 + \varepsilon)(p_v + P(v)).$$

This completes the proof of the second part. □

In order to discretize the profit, we define the approximate profit $\widetilde{\mathbb{P}}(\sigma, \hat{\pi}) = \sum_{v \in \mathcal{D}} \Phi(v) \cdot \widetilde{P}(v)$ where

$$\widetilde{P}(v) = \theta_3 \cdot \left[1 - \prod_{i \in \mathcal{R}(v)} \left(1 - \frac{p_i}{\theta_3}\right)\right] \tag{5-16}$$

Lemma 5.4 below can be used to bound the gap between the approximate profit and the original profit.

Lemma 5.4 : For any adaptive policy σ for the scaled distribution $\hat{\pi}$, we have

$$\mathbb{P}(\sigma, \hat{\pi}, \mathbb{C}) \geq \widetilde{\mathbb{P}}(\sigma, \hat{\pi}, \mathbb{C}) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma, \hat{\pi}, \mathbb{C}).$$

Proof. Fix a node v on the tree $\mathcal{T}_\sigma$. For the left side, we have

$$\widetilde{P}(v) = \theta_3 \cdot \left[1 - \prod_{i \in \mathcal{R}(v)} \left(1 - \frac{p_i}{\theta_3}\right)\right] \leq \theta_3 \cdot \left[1 - \left(1 - \sum_{i \in \mathcal{R}(v)} \frac{p_i}{\theta_3}\right)\right] = \sum_{i \in \mathcal{R}(v)} p_i = P(v).$$

For the right size, we have

$$\begin{aligned}
\widetilde{P}(v) &= \theta_3 - \theta_3 \cdot \left[\prod_{i \in \mathcal{R}(v)} \left(1 - \frac{p_i}{\theta_3}\right)\right] \\
&\geq \theta_3 - \theta_3 \cdot \left[1 - \sum_{i \in \mathcal{R}(v)} \frac{p_i}{\theta_3} + \left(\sum_{i \in \mathcal{R}(v)} \frac{p_i}{\theta_3}\right)^2\right] \\
&= \left(\sum_{i \in \mathcal{R}(v)} p_i\right) \cdot \left[1 - \frac{\sum_{i \in \mathcal{R}(v)} p_i}{\theta_3}\right] \\
&\geq (1 - O(\varepsilon))P(v),
\end{aligned}$$

where the last inequality holds by Property (P1) that $P(v) \leq \theta_1 + \theta_2$. □

Now, we choose the same discretization technique which is used in Section 5.4.1.

Lemma 5.5 : Let $\hat{\pi}$ be the distribution of size and profit for items and be $\widetilde{\pi}$ be the discretized version of $\hat{\pi}$. Then, the following statements hold:

1. For any policy σ, there exists a canonical policy $\widetilde{\sigma}$ such that

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1 + 2\varepsilon)) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma, \hat{\pi}, 1).$$

2. For any canonical policy $\widetilde{\sigma}$,

$$\mathbb{P}(\widetilde{\sigma}, \hat{\pi}, (1 + 2\varepsilon)) \geq (1 - O(\varepsilon))\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1).$$

Proof. (The proof of Lemma 5.5) For the first result, consider a randomized canonical policy $\widetilde{\sigma}$ which has the same structure as σ. If σ_r inserts an item $\widetilde{X}$ and observes a discretized size $d \in \mathcal{V}$, it chooses a random branch in $\mathcal{T}(\sigma_r, \widetilde{\pi})$ among those sizes that are mapped to d, i.e., $\{w_e \mid D_X(w_e) = d\}$ according to the probability distribution

$$\Pr[\text{branch } e \text{ is chosen}] = \frac{\Pr[X = w_e]}{\sum_{s \mid D_X(s)=d} \Pr[X = s]}.$$

Then, the probability of an edge on $\mathcal{T}_{\sigma_r}$ is the same as that of the corresponding edge on $\mathcal{T}_{\sigma}$. The only difference is two edges are labeled with different weight w_e on $\mathcal{T}_{\sigma}$ and $\widetilde{w}_e$ on $\mathcal{T}_{\sigma_r}$.

We have $P(v) = \sum_{i \in \mathcal{R}(v)} p_i$ which is less than $O(\mathsf{OPT}/\varepsilon)$ by Lemma 5.3. Define $\mathcal{D} = \{v \in \mathrm{LF} : W(v) \le 1\}$ and $\widetilde{\mathcal{D}} = \{v \in \mathrm{LF} : \widetilde{W}(v) \le 1 + 2\varepsilon\}$, where LF is the set of leaves on $\mathcal{T}(\sigma, \pi)$. Then we have

$$\mathbb{P}(\sigma, \hat{\pi}, 1) = \sum_{v \in \mathcal{D}} \Phi(v) \cdot P(v), \quad \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1 + 2\varepsilon) = \sum_{v \in \widetilde{\mathcal{D}}} \Phi(v) \cdot P(v).$$

Define $\Delta = \{v \in \mathrm{LF} : |W(v) - \widetilde{W}(v)| \ge 2\varepsilon\}$. Then we have $\mathcal{D} \setminus \widetilde{\mathcal{D}} \subseteq \Delta$. By the result of Equation (5-8), we have

$$\sum_{v \in \mathcal{D} \setminus \widetilde{\mathcal{D}}} \Phi(v) \le \sum_{v \in \Delta} \Phi(v) = O(\varepsilon^3).$$

By Property (P1), for any node v, we have $P(v) \le \theta_1 + \theta_2$.Then the gap $\mathbb{P}(\sigma, \pi, 1) - \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1 + 2\varepsilon)$ is less than

$$\sum_{v \in D \setminus \widetilde{D}} \Phi(v) \cdot P(v) \le \sum_{v \in D \setminus \widetilde{D}} \Phi(v) \cdot \frac{2\mathsf{OPT}}{\varepsilon^2} = O(\varepsilon)\mathsf{OPT}.$$

This completes the proof of the first part.

Now, we prove the second part. Since a canonical policy makes decisions based on the discretized, $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi}, 1)$ has the same tree structure as $\mathcal{T}(\widetilde{\sigma}, \hat{\pi}, 1 + 2\varepsilon)$. Define $\mathcal{D} = \{v \in \mathrm{LF} : W(v) \le 1 + 2\varepsilon\}$ and $\widetilde{\mathcal{D}} = \{v \in \mathrm{LF} : \widetilde{W}(v) \le 1\}$, where LF is the set of leaves on $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi}, 1)$. Then we have

$$\mathbb{P}(\widetilde{\sigma}, \hat{\pi}, 1 + 2\varepsilon) = \sum_{v \in \mathcal{D}} \Phi(v) \cdot P(v), \quad \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1) = \sum_{v \in \widetilde{\mathcal{D}}} \Phi(v) \cdot P(v).$$

Then the gap $\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1) - \mathbb{P}(\widetilde{\sigma}, \hat{\pi}, (1+2\varepsilon)$ is equal to

$$\sum_{v \in \widetilde{D} \setminus D} \Phi(v) \cdot P(v) \leq \sum_{v \in \widetilde{D} \setminus D} \Phi(v) \cdot \frac{2\mathsf{OPT}}{\varepsilon^2} = O(\varepsilon)\mathsf{OPT}.$$

5.5.2 Proof

Now, we ready to prove Theorem 5.3.

Proof. (The proof of Theorem 5.3) Suppose σ^* is the optimal policy with expected profit $\mathsf{OPT} = \mathbb{P}(\sigma^*, \pi, 1)$. Given an instance π, we compute the scaled distribution $\hat{pi}$ and discretized distribution $\widetilde{\pi}$. By Lemma 5.4, Lemma 5.5 (1) and Lemma 5.3 (1), there exist a policy $\widetilde{\sigma}^*$ such that

$$\begin{aligned}
&\widetilde{\mathbb{P}}(\widetilde{\sigma}^*, \widetilde{\pi}, (1+2\varepsilon)) \\
&\geq (1 - O(\varepsilon))\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1+2\varepsilon)) \quad \text{[Lemma 5.4]} \\
&\geq (1 - O(\varepsilon))\mathbb{P}(\sigma^*, \hat{\pi}, 1) \quad \text{[Lemma 5.5 (1)]} \\
&\geq (1 - O(\varepsilon))\mathbb{P}(\sigma^*, \pi, 1) \quad \text{[Lemma 5.3 (1)]} \\
&= (1 - O(\varepsilon))\mathsf{OPT}.
\end{aligned}$$

Now, we present a stochastic dynamic program for the instance $(\widetilde{\pi}, 1+2\varepsilon)$. Define the value set $\mathcal{V} = \{0, \varepsilon^{5\times 3}, 2\varepsilon^{5\times 3}, \ldots, 1+3\varepsilon\} \times \{0, 1\}$ and the item set $\mathcal{A} = \{1, 2, \ldots, n\}$. We set $T = n$ and $I_1 = 0$. When we insert an item i into the knapsack, we observe its size realization s_i and toss a coin to get a value $\widetilde{p}_i$ with $\Pr[\widetilde{p}_i = 1] = \frac{\hat{p}_i}{\theta_3}$ and $\Pr[\widetilde{p}_i = 0] = 1 - \frac{\hat{p}_i}{\theta_3}$. Then we define the system dynamics function to be

$$I_{t+1} = f(I_t, i) = (I_{t+1,1}, I_{t+1,2}) = (\min\{1+3\varepsilon, I_{t,1} + s_i\}, \max\{I_{t,2}, \widetilde{p}_i\}) \tag{5-17}$$

and $g(I_t, i) = 0$ for $I_t \in \mathcal{V}$ and $t = 1, 2, \cdots, T$. The terminal function is

$$h(I_{T+1}) = \begin{cases} \theta_3 \cdot I_{T+1,2} & \text{if } I_{T+1} \leq 1+2\varepsilon, \\ 0 & \text{otherwise;} \end{cases} \tag{5-18}$$

Then Assumption 1.1 is immediately satisfied. By Theorem 1.6, we can find a policy σ

with profit $\widetilde{\mathbb{P}}(\sigma, \widetilde{\pi}, 1+2\varepsilon)$ at least

$$\mathsf{OPT}_d - O(\varepsilon^4) \cdot \mathsf{MAX} \geq \widetilde{\mathbb{P}}(\widetilde{\sigma}^*, \widetilde{\pi}, (1+2\varepsilon)) - O(\varepsilon)\mathsf{OPT} = (1 - O(\varepsilon))\mathsf{OPT},$$

where OPT_d denotes the expected approximate profit of the optimal policy for instance $(\widetilde{\pi}, 1+2\varepsilon)$ and $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1((0,1), \mathcal{A}) = \theta_3 = \frac{\mathsf{OPT}}{\varepsilon^3}$. By Lemma 5.4, Lemma 5.5 (2) and Lemma 5.3 (2), we have

$$\begin{aligned}
&\mathbb{P}(\sigma, \pi, (1+4\varepsilon)) \\
\geq\ & (1 - O(\varepsilon))\mathbb{P}(\sigma, \hat{\pi}, (1+4\varepsilon)) \quad \text{[Lemma 5.3 (2)]} \\
\geq\ & (1 - O(\varepsilon))\mathbb{P}(\sigma, \widetilde{\pi}, (1+2\varepsilon)) \quad \text{[Lemma 5.5 (2)]} \\
\geq\ & (1 - O(\varepsilon))\widetilde{\mathbb{P}}(\sigma, \widetilde{\pi}, (1+2\varepsilon)) \quad \text{[Lemma 5.4]} \\
\geq\ & (1 - O(\varepsilon))\mathsf{OPT},
\end{aligned}$$

which completes the proof. □

5.5.3 Without Relaxing the Capacity

Before design a policy for SBK without relaxing the capacity $\mathbb{C}$, we establish a connection between adaptive policies for SKP and SBK.

