SIAM J. OPTIM.
Vol. 29, No. 2, pp. 1131–1163

TOWARD BREAKING THE CURSE OF DIMENSIONALITY: AN FPTAS FOR STOCHASTIC DYNAMIC PROGRAMS WITH MULTIDIMENSIONAL ACTIONS AND SCALAR STATES*

NIR HALMAN† AND GIACOMO NANNICINI‡

Abstract. We propose a fully polynomial-time approximation scheme (FPTAS) for stochastic dynamic programs with multidimensional action, scalar state, convex costs, and linear state transition function. The action spaces are polyhedral and described by parametric linear programs. This type of problem finds applications in the area of optimal planning under uncertainty, and can be thought of as the problem of optimally managing a single nondiscrete resource over a finite time horizon. We show that under a value oracle model for the cost functions this result for one-dimensional state space is the "best possible," because a similar dynamic programming model with two-dimensional state space does not admit a polynomial-time approximation scheme. The FPTAS relies on the solution of polynomial-sized linear programs to recursively compute an approximation of the value function at each stage. Our paper enlarges the class of dynamic programs that admit an FPTAS by showing, under suitable conditions, how to deal with multidimensional action spaces and with vectors of continuous random variables with bounded support. These results bring us one step closer to overcoming the curse of dimensionality of dynamic programming.

Key words. dynamic programming, approximation algorithms, fully polynomial-time approximation scheme, multistage linear programming

AMS subject classifications. 90C27, 90C39, 90C59

DOI. 10.1137/18M1208423

1. Introduction.

A dynamic program (DP) is a mathematical model for sequential decision making. DPs are widely used by the operations research community to model and solve a large variety of problems concerning optimal planning under uncertainty. Unfortunately, DPs are affected by the *curse of dimensionality*—an expression coined by Richard E. Bellman more than 50 years ago [2]—which makes their solution very difficult in practice. There is a large body of work devoted to ways of circumventing the curse, possibly forgoing optimality or approximation guarantees: this is discussed in section 1.1.

This paper deals with a class of discrete-time finite-horizon stochastic DPs characterized by a scalar state and a multidimensional action, where the optimal action at each stage and state can be computed as the solution of a linear program (LP). We now give a more detailed description of the underlying mathematical model. The evolution of the state of the DP is governed by the transition function f_t and the equation $I_{t+1} = f_t(I_t, \vec{x}_t, \vec{D}_t)$, $t = 1, \ldots, T$, where t is the discrete time index, $I_t \in \mathcal{S}_t$ is the state of the system at time t ($\mathcal{S}_t$ is the *state space* at stage t), $\vec{x}_t \in \mathcal{A}_t(I_t)$ is the action or decision to be selected at time t after observing state I_t ($\mathcal{A}_t(I_t)$ is the *action space* at stage t and state I_t), $\vec{D}_t$ is a vector of interstage independent random variables (r.v.s) over the sample space $\mathcal{D}_t$, and T is the number of time periods. The random vector $\vec{D}_t$ represents an exogenous information flow, and can be continuous

*Received by the editors August 20, 2018; accepted for publication (in revised form) January 14, 2019; published electronically April 16, 2019.
http://www.siam.org/journals/siopt/29-2/M120842.html
Funding: The first author was partially supported by Israel Science Foundation grant 399/17.
†Jerusalem School of Business Administration, The Hebrew University of Jerusalem, Mt. Scopus 91905, Israel (halman@huji.ac.il).
‡IBM T. J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598 (nannicini@us.ibm.com).

or discrete. The cost function $g_t(I_t, \vec{x}_t, \vec{D}_t)$ gives the cost of performing action $\vec{x}_t$ from state I_t at time t for each possible realization of $\vec{D}_t$. The cost function g_{T+1}, also called the "terminal cost function," gives the cost for leaving the system in state I_{T+1} at the end of the time horizon under consideration. Costs are accumulated over all time periods: the total incurred cost is equal to $\sum_{t=1}^{T} g_t(I_t, \vec{x}_t, \vec{D}_t) + g_{T+1}(I_{T+1})$. We use $\vec{x}_t$, $\vec{D}_t$ to emphasize vector quantities, whereas the other quantities are scalars. The problem is that of choosing policies $\pi_t : \mathcal{S}_t \to \bigcup_{I_t \in \mathcal{S}_t} \mathcal{A}(I_t)$ in order to determine a sequence of actions $\vec{x}_1 \in \mathcal{A}(I_1), \ \ldots, \ \vec{x}_T \in \mathcal{A}(I_T)$ that minimizes the expectation of the total incurred cost. This problem is called a *stochastic dynamic program*. Formally, we want to determine

$$\text{(DP)} \qquad z^*(I_1) = \min_{\pi_1,\ldots,\pi_T} \mathbb{E}\left[\sum_{t=1}^{T} g_t(I_t, \pi_t(I_t), \vec{D}_t) + g_{T+1}(I_{T+1}) \right]$$
$$\text{subject to } I_{t+1} = f_t(I_t, \pi_t(I_t), \vec{D}_t) \qquad \forall t = 1, \ldots, T,$$

where I_1 is the initial state of the system and the expectation is taken with respect to the joint probability distribution of the random variables $\vec{D}_t$. The seminal result of Bellman [1] is that the solution to this problem is given by a recursion.

THEOREM 1.1. *For every initial state I_1, the optimal value $z^*(I_1)$ of the DP is given by $z_1(I_1)$, where z_1 is the function defined by $z_{T+1}(I_{T+1}) = g_{T+1}(I_{T+1})$, together with the following recursion:*

$$z_t(I_t) = \min_{\vec{x}_t \in \mathcal{A}_t(I_t)} \mathbb{E}\{g_t(I_t, \vec{x}_t, \vec{D}_t) + z_{t+1}(f_t(I_t, \vec{x}_t, \vec{D}_t))\}, \quad t = 1, \ldots, T. \tag{1.1}$$

When the state and action spaces are finite, and the expectations can be computed with a finite process, this recursion gives a finite algorithm to compute the optimal value. However, this algorithm may require exponential time in general. If the state and action spaces are not finite (e.g., when they are intervals or polyhedra, as discussed in the present paper), or if we cannot compute expectations in finite time (e.g., when there are continuous r.v.s that do not possess a closed-form formula for the expectation), computing $z^*(I_1)$ may be intractable.

This paper proposes a fully polynomial-time approximation scheme (FPTAS) for a specific class of DPs. A polynomial-time approximation scheme (PTAS) is an approximation scheme that returns a solution whose cost is at most $(1 + \epsilon)$ times the optimal cost, where $\epsilon > 0$ is a given parameter. A PTAS runs in polynomial time in the (binary) input size. If the algorithm also runs in polynomial time in $1/\epsilon$, we call it an FPTAS. To construct an FPTAS, we must impose additional structure on the DP, because it is known that not all DPs admit a PTAS (see, e.g., [13, Theorem 9.2]). We assume that the state I_t is a scalar, whereas the action $\vec{x}_t$ can be vector valued. We show in section 3 that if the cost functions are only accessible via value oracles, the restriction that I_t is a scalar cannot be relaxed. At each stage, the vector $\vec{D}_t$ is composed of a given number of independent but not necessarily identically distributed r.v.s with bounded support that can have continuous or discrete distribution. The vectors $\vec{D}_t$ are also interstage independent. (If the vectors are not independent, it is known that the DP may be APX-hard and therefore cannot admit an FPTAS; see, e.g., [13, Theorem 10.1, Corollary 10.2].) Furthermore, we assume the following: the transition functions f_t are affine; the cost functions g_t decompose to the sum of a deterministic piecewise linear convex function and a convex function that may depend on the r.v.s via an affine transformation; and the action spaces $\mathcal{A}_t(I_t)$ are polyhedral

sets, described as the feasible region of a parametric LP with right-hand-side (R.H.S.) parameter I_t. A formal description of the assumptions is given in section 2.2. DPs satisfying these assumptions can be seen as multistage stochastic LPs (see, e.g., [4]), with a single variable linking consecutive stages (the state variable) and stochastic R.H.S. vector. The constraint matrix and the matrix linking the state between consecutive stages (also called a "technology matrix") are deterministic. However, unlike the sample average approximation approach typically used in multistage stochastic LPs, our algorithm is deterministic and its running time depends polynomially on the number of stages. Our framework allows some degree of convex nonlinearity in the objective function; see Condition 2.8(ii). One can think of the type of optimization problems addressed here as stochastic resource management problems with a single resource (see section 2.3). An example of a problem that exhibits the required characteristics in the context of energy resource allocation is described in [25]. More applications—management of water in a reservoir, of cement for a construction company, or of the cash reserve for an investment bank—are described in [23].

Organization of the paper. This paper is organized as follows. In the remainder of the introduction, we position our paper as compared to the existing literature, provide a summary of our contributions, and give an overview of the techniques used. In section 2 we define our notation, state our assumptions, and present an example of an application. In section 3 we discuss the necessity of our assumptions and show some related hardness results. Section 4 introduces an algorithm to deal with the issue of number sizes growing too rapidly. Section 5 contains an FPTAS for the sum of r.v.s, and shows how to efficiently compute the expectation in the DP in the presence of continuous r.v.s. Section 6 puts all the building blocks together to design the FPTAS for problem (DP) satisfying our assumptions. Section 7 concludes the paper.

1.1. Relevance to the existing literature. Dynamic programming is an invaluable tool for sequential decision making under uncertainty, and it has received a large amount of attention. DPs are often very difficult to solve in practice; for this reason, several approximate solution methodologies have been developed. In [24] the three *curses of dimensionality* of DPs—the large dimensionality of the state space and of the action space and the difficulty or impossibility of computing expectations—are discussed explicitly.

To deal with these curses, fixed-dimensional stochastic dynamic programs with discrete state and action spaces over a finite horizon were studied in [5]. Assuming that the cost-to-go functions are discrete $L^\natural$-convex (classes of discrete convex functions are discussed later in this subsection), Chen, Dawande, and Janakiraman [5] propose a pseudopolynomial-time approximation scheme that satisfies an arbitrary prespecified additive error guarantee. The proposed approximation algorithm is a generalization of the explicit enumeration algorithm. The main differences between our paper and [5] are (i) the latter considers discrete state and action spaces (as opposed to continuous), (ii) the latter considers fixed dimensional (as opposed to one-dimensional) state spaces, (iii) the latter gives additive (as opposed to relative) error approximation, (iv) the running time of the approximation algorithm in [5] is pseudopolynomial (as opposed to polynomial) in the binary size of the input. Both [5] and the current paper are based on a generalization of the technique of K-approximation sets and functions. Another relevant work in dealing with the curses of dimensionality in options pricing and optimal stopping is [6]. While [6] provides rigorous guarantees on the (additive and, in some cases, relative) approximation error, the assumptions and techniques are very different from those in our paper.

The discipline known as approximate dynamic programming (ADP) strives to deal with the three curses of dimensionality, while hopefully providing theoretical guarantees on the solution quality. Comprehensive references in this area are [3, 24]. In most cases, ADP approaches are content with proving asymptotic convergence to a "best possible" value function, e.g., the best value function approximation that can be obtained from a given set of basis functions [7]. This is considerably different from the approach presented in this paper: we compute an ϵ-approximate solution in polynomial time for any given $\epsilon > 0$.

