Fully Polynomial Time Approximation Schemes for Stochastic Dynamic Programs

Nir Halman*† Diego Klabjan*‡ Chung-Lun Li§ James Orlin¶†
David Simchi-Levi*†

E-mail: {halman,jorlin,dslevi}@mit.edu, d-klabjan@northwestern.edu, lgtclli@polyu.edu.hk

Abstract

We develop a framework for obtaining (deterministic) Fully Polynomial Time Approximation Schemes (FPTASs) for stochastic univariate dynamic programs with either convex or monotone single-period cost functions. Using our framework, we give the first FPTASs for several NP-hard problems in various fields of research such as knapsack-related problems, logistics, operations management, economics, and mathematical finance.

1 Introduction

Dynamic Programming (DP). Dynamic Programming is an algorithmic technique used for solving sequential, or multi-stage, decision problems and is a fundamental tool in combinatorial optimization (e.g., [17], Section 2.5 in [3], and Chapter 8 in [30]). A discrete time finite time horizon dynamic program is to find an optimal policy over a finite time horizon that minimizes the average cost. At the beginning of a time period, the state of the system is observed and an action is taken. Based on exogenous stochastic information, the state, and the action, the system incurs a single-period cost and transitions into a new state. The goal is to find a policy that realizes the minimal total expected cost over the entire time horizon.

We can formally model this by means of Bellman's optimality equation. Let $z_t(I_t)$ be the cost-to-go (also known as the value function). The value $z_t(I_t)$ is simply the cost of an optimal policy from time period t to the end of the time horizon, given that at the beginning of time period t the state is I_t. The equation reads

(1.1)
$$z_t(I_t) = \min_{x_t \in \mathcal{A}_t(I_t)} E_{D_t}\{g_t(I_t, x_t, D_t) + z_{t+1}(f_t(I_t, x_t, D_t))\}.$$

Here x_t is the action, $\mathcal{A}_t(I_t)$ is the action set, and D_t is a random variable corresponding to the stochastic exogenous information flow. The random variables are assumed to be independent, and are not necessarily identically distributed. The system dynamics are denoted by function f_t, and the incurred cost is g_t. In our context I_t and x_t are one-dimensional and D_t is a fixed-dimensional vector.

Convex/Monotone DP. We study three special cases of such dynamic programs. In the first case the system dynamics are linear in the state I_t and the action x_t, and the cost function g_t is convex, for every t. Under these assumptions, we show that z_t is a convex function for every t. We call this the *convex case*. In the second case we require that g_t is nonincreasing in I_t and can be expressed as the sum of two functions monotone in x_t. We also require that f_t is nondecreasing in I_t and monotone in x_t, and $\forall I' \leq I$, $\mathcal{A}_t(I') \subseteq \mathcal{A}_t(I)$, for every time period t. In this case, the value function is nonincreasing, and we call it the *nonincreasing case*. The third case, whose conditions are analogous to the nonincreasing case, is called the *nondecreasing case*. We refer to the second and third cases as the *monotone cases*.

Fully Polynomial Time Approximation Schemes (FPTASs). Among algorithms with performance guarantees on the maximum amount of relative error, FPTASs are by far the strongest results. For any given tolerance ϵ, an FPTAS generates a solution with a relative error guaranteed to be no more than ϵ, while the running time of the algorithm is polynomial in $1/\epsilon$ and in the size of the problem. The essence of FPTASs is to use a discrete approximation in which the number of bits of precision used for the cost functions is at most logarithmic in the data and in $1/\epsilon$ (so that the algorithm will be polynomial time), and so that no other data is approximated. It is critical to design the algorithms and the approximations so that small errors at one stage do not turn into large errors at subsequent stages. Early work on FPTASs was pioneered by Horowitz and Sahni [18], Ibarra and Kim [19], and Sahni [28], and since then, the most common techniques for constructing FPTASs are dominance (i.e., omitting states of the DP and actions which are dominated, or approximately dominated, by another state or action) and scaling and rounding the data (see, for example, Section 2.5 in [3] and [17]). Woeginger [32] made a key observation that many FPTASs were designed by modifying DPs, and he designed a framework for deriving FPTASs for deterministic dynamic programs satisfying certain regularity conditions. His framework encompassed results from a dozen of optimization problems, including the knapsack problem, for which the first FPTAS was developed in the seminal work of Ibarra and Kim [19]. At the same time, [32] did not address a number of deterministic problems that were known to have FPTASs,

*Research supported in part by NSF Contracts DMI-0085683 and DMI-0245352, and by NASA interplanetary supply chain management and logistics architecture.

†Massachusetts Institute of Technology, Cambridge, MA

‡Northwestern University, Evanston, IL

§The Hong Kong Polytechnic University

¶Research supported in part by ONR grant N00014-05-1-0165.

including treelike variants of the knapsack problem, problems involving convex or monotone functions, and stochastic optimization problems. Many FPTASs are easily constructed once the key ideas from [19, 28] are understood. But other FPTASs require great care in the algorithm design and analysis. We note that if FPTASs were easily developed whenever they exist, there would have been FPTASs for stochastic optimization prior to our result in 2006 [14].

Our results. In this paper, we introduce a general framework for obtaining FPTASs for stochastic dynamic programs, and show that by making use of this framework, we can construct FPTASs for a number of difficult stochastic and deterministic optimization problems. These problems are all NP-hard; they have no known FPTASs; and they cover a broad range of applications. Our main result is the development of an initial sufficient set of conditions that guarantee the *existence* of an FPTAS, and the *construction* of such an FPTAS. In this way we get a framework for obtaining an FPTAS to *any* stochastic univariate convex or monotone dynamic program (with independent random variables). We show that our framework can handle several well-studied cases of non-independent random variables. We show that it is *not* possible to extend our framework for general non-independent random variables, unless $P = NP$.

Our approach. The standard dynamic programming approach by means of optimality equation (1.1) gives only a pseudo-polynomial time algorithm. The running time of this algorithm is linear in the cardinality of the state and action spaces, which may be exponential in the input size. The main difficulty with dynamic programming is in the fact that all possible states are needed to be considered, which are too many. Over the past 35 years, many problems that can be optimally solved in pseudo-polynomial time via a DP formulation were shown to admit FPTASs using dominance and/or scaling. Our current work uses a different technique, one that is very similar to dominance, but which we feel is better suited for FPTAS development.

In a previous work [14], we have studied a single-item stochastic inventory control problem. In our study, we introduce the notions of K-approximation sets and K-approximation functions. In [14] we "tailor" the approximation sets to the specific functions involved in a certain formulation of the inventory control problem, and provide for it an ad-hoc FPTAS. Our current work makes use of the notion of K-approximation sets and functions as well, but we target at the development of a general framework for FPTAS construction. For this sake we provide a set of general computational rules of K-approximation functions, which we call **"Calculus of K-approximation functions"** (Proposition 4.1). The calculus appears a stronger framework than either scaling or dominance, although we cannot prove this formally.

Applications. Our newly developed framework has numerous applications. We now present eight application problems for which no known FPTAS has been reported in the literature. The first seven fall under the monotone case, while the eighth fall under the convex case. The first three problems are variants of the classical 0/1 knapsack problem:

1. Stochastic ordered adaptive knapsack [9]: The input consists of the knapsack volume B, and a description of n items. The items arrive in a predefined order. Item i has deterministic profit π_i and a stochastic volume v_i, whose distribution is known in advance. The task is to maximize the expected total profit from items successfully placed in the knapsack (i.e., whose total volume does not exceed the knapsack volume). The actual volume of an item is unknown until we instantiate it by attempting to place it in the knapsack. The problem is called adaptive since the decision about item i is made only after knowing the actual volumes of the previous items which were selected.