Lemma 5.6: For any policy σ for SKP on instance $\mathcal{J} = (\pi, \mathbb{C})$, there exists a policy σ' for *sbk* such that

$$\mathbb{P}_{\mathsf{SBK}}(\sigma', \pi, \mathbb{C}) \geq \frac{1}{4} \cdot \mathbb{P}_{\mathsf{SKP}}(\sigma, \pi, \mathbb{C}). \tag{5-19}$$

Proof. W.l.o.g, we assume that for any node $v \in \mathcal{T}_\sigma$, we have $\mathbb{P}(v) \leq \mathbb{P}_{\mathsf{SKP}}(\sigma, \pi, \mathbb{C})$. Otherwise, we use the subtree $\mathcal{T}_v$ to instead $\mathcal{T}_\sigma$ for SKP. Set $\theta = \mathbb{P}_{\mathsf{SKP}}(\sigma, \pi, \mathbb{C})/2$. We interrupt the process of the policy σ on a node v when the first time that the summation of is larger than or equal to θ to get a new policy σ', *i.e.,* we have a truncation on the node v and do not insert the item (include v) any more in the new policy σ'. Let F be the set of the nodes on which we have a truncation. Let $\bar{F} = \mathrm{LF} \setminus F$ be the set of rest leaves, where LF is the set of leaves of the tree $\mathcal{T}_{\sigma'}$. Then we have

$$2\theta = \sum_{v \in \bar{F}} \Phi(v) \cdot P(v) + \sum_{v \in F} \Phi(v) \cdot [P(v) + \mathbb{P}(v)]$$

$$\begin{aligned}
&\leq \theta \cdot \sum_{v \in \bar{F}} \Phi(v) + \sum_{v \in F} \Phi(v)[P(v) + 2\theta] \\
&= \theta + \sum_{v \in F} \Phi(v)[P(v) + \theta] \\
&\leq \theta + 2 \sum_{v \in F} \Phi(v) P(v). \qquad \square
\end{aligned}$$

Thus the expect profit of the policy σ' for SBK is equal to

$$\sum_{v \in F} \Phi(v) P(v) \geq \frac{1}{2} \cdot \theta = \frac{1}{4} \cdot \mathbb{P}_{\mathsf{SKP}}(\sigma, \pi, \mathbb{C}).$$

Lemma 5.7 : For any stochastic knapsack instance $\mathcal{J}$, we have

$$\mathsf{OPT}_{\mathsf{SKP}}(\mathcal{J}) \geq \mathsf{OPT}_{\mathsf{SBK}}(\mathcal{J}) \geq \frac{1}{4} \mathsf{OPT}_{\mathsf{SKP}}(\mathcal{J}). \tag{5-20}$$

For any fixed $\varepsilon \geq 0$ and instance $\mathcal{J}$, by the result of[22], there is a polynomial time algorithm to compute a policy σ for SKP with expected profit $(\frac{1}{2} - \varepsilon)\mathsf{OPT}_{\mathsf{SKP}}(\mathcal{J})$. By Lemma 5.6, we can find a policy σ' for SBK expected profit at least

$$\frac{1}{4} \times (\frac{1}{2} - \varepsilon)\mathsf{OPT}_{\mathsf{SKP}}(\mathcal{J}) \geq (\frac{1}{8} - \varepsilon)\mathsf{OPT}_{\mathsf{SBK}}(\mathcal{J}).$$

This completes the proof os Theorem 5.4.

5.6 Stochastic Target Problem

In this section, we consider the stochastic target problem and prove Theorem 5.5. Define the item set $\mathcal{A} = \{1, 2, \ldots, n\}$. Let the value I_t be the total profits of the items in the knapsack at time period t. Then we set $T = m$ and $I_1 = 0$. When we insert an item i into the knapsack and observe its profit realization, say X_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, i) = I_t + X_i, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} 1 & \text{if } I_{T+1} \geq \mathbb{T}, \\ 0 & \text{otherwise;} \end{cases} \tag{5-21}$$

for $t = 1, 2, \cdots, T$. Then Assumption 1.1 (2,3) is immediately satisfied. But Assumption 1.1 (1) is not satisfied for that the value space $\mathcal{V}$ is not of constant size. Hence, we need

discretization.

We use the same discretization technique as in Section 5.4.1. Let $\mathbb{P}(\sigma, \pi, 1)$ be the expected objective value of the policy σ for the instance $(\pi, 1)$, where $\pi = \{\pi_i\}$ denotes the set of reward distributions and 1 denotes the target. Let $\widetilde{\pi}$ be the discretized version of π. Then, we have following lemmas.

Lemma 5.8 : For any policy σ, there exists a canonical policy $\widetilde{\sigma}$ such that

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1-2\varepsilon)) \geq \mathbb{P}(\sigma, \pi, 1) - O(\varepsilon).$$

Lemma 5.9 : For any canonical policy $\widetilde{\sigma}$,

$$\mathbb{P}(\widetilde{\sigma}, \pi, (1-2\varepsilon)) \geq \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1) - O(\varepsilon).$$

Proof. (The Proof of Theorem 5.5) Suppose σ^* is the optimal policy with expected value $\mathsf{OPT} = \mathbb{P}(\sigma^*, \pi, 1)$. Given an instance π, we compute the discretized distribution $\widetilde{\pi}$. By Lemma 5.8, there exists a policy $\widetilde{\sigma}^*$ such that

$$\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1-2\varepsilon)) \geq \mathbb{P}(\sigma^*, \pi, 1) - O(\varepsilon) = \mathsf{OPT} - O(\varepsilon).$$

Now, we present a stochastic dynamic program for the instance $(\widetilde{\pi}, 1-2\varepsilon)$. Define the value set $\mathcal{V} = \{0, \varepsilon^5, 2\varepsilon^5, \ldots, 1\}$, the item set $\mathcal{A} = \{1, 2, \ldots, n\}$, $T = m$ and $I_1 = 0$. When we insert an item i into the knapsack and observe its profit realization, say X_i, we define the system dynamic functions to be

$$I_{t+1} = f(I_t, i) = \min\{1, I_t + X_i\}, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} 1 & \text{if } I_{T+1} \geq 1 - 2\varepsilon, \\ 0 & \text{otherwise;} \end{cases} \tag{5-22}$$

for $I_t \in \mathcal{V}$ and $t = 1, 2, \ldots, T$. Then Assumption 1.1 is immediately satisfied. By Theorem 1.6, we can find a policy σ with value $\mathbb{P}(\sigma, \widetilde{\pi}, 1-2\varepsilon)$ at least

$$\mathsf{OPT}_d - O(\varepsilon) \cdot \mathsf{MAX} \geq \mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1-2\varepsilon)) - O(\varepsilon) = \mathsf{OPT} - O(\varepsilon),$$

where OPT_d denotes the expected value of the optimal policy for instance $(\widetilde{\pi}, 1-2\varepsilon)$ and

$\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(1, \mathcal{A}) = 1$. By Lemma 5.9, we have

$$\mathbb{P}(\sigma, \pi, (1-4\varepsilon)) \geq \mathbb{P}(\sigma, \widetilde{\pi}, (1-2\varepsilon)) - O(\varepsilon) \geq \mathsf{OPT} - O(\varepsilon),$$

which completes the proof. □

Proof. (The Proof of Lemma 5.8) Consider a randomized canonical policy $\widetilde{\sigma}$ which has the same structure as σ. If σ_r inserts an item $\widetilde{X}$ and observes a discretized size $d \in \mathcal{V}$, it chooses a random branch in $\mathcal{T}(\sigma_r, \widetilde{\pi})$ among those sizes that are mapped to d, i.e., $\{w_e \mid D_X(w_e) = d\}$ according to the probability distribution

$$\Pr[\text{branch } e \text{ is chosen}] = \frac{\Pr[X = w_e]}{\sum_{s \mid D_X(s)=d} \Pr[X = s]}.$$

Then, the probability of an edge on $\mathcal{T}_{\sigma_r}$ is the same as that of the corresponding edge on $\mathcal{T}_{\sigma}$. The only difference is two edges are labeled with different weight w_e on $\mathcal{T}_{\sigma}$ and $\widetilde{w}_e$ on $\mathcal{T}_{\sigma_r}$.

Notice that $\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1-2\varepsilon))$ is the sum of all paths $\mathcal{R}(v)$ with $\widetilde{W}(v) \geq 1-2\epsilon$. Define $\mathcal{D} = \{v \in \mathrm{LF} : W(v) \geq 1\}$ and $\widetilde{\mathcal{D}} = \{v \in \mathrm{LF} : \widetilde{W}(v) \geq 1-2\varepsilon\}$, where LF is the set of leaves on $\mathcal{T}(\sigma, \pi)$. Therefore we have

$$\mathbb{P}(\sigma, \pi, 1) = \sum_{v \in \mathcal{D}} \Phi(v), \quad \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1-2\varepsilon)) = \sum_{v \in \widetilde{\mathcal{D}}} \Phi(v).$$

Consider the set $\Delta_1 = \mathcal{D} \setminus \widetilde{\mathcal{D}}$. For each $v \in \Delta_1$, we have $W(v) \geq 1$ and $\widetilde{W}(v) < 1-2\varepsilon$. Thus we claim that $|W(v) - \widetilde{W}(v)| > 2\varepsilon$, implying that $\Delta_1 \subseteq \Delta \doteq \{v \in \mathrm{LF} : |W(v) - \widetilde{W}(v)| \geq 2\epsilon\}$. Thus we have

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1-2\varepsilon)) \geq \mathbb{P}(\sigma, \pi, 1) - \sum_{v \in \Delta_1} \Phi(v) \geq \mathbb{P}(\sigma, \pi, 1) - \sum_{v \in \Delta} \Phi(v) \geq \mathbb{P}(\sigma, \pi, 1) - O(\varepsilon).$$

Proof. (The Proof of Lemma 5.9) In our case, we focus on the decision tree $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi}, 1)$ and assume all $\widetilde{w}_e$ take discretized value. $\mathcal{T}(\widetilde{\sigma}, \widetilde{\pi}, 1)$ has the same tree structure as $\mathcal{T}(\widetilde{\sigma}, \pi, 1-2\varepsilon)$.

Define $\Delta_2 = \{v \in \mathrm{LF}, W(v) < 1-2\varepsilon, \widetilde{W}(v) \geq 1\}$, where LF is the set of leaves in $\mathcal{T}_{\sigma}$. Then we see $\widetilde{W}(v) - W(v) > 2\varepsilon$, implying $\Delta_2 \subseteq \Delta = \{v \in \mathrm{LF} : |W(v) - \widetilde{W}(v)| \geq 2\varepsilon\}$.

By the result of Equation (5-8), we see

$$\sum_{v \in \Delta_2} \Phi(v) \leq \sum_{v \in \Delta} \Phi(v) = O(\varepsilon).$$

Therefore we claim that

$$\mathbb{P}(\widetilde{\sigma}, \pi, (1 - 2\varepsilon)) \geq \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1) - \sum_{v \in \Delta_2} \Phi(v) \geq \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1) - O(\varepsilon).$$

第 6 章　Conclusion

We conclude this thesis with a summary of our results as well as the techniques, and raise some open problems suggested by our studies.

In the first part of the dissertation, we formally define a model based on stochastic dynamic programs. This is a generic model. There are a number of stochastic optimization problems which fit in this model. We design a polynomial time approximation scheme for this model.

We also study two important stochastic optimization problems, stochastic probing problem and stochastic knapsack problem. We propose an approximation algorithm for stochastic probing problem. Using the stochastic dynamic programs, we design a PTAS for ProbeMax, which improves the best known approximation ratio $1-1/e$. We also design a PTAS for committed ProbeMax with a general matroid outer constraint. To improve the approximation ratio for ProbeMax with a matroid outer constraint is still a open problem. We also study the commitment gap and adaptivity gap for stochastic probing, and give a bound about them.

Next, we focus on the variants of stochastic knapsack problem: stochastic blackjack knapsack and stochastic target problem. Using the stochastic dynamic programs and discretization technique, we design a PTAS for them if allowed to relax the capacity or target. To improve the ratio for stochastic knapsack problem and variants without relaxing the capacity is still a open problem.

参考文献

[1] Dantzig G B. Linear programming under uncertainty[J]. Management Science, 1955, 1(3-4): 197–206.