Our paper gives a framework to construct an FPTAS for a continuous optimization problem. The literature contains only a few general frameworks that yield approximation schemes for nondiscrete optimization problems. The framework in [27] deals with stochastic LPs, and is a randomized scheme, whereas the algorithm given in the present paper is deterministic. The framework of [13] deals with stochastic discrete DPs in which the r.v.s are described explicitly as lists of scenarios $(d, \Pr(D = d))$, and the single-period cost functions possess either a monotone or convex structure. In [17] a subclass of the DP model of [13] is studied in which the single-period cost functions are assumed to possess a convex structure. The paper [17] provides a faster FPTAS from both theoretical worst-case upper bounds and a practical standpoint. An extension of [13] to continuous state and action spaces is given in [15]; however, [15] still deals with scalar state and action spaces, and discrete (scalar) r.v.s. Our paper lifts some of those restrictions.

To construct an approximation scheme for (DP), we study the problem of approximating the cumulative distribution function (CDF) of $\sum_{i=1}^{n} X_i$, where $X_1, \ldots, X_n$ are given r.v.s. This problem is well known to be #P-hard; see, e.g., [19], [14, Theorem 4.1]. It plays a central role in stochastic optimization because many problems in this area inherit #P-hardness from it; see, e.g., [14, 13]. The problem is related to counting knapsack solutions: given a binary knapsack $\sum_{i=1}^{n} a_i x_i \leq b$, a possible way to determine the number of feasible solutions is to define discrete r.v.s X_i equal to a_i or 0 with probability $\frac{1}{2}$ each, and then compute $\Pr(\sum_{i=1}^{n} X_i \leq b)$. Based on the technique for counting knapsack solutions developed in [9] (see also [11]), [20] gives the first FPTAS to approximate the CDF of the sum of r.v.s. The approach presented in this paper to deal with vectors of r.v.s gives, as a by-product, an FPTAS for the same problem, under weaker assumptions; the difference between [20] and our paper is discussed toward the end of the next section.

Another area of relevance to the present paper is that of discrete convexity, a discrete analogue of convexity studied in the discrete optimization literature. The first investigation of a class of discrete functions for which local optimality implies global optimality is due to Miller [21]. Favati and Tardella [8] considered a certain special way of extending functions defined over the integer lattice to piecewise-linear functions defined over the real space, and they introduced the concept of "integrally convex functions." Murota [22] introduced the concepts of "convex extensible," "$L^\natural$-convexity," and "$M^\natural$-convexity," in which "L" stands for "lattice" and "M" stands for "matroid;" see [22, section 1.4]. The relationship between several classes of discrete convex functions is depicted in Figure 1. Halman [10, section 3] shows that separable convex functions in $\mathbb{Z}^d$ admit polylogarithmic-space approximations, while any approximation of a two-dimensional nondecreasing convex function in the sense of Miller may require exponential space in the size of the domain of the function. Of the classes represented in Figure 1, [10] leaves open the approximability status of the class of convex-extensible functions; it is settled in the present paper (see Theorem 1.2).

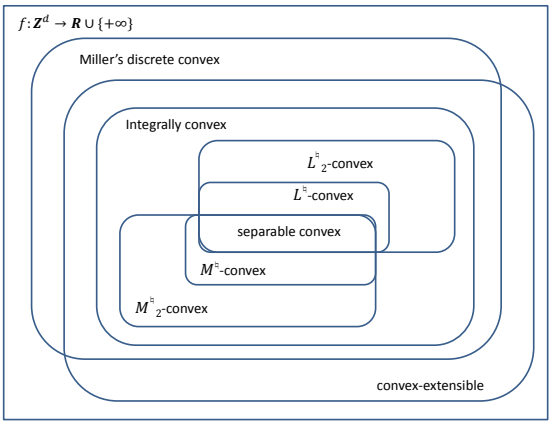

FIG. 1. *Classes of discrete convex functions (adapted from* [22, Figure 1.15]*).*

1.2. Our contribution. Our paper makes progress toward overcoming some of the curses of dimensionality while keeping the strong theoretical guarantee of finding an ϵ-approximate solution. First, the framework presented in this paper allows an action vector of arbitrary dimension, and the running time depends polynomially on such dimension. Second, we show how to handle vectors of continuous r.v.s with bounded support under suitable conditions using an oracle for their CDF, yielding an efficient approximate computation of expected values of the cost functions. We remark that continuous r.v.s are harder to handle than their discrete counterparts, because an exact computation of expected values cannot be carried out in finite time in general.

Thus, we summarize below our contribution regarding the three curses of dimensionality.

(1) We can handle scalar state spaces in polynomial time, as was first shown in [15] under different assumptions. We show that if the cost functions are only accessible via value oracles, our continuous DP model with two-dimensional state space does not admit a PTAS.
(2) We can handle vector action spaces of arbitrary dimension in polynomial time.
(3) We can handle vectors of continuous r.v.s with bounded support in polynomial time. We can also handle vectors of discrete r.v.s, extending the work of [15] on scalar discrete r.v.s to the vector case. A more precise definition of the required conditions is given in section 2.2.

To prove that a DP with multidimensional state space does not admit a PTAS when cost functions are described by an oracle, we prove in section 3.1 the following result (we define discrete convex extensible in section 3.1).

THEOREM 1.2. *There exists a family of two-dimensional positive nondecreasing discrete convex-extensible functions for which any approximation (with approximation ratio below a certain threshold) requires exponential space in the size of the domain of the function, regardless of the scheme used to represent the approximation.*

This result is of independent interest and resolves the approximability status of discrete convex-extensible functions; see Figure 1. Since discrete convex-extensible functions coincide with their convex envelope by definition, the above result implies

that two-dimensional nondecreasing piecewise-linear convex functions do not admit an approximation that is polylogarithmic in the size of the domain (i.e., polynomial in the (binary) size of the domain's description).

To handle vectors of r.v.s, we assume that such vectors appear in the cost and transition functions only via affine transformations. Our FPTAS to approximate the weighted sum of r.v.s has weaker assumptions than [20]. The work of [20] assumes that we are given integer discrete r.v.s and an exact oracle for the corresponding CDFs. It constructs an FPTAS for the sum of r.v.s by discretizing a DP recurrence relation. We relax these assumptions; the approach described in this paper only requires an approximate oracle for the CDF of the r.v.s, and it can also handle noninteger discrete r.v.s and continuous r.v.s with bounded support. The running time is essentially the same. This contribution of our paper extends the applicability of approximation schemes for the sum of r.v.s.

The main constructive result shown in this paper is that a stochastic DP satisfying some structural assumptions (Conditions 2.6–2.8) admits an FPTAS. The approximation scheme is remarkably short and simple: its pseudocode consists of less than ten lines that use three different subroutines. Most of this paper is devoted to their analysis. While the problem that we solve is continuous in nature, the technique used to build the approximation is discrete in nature, and may be useful for the construction of approximation schemes for continuous optimization problems. We remark that our approximation scheme is fully deterministic and does not rely on any form of random sampling, unlike known sample average approximation schemes for multistage optimization problems.

While the work presented in this paper is still far from the generality of ADP approaches, there are several real-world applications that fit into our framework; see section 2.3. Furthermore, our framework provides much stronger theoretical guarantees than ADP: despite the fact that an exact solution to the problem may be impossible to compute in finite time, we determine an ϵ-approximation of the optimum for arbitrary ϵ. To the best of our knowledge, the present paper is the first that uses FPTASs to tackle two curses of dimensionality of DP, i.e., multidimensional action spaces and r.v.s; other FPTASs for DPs mentioned in our literature review do not address these two aspects.

1.3. Techniques used. The approximation scheme discussed in this paper relies on the concept of K-approximation sets and functions, introduced in [14]. Intuitively, a K-approximation function (with $K = 1 + \epsilon > 1$) is a function that has a relative error of at most $K - 1$ (i.e., ϵ) with respect to a given function. A K-approximation set is a set of points that induces a K-approximation function via linear interpolation. A K-approximation set of a function with codomain $[1, U]$ can be constructed with $O(\log_K U)$ points because we can allow the function values to increase by roughly a factor K between two consecutive points. The papers [14, 13] define K-approximation sets for functions over discrete domains. The paper [15] extends this to continuous domains, but uses both relative and absolute error measures. Since we work with the continuous domain, the definition of K-approximation sets used in this paper is essentially the same as that in [15] when only relative error is allowed, i.e., the additive error is fixed to 0. Formal definitions are given in section 2.1.

The approach used in this paper to deal with vector action spaces is based on LP, and for this reason we impose a structure on the problem that allows us to use an exact LP solution methodology as a subroutine. Since our approximation scheme proceeds recursively, we must be careful to ensure that the size of the LPs does not grow too fast. This is achieved with a scaling technique, finding a K-approximation set with

integer domain and codomain values for a suitably scaled version of the function to be approximated; see section 4.

Regarding the approximation for the weighted sum of r.v.s, our construction is based not on counting knapsack solutions as in [20], but rather on constructing compact representations for the value function of a DP. As a result, we obtain a compact representation for the CDF of the sum of r.v.s over the entire domain, rather than its value at a single point. The construction is iterative and amounts to building a K-approximation set for the CDF of a sum of r.v.s, adding one r.v. at a time. This is discussed in section 5.

2. Preliminaries. In this section, we formally introduce our notation and give the necessary definitions, discussing the type of problems that can be handled by the algorithm we propose.

2.1. Definitions. Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}^+$, and $\mathbb{R}$ be the sets of nonnegative integers, integers, rational numbers, nonnegative real numbers, and real numbers respectively. We distinguish between three types of continuous random variables: continuous r.v.s that may have unbounded support and for which $\Pr(X = x) = 0$ for all x, mixed continuous r.v.s that may have $\Pr(X = x) > 0$ for some x, and truncated continuous r.v.s that have bounded support with strictly positive probability on the endpoints only. Vectors are denoted by an overhead arrow, e.g., $\vec{x}$. Subscripts are generally intended to index a stage, e.g., $\vec{x}_t$ is the action vector at stage t rather than the tth component of a vector $\vec{x}$. When it is necessary to index a component of a vector, it will be made explicit, e.g., $\vec{D}_{t,i}$ will be the ith component of the vector $\vec{D}_t$. For $\ell, u \in \mathbb{Z}$, we denote a finite set of the form $\{\ell, \ell+1, \ldots, u\}$ by $[\ell, \ldots, u]$, whereas $[\ell, u]$ denotes a real interval.

Given a real-valued function over a bounded linearly ordered domain D, $\varphi : D \to \mathbb{R}$, such that φ is not identically zero, we define $D^{\min} := \min_{x \in D}\{x\}$, $D^{\max} := \max_{x \in D}\{x\}$, $\varphi^{\min} := \min_{x \in D}\{|\varphi(x)| : \varphi(x) \neq 0\}$, and $\varphi^{\max} := \max_{x \in D}\{|\varphi(x)|\}$. We write $x^+ := \max\{0, x\}$. For any function $\varphi : D \to \mathbb{R}$, t_φ denotes an upper bound on the time needed to evaluate φ on a single point in its domain. A function $\varphi : D \subseteq \mathbb{R} \to \mathbb{R}$ is called Lipschitz continuous with a given Lipschitz constant κ (κ-Lipschitz continuous, for short) if $|\varphi(x) - \varphi(y)| \leq \kappa|x - y|$ for all $x, y \in D$. Given a Lipschitz continuous function $\varphi : D \to \mathbb{R}$, we denote by κ_φ its Lipschitz constant. Let X be a set, and let $Y(x)$ be a set for every $x \in X$. We denote by $X \otimes Y$ the set $\bigcup_{x \in X} Y(x)$. We write $\log z$ to denote $\log_2 z$. When indicating running times of algorithms, we give bounds on the number of operations, which we assume to have unitary cost. For arithmetic operations, the number of bit operations depends on the size of the numbers: because we always ensure that the size of the numbers is polynomially bounded in the (binary) input size, the number of bit operations is at most a polylogarithmic factor larger than the number of arithmetic operations.