2. Nonlinear knapsack [16, 27]: The problem is similar to the classical integer knapsack problem. Instead of fixed volumes and profits per item, we are given general nondecreasing volume and profit functions, i.e., placing j items of type i in the knapsack results in total profit $\pi_i(j)$ and requires volume $v_i(j)$. The goal is to maximize the profit from items placed in the knapsack without exceeding its volume. [16] gives an FPTAS for the special case where π_i is concave and v_i is convex for $i = 1, ..., n$. [27] gives an FPTAS for the special case where π_i is general but v_i is linear for $i = 1, ..., n$.

3. Dynamic capacity expansion [29]: This problem is best viewed as a multi-period minimum integer knapsack problem. Given a sequence of demands $d_1, ..., d_t$ and a set of items $\{1, ..., n\}$ of various volumes v_i and costs c_i, we would like to determine a combination of quantities of each of these items to be placed in a knapsack in each time period. Our objective is to satisfy the accumulated demand in minimal cost. [29] gives a pseudo-polynomial time solution for this problem.

4. Time-cost tradeoff machine scheduling [8]: There is a single machine and n jobs $J_1, ..., J_n$. Job J_j has a given due date d_j, a late penalty w_j, a "normal" processing time $\bar{p}_j$, and a nonincreasing resource consumption function ρ_j with $\rho_j(x) = 0$ for any $x \geq \bar{p}_j$. The processing time of J_j, denoted as x_j, is a nonnegative integer decision variable, and it incurs a cost of $\rho_j(x_j)$. All jobs are available for processing at time 0, and job preemption is not allowed. The objective is to determine the job processing times and to schedule the jobs onto the machine so that the total cost is minimized, Cheng *et al.* [8] have considered a special case of this problem in which ρ_j is a linear function.

5. Batch disposal [25]: Consider managing a warehouse where a single truck of capacity Q is available to dispatch the goods. The goods are received randomly based on a distribution known in advance. The question is whether we dispatch the truck and if yes, what is its load. If we use the truck, a fixed cost is incurred in addition to a per-unit cost. The goods that remain in the warehouse incur a per-unit holding cost.

6. Lifetime consumption planning with risk exposure [26]: There is a single consumer who must manage her capital in discrete time periods. She can spend some amount of capital, which is governed by the underlying non-decreasing utility function. The remaining capital yields a stochastic return rate, and, in addition, she receives a fixed amount of wealth in each time period. The problem is to find an optimal consumption strategy.

7. Deterministic and stochastic growth models [1]: Consider a single consumer with initial capital who wishes to manage her capital in order to maximize total utility over a finite discrete time horizon. In each time period the capital grows based on a production function, there is a rate of return $1-\delta$ for initial capital, and a utility function.

8. Cash management [15]: In this problem one needs to manage the cash flow of a mutual fund. In the beginning of each time period the cash balance can be changed by either selling or buying stocks. At the end of each time period the net value of deposits/withdrawals is discovered, and consequently the cash balance of the mutual fund is determined. If the balance is negative, the fund borrows money from the bank. If the balance is positive, than a cost is incurred due to the fact that the fund's money could have been invested elsewhere. This problem fits into the framework when all these costs are convex functions.

Organization of the paper. We state our model and give the sufficient conditions needed for our framework to work in Section 2. In Section 3 we review the notion of K-approximation sets and functions. In Section 4 we develop a theory which links K-approximation sets to dynamic programming. Based upon this theory, we present in Section 5 our main result for Monotone DP, namely an the FPTAS together with its analysis. (For the sake of brevity the FPTAS for Convex DP is omitted in this extended abstract). We consider extensions to random vectors in Section 7. We end with some concluding remarks.

2 Notations and model statement

Let $\mathbb{R}, \mathbb{Q}, \mathbb{Z}, \mathbb{N}$ denote the set of real numbers, rational numbers, integers, and positive integers, respectively. For every Boolean expression X, let δ_X be 1 if X is true and 0 otherwise. For any pair of integers $A < B$, let $[A, ..., B]$ denote the set of integers $\{A, A+1, ..., B\}$. We call $[A, ..., B]$ a *contiguous interval*. For any number $x \in \mathbb{R}$ we let $x^+ = \max\{0, x\}$ and $x^- = \max\{0, -x\}$. For a subset $X \subseteq \mathbb{R}$, we denote by X^+ the set of nonnegative numbers in X, i.e., $X^+ := \{x \in X \mid x \geq 0\}$. The base two logarithm of z is denoted by $\log z$. We use the standard computation model in which the binary size of a positive rational number $\frac{p}{q}$ is $\log p + \log q$, where $p, q \in \mathbb{N}$.

In this paper we deal with rational-valued functions whose domains are finite sets of integers. We formally define the domain of some of these functions. Let X be a finite set of integers. For every $x \in X$, let $Y(x)$ be a finite nonempty set of integers. Let $X \otimes Y := \bigcup_{x \in X}(\{x\} \times Y(x)) \subset \mathbb{Z}^2$, see Figure 1.

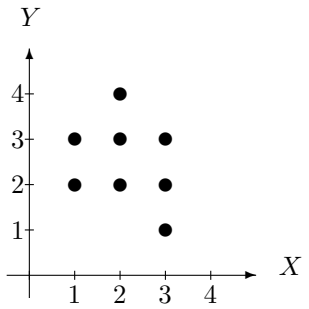

Figure 1: $X \otimes Y$ for $X = \{1,2,3\}$ and $Y(1) = \{2,3\}$, $Y(2) = \{2,3,4\}$, $Y(3) = \{1,2,3\}$.

In this section we review a basic model of decision making under stochastic uncertainty over a finite number of time periods. We consider the following formulation for a finite horizon stochastic dynamic program, as defined in Bertsekas [7]. The model has two principal features: (1) an underlying *discrete time dynamic system*, and (2) a *cost function that is additive over time*. The system dynamics are of the form

$$I_{t+1} = f_t(I_t, x_t, D_t), \quad t = 1, ..., T, \tag{2.2}$$

where

- t is the discrete time index,
- I_t is the state of the system,
- x_t is the action or decision to be selected in time period t,
- D_t is a discrete random variable,
- T is the number of time periods.

The cost function is additive in the sense that the cost incurred at time period t, denoted by $g_t(I_t, x_t, D_t)$, is accumulated over time. Let I_1 be the initial state of the system. Given a realization d_t of D_t, for $t = 1, ..., T$, the total cost is

$$g_{T+1}(I_{T+1}) + \sum_{t=1}^{T} g_t(I_t, x_t, d_t),$$

where $g_{T+1}(I_{T+1})$ is the terminal cost incurred at the end of the process. The problem is to find

$$z^*(I_1) := \min_{x_1,...,x_T} E\left\{ g_{T+1}(I_{T+1}) + \sum_{t=1}^{T} g_t(I_t, x_t, D_t) \right\}, \tag{2.3}$$

where the expectation is taken with respect to the joint distribution of the random variables involved. The optimization is over the actions $x_1, ..., x_T$ which are selected with the knowledge of the current state.

The state I_t is an element of a given *state space* $\mathcal{S}_t$, the action x_t is constrained to take values in a given *action space* $\mathcal{A}_t(I_t)$, and the discrete random variable D_t takes values in a given set $\mathcal{D}_t$. The state space and the action space are one-dimensional. Note that the domain of functions g_t and f_t is $(\mathcal{S}_t \otimes \mathcal{A}_t) \times \mathcal{D}_t$. We state now the well-known DP recursion for this model.