[2] Bellman R. Dynamic programming[C]//Princeton University Press. [S.l.: s.n.], 1957.

[3] Bertsekas D P. Dynamic programming and optimal control: volume 1[M]. [S.l.]: Athena scientific Belmont, MA, 1995

[4] Powell W B. Approximate dynamic programming: Solving the curses of dimensionality: volume 842[M]. [S.l.]: John Wiley & Sons, 2011

[5] Dean B C, Goemans M X, Vondrák J. Adaptivity and approximation for stochastic packing problems[C]//Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. [S.l.]: Society for Industrial and Applied Mathematics, 2005: 395–404.

[6] Bhalgat A, Goel A, Khanna S. Improved approximation results for stochastic knapsack problems [J]. Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms, 2011: 1647–1665.

[7] Adamczyk M. Improved analysis of the greedy algorithm for stochastic matching[J]. Information Processing Letters, 2011, 111(15): 731–737.

[8] Bansal N, Gupta A, Li J, et al. When lp is the cure for your matching woes: Improved bounds for stochastic matchings[J]. Algorithmica, 2012, 63(4): 733–762.

[9] Swamy C, Shmoys D B. Approximation algorithms for 2-stage stochastic optimization problems [J]. ACM SIGACT News, 2006, 37(1): 33–46.

[10] Gupta A, Pál M, Ravi R, et al. Boosted sampling: approximation algorithms for stochastic optimization[C]//Proceedings of the thirty-sixth annual ACM symposium on Theory of computing. [S.l.]: ACM, 2004: 417–426.

[11] Li J, Yuan W. Stochastic combinatorial optimization via poisson approximation[J]. Proceedings of the forty-fifth annual ACM symposium on Theory of computing, 2013: 971–980.

[12] Liu J L Y. Approximation algorithms for stochastic combinatorial optimization problems[J/OL]. Journal of the Operations Research Society of China, 2016, 4(1): 1. http://www.jorsc.shu.edu.cn/EN/abstract/article_6.shtml.

[13] Chen W, Hu W, Li F, et al. Combinatorial multi-armed bandit with general reward functions [J]. Advances in Neural Information Processing Systems, 2016.

[14] Munagala K. Approximation algorithms for stochastic optimization[M]. Simons Institue for the Theory of Computing: [s.n.], 2016.

[15] Vondrák J, Chekuri C, Zenklusen R. Submodular function maximization via the multilinear relaxation and contention resolution schemes[C]//Proceedings of the forty-third annual ACM symposium on Theory of computing. [S.l.]: ACM, 2011: 783–792.

[16] Gupta A, Nagarajan V. A stochastic probing problem with applications[C]//International Conference on Integer Programming and Combinatorial Optimization. [S.l.]: Springer, 2013: 205–216.

[17] Asadpour A, Nazerzadeh H. Maximizing stochastic monotone submodular functions[J]. Management Science, 2015, 62(8): 2374–2391.

[18] Gupta A, Nagarajan V, Singla S. Algorithms and adaptivity gaps for stochastic probing[M]// Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. [S.l.]: Society for Industrial and Applied Mathematics, 2016: 1731–1747.

[19] Golovin D, Krause A. Adaptive submodularity: Theory and applications in active learning and stochastic optimization[J]. Journal of Artificial Intelligence Research, 2011, 42: 427–486.

[20] Gupta A, Krishnaswamy R, Molinaro M, et al. Approximation algorithms for correlated knapsacks and non-martingale bandits[C]//Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on. [S.l.]: IEEE, 2011: 827–836.

[21] Ma W. Improvements and generalizations of stochastic knapsack and markovian bandits approximation algorithms[J]. Mathematics of Operations Research, 2017.

[22] Bhalgat A. A $(2+\varepsilon)$-approximation algorithm for the stochastic knapsack problem[J]. Unpublished Manuscript, 2011.

[23] Levin A, Vainer A. Adaptivity in the stochastic blackjack knapsack problem[J]. Theoretical Computer Science, 2014, 516: 121–126.

[24] Fu H, Li J, Xu P. A ptas for a class of stochastic dynamic programs[C]//45th International Colloquium on Automata, Languages, and Programming. [S.l.: s.n.], 2018.

[25] Shmoys D B, Swamy C. An approximation scheme for stochastic linear programming and its application to stochastic integer programs[J]. Journal of the ACM (JACM), 2006, 53(6): 978–1012.

[26] Halman N, Klabjan D, Li C L, et al. Fully polynomial time approximation schemes for stochastic dynamic programs[C]//Nineteenth Acm-Siam Symposium on Discrete Algorithms. [S.l.: s.n.], 2008: 700–709.

[27] Halman N, Klabjan D, Li C L, et al. Fully polynomial time approximation schemes for stochastic dynamic programs[J]. SIAM Journal on Discrete Mathematics, 2014, 28(4): 1725–1796.

[28] Halman N, Nannicini G, Orlin J. A computationally efficient fptas for convex stochastic dynamic programs[J]. SIAM Journal on Optimization, 2015, 25(1): 317–350.

[29] Krengel U, Sucheston L. On semiamarts, amarts, and processes with finite value[J]. Advances in Prob, 1978, 4: 197–266.

[30] Kleinberg R, Weinberg S M. Matroid prophet inequalities[C]//Proceedings of the forty-fourth annual ACM symposium on Theory of computing. [S.l.]: ACM, 2012: 123–136.

[31] Chekuri C, Vondrák J, Zenklusen R. Multi-budgeted matchings and matroid intersection via dependent rounding[C]//Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms. [S.l.]: SIAM, 2011: 1080–1097.

[32] Gupta A, Nagarajan V, Singla S. Adaptivity gaps for stochastic probing: Submodular and xos functions[C]//Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. [S.l.]: SIAM, 2017: 1688–1702.

[33] Weitzman M L. Optimal search for the best alternative[J]. Econometrica, 1979, 47(3): 641–654.

[34] Singla S. The price of information in combinatorial optimization[C]//Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. [S.l.]: SIAM, 2018: 2523–2532.

[35] Adamczyk M, Sviridenko M, Ward J. Submodular stochastic probing on matroids[J]. Mathematics of Operations Research, 2016, 41(3): 1022–1038.

[36] Chawla S, Hartline J D, Malec D L, et al. Multi-parameter mechanism design and sequential posted pricing[C]//Proceedings of the forty-second ACM symposium on Theory of computing. [S.l.]: ACM, 2010: 311–320.

[37] Yan Q. Mechanism design via correlation gap[C]//Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms. [S.l.]: Society for Industrial and Applied Mathematics, 2011: 710–719.

[38] Esfandiari H, Hajiaghayi M, Liaghat V, et al. Prophet secretary[M]//Algorithms-ESA 2015. [S.l.]: Springer, 2015: 496–508

[39] Dynkin E B. The optimum choice of the instant for stopping a markov process[C]//Soviet Math. Dokl: volume 4. [S.l.: s.n.], 1963.

[40] Kleinberg R. A multiple-choice secretary algorithm with applications to online auctions[C]// Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. [S.l.]: Society for Industrial and Applied Mathematics, 2005: 630–631.

[41] Feldman M, Svensson O, Zenklusen R. A simple o (log log (rank))-competitive algorithm for the matroid secretary problem[C]//Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms. [S.l.]: Society for Industrial and Applied Mathematics, 2015: 1189–1201.

[42] Lachish O. O (log log rank) competitive ratio for the matroid secretary problem[C]//Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on. [S.l.]: IEEE, 2014: 326–335.

[43] Vazirani V V. Approximation algorithms[M]. [S.l.]: Springer Science & Business Media, 2013

[44] Chen K, Ross S M. An adaptive stochastic knapsack problem[J/OL]. European Journal of Operational Research, 2014, 239(3): 625 – 635. http://www.sciencedirect.com/science/article/pii/S037722171400530X.

[45] İlhan T, Iravani S M, Daskin M S. The adaptive knapsack problem with stochastic rewards[J]. Operations Research, 2011, 59(1): 242–248.

致　谢

首先，我要感谢我的导师李建教授对我的精心指导。他的言传身教将使我终生受益。

我还要感谢我所有的论文评论人孙晓明教授，陈卫教授，李闽冥教授，黄志毅教授，陈旭瑾教授，栗师教授，Marco Molinaro 教授提供了许多有用的意见和建议。

在斯特拉斯堡大学进行六个月的合作研究期间，承蒙韩国牛教授热心指导与帮助，不胜感激。感谢实验室全体老师和同学们的热情帮助和支持。感谢金逸飞，曹玮，曾驭龙，戴元熙，刘艺成，何昊青，高阳，赵曦等朋友们在博士期间的照顾（我遗漏了很多）。本课题承蒙国家自然科学基金资助，特此致谢。

最后，我要感谢我的家人的支持，信任和理解。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　中文论文

A.1　简介

近年来，在很多应用场景中都产生了不确定的数据。这几乎覆盖了从电信，医药到金融工程的所有科学领域。在这些场景中，不确定性是不可避免的。如何以严谨的方式，分析和解决这些问题成为近年来一个热门的课题。该领域起源于 Dantzig[1] 的工作，并延伸出了广泛的研究。其研究成果也在许多领域都得到了应用。1957 年，Richard Bellman[2] 提出了随机动态规划模型（stochastic dynamic programming）。这是一种用来建模和解决不确定条件下决策问题的技术。因为这是一个很通用的模型，可用来设计优化算法，它在随机优化模型中扮演着很重要的角色。对于一些特定问题，可以通过直接求解 Bellman 方程获得多项式时间的闭式或精确解。另一方面，由于“维度的诅咒”和超大的状态空间，这种方法也有许多的局限。我们推荐参考书籍[3,4]。

目前，随机优化已经成为处理各种优化问题中不确定性的主要方法。对于一些特定的场景，研究人员提出了一组模型并给出了相应的多项式时间的近似算法，如随机背包[5,6]（stochastic knapsack），随机匹配[7,8]（stochastic matching），随机集覆盖[9,10]（stochastic set cover），随机容器包装[11]（stochastic bin packing）等等。感兴趣的读者可以参考[12]。

在本论文中，我们关注一类随机动态规划问题。针对这些问题，我们提出了一个基于随机动态规划的模型，并设计了一个有效的多项式时间的近似算法。通过我们的模型，可以为一群适合这个模型的随机组合优化问题设计多项式算法。接下来我们引入两类问题：随机探测问题（stochastic probing problem）和随机背包问题（stochastic knapsack），并简要陈述我们的结果。

A.1.1　随机探测问题

假设一家公司准备招聘一位秘书。当把招聘广告投放出去之后，一共收到了 n 份简历，通过简历，公司可以大致评估应聘者是否与公司的职位要求相匹配，并给一个分数，称之为匹配度。这个匹配度可能是不确定，而是一个已知分布的随机变量（例如 50% 的概率为 70，50% 的概率为 80）。想要了解应聘者真实的匹配度，公司需要安排面试。由于时间精力有限，公司只能从中挑选 $m(\leq n)$ 个应聘者来进行面试。那应该设计一个怎样的策略，能让公司可以选到一个匹配度尽可能高的秘书？这个问题可以被下面这个模型所表示。

问题 A.1 (**探测最大值问题** ProbeMax)：　给定一组非负独立的随机变量 $\{X_i\}_{i=1}^n$，并已知每个随机变量的（离散）分布 π_i。我们可以按一定顺序探测一个子集 $P \subseteq [n]$，每当探测变量 i 后，我们就可以得到来自分布 π_i 的一个独立样本值 X_i。我们可以自适应地（adaptively）探测最多 m 的变量，并且每个变量最多可以探测一次。最后的收益是这 m 个样本中的最大值。我们的目标是设计一个自适应探测策略，使得收益的期望值最大。

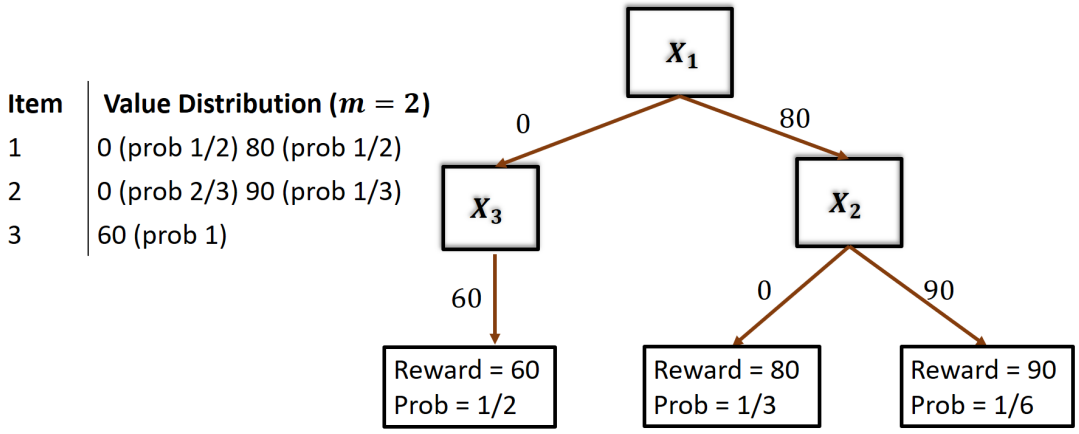

图 A.1　探测最大值问题的一个例子.