As mentioned in section 1.3, the approximation scheme discussed in this paper relies on of K-approximation sets and functions [14]. These concepts are formalized below. The definitions are based on [13, 15].

DEFINITION 2.1. *Let $K \geq 1$ and let $\varphi : D \subset \mathbb{R} \to \mathbb{R}^+$. We say that $\tilde{\varphi} : D \to \mathbb{R}^+$ is a* K-approximation function *of φ (or more briefly, a K-approximation of φ) if for all $x \in D$ we have $\varphi(x) \leq \tilde{\varphi}(x) \leq K\varphi(x)$.*

DEFINITION 2.2. *Let $\varphi : D \subset \mathbb{R} \to \mathbb{R}$ be a convex function. For every finite $E \subseteq D$, the* convex extension of φ induced by E *is the function $\hat{\varphi} : [E^{\min}, E^{\max}] \to \mathbb{R}$ defined as the lower envelope of the convex hull of $\{(x, \varphi(x)) : x \in E\}$.*

DEFINITION 2.3. *Let $\varphi : D \subset \mathbb{R} \to \mathbb{R}$ be nondecreasing. For every finite $E \subseteq D$, the* monotone extension of φ induced by E *is the function $\hat{\varphi} : [E^{\min}, E^{\max}] \to \mathbb{R}$ defined as $\hat{\varphi}(x) := \min\{\varphi(y) : y \in E,\ y \geq x\}$.*

DEFINITION 2.4. *Let $K \geq 1$ and let $\varphi : D \subset \mathbb{R} \to \mathbb{R}^+$ be a convex (respectively, nondecreasing) function. We say that a finite set $W \subset D$ is a* K-approximation set of φ *if the convex extension (respectively, the monotone extension) of φ induced by W is a K-approximation function of φ.*

An algorithm to compute a K-approximation set in polynomial time for a monotone nondecreasing function if the function is defined over integers is described in [13]. If the function is defined over a real interval and is strictly positive, such an algorithm is given in [15] (see the online appendix [16, Appendix A]). We call this algorithm APXSETINC, consistently with the aforementioned papers: whether the discrete or continuous version should be applied will be clear from the context. K-approximation functions can be combined as indicated below.

PROPOSITION 2.5 (calculus of K-approximation functions [13, Proposition 5.1]). *For $i = 1, 2$ let $K_i \geq 1$, 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 $\alpha, \beta \in \mathbb{R}^+$. Then*

1. *φ_1 is a 1-approximation of itself,*
2. *(linearity of approximation) $\alpha + \beta\tilde{\varphi}_1$ is a K_1-approximation of $\alpha + \beta\varphi_1$,*
3. *(summation of approximation) $\tilde{\varphi}_1 + \tilde{\varphi}_2$ is a $\max\{K_1, K_2\}$-approximation of $\varphi_1 + \varphi_2$,*
4. *(composition of approximation) $\tilde{\varphi}_1(\psi_1)$ is a K_1-approximation of $\varphi_1(\psi_1)$,*
5. *(minimum of approximation) $\min\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $\max\{K_1, K_2\}$-approximation of $\min\{\varphi_1, \varphi_2\}$,*
6. *(maximum of approximation) $\max\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $\max\{K_1, K_2\}$-approximation of $\max\{\varphi_1, \varphi_2\}$,*
7. *(approximation of approximation) if $\varphi_2 = \tilde{\varphi}_1$, then $\tilde{\varphi}_2$ is a K_1K_2-approximation of φ_1.*

By the *canonical representation* of an approximation set W of a function φ we mean its representation as a linearly ordered set of points and the corresponding function values. Formally, we write the canonical representation as $\{(x, \varphi(x)) \mid x \in W\}$, represented as an array sorted by its x coordinate. Such a representation corresponds to the data that is normally stored in a computer implementation of the proposed algorithms. Let φ be a convex (respectively, monotone) function, W be a K-approximation set of φ, and let $\hat{\varphi}$ be the convex (respectively, monotone) extension of φ induced by W. We say that $\hat{\varphi}$ is a K-approximation function of φ that is given in the form of a canonical representation. (Note that, by Definition 2.4, $\hat{\varphi}$ is indeed a K-approximation function of φ.) Given the canonical representation and any query point x in the domain of φ, one can compute in $O(\log |W|)$ time the value of $\hat{\varphi}$ at x. Therefore, the canonical representation serves as a value oracle for $\hat{\varphi}$ with query time $O(\log |W|)$. Given a canonical representation, we can recover the approximation set W on which it was constructed in linear time in $|W|$.

We define a routine called COMPRESSINC$(\varphi, [A, B], K)$ that, given a monotone nondecreasing function $\varphi : [A, B] \to \mathbb{R}^+$ and an approximation factor K, constructs a value oracle (in the form of a canonical representation) for a K-approximation function of φ. Details are given in the online appendix [16, Appendix A]. Similar routines will be constructed in the remainder of this paper.

The main idea for the approximation algorithm proposed in this paper is to construct a value function approximation following the backward recursion (1.1). The value function approximation is stored as a K-approximation set. At each stage, we must be able to efficiently compute the expected value appearing in (1.1) and to perform the minimization over the action space. We will show how to achieve these goals under the assumptions detailed in the next section.

2.2. Assumptions. To construct an FPTAS, we require the following conditions to be satisfied.

CONDITION 2.6 (domains). *$\mathcal{S}_{T+1}$ and $\mathcal{S}_t$ for $t = 1, \ldots, T$ are intervals on the real line. $\mathcal{A}_t(I_t) := \{\vec{x}_t : A_t\vec{x}_t \geq \vec{b}_t + \vec{\delta}_{b_t} I_t,\ \vec{x}_t \geq 0\} \subset \mathbb{R}^p$ is a p-dimensional polyhedral set that is expressed as the feasible set of a parametric LP with p variables, where the R.H.S. vector is an affine function of the parameter I_t, $A_t \in \mathbb{Q}^{m\times p}$, and $\vec{b}_t, \vec{\delta}_{b_t} \in \mathbb{Q}^m$ for $t = 1, \ldots, T$.*

CONDITION 2.7 (description of random events). *For every $t = 1, \ldots, T+1$, we have $\vec{D}_t \in \mathbb{R}^\ell$. For every $t = 1, \ldots, T+1$ and $i = 1, \ldots, \ell$, one of the following two conditions holds for the ith random variable $\vec{D}_{t,i}$ of the vector $\vec{D}_t$:*

(i) *$\vec{D}_{t,i}$ is truncated continuous, its support $\mathrm{support}(\vec{D}_{t,i}) = [\vec{D}^{\min}_{t,i}, \vec{D}^{\max}_{t,i}] \subset \mathbb{R}$ is a compact interval of length $n_{t,i} = \vec{D}^{\max}_{t,i} - \vec{D}^{\min}_{t,i}$, and its CDF is Lipschitz continuous;*

(ii) *$\vec{D}_{t,i}$ is discrete, its support $\mathrm{support}(\vec{D}_{t,i}) \subset [\vec{D}^{\min}_{t,i}, \vec{D}^{\max}_{t,i}] \subset \mathbb{R}$ consists of $n_{t,i} < \infty$ elements, and the kth largest element x in the support can be identified in polylogarithmic time in $n_{t,i}$.*

Furthermore, $\Pr(\vec{D}_{t,i} = \vec{D}^{\min}_{t,i}) > 0$, $\Pr(\vec{D}_t = \vec{D}^{\max}_{t,i}) > 0$, and the information about the random events is given via value oracles to the CDF. All $\vec{D}_{t,i}$ are independent.

CONDITION 2.8 (structure of the functions). *For every $t = 1, \ldots, T$, the function $f_t(I_t, \vec{x}_t, \vec{D}_t) : \mathcal{S}_t \otimes \mathcal{A}_t \times \mathcal{D}_t \to \mathcal{S}_{t+1}$ is linear in its variables and is expressed as $f_t(I_t, \vec{x}_t, \vec{D}_t) := \theta^I I_t + \vec{\theta}^x \cdot \vec{x}_t + \vec{\theta}^D \cdot \vec{D}_t$. The function g_t can be expressed as $g_t(I_t, \vec{x}_t, \vec{D}_t) = g^I_t(I_t, \vec{x}_t) + g^D_t(f^g_t(I_t, \vec{x}_t, \vec{D}_t))$, where $g^I_t : \mathcal{S}_t \otimes \mathcal{A}_t \to \mathbb{R}^+$ is a piecewise linear convex function described as the pointwise maximum of a set of $q_t < \infty$ given hyperplanes, and $f^g_t : \mathcal{S}_t \otimes \mathcal{A}_t \times \mathcal{D}_t \to \mathcal{G}_t$ is an affine function expressed as $f^g_t(I_t, \vec{x}_t, \vec{D}_t) := \sigma^I I_t + \vec{\sigma}^x \cdot \vec{x}_t + \vec{\sigma}^D \cdot \vec{D}_t$. One of the following conditions holds.*

(i) *The function $g_{T+1} : \mathcal{S}_{T+1} \to \mathbb{R}^+$ is positive piecewise linear convex over $\mathcal{S}_{T+1}$ with m_{T+1} given breakpoints and slopes and minimum value $g^{\min}_{T+1} > 0$. Furthermore, for all $t = 1, \ldots, T$ the function $g^D_t(\cdot) : \mathcal{G}_t \subset \mathbb{R} \to \mathbb{R}^+$ is a piecewise linear convex function over a compact interval $\mathcal{G}_t$ with m_t given breakpoints and slopes.*

(ii) *The function $g_{T+1} : \mathcal{S}_{T+1} \to \mathbb{R}^+$ is Lipschitz continuous and convex over $\mathcal{S}_{T+1}$ and is described via a value oracle. There is a given positive bound $\tilde{g}^{\min}_{T+1} > 0$ on the minimum value of $g_{T+1}(\cdot)$. Furthermore, for all $t = 1, \ldots, T$ the function $g^D_t(\cdot) : \mathcal{G}_t \subset \mathbb{R} \to \mathbb{R}^+$ is nonnegative Lipschitz continuous and convex over a compact interval $\mathcal{G}_t$ and is described via a value oracle.*

We give next a few remarks about our DP model. In Conditions 2.7(i) and 2.8(ii) the Lipschitz constant need not be given a priori. However, the running time of our algorithms depends polylogarithmically on it. In Condition 2.7(ii), we use $n_{t,i}$ to indicate either the size of the support in terms of interval length for continuous random variables or the cardinality of the support for discrete random variables: this allows us to write the running time in a more compact way. An example of a truncated

continuous r.v. satisfying Condition 2.7 is that of a Gaussian r.v. Y with given mean and standard deviation, clipped to a bounded interval: for example, if Y represents a demand, then we can truncate it from below at zero, because demand cannot be negative ($\Pr(X = 0) = \Pr(Y < 0)$), and we can truncate it from above at some positive order capacity u, because all demand above this value is treated identically ($\Pr(X = u) = \Pr(Y > u)$). Truncated mixed r.v.s can in principle be handled under assumptions similar to Condition 2.7, but their treatment is cumbersome and is not pursued here. We remark that the implicit model for the description of the r.v.s, as stated in Condition 2.7, is both more general and more compact than listing all possible values in the support of the r.v.s and the corresponding probabilities. The latter approach, while simpler and more commonly employed, would only be viable for discrete r.v.s with support of small size.