THEOREM 2.1. (THE DP RECURSION [6]) *For every initial state I_1, the optimal cost $z^*(I_1)$ of the DP is equal to $z_1(I_1)$, where the function z_1 is given by the last step of the*

following recursion, which proceeds backward from period T to period 1:

$$z_{T+1}(I_{T+1}) = g_{T+1}(I_{T+1}),$$

$$\text{For all } t = 1, ..., T, \quad z_t(I_t) = \min_{x_t \in \mathcal{A}_t(I_t)} E_{D_t}\left\{g_t(I_t, x_t, D_t) + z_{t+1}(f_t(I_t, x_t, D_t))\right\}, \tag{2.4}$$

where the expectation is taken with respect to the probability distribution of D_t.

Note that assuming $\mathcal{A}_t(I_t) \equiv \mathcal{A}$ and $\mathcal{S}_t \equiv \mathcal{S}$ for every t and $I_t \in \mathcal{S}_t$, the recurrence given in (2.4) yields an exact solution for $z_1(I_1)$ in $O(T|\mathcal{A}||\mathcal{S}|)$ time, which is pseudopolynomial in the input size, i.e., the cardinality of these sets may be exponential in the (binary) input size.

The input data of the problem consists of the number of time periods T, the initial state I_1, an oracle that evaluates g_{T+1}, and for each time period $t = 1, ..., T$, oracles that evaluate the functions g_t and f_t, and the discrete random variable D_t. For each D_t we are given n_t, the number of different values it admits with positive probability, and its support $\mathcal{D}_t := \{d_{t,1}, ..., d_{t,n_t}\}$, where $d_{t,i} < d_{t,j}$ for $i < j$. Moreover, we are also given positive integers $q_{t,1}, ..., q_{t,n_t}$ such that

$$\text{Prob}[D_t = d_{t,i}] = \frac{q_{t,i}}{\sum_{j=1}^{n_t} q_{t,j}}.$$

We define for every $t = 1, ..., T$ and $i = 1, ..., n_t$ the following values:

$p_{t,i} = \text{Prob}[D_t = d_{t,i}]$	probability that D_t has the value $d_{t,i}$ in time period t;
$n^* = \max_t n_t$	maximum number of different values D_t can take over the entire time horizon;
$D^* = \sum_{t=1}^{T} d_{t,n_t}$	maximum possible total value the random variables can take over the entire time horizon;
$Q_t = \sum_{j=1}^{n_t} q_{t,j}$	a common denominator of all the probabilities in time period t;
$M_t = \prod_{j=t}^{T} Q_j$	a common denominator of all the probabilities in all time periods following time period $t-1$;
$M_{T+1} = 1$.	

Note that with the above notation we have

$$E_{D_t}\left\{g_t(I_t, x_t, D_t) + z_{t+1}(f_t(I_t, x_t, D_t))\right\} = \sum_{j=1}^{n_t} p_{t,j}\left(g_t(I_t, x_t, d_{t,j}) + z_{t+1}(f_t(I_t, x_t, d_{t,j}))\right). \tag{2.5}$$

In order to derive an FPTAS for our dynamic program, we need the following conditions to hold.

CONDITION 1. *There exists a constant d such that $\mathcal{S}_{T+1}, \mathcal{S}_t, \mathcal{A}_t \subset \mathbb{Z}$, and $\mathcal{D}_t \subset \mathbb{Z}^d$, for every $t = 1, ..., T$. For any set X among these sets, and any $1 \le k \le |X|$, $\log \max_{x \in X} |x|$ is bounded polynomially by the (binary) input size and the kth largest element in X can be found in time logarithmic in $|X|$.*

Typically the state and action spaces are contiguous intervals, so the kth largest element in X can be found in constant time. However, as stated in Condition 1, we allow a more general setting.

Let $U_{\mathcal{S}}$ denote the least power of two such that $\max_{I_t \in \mathcal{S}_t} |I_t| \le U_{\mathcal{S}}$, $\forall t = 1, ..., T+1$. Similarly, let $U_{\mathcal{A}}$ denote the least power of two such that $\max_{x_t \in \mathcal{A}_t} |x_t| \le U_{\mathcal{A}}$, $\forall t = 1, ..., T$. Condition 1 implies that $\log U_{\mathcal{S}}$ and $\log U_{\mathcal{A}}$ are polynomially bounded by the input size.

CONDITION 2. *For every $t = 1, ..., T+1$, g_t is a function whose values are nonnegative rational numbers, and the binary size of any of its values is polynomially bounded by the (binary) size of the input.*

Let $\bar{U} := \max_t \max_{I \in I_t,\ x \in \mathcal{A}_t(I),\ d \in \mathcal{D}_t} g_t(I, x, d)$ denote the maximal single-period cost value. Algorithm 2 below makes a polynomial number of evaluations of g_t. Hence, by the condition above there exists an integer M such that $\log M$ is bounded polynomially by the input size and Mg_t is an integer for all evaluations performed during the execution of the algorithm. In this way, Condition 2 tells us that $MT\bar{U}$ is an upper bound on the value of (2.3) which is polynomially bounded by the input size.

We also need to impose on $\mathcal{S}_t, \mathcal{A}_t, g_t$ and f_t some properties to make z_t "easier" to approximate. Recall that the states and actions take values from subsets of the totally ordered set $\mathbb{Z}$, so defining a function to be either monotone or convex over these domains makes sense.

CONDITION 3. *At least one of the following cases holds.*

(*i*) **(Convex DP)** *The terminal state space $\mathcal{S}_{T+1}$ is a contiguous interval. For all $t = 1, ..., T+1$ and $I \in \mathcal{S}_t$, the state space $\mathcal{S}_t$ and the action space $\mathcal{A}(I_t)$ are contiguous intervals. g_{T+1} is convex over $\mathcal{S}_{T+1}$. For every $t = 1, ..., T$ and fixed $d \in \mathcal{D}_t$, $g_t(\cdot, \cdot, d)$ is convex over $\mathcal{S}_t \otimes \mathcal{A}_t$, and the transition function is $f_t(I_t, x_t, d) \equiv a(d)I_t + bx_t + c(d)$, where $a(\cdot)$ and $c(\cdot)$ are both functions of d, and $b \in \{-1, 0, 1\}$.*

(*ii*) **(Nondecreasing DP)** *g_{T+1} is nondecreasing. For every $t = 1, ..., T$, and fixed $d \in \mathcal{D}_t$, $g_t(\cdot, \cdot, d)$ and $f_t(\cdot, \cdot, d)$ are nondecreasing in their first variable. $f_t(\cdot, \cdot, d)$ is monotone in its second variable. $g_t(\cdot, \cdot, d) \equiv g_t^a(\cdot, \cdot, d) + g_t^b(\cdot, \cdot, d)$ where $g_t^a(\cdot, \cdot, d), g_t^b(\cdot, \cdot, d)$ are nonnegative functions monotone in their second variable. Moreover, $\forall I', I \in \mathcal{S}_t$ with $I' \le I$, $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$.*

(*iii*) **(Nonincreasing DP)** *g_{T+1} is nonincreasing. For every $t = 1, ..., T$, and fixed $d \in \mathcal{D}_t$, $g_t(\cdot, \cdot, d)$ is nonincreasing in its first variable and $f_t(\cdot, \cdot, d)$ is nondecreasing in its first variable. $f_t(\cdot, \cdot, d)$ is monotone in its second variable. $g_t(\cdot, \cdot, d) \equiv g_t^a(\cdot, \cdot, d) + g_t^b(\cdot, \cdot, d)$, where $g_t^a(\cdot, \cdot, d), g_t^b(\cdot, \cdot, d)$ are nonnegative functions monotone in their second variable. Moreover, $\forall I', I \in \mathcal{S}_t$ with $I' \le I$, $\mathcal{A}_t(I') \subseteq \mathcal{A}_t(I)$.*

We call the DP formulation (2.4) *convex* whenever it satisfies Condition 3(*i*) and *monotone* whenever either one of the last two cases holds.