备注: 图中所示的是一个探测最大值问题的一个示例。它最优的自适应策略可以表示成图中的一棵决策树。这颗决策树的期望收益为 71.67。如果考虑一个非自适应策略，那最优的策略是选择探测变量 1 和变量 3。这个策略的期望收益为 $\mathbb{E}\max\{X_1, X_3\} = 70$。

探测最大值问题及其变体已在许多论文中都研究过 (参考[13–18])。尽管是一个非常基本的随机优化问题，我们仍然没有完全理解探测最大值问题的近似性，甚至不知道获得最佳策略是不是可能的。

一个非自适应策略只是一个先验的固定的变量集合 $P \subseteq [n]$。注意到 $f(P) = \mathbb{E}[\max_{i\in P} x_i]$ 是一个次模函数，因此根据标准的做法，很容易得到一个 $1 - 1/e$ 的近似解。后来陈等[13] 找到了在所有非适应性策略中的第一个 PTAS。参见图 A.1这个简单的例子，相比非自适应策略，我们可以很明显的对比出适应性策略的优势。对于探测最大值问题，目前所知道的最优的近似解是 $1 - \frac{1}{e}$ 近似[17,19]，这可以通过使用 Asadpour 等人[17] 的随机最大化单调次模算法得到，不过这也是一个非适应性策略，这就意味着最优的自适应策略和最优的非自适应策略的比值最多为 $\frac{e}{e-1}$。后来 Golovin 等人[19] 介绍一个自适应单调次模的概念，同样可以得到 $1 - \frac{1}{e}$ 近似。在本文中，我们提供第一个在所有适应性策略中的 PTAS，需要强调是我们的策略确实是一个自适应策略。

定理 A.1：　对于所有自适应策略的探测最大值问题，都存在一个 PTAS。也就是说，对于任何固定的常数 $\varepsilon > 0$，有一个可以用于探测最大值问题的多项式时间近似算法，它可以找到一个期望收益值至少为 $(1-\varepsilon)\mathsf{OPT}$ 的策略，这里的 OPT 是最优自适应策略的期望收益值。

Gupta 等[18] 提出了一个随机探测问题（stochastic probing problem）的框架。其中探测最大值问题就是这个通用框架的一个特例。

问题 A.2 (**随机探测问题** stochastic probing problem)：　随机探测问题的实例 $\mathcal{J}$ 由三元组 $\mathcal{J} = (U, \pi, \mathcal{I})$ 组成，其中 U 是一组非负随机变量 $\{X_i\}_{i=1}^{n}$，并已知每个随机变量的（离散）分布 π_i 以及 $\pi = \{\pi_i\}_{[i\in U]}$，约束条件 $\mathcal{I}$ 由两个独立的集合系统 $\mathcal{I}_{out}$ 和 $\mathcal{I}_{in}$ 组成，分别表示外部（打包）约束和内部（打包）约束。我们可以按一定顺序自适应地探测一个子集 $P \subseteq U$，并选择另一个子集 $C \subseteq P$ 作为最终输出集合。每当探测变量 i 后，我们就可以得到来自分布 π_i 的一个独立样本值 X_i。约束条件是我们探测的变量序列集合 P 必须在 $\mathcal{I}_{out}$ 中，我们输出的最终集合 C 必须在 $\mathcal{I}_{in}$ 中。我们的目标是设计一个策略，使预期值 $\mathbb{E}[\sum_{i\in C} X_i]$ 最大。

这里有两种约束条件：

- 外部约束 $\mathcal{I}_{out}$[①]: 这是我们可以探测变量序列的集合。这要求我们探测的变量集合 P 必须属于 $\mathcal{I}_{out}$，并每个变量最多被探测一次。
- 内部约束 $\mathcal{I}_{in}$: 这是我们最终可以输出的变量集合 C 的约束条件。它要求集合 C 必须属于 $\mathcal{I}_{in}$。

根据我们如何选择最终集合 C，可以自然的分成两个模型：承诺模型（committed）和非承诺模型（non-committed）[②]。

- 承诺模型（committed）：每当我们探测一个变量，并得到它的样本值，我们必须立即做出一个是否选择它的决定，并且不可撤销。也就是说，我们必须立即将它添加到最终选择的集合 C 或者永远抛弃它。
- 未承诺模型（non-committed）：我们可以在得到 P 中所有探测变量的样本值后再选择集合 C。也就是说，C 的选择可以在 P 中的变量都被探测之后进行。

实际上，每一个承诺策略都是一个非承诺策略的特例。随机探测问题可以很自然地推广到非承诺模型的探测最大值问题（ProbeMax ），这里，外部约束 $\mathcal{I}_{out}$ 是简单的 m-均匀拟阵（m- uniform matroid），也就是说 $\mathcal{I}_{out} = \{P \mid |P| \le m\}$。内部约束 $\mathcal{I}_{in}$ 是 1-均匀拟阵，也就是说我们只输出一个最大值。我们可以用一些更一般的约

① $\mathcal{I}_{out}$ 可以理解为一些序列的集合。如果 $\mathcal{I}_{out}$ 由集合的组成, 那么集合以任何顺序组成的序列都可以被探测。

② Gupta 等[18] 把它们称为在线和离线决策模型。

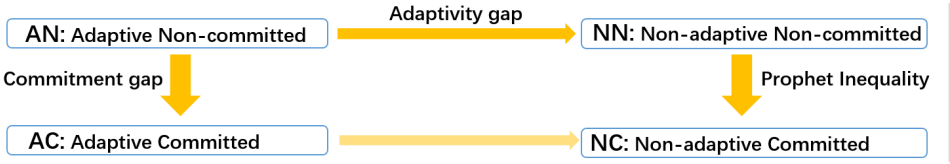

图 A.2　承诺差距与自适应差距

束来替换外部约束，例如一个拟阵。如果用 k-均匀拟阵代替探测最大值问题中的内部约束（*i.e.*, $\mathcal{I}_{in} = \{C \mid |C| \leq k\}$），这就是探测前 k 大和问题（ProbeTop-k）。在本文中，我们假设 k 是一个常数。

如图A.2所示，有四种不同类型的 (AN, NN, AC, NC) 策略用于随机探测问题，其中第一个符号指示策略是否是自适应的, A 表示是，N 表示不是，第二个符号指示是否是承诺策略，C 表示是，N 表示不是。对于特定的随机探测实例 $\mathcal{J}$，我们使用 $\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})$ 来表示为类型 AN 的最优策略的期望值，如果上下文清楚，我们还使用简写符号 $\mathsf{OPT}_{\mathsf{AN}}$ 表示，类似我们有 $\mathsf{OPT}_{\mathsf{AC}}(\mathcal{J}), \mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})$ 和 $\mathsf{OPT}_{\mathsf{NC}}(\mathcal{J})$。

注意到对于特定的随机探测实例 $\mathcal{J}$，$\mathsf{OPT}_{\mathsf{AN}}$ 值不小于 $\mathsf{OPT}_{\mathsf{NN}}$ 和 $\mathsf{OPT}_{\mathsf{AC}}$。我们用承诺差距（commitment gap）和自适应差距（adaptivity gap）来测量之间的差距，定义如下：

定义 A.1：（自适应差距）对于特定的随机探测实例 $\mathcal{J}$，我们定义在非承诺模型的自适应差距

$$\mathsf{AdaptiveGap}(\mathcal{J}) = \frac{\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{NN}}(\mathcal{J})} \tag{A-1}$$

定义 A.2：（承诺差距）对于特定的随机探测实例 $\mathcal{J}$，我们定义承诺差距为

$$\mathsf{CommitGap}(\mathcal{J}) = \frac{\mathsf{OPT}_{\mathsf{AN}}(\mathcal{J})}{\mathsf{OPT}_{\mathsf{AC}}(\mathcal{J})} \tag{A-2}$$

现在，我们在表格A.1中列出我们的主要结果。这个表格A.1中的所有近似值都是与最优自适应策略进行比较。

定理 A.2：　当内部约束是均匀拟阵，外部约束是任意前缀闭包约束时，任何随机探测问题的承诺差距最多为 2。

我们的技术可以从定理A.2的证明中推导出以下有用的结果，该结果还改进了[18]中 3.51 的结果。

推论 A.1：　当内部约束是均匀拟阵，外部约束是任意前缀闭包约束时，任何随机探测问题的适应性差距最多为 2。

表 A.1　随机探测问题（stochastic probing problem）近似结果总结

内部约束	外部约束	承诺模型	未承诺模型
1-均匀拟阵	均匀拟阵	PTAS [Thm 1.5 in[11]]	$1-1/e$[17], $1/8$[14], PTAS [定理 A.1 in $\mathbb{S}$A.3]
	拟阵	PTAS [Thm 4.2 in $\mathbb{S}$4.5.2]	$1-1/e$[17]
	2 个拟阵	PTAS [Thm 4.2 in $\mathbb{S}$4.5.2]	$1/3$ [Thm 4.6], $1/2+\varepsilon$ [Thm 1.2 in $\mathbb{S}$4.5.3]
k-均匀拟阵	均匀拟阵	PTAS [Thm 4.3 in $\mathbb{S}$4.5.1]	$1-1/e$[17], PTAS [Corollary 4.1 in $\mathbb{S}$4.4.2]
拟阵	拟阵	$1/2$ [Thm 4.6 in $\mathbb{S}$4.6]	$1-1/e$[17]
k^{in} 个拟阵	k^{out} 个拟阵	$\frac{1}{k^{in}+k^{out}}$ [Thm 4.6 in $\mathbb{S}$4.6]	

A.1.2　随机背包问题及其变种

问题 A.3 (**随机背包问题** (SKP))：　给我们一个容量为 $\mathbb{C}$ 的背包和 n 个物品，每个物品 $i \in [n]$ 的大小 X_i 是随机的，由已知分布 π_i 决定，每个物品 i 的价值为 p_i。当我们决定将一个物品 i 放入背包中，我们就知道其实际的大小 s_i。它与分布 π_i 有关。只要不超过容量限制，我们可以自适应地将物品放入背包。一旦超过容量，我们就停止放入物品。我们的目标是设计一个策略，使得背包中物品总价值最大。