Condition 2.8(i) is the traditional model, in which the cost functions are analytically known in explicit form. Condition 2.8(ii) is a more flexible model that allows some of the cost functions to be described by value oracles. The FPTAS proposed in this paper is essentially the same under both Conditions 2.8(i) and 2.8(ii), but some of our hardness results assume the oracle model. Studying the oracle model is appealing because positive results that hold in this setting have weak assumptions and can therefore be broadly applied. Value oracles can also allow strong negative results (see section 3).

The input data of the problem include the following: the number of time periods T; the initial state I_1; the constant

$$\gamma = \min\{\Pr(\vec{D}_{t,i}^{\max}), \Pr(\vec{D}_{t,i}^{\min}) \mid t = 1, \dots, T,\ i = 1, \dots, \ell\}$$

(Condition 2.7 implies $\gamma > 0$); the maximum cost in a time period U_g; the maximum Lipschitz constant κ of the cost functions g_t^I, g_t^D (and of the CDF of $\vec{D}_{t,i}$ in the case of Condition 2.7(i)) for $t = 1, \dots, T+1$; the minimum value $\mu := g_{T+1}^{\min}$ of the terminal cost function; the maximum length U_S of the state spaces; the maximum size U_A of the coefficients in the constraint matrix or R.H.S. vector of the LPs describing $A_t(I_t)$; the maximum size U_f of the coefficients in the vectors $\vec{\theta}^x$, $\vec{\sigma}^x$, $\vec{\theta}^D$, $\vec{\sigma}^D$ and of the scalars θ^I, σ^I in the description of f_t and f_t^g; the maximum numbers of pieces $m^* = \max_t m_t$ and $q^* = \max_t q_t$ in the piecewise description of the cost functions; and the maximum size of the support $n^* = \max_{t,i} n_{t,i}$ of the r.v.s. Our algorithms run in polynomial time in $\log I_1$, $\log 1/\gamma$, $\log 1/\mu$, $\log U_g$, $\log \kappa$, $\log U_S$, $\log U_A$, $\log U_f$, m^*, q^*, p, ℓ, and $T \log n^*$. Throughout this paper, the Lipschitz constants of the functions involved need not be known, but the running time of the algorithms depends on them: intuitively, a function with high Lipschitz constant may change quickly, and it may take longer for the algorithms to identify sudden "jumps" in function value with the required precision. The necessity of our assumptions is discussed in section 3.

2.3. An application to resource management under uncertainty. We describe here an example of the type of models allowed under the assumptions given in the previous section. The example involves managing a resource of nondiscrete nature, e.g., liquid natural gas or cement, at a central storage facility. Let $G = (V, E)$ be a directed graph with $|V| = m+1$ nodes, with a special node v_0 that is the central storage facility; for $v \in V$, we denote by $\delta^+(v)$ its set of outgoing edges and by $\delta^-(v)$ its set of ingoing edges. The optimization is performed over a finite time horizon T. The state of the system I_t is the amount of resource available at the central storage facility at the beginning of time period t. At the end of every time period, there is an unknown arrival of the resource encoded by an r.v. $\vec{D}_{t,0}$, e.g., cargo ships (in

which case, the r.v. is discrete). There are m locations $v_1, \ldots, v_m$ that consume the resource but do not have (between time periods) storage capabilities. Each location has unknown demand at time period t, distributed according to a continuous r.v. $\vec{D}_{t,i}$ with $i = 1, \ldots, m$. The demand cannot be transferred to other locations: unsatisfied demand or excess inventory at each of the locations is lost. The resource can be moved from the central storage facility to the m locations, as well as between locations, over a capacitated flow network described by G, potentially with losses $1 - a_{ij}$ over the arcs $(i, j) \in E$ (i.e., the flow variables may appear with coefficients < 1 in the flow balance constraints at some nodes). There is no backlogging at the central storage facility: the inventory must stay nonnegative (but we could easily amend the formulation to allow backlogging). At each time period t, one observes the state of the system I_t and, based on the distribution of the demand in time periods $t, t+1, \ldots, T$, decides on the flow $f_{t,ij}$ over each arc $(i, j) \in E$. At the end of the period a cost is incurred that consists of the transportation cost given by the flow network with unit cost c_{ij} on each arc and shortage (respectively, disposal) cost s_i (respectively, h_i) that is accumulated for every unit by which the demand $\vec{D}_{t,i}$ at location i is not met (respectively, is exceeded). In the final time period $T + 1$, there is a disposal cost h_0 for every unit left in the inventory. We denote by $y_{t,i}$, $i = 0, \ldots, m$, the flow balance (exiting minus entering) at location i; therefore, $y_{t,i}$ is positive if the node sends out more than it receives and negative otherwise. Furthermore, we denote by $w_{t,i}$ an auxiliary variable to encode the cost incurred at the end of time period $t - 1$ at location i, i.e., the maximum between the shortage and disposal costs.

This problem can be formulated as a multistage stochastic linear program as follows:

$$\min \mathbb{E}_{\vec{D}} \left[\sum_{t=1}^{T} \left(\sum_{(i,j)\in E} c_{ij} f_{t,ij}(\vec{D}_t) + \sum_{i=1}^{m} w_{t+1,i}(\vec{D}_t) \right) + h_0 I_{T+1}(\vec{D}_T) \right], \tag{2.1}$$

$$t = 1, \ldots, T, \qquad -I_t(\vec{D}_{t-1}) + I_{t+1}(\vec{D}_t) + y_{t,0}(\vec{D}_t) = \vec{D}_{t,0}, \tag{2.2}$$

$$t = 1, \ldots, T, \ i = 0, \ldots, m, \quad \sum_{(i,j)\in\delta^+(i)} f_{t,ij}(\vec{D}_t) - \sum_{(j,i)\in\delta^-(i)} a_{ji} f_{t,ji}(\vec{D}_t) = y_{t,i}(\vec{D}_t), \tag{2.3}$$

$$t = 1, \ldots, T, \qquad y_{t,0}(\vec{D}_t) \le I_t(\vec{D}_{t-1}), \tag{2.4}$$

$$t = 2, \ldots, T+1, \ i = 1, \ldots, m, \qquad -h_i y_{t-1,i}(\vec{D}_{t-1}) - h_i \vec{D}_{t-1,i} \le w_{t,i}(\vec{D}_t), \tag{2.5}$$

$$t = 2, \ldots, T+1, \ i = 1, \ldots, m, \qquad s_i y_{t-1,i}(\vec{D}_{t-1}) + s_i \vec{D}_{t-1,i} \le w_{t,i}(\vec{D}_t), \tag{2.6}$$

$$t = 1, \ldots, T, \ (i, j) \in E, \qquad f_{t,ij}(\vec{D}_t) \ge 0, \tag{2.7}$$

$$t = 1, \ldots, T, \ i = 1, \ldots, m, \qquad y_{t,i}(\vec{D}_t) \le 0, \tag{2.8}$$

$$I_1(\vec{D}_0) = \bar{I}_1. \tag{2.9}$$

In the above formulation, $\bar{I}_1$ is the initial inventory. The inventory level $I_{t+1}(\vec{D}_t)$ depends on the random variable at time period t, as indicated in constraint (2.2): I_{t+1} is automatically determined after all decisions at time period t are taken and $\vec{D}_t$ is revealed. Similarly, the auxiliary variable $w_{t,i}(\vec{D}_t)$ encodes the cost at location i at the end of the previous time period, because such a cost can only be computed after the demand at a given time period is revealed. $\vec{D}_0$ only serves the purpose of keeping this notation consistent for $I_1(\vec{D}_0)$ and is otherwise not used in the model. Constraint (2.4) implies $I_t(\vec{D}_{t-1}) \ge 0$.

If the $\vec{D}_{t,i}$ are all discrete r.v.s, this problem admits an equivalent deterministic formulation by introducing a copy of the decision variables for each sample path, but the number of sample paths is exponential in T. We now show that a DP formulation of this problem satisfies Conditions 2.6–2.8.

The state variable is I_t, as in our usual notation. The action vector is defined by $\vec{x}_t = ((y_{t,i})_{i=1,\dots,m}, (f_{t,ij})_{(i,j)\in E})$; for simplicity, we use $y_{t,i}$, $f_{t,ij}$ to refer to the variables in the action vector $\vec{x}_t$ corresponding to the variables with the same name in (2.1)–(2.9). The transition function is $f_t(I_t, \vec{x}_t, \vec{D}_t) = I_t - y_{t,0} + \vec{D}_{t,0}$. The action space is

$$\mathcal{A}_t(I_t) = \Big\{ \vec{x}_t : \sum_{(i,j)\in\delta^+(i)} f_{t,ij} - \sum_{(j,i)\in\delta^-(i)} a_{ji} f_{t,ji} = y_{t,i} \ \forall i = 0,\dots,m; \\ y_{t,0} \le I_t;\ f_{t,ij} \ge 0 \ \forall (i,j) \in E;\ y_{t,i} \le 0 \ \forall i = 1,\dots,m \Big\}.$$

The cost functions are $g_t^I(I_t, \vec{x}) = \sum_{(i,j)\in E} c_{ij} f_{t,ij}$, $g_t^D(I_t, \vec{x}, \vec{D}_t) = h_i(-y_{t,i} - \vec{D}_{t,i})^+ + s_i(y_{t,i} + \vec{D}_{t,i})^+$, $g_{T+1}(I_{T+1}) = h_0 I_{T+1}$. It is easy to verify that, with the cost functions defined as given, we obtain an equivalent description to problem (2.1)–(2.9), and Conditions 2.6–2.8(i) are satisfied.

3. Hardness results. We now discuss the necessity of our assumptions. Regarding Condition 2.6, it is an open problem to determine additional assumptions under which one can lift the restriction in our DP model that the state spaces $\mathcal{S}_t$ are intervals on the real line. However, subsection 3.2 shows that constructing an approximation of the value function is difficult already whenever the $\mathcal{S}_t$'s are two-dimensional boxes (Corollary 3.3). We show this by first proving in subsection 3.1 the existence of a class of two-dimensional piecewise linear convex functions that are hard to approximate.