At a first glace Condition 3, with its three cases, appears quite cumbersome. It is possibly due to our effort to formulate it in a general way. We try to justify this formulation by showing that each of its various cases has applications. Case (i) seems particularly "clumsy". Unfortunately, dropping either one of the conditions about either $\mathcal{A}_t(I)$ or f_t is unlikely, as seen in the theorem below.

THEOREM 2.2. *A convex DP where either $b \notin \{-1, 0, 1\}$, or the action space is not a contiguous interval, does not admit an FPTAS unless P=NP.*

We aim at providing an FPTAS for the optimal solution $z_1(I_1)$. Note that even in the very restrictive case where the number of the states of the system is constant, computing the optimal solution by Theorem 2.1 can take up to $\sum_{t=1}^{T} \max_I |\mathcal{A}_t(I)|$ evaluations of function g_t. When the action spaces are "large" this number of calls can be exponential in the input size. In [32], Woeginger designs a framework for deriving an FPTAS for deterministic DP. Among other assumptions, he requires the cardinality of the action space to be bounded by a polynomial over the binary input size (Condition C.4(ii) in [32]). We do not require this. For this reason our framework, when applied on deterministic DP, is not a special case of Woeginger's framework.

3 K-approximation sets and functions

Since we build our framework upon the notion of approximation sets introduced in [14], we base this section mainly on definitions and results from [14]. The notion of weak K-approximation sets presented here is new. For the ease and brevity of exposition we consider mainly the definitions and results concerning Monotone DP.

Let $K > 1$ be an arbitrary value. Recall that a K-approximation algorithm for a minimization problem guarantees its output to be no more than K times the optimal solution.

DEFINITION 3.1. ([14]) *Let $K \geq 1$ and let $\varphi : D \rightarrow \mathbb{R}^+$ be a function. We say that $\tilde{\varphi} : D \rightarrow \mathbb{R}$ is a* K-approximation function of φ *(K-approximation of φ, in short) if for all $x \in D$ we have $\varphi(x) \leq \tilde{\varphi}(x) \leq K\varphi(x)$.*

For the ease of exposition we assume in the sequel that the domain D of $\varphi(\cdot)$ is the contiguous interval $[0, ..., U]$. All the definitions below and all the results regarding monotone functions can be extended in a natural way to any general totally ordered finite domain D. The results concerning convex functions require D to be a contiguous interval.

In order to get an FPTAS, we consider only a subset of all possible states, whose cardinality is polynomially bounded in the input size. Of course this can only be done by sacrificing accuracy in the final solution.

DEFINITION 3.2. *Let $K > 1$ and let $\varphi : [0, ..., U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function. A* weak K-approximation set of φ *is an ordered set $W = \{i_1 < ... < i_r\}$ of integers satisfying the following two properties:*

1. *$0, U \in W \subseteq [0, ..., U]$;*
2. *for each $k = 1$ to $r - 1$, if $i_{k+1} > i_k + 1$, then $\varphi(i_{k+1}) \leq K\varphi(i_k)$;*

Note that this definition implies that for every nonnegative integer $x \leq U$ there is an element $i_k \in W$ such that $i_k \geq x$ and $\varphi(x) \leq \varphi(i_k) \leq K\varphi(x)$.

DEFINITION 3.3. ([14]) *Let $K > 1$ and let $\varphi : [0, ..., U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function. A K-*approximation set of φ *is a weak K-approximation set of φ where for every $k = 1, ..., r - 1$, there exists a nonnegative integer j such that $i_{k+1} - i_k = 2^j$.*

The additional property of K-approximation sets is technical and is needed for the integrality of $U\hat{\varphi}$ in Proposition 3.1 below.

EXAMPLE 3.1. *Let $U = 11$, $K = 1\frac{1}{2}$, and φ be a function defined for $i = 0, 1, ..., 11$ as $\varphi(i) = \lfloor \frac{i}{2} \rfloor$. It is easy to check that $S_1 := \{i_1 = 0, i_2 = 1, i_3 = 2, i_4 = 3, i_5 = 4, i_6 = 5, i_7 = 7, i_8 = 9, i_9 = 11\}$ is a minimal (by set inclusion) $1\frac{1}{2}$-approximation set of φ, and $W_1 := \{i_1 = 0, i_2 = 1, i_3 = 2, i_4 = 3, i_5 = 4, i_6 = 6, i_7 = 9, i_8 = 11\}$ is a minimal (by set inclusion) weak $1\frac{1}{2}$-approximation set of φ, see Table 1. Note that W_1 is not a K-approximation set of φ.*

Objects / i	0	1	2	3	4	5	6	7	8	9	10	11
$\varphi(i)$	0	0	1	1	2	2	3	3	4	4	5	5
S_1	*	*	*	*	*	*		*		*		*
W_1	*	*	*	*	*		*			*		*

Table 1: S_1 is a $1\frac{1}{2}$-approximation set of φ and W_1 is a weak $1\frac{1}{2}$-approximation set of φ.

We are interested in finding "small" K-approximation sets for φ. Let $\bar{U} \geq \varphi(U)$ be an arbitrary upper bound for the values of φ. The next lemma tells us that there exist "small" K-approximation sets which can be constructed in time logarithmic in U and $\bar{U}$.

LEMMA 3.1. *Let $\varphi : [0, ..., U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function. For every $K > 1$ there exists a K-approximation set S of φ of cardinality $O(\log_K \bar{U} \log U)$. Furthermore, it takes $O((1 + t_\varphi) \log_K \bar{U} \log U)$ time to construct this set, where t_φ is the time needed to evaluate φ. Moreover, for every $K > 1$ there exists a weak K-approximation set W of φ of cardinality $O(\log_K \bar{U})$ which can be constructed in $O((1 + t_\varphi) \log_K \bar{U} \log U)$ time.*

Proof. The Lemma's assertion about S is proved in [14]. It is easy to see that the simple Algorithm 1 below is well defined, and correctly builds a weak K-approximation set W for f of size $O(\log_K \bar{U})$ in time $O((1+t_\varphi)(\log_K \bar{U} \log U)$.

We use K-approximation sets for φ in order to build approximation functions for φ in the following way.

```
1: Function WeakApxSet(K, f)
2: Let W := {U} and x := U
3: while x ≠ 0 do
4:    if f(x) ≤ Kf(0) then
5:       Let y := 0
6:    else
7:       if f(x) > Kf(x − 1) then
8:          Let y := x − 1
9:       else
10:         By bisection let y be the first element in [0, ..., x]
            with f(x) ≤ Kf(y)
11:      end if
12:   end if
13:   Insert y into W
14:   Let x := y
15: end while
16: Return W
```

Algorithm 1: Constructing a weak K-approximation set for f.

DEFINITION 3.4. *Let $K > 1$ and let $\varphi : [0, ..., U] \to \mathbb{Z}^+$ be a nondecreasing function. Let $V \subseteq [0, ..., U]$ be a subset which contains $0, U$. For any integer $0 \le x \le U$ and successive elements $i_k, i_{k+1} \in V$ with $i_k < x \le i_{k+1}$ let*

$$\hat{\varphi}(x) := \frac{i_{k+1} - x}{i_{k+1} - i_k}\varphi(i_k) + \frac{x - i_k}{i_{k+1} - i_k}\varphi(i_{k+1});\ \check{\varphi}(x) := \varphi(i_{k+1}).$$

We call $\hat{\varphi}, \check{\varphi}$ the approximations of φ corresponding to V.

From its definition, $\hat{\varphi}$ is a piecewise linear function, and $\check{\varphi}$ is a step function.

In what follows we assume that U is a power of 2. We denote by V any subset of $[0, ..., U]$ which contains $0, U$, by $\tilde{\varphi}$ an arbitrary K-approximation of φ, and by $\hat{\varphi}, \check{\varphi}$ the approximations as defined above.