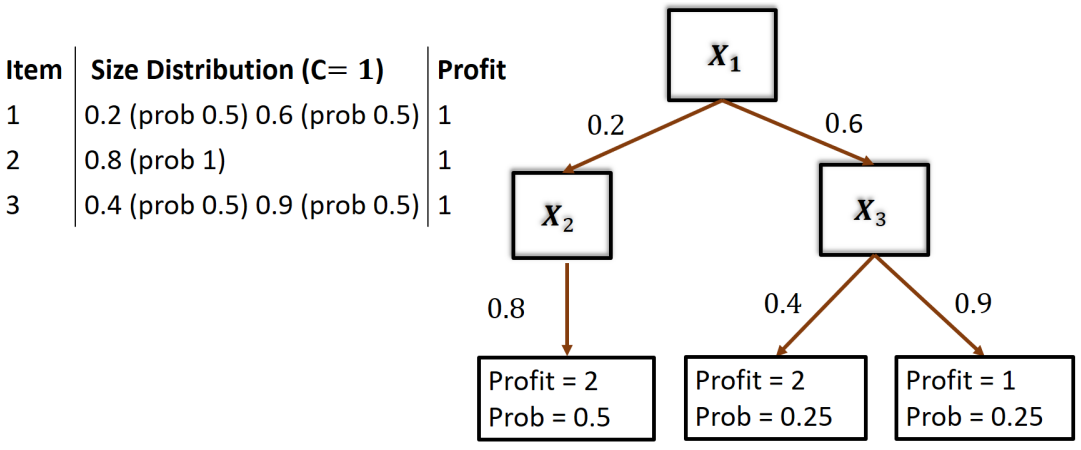

图 A.3　随机背包问题的一个例子

备注: 图中所示的是一个随机背包问题的一个示例。它的最优的自适应策略可以表示成图中的一棵决策树。这颗决策树的期望价值为 1.75。如果考虑一个非自适应策略，那最优的非适应性策略是依次放入物品 $1, 2, 3$，可以实现的期望价值为 1.5。

这个问题是随机组合优化中的一个经典问题。一个自然的动机是来自如下的

随机调度问题。假设我们想要在单台机器上安排 n 份作业。每项作业都有一个随机的处理时间（已知其分布）和相应完成所获得的固定利润。给定一个固定的截止日期，我们希望在截止日期前获得尽可能多的利润。

我们考虑自适应策略（参见图A.3），这个问题是 PSPACE 难题（这个问题的一个变体已被证明是 PASACE-难[5]）。为这个问题设计好的近似算法近年来受到了相当多的关注[5,6,11,20–22]。接下来，Levin 等人[23] 引入了随机背包问题的一种变体：一旦超过背包容量，所有的物品的价值都会失去，这被形象的称为随机二十一点背包问题。

问题 A.4 (**随机二十一点背包问题** (SBK))：　在这个问题中，我们给出了一个实例 $\mathcal{J} = (\pi, \mathbb{C})$。我们的目标是设计一个适应性策略，使得放入背包的预期物品总价值最大。与随机背包问题的区别是，如果放入的总物品大小超过容量 $\mathbb{C}$，我们获得的价值为 0。

定理 A.3：　如果我们将背包容量放宽到 $(1+\varepsilon)\mathbb{C}$，那么存在一个随机二十一点背包问题的 PTAS。换句话说，对于任何给定的常量 $\varepsilon > 0$，都有一个多项式时间近似算法，它找到一个期望价值至少为 $(1-\varepsilon)\mathsf{OPT}$ 的策略，其中 OPT 是为最佳适应性策略的期望价值。

接下来的一个问题的动机来自选择渔场，这也是随机背包问题的一个变种。假设杰克船长有一艘渔船并准备去捕鱼，他有几个渔场可供选择。通过近乎实时的海洋学分析和历史数据，可以预先估计渔场的产量。但是，预估的也仅仅是一个产量的分布，真实的产量只有到捕获结束时才能知道。给定一个捕鱼目标，如何选择正确的渔场可以以最高的概率达到目标。这个问题可以用下面的模型来表示。

问题 A.5 (**随机目标问题（STP）**)：　我们给定预定目标 $\mathbb{T}$ 和一组非负的随机变量 $\{X_i\}_{i\in[n]}$，并已知每个随机变量的（离散）分布 π_i。每当选择一个变量 i 时，我们就可以得到来自分布 π_i 的一个独立样本值 X_i。可以自适应的选择 m 变量，我们的目标是设计一个适应性策略，使得 $\Pr[\sum_{i\in P} x_i \geq \mathbb{T}]$ 是最大，其中 $P \subseteq [n]$ 是选择的变量集合。

定理 A.4：　如果我们将目标放松到 $(1-\varepsilon)\mathbb{T}$，那么存在一个随机目标问题的 PTAS。换句话说，任何给定的常量 $\varepsilon > 0$，都有一个多项式时间近似算法，该算法可以找到一个策略，使得变量总和超过 $(1-\varepsilon)\mathbb{T}$ 的概率至少为 $\mathsf{OPT} - \varepsilon$，其中 OPT 是最优自适应策略能达到目标的概率。

Item	Value Distribution (target = 30)
1	10 (prob 1)
2	10 (prob 1/2) 20 (prob 1/2)
3	0 (prob 1/2) 20 (prob 1/2)

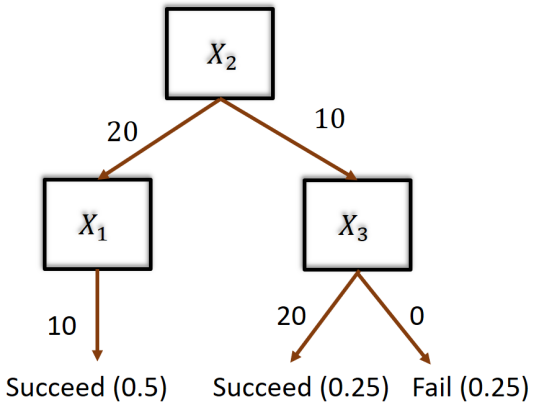

图 A.4　随机目标问题的例子

备注: 图中所示的是一个随机目标问题的一个示例，这里只能选择 2 个变量。它的最优的自适应策略可以表示成图中的一棵决策树。这颗决策树能达到目标的概率为 0.75。如果考虑一个非自适应策略，那最优的非适应性策略是选择 1, 3，能达到目标的概率为 0.5。

通过图A.4中一个简单的例子，我们可以看到自适应策略的优势。如本例所示，如果使用自适应策略，它会显著的提高最优解。现在，我们在表A.2中列出我们的主要结果。

表 A.2　随机背包问题（stochastic knapsack）及其变种的近似结果总结

问题	单参数近似	双参数近似 ($\mathbb{C}$ 放宽至 $(1+\varepsilon)\mathbb{C}$ 或者 $\mathbb{T}$ 放宽至 $(1-\varepsilon)\mathbb{T}$)
随机背包问题	$\frac{1}{3}-\varepsilon$[5], $\frac{3}{8}-\varepsilon$[6], $\frac{1}{2}-\varepsilon$[22]	$1-\varepsilon$[6,11] [定理 5.1 在 $\mathbb{S}$5.4]
随机二十一点背包	$(\sqrt{2}-1)^2/2 \approx 1/11.66$[23] $1/8-\varepsilon$ [定理 5.4 在 $\mathbb{S}$5.5.3]	$1-\varepsilon$ [定理A.3 在 $\mathbb{S}$A.5]
随机目标问题	-	OPT $-\varepsilon$ [定理 A.4 在 $\mathbb{S}$A.4]

如果随机背包问题中的物品大小有界，我们有以下定理。

定理 A.5：　对于任何 $\varepsilon \in (0,1)$，当满足 $i \in [n]$，$\Pr[s_i \le \varepsilon] = 1$，存在有一个多项式算法，可以为随机背包问题找到一个 $(1-4\varepsilon)$ 的多项式近似算法。

A.1.3　随机动态规划

上面讨论的问题都是重要的随机优化问题，从这三个展示的示例（图A.1，A.3，A.4），我们可以看到自适应策略的优势，但是设计一个自适应策略是相当困难的，这是因为对应最优策略的决策树可能是指数级大，并且非常复杂的，因此，在多项式空间中表示最佳决策树几乎是不可能的，这就是研究人员热衷于设计非适应

性策略的原因。在本文中，我们提供了一个解决方案，为了解决这些问题，我们开发了一个基于随机动态规划的通用模型，我们还设计了一个有效的多项式时间近似算法来寻找自适应策略，这个模型可以用于很多包括前面提到的随机优化问题。

考虑一个有限轮次的在线随机优化问题，给定一组任务（或项目，物品，工作，副本或行动），在每一轮中，我们都可以选择一项任务，每个任务最多可以选择一次。我们有一个初始的系统“状态”（称为系统的价值）。在每一轮，我们可以选择一个任务，完成任务会产生一些（可能是随机的）反馈，包括改变系统的价值并为本轮提供一些收益。我们的目标是设计一个策略，使得我们的总（预期）收益最大。这个模型可以被形式化表示如下：

问题 A.6 (**随机动态规划**（Stochastic Dynamic Programs）)：给定一个 6 元组 $(\mathcal{V}, \mathcal{A}, f, g, h, T)$。这里，$\mathcal{V}$ 是系统所有可能价值的集合，$\mathcal{A}$ 是项目或者任务集合，每个项目最多可以选择一次。这个模型最多可以进行 T 轮，在每一轮 $t \in [T]$，我们用 $I_t \in \mathcal{V}$ 表示系统的当前价值，$\mathcal{A}_t \subseteq \mathcal{A}$ 表示剩余可用的项目集合。如果我们选择一个项目 $a_t \in \mathcal{A}_t$，系统的价值更改为 $f(I_t, a_t)$，这里 f 可能是随机的，并对于每个项目 $a_t \in \mathcal{A}$ 是独立的。系统同时产生一个随机收益 $g(I_t a_t)$。函数 $h(I_{T+1})$ 是过程结束时的最终收益函数。给定系统初始价值 $I_1 \in \mathcal{V}$，我们的目标是设计一个适应性策略，使得收益总额的期望值 $\mathbb{E}_{t=1}^{T} g(I_t, a_t) + h(I_{T+1})]$ 最大。

对于随机动态规划，为了获得多项式时间近似方案（PTAS），我们需要以下假设。

假设 A.1：我们有如下假设：

1. 价值空间 $\mathcal{V}$ 是离散的和有序的，并且其空间大小 $|\mathcal{V}|$ 为常数。不失一般性，假设 $\mathcal{V} = \{0, 1, \ldots, |\mathcal{V}| - 1\}$。
2. 函数 f 满足 $f(I_t, a_t) \geq I_t$，也就是说系统价值是非减的。
3. 函数 $h : \mathcal{V} \to R^{\geq 0}$ 是一个非负函数。期望收益 $\mathbb{E}[g(I_t, a_t)]$ 也是非负的（即使 $g(I_t, a_t)$ 可能为负）

假设（1）看起来似乎是相当严格的。但是，对于几个具体问题，当它们的价值空间的大小不是常数（例如探测最大值问题，参见第 A.3节），可以利用离散技术，可以将价值空间离散化，将其规模缩小到一个常数，同时不会损失太多的收益。假设（2）和（3）对于许多问题都是很自然的。现在，我们阐述我们的主要结果。

定理 A.6：对于任何固定的 $\varepsilon > 0$，如果假设 A.1成立，我们可以在多项式时间 $n^{2^{O(\varepsilon^{-3})}}$ 中找到一个自适应策略，它的期望收益至少为 $\mathsf{OPT} - O(\varepsilon) \cdot \mathsf{MAX}$，这里

$\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}(I, \mathcal{A})$ 是所有初始值 $I \in \mathcal{V}$ 中的上界，其中 $\mathrm{DP}(I, \mathcal{A})$ 是对于初始值 I 的最优自适应策略的期望收益和 $\mathsf{OPT} = \mathrm{DP}(I_1, \mathcal{A})$ 的给定初始值 I_1 的最优自适应策略的预期收益。

通过定理A.6和不同的离散化技术，我们得到第一个用于探测最大值问题（定理A.1），随机二十一点背包问题（定理A.3）和随机目标问题（定理 A.4）的PTAS。定理A.6也可以用于其他问题，例如探测前 k 大和问题（请参见第4.5.1节），Pandora's Box 问题（请参见第4.5.4节）和 SKP（参见第5.4节）。这项工作发表在第 45 届国际自动机，语言和编程国际学术讨论会上（ICALP2018）[24]。

A.2　主要技术

对于随机动态规划，假设系统初始状态为 $s_1 = (I_1, \mathcal{A})$，当我们选择一个项目 $a_1 \in \mathcal{A}$，系统就产生收益 $g(I_1, a_1)$，并转移到下一状态 $s_2 = (I_2, \mathcal{A}_2)$，其中 I_2 遵循分布 $f(I_1, a_1)$，$\mathcal{A}_2 = \mathcal{A} \setminus a_1$，这个过程迭代产生一个随机序列