Regarding Condition 2.7, the conditions on truncated r.v.s seem difficult to relax under the oracle model for the CDFs: Proposition 2.6 of [15] shows that a continuous function over a compact interval may not admit an efficient approximation unless its values at the endpoints of the domain are bounded away from zero. This result holds even if the function is monotone and Lipschitz continuous. Hence, approximating the CDF of a continuous r.v. that does not have positive probability mass at the endpoints seems difficult (in [15] it is also shown that for bounded Lipschitz continuous functions we can drop the strict nonnegativity requirement if we allow both relative and absolute error at the same time; however, in this paper we aim for an FPTAS in the usual sense). Regarding Condition 2.8, the nonnegativity requirement on the cost functions g_t cannot be relaxed: in [18] a DP is exhibited that is #P-hard to approximate to any constant factor, and such a problem can be cast into our framework if we allow the cost functions to be of either sign. In other words, the #P-hard DP in [18] satisfies Conditions 2.6–2.8 except for the nonnegativity requirement on g_t. It is an open question as to whether the restriction $g_{T+1}^{\min} > 0$ (or $\tilde{g}_{T+1}^{\min} > 0$ in the setting of Condition 2.8(ii)) can be relaxed to $g_{T+1}^{\min} \ge 0$ (to $\tilde{g}_{T+1}^{\min} \ge 0$ in the setting of Condition 2.8(ii)). The difficulty stemming from $g_{T+1}^{\min} = 0$ is that we would have to keep track of the exact location of the zeros of the value function at each stage.

3.1. Hardness of the approximation of multivariate convex functions. The DP model discussed in this paper naturally yields piecewise linear convex value functions. To extend the framework to a multidimensional state-space setting we

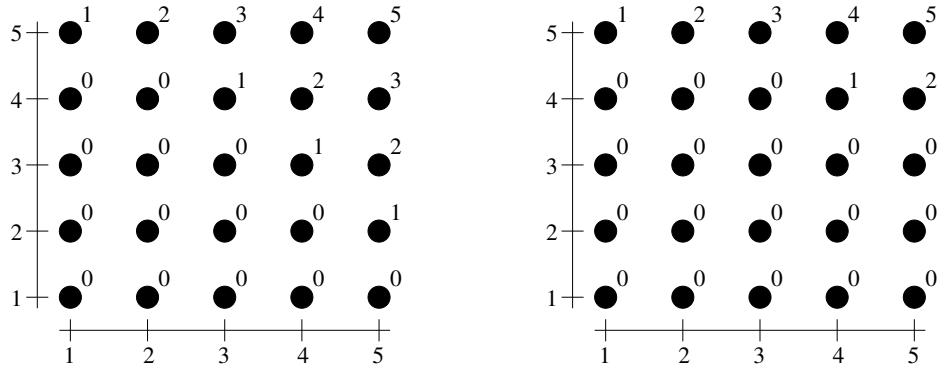

FIG. 2. *Values of the function $\varphi_{0,1,1,1,2}$ (left) and $\varphi_{0,0,0,2,3}$ (right) on the integer grid $[1,\dots,5]^2$, with $A = 0$. At each integer point in the plane, we indicate the corresponding function value next to it.*

would have to efficiently build approximations of such functions. However, the next results show that some piecewise linear convex functions do not admit an efficient approximation. We first show that there is an exponential number of two-dimensional piecewise linear convex functions with distinct values at integer points and use this to prove that approximations of such functions require exponential space in the size of the domain.

THEOREM 3.1. *For any nonnegative integer A, there exist $\Omega(2^{\sqrt{U}})$ distinct nondecreasing piecewise linear convex functions $\varphi : [1,U]^2 \to [A, A+U]$ that attain the value A at distinct sets of points in $[1,\dots,U]^2$, and at least $A+1$ at the other sets of integer points. These functions are described as the pointwise maximum of $O(\sqrt{U})$ hyperplanes.*

Proof. Let

$$\Phi := \left\{ \varphi_{r_1,\dots,r_U} \;\middle|\; r_1,\dots,r_U \in [0,U];\ \sum_{i=1}^{U} r_i = U,\ r_1 \le r_2 \le \cdots \le r_U \right\}$$

be a family of bivariate discrete functions $[1,U]^2 \to [A, A+U]$ defined as follows: for $x, y \in [1,\dots,U]$,

$$\varphi_{r_1,\dots,r_U}(x,y) = \begin{cases} A & \text{if } x = 1,\dots,U - \sum_{i=1}^{y} r_i, \\ A + x - U + \sum_{i=1}^{y} r_i & \text{otherwise.} \end{cases} \tag{3.1}$$

Proposition 3.4 of [10] shows (taking $A = 0$) that any pair of elements of Φ attains the value A at different points on the integer grid. Furthermore, it shows that the number of such functions is equal to the number of partitions of the integer U, which is bounded below by $(H/U)e^{2\sqrt{U}} \ge H2^{\sqrt{U}}$. In Figure 2 we illustrate the shape of $\varphi_{r_1,\dots,r_U}$ by plotting its values for two choices of $r_1,\dots,r_U$.

Let $\mathrm{diff}(r_1,\dots,r_U)$ be the number of distinct values among $r_1,\dots,r_U$. We show next that each function $\varphi_{r_1,\dots,r_U}$ can be described as the pointwise maximum of at most $\mathrm{diff}(r_1,\dots,r_U)+1$ hyperplanes in a such a way that the value of the pointwise maximum on the integer grid matches that of $\varphi_{r_1,\dots,r_U}$. Note that the slopes of

$\varphi_{r_1,\dots,r_U}$ satisfy the following relationships:

$$\varphi_{r_1,\dots,r_U}(x+1,y)-\varphi_{r_1,\dots,r_U}(x,y)=\begin{cases}0 & \text{if } x<U-\sum_{i=1}^{y} r_i,\\ 1 & \text{otherwise,}\end{cases} \tag{3.2}$$

$$\varphi_{r_1,\dots,r_U}(x,y+1)-\varphi_{r_1,\dots,r_U}(x,y) = \begin{cases}0 & \text{if } x\le U-\sum_{i=1}^{y+1} r_i,\\ r_{y+1}+\min\{x-U+\sum_{i=1}^{y} r_i,0\} & \text{otherwise.}\end{cases} \tag{3.3}$$

Furthermore, $\varphi_{r_1,\dots,r_U}(x,U)=A+x$ for all valid choices of $r_1,\dots,r_U$.

Claim. For $1\le k\le U-1$, there exists $S_k:=\{(a,b,c)\in\mathbb{N}\}$, with $|S_k|\le \text{diff}(r_k,\dots,r_U)+1$, such that $\varphi_{r_1,\dots,r_U}(x,y)=\max_{(a,b,c)\in S_k}\{ax+by+c\}$ for all $1\le x\le U$, $y\ge k$.

Proof of the claim (by backward induction). We start with $k=U-1$. Consider the set $S_k:=\{(0,0,A),(1,r_U,-r_UU)\}$, which corresponds to two hyperplanes given in the form $ax+by+c$: one "flat" hyperplane at the value A, and one hyperplane with slope 1 along the x-axis, r_U along the y-axis. Consider the latter hyperplane, which we call H. By the choice of $c=-r_UU$, H interpolates $\varphi_{r_1,\dots,r_U}$ at the points (x,U) for all $1\le x\le U$. Furthermore, by (3.3), H interpolates $\varphi_{r_1,\dots,r_U}$ at the points $(x,U-1)$ for which $x\ge U-\sum_{i=1}^{k} r_i$, because for such points its slope in the y-direction matches the slope of $\varphi_{r_1,\dots,r_U}$. It remains to analyze the points $(x,U-1)$ with $x<U-\sum_{i=1}^{k} r_i$. Let P be the set of such points. For $(x,y)\in P$, $\varphi_{r_1,\dots,r_U}(x,y)$ has value A by definition; see (3.1). Furthermore, $\varphi_{r_1,\dots,r_U}(x,y+1)=\varphi_{r_1,\dots,r_U}(x,U)=x+A<U-\sum_{i=1}^{k} r_i+A=r_U+A$. Then, since H interpolates $\varphi_{r_1,\dots,r_U}$ at $(x,y+1)$, and its slope is r_U in the y-direction, we have that the value of H at (x,y) is equal to $x+A-r_U<A$. But then the maximum of the two hyperplanes in S_k at $(x,y)\in P$ is given by A (i.e., it is given by the "flat" hyperplane), which is exactly the value of $\varphi_{r_1,\dots,r_U}(x,y)$ by (3.1). This concludes the initial step of the induction.

We now need to prove the induction step k, assuming the claim is true for $k+1$. Suppose first that $r_k=r_{k+1}$. In this case, a similar argument to the one above shows that no further hyperplane is needed in S_k, because the slope of $\varphi_{r_1,\dots,r_U}$ along the y-axis for points with $x\ge U-\sum_{i=1}^{k} r_i$ and $y=k$ is the same as for $y=k+1$, and for the remaining points the "flat" hyperplane suffices. We need to analyze the case when $r_k<r_{k+1}$. In this case, let $S_k=S_{k+1}\cup\{(1,r_{k+1},\varphi_{r_1,\dots,r_U}(U,k+1)-U-r_{k+1}(k+1))\}$. Let H be the hyperplane defined by the coefficients

$$(1,r_{k+1},\varphi_{r_1,\dots,r_U}(U,k+1)-U-r_{k+1}(k+1));$$

it is easy to verify that H interpolates $\varphi_{r_1,\dots,r_U}$ at $(x,k+1)$ for all $1\le x\le U$ by construction. The argument now proceeds as before: the hyperplane interpolates $\varphi_{r_1,\dots,r_U}$ at points (x,k) for which $x\ge U-\sum_{i=1}^{k} r_i$, because for such points its slope on the y-direction matches the slope r_{k+1} of $\varphi_{r_1,\dots,r_U}$. The remaining points with $y=k$ have value A, and therefore the "flat" hyperplane interpolates them and H has value $<A$ at them. Furthermore, because the $r_{k+1}<r_{k+2}\le\cdots\le r_U$, the value of H at all points (x,y) with $y>k$ is smaller than $\varphi_{r_1,\dots,r_U}(x,y)$. This concludes the inductive claim. ∎

The above claim shows that we can construct a set S_1 containing hyperplanes such that, for every $(x, y) \in [1, \ldots, U]^2$, the pointwise maximum of the hyperplanes at (x, y) is equal to $\varphi_{r_1,\ldots,r_U}(x, y)$. Note that all hyperplanes except the "flat" hyperplane have positive integer slopes. Thus, the pointwise maximum of the hyperplanes at integer points is integer valued. This implies that the value of the pointwise maximum of such hyperplanes on the integer grid is either A or $\geq A + 1$. To conclude the proof, we need a bound on $|S_1| = \text{diff}(r_1, \ldots, r_U) + 1$. Recall that $\sum_{i=1}^{U} r_i = U$, $r_1 \leq r_2 \leq \cdots \leq r_U$. The largest possible number of distinct values of r_i is attained when all r_i that increase do so by exactly 1. This implies that it is the largest number d such that $\sum_{j=1}^{d} j = \frac{d}{2}(d + 1) \leq U$. Then, $\text{diff}(r_1, \ldots, r_U) = O(\sqrt{U})$. ∎

COROLLARY 3.2. *Let A and U be nonnegative integers, and let Φ be a family of nondecreasing convex functions $\varphi : [1, U]^2 \to [A, A + U]$ that are described as a pointwise maximum of hyperplanes. Then, any approximation of $\varphi \in \Phi$ that attains relative error less than $\frac{A+1}{A}$, or absolute error less than 1, requires $\Omega(\sqrt{U})$ space regardless of the scheme used to represent the function.*

Proof. By Theorem 3.1, there are $\Omega(2^{\sqrt{U}})$ different nondecreasing convex functions that attain the value A at distinct points in $[1, \ldots, U]^2$ and a value at least $A+1$ at the remaining integer points. Any approximation of φ that attains a relative error less than $\frac{A+1}{A}$, or absolute error less than 1, characterizes exactly the points at which φ attains the value A. Thus, the approximation must distinguish between all functions in the family Φ described in the proof of Theorem 3.1. This implies that every function in Φ must have a different description, regardless of the scheme used to describe a function. Since $|\Phi| \in \Omega(2^{\sqrt{U}})$, it follows that any approximation of φ must take $\Omega(\sqrt{U})$ space. ∎

As mentioned in section 1.1, there exist several special classes of multivariate discrete convex functions; the inclusion relationships among these classes are depicted in Figure 1. Corollary 3.2 has implications on the approximability of convex-extensible functions, which we define next. Let $f : \mathbb{Z}^d \to \mathbb{R}^+$ be a function defined over the d-dimensional lattice of integer points. The convex closure of f is defined to be a function $\bar{f} : \mathbb{R}^n \to \mathbb{R}^+$ given by

$$\bar{f}(x) = \sup_{p \in \mathbb{R}^n, \alpha \in \mathbb{R}} \{\langle p, x\rangle + \alpha \mid \langle p, y\rangle + \alpha \leq f(y)\ \forall y \in \mathbb{Z}^n\} \quad \forall x \in \mathbb{R}^n.$$

If this function $\bar{f}$ coincides with f on integer points, i.e., if $\bar{f}(x) = f(x)\ \forall x \in \mathbb{Z}^n$, we say that f is *convex extensible* and call $\bar{f}$ the *convex extension of* f [22, Chapter 3.4].