Note that φ, $\hat{\varphi}$ and $\check{\varphi}$ coincide on V, so since $0 \in V$, $\min_{x \in [0,...,U]} \varphi(x) = \min_{x \in V} \varphi(x) = \varphi(0)$, while for an arbitrary approximation function $\tilde{\varphi}$, $\min_{x \in [0,...,U]} \tilde{\varphi}(x)$ is not necessarily realized in $x = 0$. Moreover, while $\hat{\varphi}$ and $\check{\varphi}$ are nondecreasing, $\tilde{\varphi}$ may not be such. We consider $\hat{\varphi}$ and $\check{\varphi}$ as special cases of approximation functions of φ.

Also note that if we calculate and store the values of φ on V in advance, then any query for the value of either $\hat{\varphi}(x)$ or $\check{\varphi}(x)$ for any x, can be calculated in $O(\log |V|)$ time. This is done by performing binary search on V to find the consecutive elements $i_k, i_{k+1} \in V$ such that $i_k \le x \le i_{k+1}$.

The next proposition tells us that an approximation $\hat{\varphi}$ of a convex nondecreasing function φ, corresponding to a given K-approximation set S of φ, is a convex K-approximation of φ.

PROPOSITION 3.1. ([14]) *Let $K > 1$, let $\varphi : [0, ..., U] \to \mathbb{Z}^+$ be a nondecreasing function, and let S be a K-approximation set of φ. The approximation $\hat{\varphi}$ of φ corresponding to S is a piecewise-linear nonnegative nondecreasing function satisfying $\hat{\varphi}(x) \le K\varphi(x)$ for any $0 \le x \le U$, and $U\hat{\varphi}$ is an integer-valued function. Moreover, if φ is convex, then $\hat{\varphi}$ is convex and $\hat{\varphi} \ge \varphi$, and therefore $\hat{\varphi}$ is a K-approximation of φ.*

Similarly, the next proposition tells us that an approximation $\check{\varphi}$ of a (general) nondecreasing function φ, corresponding to a given weak K-approximation set W of φ, is a K-approximation of φ:

PROPOSITION 3.2. *Let $K > 1$, let $\varphi : [0, ..., U] \to \mathbb{Z}^+$ be a nondecreasing function, and let W be a weak K-approximation set of φ. The approximation $\check{\varphi}$ of φ corresponding to W is a nonnegative nondecreasing integer-valued step function, and is a K-approximation of φ.*

We extend the definition of K-approximation sets and weak K-approximation sets to nonnegative nonincreasing functions in a natural way.

DEFINITION 3.5. *Let $K > 1$ and let $\varphi : [-U, ..., 0] \to \mathbb{Z}^+$ be a nonincreasing function. Let $\dot{\varphi}(x) := \varphi(-x)$, for x in $[0, ..., U]$. A* K-approximation set of φ *is an ordered set $S = \{i_1 > ... > i_r\}$ of integers such that $= \{-i_1, ..., -i_r\}$ is a K-approximation set of $\dot{\varphi}$. A* weak K-approximation set of φ *is an ordered set $W = \{i_1 > ... > i_r\}$ of integers such that $= \{-i_1, ..., -i_r\}$ is a weak K-approximation set of $\dot{\varphi}$.*

4 From K-approximation sets to dynamic programming

In this section we develop a theory, based upon the notion of K-approximation sets. In the next section we will use this theory to build a framework for approximating the basic stochastic dynamic programming model considered in Section 2.

The following proposition, which we call Calculus of K-approximation Functions, follows directly from the definition of K-approximation functions, and its proof is therefore omitted. (The first two properties can also be derived from [14].)

PROPOSITION 4.1. (**Calculus of K-approximation Functions**) *For $i = 1, 2$ let $K_i > 1$, let $\varphi_i : D \to \mathbb{R}^+$ be an arbitrary function over domain D, and let $\tilde{\varphi}_i : D \to \mathbb{R}$ be a K_i-approximation of φ_i. Let $\psi_1 : D \to D$, and let $\alpha, \beta \in \mathbb{R}^+$. The following properties hold:*

1. *(summation of approximation) $\alpha\tilde{\varphi}_1 + \beta\tilde{\varphi}_2$ is a $\max\{K_1, K_2\}$-approximation of $\alpha\varphi_1 + \beta\varphi_2$,*
2. *(composition of approximation) $\tilde{\varphi}_1(\psi_1)$ is a K_1-approximation of $\varphi_1(\psi_1)$,*
3. *(minimization of approximation) $\min\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $\max\{K_1, K_2\}$-approximation of $\min\{\varphi_1, \varphi_2\}$,*
4. *(approximation of approximation) If $\varphi_2 = \tilde{\varphi}_1$ then $\tilde{\varphi}_2$ is a K_1K_2-approximation of φ_1.*

The Calculus of K-approximation Functions turns out to be very handy. For example, for proving that Lemma 3.1 and Propositions 3.1 and 3.2 hold also for cases where the domain of φ is a general interval $[A, ..., B]$. All we need is to define $\psi(x) = x - A$ and use composition of approximation.

Suppose for a moment that we have at hand an oracle which computes K-approximations for functions g_t, f_t

and z_{t+1} in (2.4). Note that (2.4) consists of minimization, summation, and composition of these functions. The question is, can we then approximate function z_t in polynomial time? The following proposition gives us a partial answer.

PROPOSITION 4.2. (**minimization of summation of composition**) *For $i = 1, 2$ let $K_i > 1$, let $\varphi_i : D \to \mathbb{R}^+$ be an arbitrary function over domain D, let $\tilde{\varphi}_i : D \to \mathbb{R}$ be a K_i-approximation of φ_i and let $\psi_i : D \times E \to D$. Then*

$$\tilde{\varphi}_3(x) := \min_{y \in E}\{\tilde{\varphi}_1(\psi_1(x,y)) + \tilde{\varphi}_2(\psi_2(x,y))\}$$

is a $\max\{K_1, K_2\}$-approximation of $\varphi_3(x) := \min_{y \in E}\{\varphi_1(\psi_1(x,y)) + \varphi_2(\psi_2(x,y))\}$.

The proof of Proposition 4.2 (as well as its name), is due to summation of approximation, minimization of approximation, and composition of approximation (Properties 1-3 in Proposition 4.1).

Since the cardinality of E may be "big", applying Proposition 4.2 and calculating the minimum over all the elements of E may take time exponential in the input size. For this reason we would like to perform the minimization over a subset of E, whose cardinality is at most logarithmic in the cardinality of E. The question now is whether the approximation ratio for the resulting function remains "under control". The following theorem tells us that this is indeed possible, whenever the functions we want to approximate are either convex or monotone.

THEOREM 4.1. *For $i = 1, 2$ let $K_i \geq 1$, $L_i > 1$ and let $\varphi_i : D \to \mathbb{R}^+$ be a function over domain D. Let $\tilde{\varphi}_i : D \to \mathbb{R}$ be a K_i-approximation of φ_i. For every fixed $x \in D$, let $\psi_i : D \times E \to D$ be a function such that $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is either monotone or convex over the totally ordered domain E. If $W_i(x) \subseteq D$ is a weak L_i-approximation set of $\tilde{\varphi}_i(\psi_i(x, \cdot))$, then*

$$\tilde{\varphi}_4(x) := \min_{y \in W_1(x) \cup W_2(x)}\{\tilde{\varphi}_1(\psi_1(x,y)) + \tilde{\varphi}_2(\psi_2(x,y))\}$$

is a $\max\{K_1, K_2, \min\{K_1 L_1, K_2 L_2\}\}$-approximation of $\varphi_4(x) := \min_{y \in E}\{\varphi_1(\psi_1(x,y)) + \varphi_2(\psi_2(x,y))\}$.