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

总共收益可以累计 T 步[①]，这里的目标是找到一个使收益总额最大的策略。在形式化上，我们想求解如下方程：

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

根据 Bellman 方程[2]，对于每个初始状态 $s_1 = (I_1, \mathcal{A})$，最优解 $\mathrm{DP}^*(s_1)$ 由函数 $\mathrm{DP}_1(I_1, \mathcal{A})$ 计算，这里 DP_1 是由 $\mathrm{DP}_{T+1}(I_{T+1}) = h(I_{T+1})$ 和递归函数：

$$\mathrm{DP}_t(I_t, \mathcal{A}_t) = \max_{a_t \in \mathcal{A}_t} \mathbb{E}\Big[\mathrm{DP}_{t+1}(f(I_t, a_t), \mathcal{A}_t \setminus a_t) + g(I_t, a_t)\Big], \quad t = 1, \ldots, T. \qquad \text{(A-3)}$$

当价值空间和项目空间是有限的，并且这里的期望是可以计算时，这个递归函数产生的算法可以用来计算最优值。然而，由于状态空间 $\mathcal{S} = \mathcal{V} \times 2^{\mathcal{A}}$ 是指数级大的，所以这个算法需要指数时间。由于这个模型可以归纳的几个随机优化问题

① 如果步骤小于 T，我们可以使用一些特殊项目来填充。这个项目对于任何值 $I \in \mathcal{V}$，满足 $f(I, a) = I$ 和 $g(I, a) = 0$。

是（或相信为）#P-难或者 PSPACE-难，我们更关注于多项式时间可证明性能保证的近似算法。

对于随机动态规划，最优自适应策略 σ 可以表示为决策树 $\mathcal{T}$（更多细节参见第3.5节）。最优策略的决策树可能是指数级大小，并且非常复杂的，甚至不可能在多项式空间中表示。为了减少空间，我们专注于一类特殊的策略，称为块自适应策略（block adaptive policy）。块自适应策略的概念首先由 Bhalgat 等[6] 引入，并且进一步被 Li 等[11] 推广到随机背包问题中。据我们所知，这个想法还没有扩展到其他应用。在本文中，我们也利用了块自适应策略的概念，但我们的目标是制定一个总体框架。为此，我们提供了一个通用的块策略模型（参见第A.2.1节）。由于我们需要使用更抽象的动态规划，我们建立的块自适应策略与[6,11] 有所不同。

粗略地讲，在一个块适应策略中，我们每次都会同时选择一组项目，而不是单个项目，这可以显着减少决策树的大小。此外，我们证明存在一个块自适应策略，它可以近似于最佳自适应策略，并且在决策树上只有常数数量的块（常数取决于 ε）。由于对应于块自适应策略的决策树具有常数数量的节点，所以块决策树的所有拓扑结构的数量是常数。固定一个对应于块自适应策略决策树的拓扑结构，我们仍需要决定要放置在每个块中的项目的子集，这可能的选择再次的指数数量的。对于每个块，我们可以为它定义一个签名（signature），并且我们只使用多项式数量的签名。根据签名的定义，两个具有相同签名的子集有大致相同的价值分配。最后，我们证明可以使用动态规划在多项式时间中枚举所有块的签名并找到一个近乎最佳的块自适应策略。这里高层次的思想有点类似于[11] 中的思想，但细节完全不同。因此定理A.6的证明可以总结为如下步骤：

1. 证明对随机动态规划，存在一个块自适应策略，它的期望收益值与最优自适应策略的期望值差距很小，并且这个块自适应策略的决策树上只有常数数量的块（常数取决于 ε）。
2. 固定一个决策树的拓扑结构，证明对于两个不同，但签名相同的方案，它们的期望收益差距同样很小。
3. 枚举所有的决策树的拓扑结构和所有的签名，找到一个近似最优的块自适应策略。

在中文论文中，我们主要从高层次方面描述主要方法和技术，忽略一些技术细节，具体的技术细节请参考相应的英文原文。

A.2.1　块自适应策略

这节我们来介绍块自适应策略（block adaptive policy）。由于对应于最优策略的决策树可能是指数级大的，并且非常复杂，我们才考虑块自适应策略，这个概念最初由 Bhalgat 等[6] 引入。我们的结构与[6,11] 有所不同，在这里，我们需要为每个块定义一个顺序，并引进近似区块策略的概念。

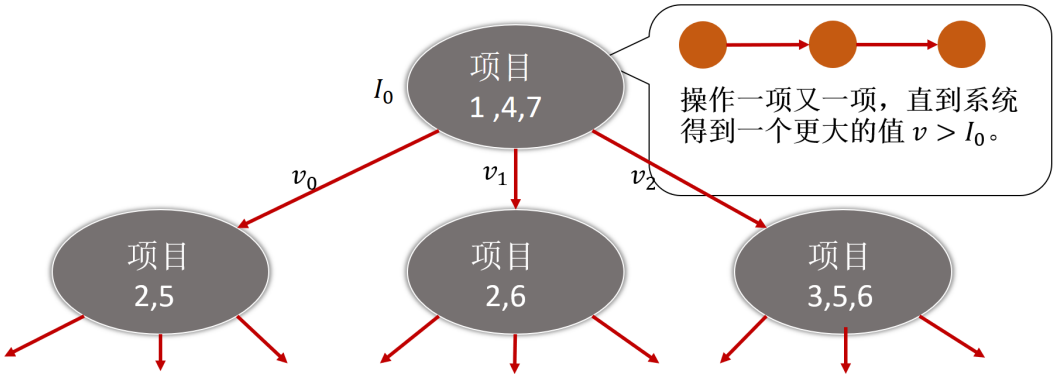

图 A.5　块自适应策略

这里，块自适应策略 $\hat{\sigma}$ 也可以表示成决策树 $\mathcal{T}_{\hat{\sigma}}$（见图A.5）。每一个块是一组项目，树上的每个节点都是一个块。当操作一个块时块，我们为块中的项目选择任意顺序 φ，根据 φ 的顺序，我们逐个操作项目，直到我们系统转移到一个更大的价值，或者块中的所有项目都被操作，系统的价值没有改变（从假设 A.1知道系统的价值不递减），然后我们访问与价值相对应的子块。

A.2.2　创建块自适应策略

在本节中，我们证明存在一个近似于最优自适应策略的块自适应策略。为了证明这一点，从最优（或接近最优）的自适应策略 σ 出发，我们构造了一个块自适应策略 $\hat{\sigma}$，并且它满足了一些很好的属性，还能获得与 σ 几乎一样多的收益。因此，我们将搜索限制在块自适应策略中就足够了。该结构与[11] 中的结构相似。这里的符号定义参考第3.5节。

引理 A.1：　一个最优的策略 σ 可以转换成一个块自适应策略 $\hat{\sigma}$，其期望收益为 $\mathbb{P}(\hat{\sigma})$ 至少为 $\mathrm{OPT} - O(\varepsilon) \cdot \mathrm{MAX}$。此外，这个块自适应策略 $\hat{\sigma}$ 满足性质 (P1) 和 (P2)：

(P1) 每个有多于一个项目的块 M 满足 $\mu(M) \le \varepsilon^2$。

(P2) 在决策树的任何根到叶的路径上至多有 $O(\varepsilon^{-3})$ 块。

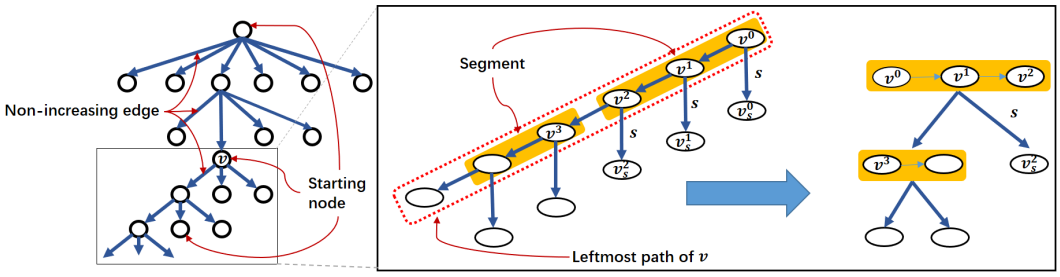

图 A.6　决策树与自适应块策略

Proof. 对于决策树 $\mathcal{T}_\sigma$ 的节点 v 和值 $s \in \mathcal{V}$ 中的，我们使用 v_s 来表示 v 的 s-字节点。如果 $I_v = I_u$，我们说边 $e_{v,u}$ 是非增加的（non-increasing），并且定义 v 的最左路径（leftmost path）为，开始于节点 v，结束于叶节点，并且只包含非增加的边。

如果 v 是根节点或者 v 对应于其父节点 v' 的值是增加的（也就是说 $I_v > I_{v'}$），那么我们说节点 v 是一个起始节点（starting node）。对于每个起始节点 v，我们贪心地将 v 的最左边路径分成若干段，这样对于任何两个节点 u, w 在同一个段中和任何值 $s \in \mathcal{V}$ ，满足

$$|\mathbb{P}(u_s) - \mathbb{P}(w_s)| \leq \varepsilon^2 \cdot \mathsf{MAX} \text{ and } \mu(M) \leq \varepsilon^2. \tag{A-4}$$

由于根据引理3.1，(A-4) 中的第二个不等式最多可以产生 $O(\varepsilon^{-3})$ 块。现在关注 (A-4) 中的第一个不等式。固定起始节点 $v (v = v^0)$ 最左边路径 $\mathcal{R}^v = (v^0, v^1, \ldots, v^m)$on $\mathcal{T}_\sigma$，对于每个值 $s \in \mathcal{V}$，我们有

$$\mathsf{MAX} \geq \mathrm{DP}_1(s, \mathcal{A}) \geq \mathbb{P}(v_s^0) \geq \mathbb{P}(v_s^1) \geq \cdots \geq \mathbb{P}(v_s^m) \geq 0.$$

否则，对某些 $i < j \leq m$，如果 $\mathbb{P}(v_s^i) < \mathbb{P}(v_s^j)$，用 $T_{v_s^i}$ 替换子树 $T_{v_s^j}$ 会增加策略 σ 的收益。因此，对于每个特定的大小 $s \in \mathcal{V}$，路径 $\mathcal{R}^v$ 至多被分 ε^{-2} 次。由于 $|\mathcal{V}| = O(1)$，所以最左路径 $\mathcal{R}^v$ 中至多有 $O(\varepsilon^{-2})$ 段。现在，固定一个特定的根到叶的路径，由假设A.1（2），该值不会降低，所以路径上至多有 $|\mathcal{V}| = O(1)$ 个起始节点。因此，(A-4) 中的第一个不等式可以从根到叶路径上产生最多 $O(\varepsilon^{-2})$ 段。总的来说，决策树上的任何根到叶路径上最多有 $O(\varepsilon^{-3})$ 段。

现在，我们准备描述算法，该算法把策略 σ 作为输入，输出块自适应策略 $\hat{\sigma}$。对于每个节点 v，我们用 $\mathsf{seg}(v)$ 表示其段，并使用 $l(v)$ 表示 $\mathsf{seg}(v)$ 中的最后一个节点。在算法6中，我们可以看到策略 $\hat{\sigma}$ 尝试操作的项目集总是对应于原始策略 σ 中的某个实现路径，因此根据我们的分段，性质（P1）和（P2）容易满足。现在，我们证明新策略 $\hat{\sigma}$ 可以获得的预期收益 $\mathbb{P}(\hat{\sigma})$ 至少为 $\mathsf{OPT} - O(\varepsilon^2) \cdot \mathsf{MAX}$。

Algorithm 6 策略 $\hat{\sigma}$

Input: 策略 σ.