The starting point for the proof of Theorem 3.1 is the family Φ of bivariate discrete convex functions (in the sense of Miller [21]) over domain $[1, \ldots, U]^2$ as defined in the proof of [10, Proposition 3.4]. For each function $f \in \Phi$ we construct a set S_1 of $O(\sqrt{U})$ hyperplanes and define a (continuous) convex function $\bar{f} : [1, U] \to \mathbb{R}^+$ as the pointwise maximum of the hyperplanes in S_1. We then show that f coincides with $\bar{f}$ on the integer square $[1, \ldots, U]^2$ and therefore the family Φ consists of discrete functions that are convex extensible (and are also convex in the sense of Miller at the same time). Using the fact that $|\Phi| = \Omega(\sqrt{U})$ [10, Proposition 3.4], and following similar arguments as in the proof of Corollary 3.2, Theorem 1.2 follows. We remark that the functions in [10, Proposition 3.4] are discrete convex as opposed to convex in the ordinary sense, and the two concepts do not necessarily coincide; this paper settles the status of discrete convex-extensible and continuous (ordinary) piecewise

linear convex functions, showing their inapproximability in space polynomial in the (binary) size of the domain.

3.2. Hardness of multidimensional DP. In this section we make some remarks about the approximability of (DP) when functions g_{T+1}, g_t^D are described by a value oracle and are not necessarily piecewise linear, as in Condition 2.8(ii). We focus on these two components of the cost function because the remaining terms g_t^I appear in the minimization over an ℓ-dimensional polyhedron; achieving an FPTAS for the case with g_t^I described by a value oracle would therefore require many additional mathematical tools, and is outside the scope of this paper.

As a consequence of Corollary 3.2, a two-dimensional generalization of problem (DP) is hard to approximate under the oracle model. We formalize this below. Consider the following problem:

$$\text{(MD-DP)} \qquad z^*(\vec{I}_1) = \min_{\pi_1,\dots,\pi_T} \mathbb{E}\Big[\sum_{t=1}^{T} g_t(\vec{I}_t, \pi_t(\vec{I}_t), \vec{D}_t) + g_{T+1}(\vec{I}_{T+1})\Big]$$
$$\text{subject to } \vec{I}_{t+1} = \vec{f}_t(\vec{I}_t, \pi_t(\vec{I}_t), \vec{D}_t) \qquad \forall t = 1, \dots, T,$$

where $\vec{I}_t \in \mathbb{R}^d$, $\vec{f}_t$ is a vector-valued function, each component of $\vec{f}_t$ is affine, g_{T+1} is strictly positive convex, and g_t can be expressed as

$$g_t(\vec{I}_t, \vec{x}_t, \vec{D}_t) = g_t^I(\vec{I}_t, \vec{x}_t) + g_t^D(\vec{f}_t^{\,g}(\vec{I}_{t,i}, \vec{x}_t, \vec{D}_t)),$$

where g_t^I and $g_t^D(\cdot)$ are nonnegative convex, and $\vec{f}_t^{\,g}$ has d components, each of which is affine. As usual, an approximation algorithm for (MD-DP) should output a function that is stored succinctly (in poly-space) and serves as an approximate value oracle for z^*. If g_{T+1} or g_t^D are described by value oracles, Corollary 3.2 immediately implies the following.

COROLLARY 3.3. *There does not exist a PTAS for continuous dynamic programs of the form* (MD-DP) *with cost functions described by a value oracle, even if* $d = 2$, $T = 1$, *and the cost functions are nonnegative, nondecreasing, and piecewise linear convex.*

By contrast, this paper provides an FPTAS when $d = 1$ under the same assumptions.

3.3. Hardness of approximating the value function of two-stage stochastic linear programs. In addition to DP, a widely used paradigm for optimization under uncertainty is stochastic programming. Due its importance and applicability, we find it worthwhile to report that Theorem 3.1 has implications on the hardness of approximation of stochastic programs. The concept of value functions is widely used in stochastic programming just as in DP. It is well known that the value function of a stochastic linear program may have a number of hyperplanes that is proportional to the number of values in the support of the r.v.s. In this section we show that if there are two variables linking consecutive stages, even an approximation of the value function may require a large number of hyperplanes. In particular, if the r.v.s are described in a compact form, i.e., as a value oracle to their CDF, then such a number is exponential in the input size: this is formalized in the next theorem. By contrast, if there is only one linking variable, we can construct such an approximation with the approach discussed in the previous section.

THEOREM 3.4. *There does not exist a PTAS to compute the optimal value function for the second stage of a two-stage stochastic linear program with a single discrete r.v. described via an oracle to its CDF, even if there are only two variables linking the first and second stages and the value function is bounded from below by a given positive number.*

Proof. We show how to construct a two-stage stochastic linear program such that, for any partitioning $r_1, \dots, r_U$ of a given integer U, the optimal second-stage value function $Q(x, y)$ is exactly the function $\varphi_{r_1,\dots,r_U}$ described in the proof of Theorem 3.1. At the same time, the input size is $O(\log U)$. By Corollary 3.2, this implies that there cannot be a PTAS for the problem.

Let the first stage of the problem be the following:

$$\begin{aligned} &\min c_1 x + c_2 y + Q(x, y), \\ &\qquad B \begin{pmatrix} x \\ y \end{pmatrix} = \vec{b}, \\ &\qquad x, y \in [1, U], \end{aligned}$$

where $Q(x, y)$ is the second-stage value function, B is a suitable constraint matrix and $\vec{b}$ is a corresponding R.H.S. vector. Define the second stage problem as

$$\begin{aligned} Q(x, y) := \min\ & z(D) + A \\ \text{subject to } & w(D) \geq y - D, \\ & z(D) \geq U\mathbb{E}_D[w(D)] + x - U, \\ & w(D) \geq 0, \\ & z(D) \geq 0. \end{aligned} \tag{3.4}$$

This is a stochastic linear program with an expected-value constraint. In (3.4), $A > 0$ is the minimum of the value function.

For a given partition $r_1, \dots, r_U$ of the integer U, the slopes of $\varphi_{r_1,\dots,r_U}$ along its two dimensions are given in (3.2) and (3.3). Notice that the x-axis slope is always 0 or 1. Let $b_1 = 1, \ \dots, \ b_m \in [1, \dots, U-1]$ be the integer values at which the slope of $\varphi_{r_1,\dots,r_U}(U, y)$ changes along the y-axis. We define $b_{m+1} = U$ for notational convenience. For $i = 1, \dots, m$, let Δ_i be the y-slope of $\varphi_{r_1,\dots,r_U}(U, y)$ in $[b_i, b_{i+1}]$, which is constant by the definition of b_i. We define $\Delta_0 = 0$ for notational convenience. Let D be a discrete r.v. with support $\{b_1, \dots, b_{m+1}\}$ and $\Pr(D = b_i) = \frac{\Delta_i - \Delta_{i-1}}{U}$ for $i = 1, \dots, m$, $\Pr(D = b_{m+1}) = \frac{U - \Delta_m}{U}$. Notice that problem (3.4) can be described in $O(\log U)$ bits.

With this construction, the optimal value function $Q(x, y)$ is the following:

$$\begin{aligned} Q(x, y) &= \min\{z(D) + A : w(D) \geq y - D,\ z(D) \geq U\mathbb{E}_D[w(D)] + x - U, \\ &\qquad w(D) \geq 0,\ z(D) \geq 0\} \\ &= \min\{(U\mathbb{E}_D[w(D)] + x - U)^+ + A : w(D) \geq y - D,\ w(D) \geq 0\} \\ &= \left(U \sum_{i=1}^{m+1} \Pr(D = b_i)(y - b_i)^+ + x - U \right)^+ + A \\ &= \left(\sum_{i=1}^{m} (\Delta_i - \Delta_{i-1})(y - b_i)^+ + x - U \right)^+ + A. \end{aligned}$$

In the above expression, the last equality is due to the fact that for $i = m+1$ we have $(y - b_i)^+ = 0$, so the last term of the summation vanishes. We claim that $Q(x, y) = \varphi_{r_1,\dots,r_U}(x, y)$. Clearly $Q(x, y)$ is piecewise linear convex, nondecreasing, and its range is $[A, A + U]$ over the domain $[1, U]^2$. Since its slope along the x-axis is 1 whenever the function value is $> A$, we just need to show that $Q(U, y) = \varphi_{r_1,\dots,r_U}(U, y)$; equality over the rest of the domain then follows from the analysis carried out in the proof of Theorem 3.1. We show by induction on $k = 1, \dots, m+1$ that $Q(U, y) = \varphi_{r_1,\dots,r_U}(U, y)$ for $y \in [1, b_k]$. For $k = 1$ this is trivial because $Q(U, 1) = A = \varphi_{r_1,\dots,r_U}(U, 1)$. For the induction step, we need to ensure that the y-slope of $Q(U, y)$ over the interval $[b_{k-1}, b_k]$ matches that of $\varphi_{r_1,\dots,r_U}(U, y)$. The terms in the summation $\sum_{i=1}^{m+1}(\Delta_i - \Delta_{i-1})(y - b_i)^+ + x - U$ for $i \geq k$ are clearly 0, whereas for $i < k$ the slope is $\Delta_i - \Delta_{i-1}$. So, the overall slope is $\sum_{i=1}^{k-1}(\Delta_i - \Delta_{i-1}) = \Delta_{k-1}$, as desired. This concludes the proof. ∎

We remark that the stochastic LP (3.4) does not satisfy the conditions of our DP framework because of the term $(U\mathbb{E}_D[(y - D)^+] + x - U)^+$ in the objective function, which is unusual for DP. More specifically, Condition 2.8 allows a cost function g_t of the form $(U(y - D)^+ + x - U)^+$, and our DP model (as is customary in DP) would take the expectation of the whole expression: $\mathbb{E}_D[(U(y - D)^+ + x - U)^+]$. In other words, it is unclear how to formulate as a DP the cost $(U\mathbb{E}_D[(y - D)^+] + x - U)^+$, where a piecewise linear convex function is applied to the expectation. In any case, Theorem 3.4 is indicative of the difficulties faced when dealing with two-dimensional value functions.