The following corollary links the notion of K-approximation sets to dynamic programming. We use it in the next section in order to construct FPTASs for Monotone DP.

COROLLARY 4.1. *Let $K_2 > 1$, $K_2 \geq L_1 > 1$, $L_2 > 1$. In the setting of Theorem 2.1, for fixed $1 \leq t \leq T$, I_t and $d \in D_t$, let $\tilde{z}_{t+1}$ be a K_2-approximation of z_{t+1}. Suppose each of $g_t(I_t, \cdot, d)$ and $\tilde{z}_{t+1}(f_t(I_t, \cdot, d))$ is either monotone or convex function. Let $W_t^g(I_t)$ be a weak L_1-approximation set of $E_{D_t} g_t(I_t, \cdot, D_t)$, $W_{t+1}^z(I_t)$ be a weak L_2-approximation set of $E_{D_t} \tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))$, and $X_t(I_t) = W_t^g(I_t) \cup W_{t+1}^z(I_t)$. Then*

$$\bar{z}_t(I_t) = \min_{x_t \in X_t(I_t)}\{E_{D_t} g_t(I_t, x_t, D_t) + E_{D_t} \tilde{z}_{t+1}(f_t(I_t, x_t, D_t))\}$$

is a K_2-approximation of z_t.

5 An FPTAS for Monotone DP

In this section we present an FPTAS for monotone DP. For ease of exposition, we multiply the functions g_t by $M_1 M$, for every $t = 1, ..., T+1$, so their values are nonnegative integers.

5.1 Algorithm. We consider here nondecreasing DP where $g_t(\cdot, \cdot, d)$ is monotone in its second variable, for every fixed d. The FPTAS for the general case of nondecreasing DP and for nonincreasing DP is similar.

1: **Procedure FPTASNondecreasingDP**
2: Let $K := 1 + \frac{\epsilon}{2T}$ and $z_{T+1} := g_{T+1}$
3: Let W_{T+1} be a weak K-approximation set of z_{T+1}
4: Let $\check{z}_{T+1}$ be the K-approximation of z_{T+1} corr. to W_{T+1}
5: **for** $t := T$ **downto** 1 **do**
6: Let $W_t :=$WeakApxSet'(K,EvalMon$\bar{z}(\cdot, t)$)
7: Modify $\bar{z}_t$ to be nondecreasing on W_t
8: Let $\check{z}_t$ be the K-approximation of $\bar{z}_t$ corresponding to W_t
9: **end for**

Algorithm 2: FPTAS for Nondecreasing DP.

1: **Function EvalMon**$\bar{z}(I, t)$
2: Let $W_t^g :=$ WeakApxSet$(K, E_{D_t} g_t(I, \cdot, D_t))$
3: Let $W_{t+1}^z :=$ WeakApxSet$(K, E_{D_t} \check{z}_{t+1}(f_t(I, \cdot, D_t)))$
4: Let $X_t := W_t^g \cup W_{t+1}^z$
5: Return $\min_{x_t \in X_t}\{E_{D_t} g_t(I, x_t, D_t) + E_{D_t} \check{z}_{t+1}(f_t(I, x_t, D_t))\}$

Algorithm 3: Evaluating $\bar{z}$ for Monotone DP.

We give a few remarks about the algorithm. Our first remark is about Step 7 in Algorithm 2. Since $\bar{z}_t$ is not necessarily monotone, we modify it to be nondecreasing by scanning W_t backwards, and setting $\bar{z}_t(x)$ for every pair of successive elements $x < y$ in W_t to be $\bar{z}_t(x) \leftarrow \min\{\bar{z}_t(x), \bar{z}_t(y)\}$. Considering Step 6 of Algorithm 2, since we have no explicit formula, nor oracle access to $\bar{z}_t$, we calculate it by calling Function WeakApxSet'$(K,$ EvalMon$\bar{z}(\cdot, t))$. We define Function WeakApxSet' to be identical to Function WeakApxSet except for that in the "while loop" of Algorithm 1 the calls to $f(\cdot)$ are replaced with calls to $\min\{f(\cdot), f(x)\}$. (This replacement is necessary since $\bar{z}_t$ is not necessarily monotone.) The identical analysis shows that W_t is a weak K-approximation set of the modified $\bar{z}_t$. Using the monotonicity of the optimal z_t it is easy to see that the modified $\bar{z}_t$ has the same approximation ratio to z_t has the unmodified one. Last remark is about the various resulted weak K-approximation sets. Note that for every t, W_t is a subset of the state space and W_t^g, W_{t+1}^z are subsets of the action space.

5.2 Invariants. There are several properties which remain invariant throughout the execution of our FPTAS.

PROPOSITION 5.1. (INTEGRALITY INVARIANT) *For every $t = 1, ..., T+1$ and I_t, $M_t M z_t(I_t)$ is a nonnegative integer. Moreover, so are $M_t M \check{z}_t$ in Step 8 of Algorithm 2, and the $M_t M \bar{z}_t$ which is calculated by Function EvalMon$\bar{z}$.*

PROPOSITION 5.2. (NONDECREASING INVARIANT) *If Condition 3(ii) is satisfied, then for every $t = 1, ..., T$, function z_t in (2.4) is nondecreasing over $\mathcal{S}_t$. Moreover, so are the modified $\bar{z}_t$ in Step 7 of Algorithm 2, and the $\check{z}_t$ in Step 8 of Algorithm 2.*

5.3 Analysis. Finally, we present our main result.

THEOREM 5.1. (FPTAS FOR MONOTONE DP) *Every monotone dynamic program admits an FPTAS.*

We only provide here the outline of the proof for nondecreasing DP. It suffices to prove that for every $1 > \epsilon > 0$ and every initial state I_1, a $(1+\epsilon)$-approximation of the optimal cost $z^*(I_1)$ is equal to $\bar{z}_1(I_1)$, where $\bar{z}_1(I_1)$ is given in Step 7 in the last iteration of Algorithm 2, and that Algorithm 2 runs in time polynomial in both $\frac{1}{\epsilon}$ and the (binary) input size.

We first show that the algorithm is well defined. The Integrality Invariant, Proposition 5.1, together with the Nondecreasing Invariant, Proposition 5.2, assure us that all the functions for which the algorithm builds weak K-approximation sets are nonnegative nondecreasing integer-valued. In this way the weak K-approximation sets for them are well defined.

The correctness of the algorithm follows from the correctness of the DP (exact) algorithm, Theorem 2.1, and is proved by induction. We observe that when the algorithm first enters the "for loop", all the conditions required by Corollary 4.1 are satisfied with $L_1 = L_2 = K$ and $K_2 = 1$. Hence, $\bar{z}_T(I)$ is a K-approximation of $z_T(I)$. In every additional iteration, by approximation of approximation (Property 4 in Proposition 4.1), the error is multiplied by K, resulting in a total accumulated error of K^T. The choice of $K = 1 + \frac{\epsilon}{2T}$ results in the required $(1+\epsilon)$-approximation by applying the inequality inequality $(1 + \frac{x}{n})^n \leq 1 + 2x$, which holds for every $0 \leq x \leq 1$.