1: 我们从 $\mathcal{T}_\sigma$ 的根节点开始
2: **repeat**
3: 假设现在在 $\mathcal{T}_\sigma$ 的节点 v 上，按照之前策略 σ 中的顺序一个一个操作 $\mathsf{seg}(v)$ 中项目直到某个节点 u 产生了一个更大的价值，记做 s.
4: 访问 $l(v)$ 的 s 子节点 $l(v)_s$
5: 如果所有的项目操作完成之后，系统价值不变，则访问 $l(v)_{I_v}$.
6: **until** 到达 $\mathcal{T}_\sigma$ 的叶节点

当段 $\mathsf{seg}(v)$ 中的节点 u 第一次转换为增加值（例如 s）时，我们的算法偏离了策略 σ。在这种情况下，σ 将访问 u 的 s 的子节点 u_s，并遵循 $\mathcal{T}_{u_s}$ 之后。但是我们的算法访问 $l(v)_s$，并且遵循 $\mathcal{T}_{l(v)_s}$。每个此类事件的预期收益差距为

$$\mathbb{P}(u_s) - \mathbb{P}(l(v)_s) \le \varepsilon^2 \cdot \mathsf{MAX},$$

这由于 (A-4) 中的第一个不等式。假设 σ 支付这样的收益损失，并切换到访问 $l(v)_s$，然后，σ 和我们的算法总是停留在同一个节点上。注意在任何根到叶的路径上最多有 $|\mathcal{V}| = O(1)$ 个起始节点，σ 最多支付 $O(1)$ 次收益损失。因此，总利润损失最多为 $O(\varepsilon^2) \cdot \mathsf{MAX}$。证毕！ □

A.2.3 枚举签名

为了搜索（几乎）最优的块自适应策略，我们想列举出所有可能的块决策树的拓扑结构。固定一个决策树的拓扑结构，我们需要决定放在每个块中的项目集合。为此，我们为每个块定义签名（signature），使两个具有相同签名的子集具有大致相同的收益分布，然后，我们可以在多项式时间内枚举所有块的签名，并找到一个近似最优的块自适应策略。具体的，对于每个项目 $a \in \mathcal{A}$ 和价值 $I \in \mathcal{V} = (0, 1, \ldots, |\mathcal{V}| - 1)$, 我们定义 a 在 I 的签名为下面的向量

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

其中

$$\bar{\Phi}_a(I, J) = \left\lfloor \Phi_a(I, J) \cdot \frac{n}{\varepsilon^4} \right\rfloor \cdot \frac{\varepsilon^4}{n} \quad \text{和} \quad \bar{\mathcal{G}}_a(I) = \left\lfloor \mathcal{G}_a(I) \cdot \frac{n}{\varepsilon^4 \mathsf{MAX}} \right\rfloor \cdot \frac{\varepsilon^4 \mathsf{MAX}}{n}$$

对于任意 $J \in \mathcal{V}$。[①] 对于集合块 M，我们定义其在 I 的签名为

$$\mathsf{Sg}_I(M) = \sum_{a \in M} \mathsf{Sg}_I(a).$$

引理 A.2：　给定两个相同拓扑结构的自适应块策略的决策树 $\mathcal{T}_1, \mathcal{T}_2$ （也就是说 $\mathcal{T}_1$ 和 $\mathcal{T}_2$ 是同构的），并且满足性质 (P1) 和 (P2)。如果对于 $\mathcal{T}_1$ 的任何块 M_1 和在 $\mathcal{T}_2$ 的相应块 M_2，满足 $\mathsf{Sg}_I(M_1) = \mathsf{Sg}_I(M_2)$，我们有 $|\widetilde{\mathbb{P}}(\mathcal{T}_1) - \widetilde{\mathbb{P}}(\mathcal{T}_2)| \le O(\varepsilon) \cdot \mathsf{MAX}$。

由于 $|V| = O(1)$，区块所有可能的签名有 $O\left((n/\varepsilon^4)^{|\mathcal{V}|}\right) = n^{O(1)}$ 种，这是关于 n 的多项式大小。通过引理A.1, 每个自适应块策略最多有 $(|\mathcal{V}|)^{O(\varepsilon^{-3})} = 2^{O(\varepsilon^{-3})}$ 个区块，这也是一个常数。

A.2.4　寻找近似最优的块自适应策略

在本节中，我们寻找一个近似最优的块自适应策略，并证明了定理A.6。为此，我们列举决策树的所有拓扑结构以及每个块的所有可能的签名，然后可以通过标准的动态规划来完成。

固定一个树形拓扑 $\mathcal{T}$ 结构，定义配置 C 是一组签名，每个签名对应一个块。假设 t_1 和 t_2 分别为 $\mathcal{T}$ 上的路径和块的数量。我们定义一个向量 $\mathsf{CA} = (u_1, u_2, \ldots, u_{t_1})$，其中 u_j 是 jth 路径上能放的项目数上限。对于每个给定的 $i \in [n], \mathsf{C}$ 和 CA，设置 $\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = 1$ 表示我们可以使用项目子集 $\{a_1, \ldots, a_i\}$ 就可以到达配置 C，并且在每个路径上 j 的项目总数不超过 u_j，否则设置 $\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = 0$。设置初始值 $\mathcal{M}(0, \mathbf{0}, \mathbf{0}) = 1$，我们以 (i, C) 的字典顺序计算 $\mathcal{M}(i, \mathsf{C}, \mathsf{CA})$：

$$\mathcal{M}(i, \mathsf{C}, \mathsf{CA}) = \max\left\{\mathcal{M}(i-1, \mathsf{C}, \mathsf{CA}), \mathcal{M}(i-1, \mathsf{C}', \mathsf{CA}')\right\} \tag{A-5}$$

现在，我们解释如上的递归，在每一步中，我们都应该决定如何将项目 a_i 放置在树 $\mathcal{T}$ 上。请注意，至多有 $t_2 = (|\mathcal{V}|)^{O(\varepsilon^{-3})} = 2^{O(\varepsilon^{-3})}$ 块，因此最多有 2^{t_2} 种可能的放置 a_i 位置。如果没有两个块有祖先后裔关系，且都放置该项目 a_i，称这个布局是有效的。对于 a_i 的可行布局，我们从 C 中减去放置 a_i 块的签名 $\mathsf{Sg}(a_i)$，CA 中放置 a_i 的对应路径减去 1，这样我们得到配置 C' 和 CA'。这里最大值是遍历所有可能的可行布局相对应的 $\mathsf{C}', \mathsf{CA}'$。

① 如果 $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A})$ 是未知的，对于具体问题（例如探测最大值问题），我们可以得到 MAX 的常数近似结果，这对我们来说已经足够了。更一般来，我们可以用二元搜索来得到 MAX 的一个常数近似结果。

$\mathcal{T}$ 所有可能的配置个数为 n^{t_2}，向量 CA 可能总数为 $T^{t_1} \leq n^{t_1} \leq n^{t_2} = n^{t_2}$，这里 T 是系统轮数。对于给定的 $(i, \mathsf{C}, \mathsf{CA})$，可以通过常数时间 $O(2^{t_2})$ 计算所得。因此，给定一个树形拓扑结构，可以在 $O(n^{2^{O(\varepsilon^{-3})}})$ 内找到一个最优的配置。

Proof. (**定理A.6的证明**) 假设 σ^* 最优策略，期望收益为 $\mathbb{P}(\sigma^*) = \mathsf{OPT}$。我们使用上面的动态规划来找到一个近似最优的块自适应策略 σ。通过引理A.1，存在一个块自适应策略 $\hat{\sigma}$，使得

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

由于在算法的某个步骤枚举了 $\hat{\sigma}$ 的配置，我们的动态规划能够找到一个具有相同配置的块自适应策略 σ（相同的树形拓扑和相同的签名）。通过引理A.2，我们有

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

证毕！ □

A.3　探测最大值问题

在本节中，我们演示了我们的框架在探测最大值问题中的应用。定义价值集合 $S = \bigcup_{i\in[n]} S_i$ 其中 S_i 是随机变量 X_i 的支撑集和变量集合 $\mathcal{A} = \{1, 2, \ldots, n\}$。假设价值 I_t 是在时间段 t 内被探测项目的最大值。因此，我们的初始值为 $I_1 = 0$。由于我们可以探测最多 m 物品，我们将回合数设置为 $T = m$。当我们探测一个变量 i 并得到一个独立样本值 X_i，我们有系统动态规划函数：

$$I_{t+1} = f(I_t, i) = \max\{I_t, X_i\}, \quad g(I_t, i) = 0, \text{ 和} h(I_{T+1}) = I_{T+1} \tag{A-6}$$

对于 $I_t \in S$ 和 $t = 1, 2, \ldots, T$。这个模型满足假设 A.1 (2,3)。但是由于价值空间 S 不是常数大小，假设A.1 (1) 不满足。因此我们需要离散化技术。

现在，我们需要使用参数 ε 离散价值空间。首先，我们求解一个探测最大值问题的常数近似解 $\widetilde{\mathsf{OPT}}$，它满足 $\mathsf{OPT} \geq \widetilde{\mathsf{OPT}} \geq (1-1/e)^2\mathsf{OPT}$（这可以通过简单的贪心算法获得，参见[13] 的附录 C）。假设离散随机变量 X 的支撑集为 $S = (s_1, s_2, \ldots, s_l)$，并且 $p_{s_i} = \Pr[X = s_i]$ 的。令阈值 $\theta = \frac{\widetilde{\mathsf{OPT}}}{\varepsilon}$。对于“大”的值 s_i，也就是 $s_i \geq \theta$，设置 $D_X(s_i) = \theta$。对于“小”的值 s_i，也就是 $s_i < \theta$，设置 $D_X(s_i) = \left\lfloor \frac{s_i}{\varepsilon\widetilde{\mathsf{OPT}}} \right\rfloor \varepsilon\widetilde{\mathsf{OPT}}$。我

们使用 $\mathcal{V} = \{0, \varepsilon\widetilde{\mathsf{OPT}}, \ldots, \widetilde{\mathsf{OPT}}/\varepsilon\}$ 来表示离散支撑集。现在，我们来定义支撑集为 $\mathcal{V}$ 的离散随机变量 $\widetilde{X}$。对于“大”的值，我们设定

$$\widetilde{p}_\theta = \Pr[\widetilde{X} = \theta] = \Pr[X \geq \theta] \cdot \frac{\mathbb{E}[X \mid X \geq \theta]}{\theta}. \tag{A-7}$$

在概率总和保持为 1 的约束下，对于‘‘小’’的值 $d \in \mathcal{V} \setminus \{\theta\}$，我们通过设置

$$\widetilde{p}_d = \Pr[\widetilde{X} = d] = \frac{1 - \Pr[\widetilde{X} = \theta]}{\Pr[X < \theta]} \cdot \left(\sum_{s \in S, D_X(s) = d} \Pr[X = s] \right) \leq \sum_{s \in S, D_X(s) = d} \Pr[X = s]. \tag{A-8}$$

虽然上面的离散化很自然，但也有一些技术细节。我们知道如何解决 $\mathcal{V}$ 支持的离散随机变量的问题，但实现的值是在 S。因此，我们需要介绍规范策略（canonical policy），该概念在 Bhalgat 等[6] 介绍随机背包中所引用。该策略根据变量的离散化之后的大小做出决策，而不是其真实大小。更准确地说，当规范策略 $\widetilde{\sigma}$ 探测一个在 S 中样本为 s 的项目 X 时，该策略将根据离散大小 $D_X(s)$ 作出决策。在下面的引理中，我们证明只考虑规范政策就足够了。我们使用 $\mathbb{P}(\sigma, \pi)$ 来表示策略 σ 在分布 π 获得的预期收益。

引理 A.3：　假设 $\pi = \{\pi_i\}$ 为随机变量的分布，$\widetilde{\pi}$ 为分布 π 的离散版本。我们有如下结论：

1. 对于任意的策略 σ，都存在一个规范策略 $\widetilde{\sigma}$ 满足

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}) \geq (1 - O(\varepsilon))\mathbb{P}(\sigma, \pi) - O(\varepsilon)\mathsf{OPT}$$

2. 对于任意的规范策略 $\widetilde{\sigma}$,

$$\mathbb{P}(\widetilde{\sigma}, \pi) \geq \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}).$$