4. Constructing approximation sets with bounded-size numbers. The FPTAS that we build in section 6 relies on solving several LPs to compute an approximation of the value function at each stage $t = T, \dots, 1$. The input to these LPs at stage t depends on data computed by the LPs at stage $t+1$: the data are the domain and codomain values of a K-approximation set for the value function. It is well known that the size of the output of an LP (i.e., the optimal solution and its value) is, in the worst case, superlinear in the size of the input data, although still polynomial. In other words, every time an LP is solved, the size of the output numbers may increase polynomially compared to the size of the input numbers. We therefore need a device to limit the growth of the number sizes, to ensure that they do not take an exponential number of bits. As will become clearer in section 6, to limit the growth it is sufficient to have the ability to compute K-approximation sets that involve numbers that are "not too large." The next result shows that this can be done efficiently (recall from section 2.1 that t_φ denotes an upper bound on the time needed to evaluate φ on a single point in its domain).

PROPOSITION 4.1. *Let $\varphi : [A, B] \to \mathbb{R}^+$ with $A, B \in \mathbb{Z}$ be a positive convex nondecreasing function with Lipschitz constant κ_φ. Then, for any $K := 1 + \epsilon$, Algorithm 4.1 returns a canonical representation of a K-approximation set of φ of $O(\log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ size consisting of points that have domain and codomain values representable in $O(\log \frac{1}{\epsilon} + \log \frac{\varphi^{\max}}{\varphi^{\min}} + \log \kappa_\varphi + \log |A| + \log |B|)$ space. The algorithm runs in $O(t_\varphi \log \frac{\kappa_\varphi (B-A)}{\epsilon \varphi^{\min}} \log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ time.*

Proof. To simplify the exposition, we assume $\epsilon \leq 1/4$. The idea for the proof is to construct an appropriately scaled version of φ that admits a fully integer K-approximation set, i.e., with integer domain and codomain values. After constructing such an integer K-approximation set, we apply the inverse scaling to obtain a

Algorithm 4.1 Function ScaledCompressConvInc($\varphi, [A, B], K = 1 + \epsilon$).

1: Let $A' \leftarrow \frac{100\kappa_\varphi}{\epsilon^2\varphi^{\min}} A$, $B' \leftarrow \frac{100\kappa_\varphi}{\epsilon^2\varphi^{\min}} B$
2: Define $\rho(x) := \frac{10\varphi\left(\frac{\epsilon^2\varphi^{\min}}{100\kappa_\varphi} x\right)}{\epsilon\varphi^{\min}}$
3: $W' \leftarrow \{(A', \lceil\rho(A')\rceil)\}$
4: $x \leftarrow A'$
5: **while** $(1+\epsilon)\rho(x) < \rho(B')$ **do**
6: find a point $y' \in [A', B']$ such that $(1 + \frac{3}{10}\epsilon)\rho(x) \le \rho(y') \le (1 + \frac{4}{10}\epsilon)\rho(x)$
7: find a point $y'' \in [A', B']$ such that $(1 + \frac{6}{10}\epsilon)\rho(x) \le \rho(y'') \le (1 + \frac{7}{10}\epsilon)\rho(x)$
8: find an integer point $y \in [y', y'']$
9: $W' \leftarrow W' \cup \{(y, \lceil\rho(y)\rceil)\}$
10: $x \leftarrow y$
11: $W' \leftarrow W' \cup \{(B', \lceil\rho(B')\rceil)\}$
12: $W \leftarrow \left\{ \left(\frac{\epsilon^2\varphi^{\min}}{100\kappa_\varphi} y, \frac{\epsilon\varphi^{\min}}{10} v^* \right) : (y, v^*) \in W' \right\}$
13: **return** W as an array of tuples sorted by their first coordinate

K-approximation set for the original function, and show that the size of the points is bounded. The construction that we follow is summarized in Algorithm 4.1, called ScaledCompressConvInc. The function ρ defined therein (step 2) is the scaled version of φ mentioned above.

We claim that a canonical representation W' of a K-approximation set for ρ that is fully integer (except at its endpoints A', B') is constructed by routine ScaledCompressConvInc in $O(t_\varphi \log \frac{\kappa_\varphi(B-A)}{\epsilon\varphi^{\min}} \log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ time. We start by showing that, in every iteration of the "while" loop at lines 5–10, we find a point $y \in [x, B']$ and a value v^* such that (i) $\rho(y) \ge (1 + \frac{3}{10}\epsilon)\rho(x)$, (ii) $\rho(x) \le v^* \le (1+\epsilon)\rho(x)$, and (iii) $\rho(x) \le \rho(w) \le (1+\epsilon)\rho(x)$ for all $w \in [x, y]$.

Because φ is nondecreasing, lines 6 and 7 of ScaledCompressConvInc can be executed by binary search. We are looking for a point within an interval of size at least 1 starting with an interval of size $\frac{100\kappa_\varphi}{\epsilon^2\varphi^{\min}}(B-A)$; hence, the running time for each of the two steps is $O(t_\varphi \log \frac{\kappa_\varphi(B-A)}{\epsilon\varphi^{\min}})$. Line 8 can be performed in constant time by setting $y = \lceil y' \rceil$, as we will show next. By construction, $\rho(y'') - \rho(y') \ge \frac{2}{10}\epsilon\rho(x)$. Furthermore, $\kappa_\rho \le \frac{\epsilon}{10}$ by construction and the definition of κ_φ, and $\rho(x) \ge \frac{10}{\epsilon} \ge 1$. Therefore, $y'' - y' \ge \frac{2}{10}\epsilon\frac{\rho(x)}{\kappa_\rho} \ge 2$, and there must exist at least two integer points between y' and y''. The chain $y' \le \lceil y' \rceil < y''$ implies $(1 + \frac{3}{10}\epsilon)\rho(x) \le \rho(y') \le \rho(\lceil y' \rceil) < \rho(y'') \le (1 + \frac{7}{10}\epsilon)\rho(x)$, showing that $y = \lceil y' \rceil$ satisfies property (i). We claim that $\lceil\rho(y)\rceil$, used in line 9 of ScaledCompressConvInc, is such that $\rho(y) \le \lceil\rho(y)\rceil \le \rho(y)(1 + \frac{1}{10}\epsilon)$. Indeed,

$$\frac{\lceil\rho(y)\rceil}{\rho(y)} \le \frac{1 + \rho(y)}{\rho(y)} \le 1 + \frac{\epsilon\varphi^{\min}}{10\varphi(\frac{\epsilon^2\varphi^{\min}}{100\kappa_\varphi} y)} \le 1 + \frac{1}{10}\epsilon,$$

where the last inequality follows by definition of $\varphi^{\min}$. Finally, we have

$$\frac{\lceil\rho(y)\rceil}{\rho(x)} \le \left(1 + \frac{1}{10}\epsilon\right)\frac{\rho(y)}{\rho(x)} \le \left(1 + \frac{1}{10}\epsilon\right)\left(1 + \frac{7}{10}\epsilon\right) \le 1 + \epsilon,$$

showing property (ii). For the last inequality we used the assumption that $\epsilon \le 1/4$; larger ϵ can be handled by adjusting lines 6–7 of the algorithm. Property (iii) now follows immediately because ρ is convex nondecreasing.

Algorithm 4.2 Function ScaledCompressConv($\varphi, [A, B], K = 1 + \epsilon$).

1: $x^* \leftarrow \arg\min_{x \in [A,B]} \varphi(x)$ {we assume that a minimum of φ can be found efficiently}
2: $W_A \leftarrow$ ScaledCompressConvDec$(\varphi, [A, x^*], K)$
3: $W_B \leftarrow$ ScaledCompressConvInc$(\varphi, [x^*, B], K)$
4: $W \leftarrow (W_A \cup W_B) \setminus \{(x^*, \lceil \varphi(x^*) \rceil)\}$
5: **return** W as an array of tuples sorted by their first coordinate

By our discussion above, all (domain) points in W' except the two endpoints A' and B' have integer values. By properties (i), (ii), and (iii), the set W' is a canonical representation of a K-approximation set for ρ (the ceiling function at the endpoints A', B' adds only $\frac{1}{10}\epsilon$ relative error, by the same argument used in showing property (ii)). Clearly, the "while" loop at lines 5–10 is executed at most $O(\log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ times because the value of $\rho(x)$ increases by at least a bounded-below fraction of the multiplication factor K. Thus, W' contains $O(\log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ points and can be constructed in $O(t_\varphi \log \frac{\kappa_\varphi (B-A)}{\epsilon \varphi^{\min}} \log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ time.

The final step in ScaledCompressConvInc consists of applying to W' the inverse scaling that transforms φ into ρ to obtain W. It follows that the points in W belong to the domain and codomain of φ. The error bound, i.e., the K-approximation property, is trivially satisfied because it is satisfied for ρ and the relative error is invariant to multiplicative scaling. To conclude the proof, note that the size of the numbers in W', both domain and codomain, is

$$O\left(\log \frac{1}{\epsilon} + \log \frac{\varphi^{\max}}{\varphi^{\min}} + \log \kappa_\varphi + \log |A| + \log |B| \right)$$

because the domain is $[A', B']$ and the codomain is $[\frac{10}{\epsilon}, \frac{10\varphi^{\max}}{\epsilon \varphi^{\min}}]$; the multiplication factor when rescaling W' to W is of the same size, which gives the desired bound. ∎

We define function ScaledCompressConvDec for positive convex nonincreasing Lipschitz continuous functions in a similar way. We define function ScaledCompressConv for (not necessarily monotone) convex Lipschitz continuous functions as described in Algorithm 4.2. The algorithm assumes that a minimum of φ can be found exactly; in the FPTAS, this will be the case, as the minima will be computable by solving an LP.

Theorem 4.2. *Let $\varphi : [A, B] \to \mathbb{R}^+$ with $A, B \in \mathbb{Z}$ be a positive convex function with Lipschitz constant κ_φ, and assume that $x^* \in \arg\min_{x \in [A,B]} \varphi(x)$ can be efficiently computed. Then, for any $K := 1+\epsilon$, Algorithm 4.2 returns a canonical representation of a K-approximation set of φ of size $O(\log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ consisting of points having domain and codomain values representable in*

$$O\left(\log \frac{1}{\epsilon} + \log \frac{\varphi^{\max}}{\varphi^{\min}} + \log \kappa_\varphi + \log |A| + \log |B| \right)$$

space. The algorithm runs in $O(t_\varphi \log \frac{\kappa_\varphi (B-A)}{\epsilon \varphi^{\min}} \log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ time.