We show now that the algorithm runs in time polynomial in both the input size and $\frac{1}{\epsilon}$. By Lemma 3.1 the first two steps are executed in $O((1 + t_g) \log_K \bar{U} \log U_{\mathcal{S}})$ time, where t_g is the time needed to evaluate g. Clearly, the "for loop" is executed T times, so by Lemma 3.1 Function EvalMon$\bar{z}$ is executed $O(T \log_K \bar{U} \log U_{\mathcal{S}})$ times. The execution time of Function EvalMon$\bar{z}$ is dominated by the time needed to build the various weak K-approximation sets, and is $O((t_g + t_f + t_{\check{z}_{t+1}})n^* \log_K \bar{U} \log U_{\mathcal{A}})$, where $t_g, t_f, t_{\check{z}_{t+1}}$ is the time needed to evaluate $g, f, \check{z}_{t+1}$, respectively. We conclude that the running time of the algorithm is $O((t_g + t_f + \log(\log_K \bar{U} \log U_{\mathcal{S}}))n^* T \log_K^2 \bar{U} \log U_{\mathcal{S}} \log U_{\mathcal{A}})$. Since $\epsilon < 1$, we get that $K < 2$, so $O(\log_K \bar{U}) = O(\frac{1}{K-1} \log \bar{U})$. By replacing K with $1 + \frac{\epsilon}{2T}$, we conclude that the running time of the algorithm is

$$O((t_g+t_f+\log(\frac{T}{\epsilon} \log \bar{U} \log U_{\mathcal{S}}))\frac{n^* T^3}{\epsilon^2} \log^2 \bar{U} \log U_{\mathcal{S}} \log U_{\mathcal{A}}).$$

Since $\bar{U}$ depends linearly on the multiplying factor, which is at most $M_1 M$, and since $\log U_{\mathcal{S}}, \log U_{\mathcal{A}}$, $\log M_1$ and $\log M$ are all polynomially bounded by the input size, the algorithm runs in polynomial time in both the input size and $\frac{1}{\epsilon}$.

6 The stochastic ordered adaptive knapsack problem

In this section we demonstrate the application of our framework using the stochastic ordered adaptive knapsack problem. Although this is the only application presented in the Introduction where the action space is not exponential in the size of the input (it is rather constant), we choose it because it is the most recent application, and appears to draw a lot of interest in the community.

The classical knapsack problem can be formulated as follows. We are given n items, each is associated with integer profit π_i and volume v_i, $i = 1, ..., n$. We are also given an integer knapsack volume B. The goal is to select a subset of this items with maximum profit without exceeding the knapsack volume. Formally, the problem is

$$\begin{array}{lll} \max & \sum_{i=1}^n \pi_i x_i & \\ \text{subject to} & \sum_{i=1}^n v_i x_i \leq B & \\ & x_i \in \{0,1\} & i = 1, ..., n. \end{array} \tag{6.6}$$

In this formulation, x_i indicates the selected items.

This problem is also known as the 0/1 knapsack problem, since each item can appear in the knapsack 0 or 1 times. The 0/1 knapsack problem is NP-hard [22] and a first FPTAS for it was developed by Ibarra and Kim [19].

Many stochastic variants of the knapsack problem have been studied in the literature. We consider the ordered adaptive model which is discussed in [9]. In this model we are given a sequence (i.e., ordered set) of n items. While the profit of item i, π_i, is a constant, its volume, v_i, is a random discrete variable with a known distribution. The distribution of v_i is given as the one of D_t in Section 2. The problem is about which of the items to place in the knapsack. The actual volume of an item is unknown until we instantiate it by attempting to place it in the knapsack. The goal is to maximize the expected profit from items successfully placed in the knapsack. In the ordered *adaptive* model the decision whether to put item i in the knapsack is made after knowing the available capacity of the knapsack after executing the previous $i-1$ decisions. In the ordered *nonadaptive* model all n decisions are made in advance. In [9] the authors give a polynomial time algorithm for the stochastic ordered adaptive knapsack problem. For every $\epsilon > 0$, their algorithm gives a solution whose value is at least the optimal value, at the expense of a slight loss in terms of feasibility, i.e., the total volume of the items placed in the knapsack does not exceed $(1 + \epsilon)B$. While valuable, their algorithm is not in the spirit of FPTASs, in which constraints are treated as "hard" and feasibility is always maintained.

We now present a dynamic program for the problem. Let $z_t(I_t)$ be the expected profit when considering only items t to n, where the remaining available volume in the knapsack is I_t. The recurrence relation is

$$z_t(I_t) = \max\{E_{v_T}\{\pi_t \delta_{v_t \leq I_t} + z_{t+1}((I_t - v_t)^+)\}, z_{t+1}(I_t)\}, \tag{6.7}$$

for $I_t = 1, ..., B$ and $t = 1, ..., n$. The boundary conditions are $z_{n+1} \equiv 0$ and $z_t(0) = 0$ for $t = 1, ..., n$. The optimal solution is $z_1(B)$.

We note that the first term in the set in (6.7) is the outcome of placing item t in the knapsack, and the second term is the outcome of not doing so. In order to show that this problem fits in our framework we need to reformulate (6.7) as a maximization over a function of the action space. It is easy to see that (6.8) below is equivalent to (6.7), and that it is indeed a maximization over a function of the action space.

$$(6.8)\quad z_t(I_t) = \max_{x_t=0,1} E_{v_t}\{x_t\pi_t\delta_{v_t\le I_t} + z_{t+1}((I_t - x_t v_t)^+)\}.$$

We next show that problem (6.8) is a maximization nondecreasing dynamic program which fits into our framework, i.e., a dynamic program satisfying Conditions 1, 2 and a condition which is the counterpart of Condition 3(ii) for maximization problems (i.e., g_{T+1} is nondecreasing. For every $t = 1, ..., T$, and fixed $d \in \mathcal{D}_t$, $g_t(\cdot,\cdot,d)$ and $f_t(\cdot,\cdot,d)$ are nondecreasing in their first variable. $f_t(\cdot,\cdot,d)$ is monotone in its second variable. $g_t(\cdot,\cdot,d) \equiv g_t^a(\cdot,\cdot,d) + g_t^b(\cdot,\cdot,d)$, where $g_t^a(\cdot,\cdot,d), g_t^b(\cdot,\cdot,d)$ are nonnegative functions monotone in their second variable. Moreover, $\forall I', I \in \mathcal{S}_t$ with $I' \le I$, $\mathcal{A}_t(I') \subseteq \mathcal{A}_t(I)$ holds).

We define $D_t = v_t$, $\mathcal{S}_t = [0, ..., B]$, $\mathcal{A}_t = \mathcal{A}_t(I_t) = [0, 1]$ for all I_t and $t = 1, ..., n$. We define the system dynamics function to be $f_t(I_t, x_t, D_t) = \max\{I_t - x_t D_t, 0\}$, $t = 1, ...n$, and the single-period cost functions to be $g_{T+1}(I_{T+1}) = 0$ and $g_t(I_t, x_t, D_t) = x_t\pi_t\delta_{D_t\le I_t}$, $t = 1, ..., n$.

We need to show that Conditions 1, 2 and the maximization counterpart of Condition 3(ii) are satisfied. Let us fix the time period t. Clearly, $\mathcal{D}_t, \mathcal{S}_t, \mathcal{A}_t \in \mathbb{Z}$. The logarithm of the cardinality of the maximum element in $\mathcal{S}_t$ is linear in the input size since the input size of B is $\log B$. Hence Condition 1 is satisfied. Clearly, Condition 2 is satisfied as well. As for the maximization counterpart of Condition 3(ii), note that both g_t and f_t are nondecreasing in I_t, f_t is nonincreasing in x_t, g_t is nondecreasing in x_t, and $\mathcal{A}_t(I_t)$ does not depend on I_t. Therefore, this condition is satisfied as well.

7 Extensions to random vectors and structure of optimal policies

Until now we have assumed that the D_t are independent random one-dimensional variables. We first observe that dealing with multi-dimensional random variable is straightforward. Consider, for example, a more general version of the stochastic adaptive ordered knapsack problem studied in Section 6, where not only the volume V_t is a random variable, but also the profit Π_t. In this case the input includes the mutual distribution of $(V_t, \Pi_t) = D_t$ (for every t we allow V_t and Π_t to be non-independent). The domain of the single-period cost function g_t and the transition function f_t is then 4-dimensional, where $g_t(I_t, x_t, V_t, \Pi_t) = x_t\Pi_t\delta_{V_t\le I_t}$ and $f_t(I_t, x_t, V_t, \Pi_t) = (I_t - x_t V_t)^+$.