Proof. (**定理**A.1**的证明**) 假设 σ^* 是最优策略，它的期望收益为 $\mathbb{P}(\sigma^*, \pi) = \mathsf{OPT}$。给定一个分布 π，我们计算离散分布 $\widetilde{\pi}$。通过引理A.3 (1), 存在一个规范策略 $\widetilde{\sigma}^*$ 使得

$$\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}) \geq (1 - O(\varepsilon)) \cdot \mathbb{P}(\sigma^*, \pi) - O(\varepsilon)\mathsf{OPT} = (1 - O(\varepsilon))\mathsf{OPT}.$$

现在，我们描述探测最大值问题在离散分布 $\widetilde{\pi}$ 的随机动态规划。设置价值集合 $\mathcal{V} = \{0, \varepsilon\widetilde{\mathsf{OPT}}, \ldots, \widetilde{\mathsf{OPT}}/\varepsilon\}$，项目集合 $\mathcal{A} = \{1, 2, \ldots, n\}$。设置 $T = m$ 和 $I_1 = 0$。当

我们探测变量 i，并得到其样本值 X_i，那我们系统动态规划函数为

$$I_{t+1} = f(I_t, i) = \max\{I_t, X_i\}, \quad g(I_t, i) = 0, \text{ 和} h(I_{T+1}) = I_{T+1} \tag{A-9}$$

对于 $I_t \in S$ 和 $t = 1, 2, \ldots, T$。这个模型满足假设 A.1 (1,2,3)。通过定理A.6，我们可以得到一个策略 σ，它的收益至少为

$$\mathsf{OPT}_d - O(\varepsilon^2) \cdot \mathsf{MAX}$$

这里 OPT_d 表示在离散分布 $\widetilde{\pi}$ 下的最优策略的期望值，$\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(\widetilde{\mathsf{OPT}}/\varepsilon, \mathcal{A}) = \widetilde{\mathsf{OPT}}/\varepsilon$。我们可以得到 $\mathsf{OPT}_d \geq \mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}) \geq (1 - O(\varepsilon))\mathsf{OPT}$ 。因此通过引理A.3 (2)，有

$$\mathbb{P}(\sigma, \pi) \geq \mathbb{P}(\sigma, \widetilde{\pi}) \geq \mathsf{OPT}_d - O(\varepsilon^2)\mathsf{MAX} \geq (1 - O(\varepsilon))\mathsf{OPT} - O(\varepsilon)\mathsf{OPT} = (1 - O(\varepsilon))\mathsf{OPT}$$

证毕！ □

A.4　随机目标问题

这一节我们讨论随机目标问题，并证明定理A.4。定义变量集合 $\mathcal{A} = \{1, 2, \ldots, n\}$。设置系统价值 I_t 是在 t 轮已经被选择的随机变量之和。然后我们设置 $T = m$ 和 $I_1 = 0$。当我们探测一个变量 i 并得到一个独立样本值 X_i，我们有系统动态规划函数：

$$I_{t+1} = f(I_t, i) = I_t + X_i, \quad g(I_t, i) = 0, \text{ 和} h(I_{T+1}) = \begin{cases} 1 & \text{if } I_{T+1} \geq \mathbb{T}, \\ 0 & \text{otherwise;} \end{cases} \tag{A-10}$$

对于 $t = 1, 2, \cdots, T$。这满足假设A.1 (2,3)，但是由于价值空间 $\mathcal{V}$ 不是常数，假设 A.1 (1) 不被满足，我们需要离散化。

这里利用第5.4.1节中描述的离散技术。令 $\mathbb{P}(\sigma, \pi, 1)$ 为策略 σ 达到目标的概率，其中 $\pi = \{\pi_i\}$ 表示变量分布集合，1 表示目标。令 $\widetilde{\pi}$ 是 π 的离散版本，我们有以下引理。

引理 A.4：　对于任意的策略 σ，存在一个规范策略 $\widetilde{\sigma}$ 满足

$$\mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, (1 - 2\varepsilon)) \geq \mathbb{P}(\sigma, \pi, 1) - O(\varepsilon).$$

引理 A.5：　对于任意的规范策略 $\widetilde{\sigma}$,

$$\mathbb{P}(\widetilde{\sigma}, \pi, (1-2\varepsilon)) \geq \mathbb{P}(\widetilde{\sigma}, \widetilde{\pi}, 1) - O(\varepsilon).$$

Proof. (**定理**A.4**的证明**) 假设 σ^* 是最优策略，它到达目标的概率为 $\mathbb{P}(\sigma^*, \pi, 1) = \mathsf{OPT}$。给定一个分布 π，我们计算离散分布 $\widetilde{\pi}$。通过引理A.4，存在一个策略 $\widetilde{\sigma}^*$ 使得

$$\mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1-2\varepsilon)) \geq \mathbb{P}(\sigma^*, \pi, 1) - O(\varepsilon) = \mathsf{OPT} - O(\varepsilon).$$

现在，我们为实例 $(\widetilde{\pi}, 1-2\varepsilon)$ 提供一个随机动态规划。定义价值集合 $\mathcal{V} = \{0, \varepsilon^5, 2\varepsilon^5, \ldots, 1\}$，项目集合 $\mathcal{A} = \{1, 2, \ldots, n\}$，$T = m$ 和 $I_1 = 0$。当我们探测一个变量 i 并得到一个独立样本值 X_i，我们有系统动态规划函数：

$$I_{t+1} = f(I_t, i) = \min\{1, I_t + X_i\}, \quad g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} 1 & \text{if } I_{T+1} \geq 1-2\varepsilon, \\ 0 & \text{otherwise;} \end{cases} \tag{A-11}$$

对于 $I_t \in \mathcal{V}$，$t = 1, 2, \ldots, T$。这满足假设A.1。通过定理A.6，我们找到一个策略 σ，它到达目标的概率 $\mathbb{P}(\sigma, \widetilde{\pi}, 1-2\varepsilon)$ 至少为

$$\mathsf{OPT}_d - O(\varepsilon) \cdot \mathsf{MAX} \geq \mathbb{P}(\widetilde{\sigma}^*, \widetilde{\pi}, (1-2\varepsilon)) - O(\varepsilon) = \mathsf{OPT} - O(\varepsilon),$$

这里 OPT_d 表示在实例 $(\widetilde{\pi}, 1-2\varepsilon)$ 下的最优策略的期望值，和 $\mathsf{MAX} = \max_{I \in \mathcal{V}} \mathrm{DP}_1(I, \mathcal{A}) = \mathrm{DP}_1(1, \mathcal{A}) = 1$。通过引理A.5，有

$$\mathbb{P}(\sigma, \pi, (1-4\varepsilon)) \geq \mathbb{P}(\sigma, \widetilde{\pi}, (1-2\varepsilon)) - O(\varepsilon) \geq \mathsf{OPT} - O(\varepsilon),$$

证毕！ □

A.5　随机二十一点背包问题

在本节中，我们考虑随机二十一点背包问题。定义项目集 $\mathcal{A} = \{1, 2, \ldots, n\}$。在 t 轮，定义系统价值 $I_t = (I_{t,1}, I_{t,2})$，其中 $I_{t,1}, I_{t,2}$ 是背包中物品的总大小和总价值。我们设置 $T = n$ 和 $I_1 = (0, 0)$。当我们将一个项目 i 插入背包并得到其实际大小 s_i，

我们定义系统动态规划函数为

$$I_{t+1} = f(I_t, i) = (I_{t,1} + s_i, I_{t,2} + p_i), g(I_t, i) = 0, \text{ and } h(I_{T+1}) = \begin{cases} I_{T+1,2} & \text{if } I_{T+1,1} \le \mathbb{C}, \\ 0 & \text{otherwise;} \end{cases} \tag{A-12}$$

对于 $t = 1, 2, \cdots, T$。这满足假设A.1 (2,3)，由于价值空间 $\mathcal{V}$ 不是常数大小，故不满足假设A.1 (1)。因此，我们需要离散技术，具体细节请参考第5.5节。

A.6　相关工作

有一些框架为某些类别的随机动态规划提供近似方案。Shmoys 等[25] 处理的是随机线性规划。Halman 等[26–28] 研究具有标量状态随机离散 DPs，并为其框架设计了 FPTAS。作为应用之一，他们用它来解决随机有序自适应背包问题。相比之下，在我们的模型中，状态空间 $\mathcal{S} = \mathcal{V} \times 2^{\mathcal{A}}$ 指数规模很大，因此不能由以前的框架解决。

在随机探测问题中，如果在承诺模型上没有外部约束，即我们可以探测任何我们需要的项目序列，这就是所谓的贝叶斯在线选择问题（BOSP）。Chawla 等[36] 提出了一种简单的机制，称为连续发布定价机制（SPM），用于贝叶斯单参数拍卖。当内部约束条件是一个拟阵，他们给出了一个近似比率为 $\frac{1}{2}$ 的承诺算法（相比于最佳非承诺策略），Yan[37] 将该比率提高到 $1 - \frac{1}{e}$，这个 $1 - \frac{1}{e}$ 最优的，无法提高，这意味着当没有外部约束时，对于拟阵内部约束的承诺差距最多为 $1 - \frac{1}{e}$。

Esfandiari 等[38] 介绍了另外一种设置，这里变量顺序是随机的，而不是由策略决定。相比顺序是固定的，这里的近似比率可以从 $\frac{1}{2}$ 提高到 $1 - \frac{1}{e}$。他们称这个问题为先知秘书，并证明该比率的上限为 0.75。当变量的分布是未知的，顺序是随机的，这是由 Dynkin 在 20 世纪 60 年代引入的传统的秘书问题[39]。他们设计了一个简单的承诺策略，当 n 很大，其近似比率为 $\frac{1}{e}$。Kleinberg[40] 提供了一个算法，当内部约束是一个 k-均匀拟阵时，近似比率为 $1 - O(\sqrt{1/k})$。然而，对于拟阵的内部约束条件，当前的最佳近似比率是 $O(1/\log\log r)$[41,42]，其中 r 是拟阵的等级。

A.7　总结

在最后，我们总结本文的研究结果和技术，并给出一些在我们研究过程中未解决的问题。在论文的第一部分，我们正式定义了一个基于随机动态规划的模型。这是一个通用的模型，有许多随机优化问题适合这个模型，我们为这个模型设计了一个多项式时间近似方案。

我们还研究了两个重要的随机优化问题，即随机探测问题和随机背包问题，我们为随机探测问题设计近似算法。使用随机动态规划，我们为探测最大值问题设计了一个 PTAS，它提高了之前最好的近似比率 $1-1/e$。我们还为通用拟阵外部约束的承诺模型下的探测最大值问题设计了 PTAS。当外部约束是拟阵，提高非承诺模型下的探测最大值问题近似比率仍然是一个悬而未决的问题。我们还研究随机探测问题的承诺差距和自适应差距，并给出一个上界。

接下来，我们重点研究随机背包问题及其变体：随机二十一点背包问题和随机目标问题。使用随机动态规划和离散化技术，如果允许放松容量或目标，我们为他们设计一个 PTAS。在不放宽容量的情况下提高随机背包问题近似比例仍然是一个悬而未决的问题。

个人简历、在学期间发表的学术论文与研究成果

个人简历

1992 年 06 月 13 日出生于湖南省宁乡县。

2009 年 9 月考入清华大学交叉信息院计算机科学与技术专业，2013 年 7 月本科毕业并获得工学学士学位。

2013 年 9 月免试进入清华大学交叉信息院攻读博士学位至今。

发表的学术论文

[1] H. Fu, G.-N. Han, On t-extensions of the Hankel determinants of certain automatic sequences, Theoretical Computer Science, 562, 2015, pp. 46-56.

[2] H. Fu, G.-N.Han, Computer assisted proof for Apwenian sequences, The 41st International Symposium on Symbolic and Algebraic Computation (ISSAC2016), Waterloo, Ontario, Canada, 2016

[3] H. Fu, J. Li, P. Xu, A PTAS for a Class of stochastic dynamic programs, The 45th International Colloquium on Automata, Languages, and Programming (ICALP2018)