In this paper we mainly deal with complexity and computational issues of our framework. A natural issue to explore is the structure of optimal policies for problems in our framework. Recall that a real-valued function $f : \mathbb{R}^d \to \mathbb{R}^+$ is *V-shaped* on its first variable x if it is linear with non-positive slope for $x < 0$, and linear with nonnegative slope for $x > 0$. Also recall that a *limit policy* (r, s) $(-\infty \le r \le s \le \infty)$ is a policy under which whenever the state I falls below r we augment it to r by ordering $r - I$ units, and whenever the state I exceeds s we decrease it to s by discarding of $I - s$ units. (When the state I is between r and s we do nothing.)

THEOREM 7.1. *Suppose a given convex DP satisfies the following. For every time period t and fixed D_t the transition cost function $f_t(\cdot,\cdot,D_t)$ is linear and the absolute values of the coefficients for both its variables are the same. Moreover, the single-period cost function can be expressed as $g_t(I_t, x_t, D_t) = v_t(x_t, D_t) + u_t(f_t(I_t, x_t, D_t), D_t)$, where v_t is V-shaped on x_t, and u_t is convex. Then this convex DP admits an optimal limit policy (r_t, s_t)*

The cash management problem mentioned in the Introduction satisfies the conditions of the theorem above.

8 Concluding remarks and future research

In this paper, we introduce a general framework for obtaining FPTASs for stochastic dynamic programs, and show that by making use of this framework, we can construct FPTASs for a number of difficult stochastic and deterministic optimization problems. These problems are all NP-hard; they have no known FPTASs; and they cover a broad range of applications.

Previous works tried to determine sufficient and necessary conditions for a dynamic program to admit an FPTAS. While most of these works are not constructive, the one of Woeginger [32] gives sufficient conditions for a deterministic dynamic program to admit an FPTAS, and states a clear FPTAS for such a DP. He demonstrates a number of examples that fit into his framework, where all of those examples have already (other) known FPTASs. We note that none of the problems discussed in this extended abstract appears to fit into his framework, either because the problem is stochastic, or it does not satisfy his Condition C4(ii).

It is interesting to relax any of Conditions 1-3. Regarding Condition 1, we would have liked to extend our framework to deal with multi-variate DP, i.e., to allow fixed-dimensional state and action spaces. Unfortunately, this is unlikely to be successful, since it is known that the existence of an FPTAS for the 2-dimensional 0/1 knapsack problem (which can be formulated as a 2-dimensional nondecreasing DP) would imply $P = NP$ (see p. 252 in [23] and the references therein).

We conclude with two complexity remarks. First, Alekhnovich *et al.* [2] present a model for backtracking and dynamic programming. They prove several upper and lower bounds on the capabilities of algorithms in their model, and show that it captures the simple dynamic programming framework of Woeginger [32]. In their paper they question whether their model captures other dynamic programming algorithms. It would be interesting to check the capabilities of our framework in this context.

Second, many #P-complete problems exhibit fully polynomial *randomized* approximation schemes (FPRASs),

for example, counting Hamiltonian cycles in dense graphs [11], counting knapsack solutions [10], counting Eulerian orientations of a directed graph [24], counting perfect matchings in a bipartite graph [20], and computing the permanent [21]. To the best of our knowledge, the only deterministic FPTASs for #P-hard problems known up-to-date and published in the literature are the very recent works of [31, 4, 5, 13], which are all developed by applying methods from statistical physics. Our FPTAS, which uses different methods, is another rare example in the literature of a (deterministic) FPTAS for #P-hard problems. Dyer *et al.* [12] investigate classes of counting problems that are interreducible under approximation-preserving reductions. One of these classes is the class of counting problems that admit (randomized) FPRASs. It is therefore interesting in this context to investigate the class of counting problems that admit FPTASs.

Acknowledgments. The first author would like to thank Oded Goldreich, Sudipto Guha and Asaf Levin for inspiring discussions, and Leslie Ann Goldberg for pointing out selected references.

References

[1] J. Adda and R. Cooper. *Dynamic Economics: Quantitive Methods and Applications*. MIT Press, 2003.

[2] M. Alekhnovich, A. Borodin, J. Buresh-Oppenheim, R. Impagliazzo, A. Magen, and T. Pitassi. Towards a model for backtracking and dynamic programming. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, 2005.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer, 1999.

[4] A. Bandyopadhyay and D. Gamarnik. Counting without sampling. new algorithms for enumeration problems using statistical physics. *To Appear in Random Structures and Algorithms*, 2007.

[5] M. Bayati, D. Gamarnik, D. Katz, C. Nair, and P. Tetali. Simple deterministic approximation algorithms for counting matchings. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, 2007.

[6] R. Bellman. *Applied Dynamic Programming*. Princeton University MIT Press, Princeton, NJ, 1957.

[7] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 1995.

[8] T.C.E. Cheng, Z.-L. Chen, C.-L. Li, and B.M.-T. Lin. Scheduling to minimize the total compression and late costs. *Naval Research Logistics*, 45:67–82, 1998.

[9] B.C. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: the benefit of adaptivity. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[10] M. Dyer. Approximate counting by dynamic programming. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 693–699, 2003.

[11] M. Dyer, A. Frieze, and M. Jerrum. Approximately counting Hamilton paths and cycles in dense graphs. *SIAM Journal on Computing*, 27:1262–1272, 1998.

[12] M. Dyer, L.A. Goldberg, C. Greenhill, and M. Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38:471–500, 2003.

[13] D. Gamarnik and D.Katz. Correlation decay and deterministic fptas for counting list-colorings of a graph. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[14] N. Halman, D. Klabjan, M. Mostagir, J. Orlin, and D. Simchi-Levi. A fully polynomial time approximation scheme for single-item stochastic inventory control with discrete demand. Technical report, Massachusetts Institute of Technology, 2006. Submitted for journal publication.

[15] K. Hinderer and K.-H. Waldmann. Cash management in a randomly varying environment. *European Journal of Operational Research*, 130:468–485, 2001.

[16] D.S. Hochbaum. A nonlinear knapsack problem. *Operations Research Letters*, 17:103–110, 1995.

[17] D.S. Hochbaum. Various notions of approximations: Good, better, best, and more. In D.S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 346–398. PWS Publishing company, 1997.

[18] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23:317–327, 1976.

[19] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[20] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18:1149–1178, 1989.

[21] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

[22] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[23] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springler-Verlag, 2004.

[24] M. Mihail and P. Winkler. On the number of Eulerian orientations of a graph. *Algorithmica*, 16:402–414, 1995.

[25] K.P. Papadaki and W.B. Powell. An adaptive dynamic programming algorithm for stochastic multiproduct batch dispatch problem. *Naval Research Logistics*, 50:742–769, 2003.

[26] E. Phelps. The accumulation of risky capital: A sequential utility analysis. *Econometrica*, 30:729–743, 1962.

[27] H. Safer and J. Orlin. Fast approximation schemes for multi-criteria flow, knapsack, and scheduling problems. Sloan school working paper 3757-95, Massachusetts Institute of Technology, 1995.

[28] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

[29] I. Saniee. An efficient algorithm for the multiperiod capacity expansion of one location in telecommunications. *Operations Research*, 43:187–190, 1995.

[30] V.J. Vazirani. *Approximation Algorithms*. Springler-Verlag, 2001.

[31] D. Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 2006.

[32] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12:57–75, 2000.