Proof. Let $\ell := \max\{x \in W : x \le x^*\}$, $r := \min\{x \in W : x \ge x^*\}$. Let $\hat{\varphi}$ be the convex extension of φ induced by W. Clearly $\varphi(x) \le \hat{\varphi}(x) \le K\varphi(x)$ for $x \in [A, \ell] \cup [r, B]$ because W_A and W_B are K-approximation sets over the respective domains. By construction (see the proof of Proposition 4.1), we have $\varphi(\ell) \le K\varphi(x^*)$ and $\varphi(r) \le K\varphi(x^*)$. But then the convex extension $\hat{\varphi}$ induced by W is such that $\varphi(x) \le \hat{\varphi}(x) \le K\varphi(x)$ also for $x \in [\ell, r]$. The size of the numbers follows directly from Proposition 4.1. ∎

We remark that the main reason for discarding $(x^*, \lceil \varphi(x^*) \rceil)$ from W in Algorithm 4.2 is to ensure that the size of the numbers remains bounded as indicated by Proposition 4.1; in a practical implementation of the routine, it is recommended to keep some bounded-size rounding of $(x^*, \lceil \varphi(x^*) \rceil)$ in the canonical representation.

5. Approximating expectations with vectors of continuous random variables. To construct an approximation of the value function (1.1), we must be able to compute expected values efficiently using the given representation of the r.v.s. Under Condition 2.7, we can compute a K-approximation set for the CDF of each of the components of the vectors of r.v.s. More precisely, in case (i) of Condition 2.7 we can use the algorithm given in [15, section 8.2], and in case (ii) we can use the algorithm given in [13, section 4.1] (see the online appendix [16, Appendix A]). Within a DP setting, the expression of the value function contains the expectation of a composition of functions, and we need to find a way to efficiently compute this expectation. Under Conditions 2.7 and 2.8, at each stage the vector of random variables $\vec{D}_t$ is mapped to a scalar r.v. via a linear transformation, i.e., $\vec{\theta}^D$ or $\vec{\sigma}^D$. However, we do not have explicit access to the CDF of the corresponding scalar r.v., but only to the CDF of each component of the sum. We now show how to efficiently construct an approximate oracle for the CDF of a sum of r.v.s, using only oracles to their CDFs. We focus on the case of an unweighted sum of continuous r.v.s, remarking that a similar construction for a weighted sum of continuous or discrete r.v.s is straightforward. At the same time, our approach can be applied to the case where, instead of having exact access to the CDFs of the r.v.s, one has an FPTAS for it. Below, by "generalized PDF" we mean a probability distribution function (PDF) that may include the Dirac delta function δ in its definition. Recall that the Dirac delta function can be informally thought of as a function such that $\delta(0) = \infty$, $\delta(x) = 0\ \forall x \neq 0$ and $\int_{-\infty}^{+\infty} \delta(x)\,\mathrm{d}x = 1$; a formal definition is beyond the scope of this paper.

PROPOSITION 5.1. *For $i = 1, \ldots, \ell$, let X_i be a truncated continuous r.v. with support $[X_i^{\min}, X_i^{\max}]$. Suppose X_i admits a (generalized) PDF f_i and κ-Lipschitz continuous CDF F_i. Assume*

$$\gamma = \min_{i=1,\ldots,\ell} \{\Pr(X_i = X_i^{\min}), \Pr(X_i = X_i^{\max})\} > 0.$$

Define

$$X^{\min} = \min_{i=1,\ldots,\ell} X_i^{\min}, \qquad X^{\max} = \max_{i=1,\ldots,\ell} X_i^{\max},$$

and let $t_F = \max_{i=1,\ldots,\ell} t_{F_i}$. If $X_1, \ldots, X_\ell$ are independent, then, for any $K = 1 + \epsilon$, we can compute a K-approximation set with $O(\frac{\ell}{\epsilon} \log \frac{1}{\gamma})$ points for the CDF of $\sum_{i=1}^{\ell} X_i$ in $O(\frac{t_F \ell^3}{\epsilon^2} \log^2 \frac{1}{\gamma} \log(\kappa \ell (X^{\max} - X^{\min})))$ time.

The proof technique is based on building an approximation set of the sum of r.v.s by adding one variable at a time: assuming we have an approximation set W for the CDF of $\sum_{i=1}^{k-1} X_i$, we show that an approximation for the CDF of $\sum_{i=1}^{k} X_i$ can be constructed by iterating over the points in W.

Proof. Let $\bar{K} := \sqrt[\ell]{1+\epsilon}$. We proceed by induction on $k = 1, \ldots, \ell$, computing a $\bar{K}^k$-approximation set for the CDF of $\sum_{i=1}^{k} X_i$ that contains $O(\frac{\ell}{\epsilon} \log \frac{1}{\gamma})$ points and requires $O(\frac{t_F \ell^2}{\epsilon^2} \log^2 \frac{1}{\gamma} \log(\kappa k (X^{\max} - X^{\min})))$ time at each induction step. In these expressions, the factor ℓ/ϵ comes from the fact that $\log_{\bar{K}}(\cdot)$ is $O(\frac{\ell}{\epsilon} \log(\cdot))$ because $K = \sqrt[\ell]{1+\epsilon}$. For $k = 1$, we simply apply CompressInc, defined in [13] if D_t is a discrete random variable, and in [15] if D_t is a continuous random variable (see the

online appendix [16, Appendix A]). This yields a $\bar{K}$-approximation set W_1 of X_1 with $O(\frac{\ell}{\epsilon}\log\frac{1}{\gamma})$ points in $O(\frac{\ell t_F}{\epsilon}\log\frac{1}{\gamma}\log(\kappa(X^{\max}-X^{\min})))$ time.

Define $Y = \sum_{i=1}^{k-1} X_i$. Now assume that we have a $\bar{K}^{k-1}$-approximation set $W_{k-1} := \{b_1, \dots, b_m\}$ for the CDF F_Y of Y, and let $\tilde{F}_Y$ be the corresponding monotone extension. Recall that $\tilde{F}_Y$ can be seen as a summation of step functions. Let $h(x)$ be the step function $h(x) = 1$ if $x \geq 0$, and $h(x) = 0$ otherwise. Using this notation, we can express $\tilde{F}_Y(x) = \sum_{i=1}^m a_i h(x - b_i)$ for some $a_i \in \mathbb{R}^+$.

We want to compute $\varphi(z) := \Pr(X_k + Y \leq z)$ for any given z. Because X_k and Y are independent, we can write

$$\begin{aligned}\int_{\mathbb{R}}\int_{-\infty}^{z-x} f_{X_k,Y}(x,y)\,\mathrm{d}y\,\mathrm{d}x &= \int_{\mathbb{R}} F_Y(z-x)f_k(x)\,\mathrm{d}x \\ &\leq \int_{\mathbb{R}} \tilde{F}_Y(z-x)f_k(x)\,\mathrm{d}x \\ &\leq \bar{K}^{k-1}\int_{\mathbb{R}} F_Y(z-x)f_k(x)\,\mathrm{d}x,\end{aligned}$$

where the inequalities follow from the facts that $\tilde{F}_Y$ is a $\bar{K}^{k-1}$-approximation of F_Y and f_k is nonnegative. This shows that $\tilde{\varphi}(z) := \int_{\mathbb{R}} \tilde{F}_Y(z-x)f_k(x)\,\mathrm{d}x$ is a $\bar{K}^{k-1}$-approximation of $\varphi(z)$, i.e., the CDF of $X_k + Y$. Breaking down $\tilde{F}_Y$ as a summation of step functions, we can expand this integral as

$$\int_{\mathbb{R}}\sum_{i=1}^m a_i h(z-x-b_i)f_k(x)\,\mathrm{d}x = \sum_{i=1}^m a_i\int_{-\infty}^{z-b_i} f_k(x)\,\mathrm{d}x = \sum_{i=1}^m a_i \Pr(X_k \leq z - b_i).$$

It follows that $\tilde{\varphi}(z)$ can be computed in $O(mt_{F_k})$ time. By the induction hypothesis, this is $O(\frac{t_F\ell}{\epsilon}\log\frac{1}{\gamma})$. We now apply CompressInc to $\tilde{\varphi}$ with approximation factor $\bar{K}$. By approximation of approximation (Proposition 2.5(7)), this yields an approximation set W_k that induces a $\bar{K}^k$-approximation function of φ. W_k has $O(\frac{\ell}{\epsilon}\log\frac{1}{\gamma})$ points, and the computation takes $O(\frac{t_F\ell^2}{\epsilon^2}\log^2\frac{1}{\gamma}\log(\kappa k(X^{\max}-X^{\min})))$ time. This concludes the inductive claim.

To conclude the proof, we note that the loop discussed above is repeated for $k = 1, \dots, \ell$. Therefore, the total runtime is $O(\frac{t_F\ell^3}{\epsilon^2}\log^2\frac{1}{\gamma}\log(\kappa\ell(X^{\max}-X^{\min})))$. ∎

Proposition 5.1 is given for truncated continuous r.v.s, but it is straightforward to show that the same approach works for discrete r.v.s. In this case, instead of the PDF of the r.v.s, we simply employ the corresponding probability mass function, obtaining the same result. Furthermore, we note that the proof can be adapted to the case in which only approximate access to the CDF of the r.v.s is available. Indeed, suppose we have an FPTAS for the CDF of each of the r.v.s. In this case, we simply need to decrease the approximation factor $\bar{K}$ used in the proof to, say, $\bar{K} = \sqrt[2\ell]{1+\epsilon}$, and replace each call to the oracles for the CDF to a $\sqrt[2\ell]{1+\epsilon}$-approximate call to the FPTAS for the CDF. In this respect we improve upon [20]. The running time given by [20] to compute $\Pr(\sum_{i=1}^\ell X_i \leq \rho)$ for discrete integer r.v.s is $O(\frac{t_F\ell^3}{\epsilon^2}\log^2\frac{1}{\gamma}\log\rho)$, essentially the same as ours if we restrict our algorithm to the discrete case: we have $\log\ell(X^{\max}-X^{\min})$ rather than $\log\rho$ because we compute the CDF for all values of ρ simultaneously.

We define in Algorithm 5.1 a routine CompressConvolution$(\vec{X}, \vec{c}, K)$ that implements the constructive proof of Proposition 5.1. The routine takes as input a

Algorithm 5.1 Function CompressConvolution$((X_1,\dots,X_\ell),(c_1,\dots,c_\ell),K)$.

$\bar{K} \leftarrow \sqrt[\ell]{K}$
$W_1 \leftarrow$ CompressInc$(F_1(\frac{\cdot}{c_1}),[X_1^{\min},X_1^{\max}],\bar{K})$ $\{F_1(\cdot)$ is the CDF of $X_1\}$
for $k=2,\dots,\ell$ **do**
  denote by $\{b_1,\dots,b_m\}$ the points in the approximation set W_{k-1}
  let $a_1,\dots,a_m$ be the coefficients of the monotone extension $\tilde{F}_{k-1}(x)=\sum_{i=1}^m a_i h(x-b_i)$ of W_{k-1}
    $\{h(x)$ is the unit step function: $h(x)=1$ for $x\ge 0$, 0 otherwise$\}$
    $\{\tilde{F}_{k-1}(\cdot)$ is a $\bar{K}^{k-1}$-approximation of the CDF of $\sum_{j=1}^{k-1} c_j X_j\}$
  define $\tilde{\varphi}(z):=\sum_{i=1}^m a_i F_k(\frac{z-b_i}{c_k})$ $\{F_k(\cdot)$ is the CDF of $X_k\}$
  $W_k \leftarrow$ CompressInc$(\tilde{\varphi},[\sum_{j=1}^k X_j^{\min},\sum_{j=1}^k X_j^{\max}],\bar{K})$
return W_ℓ as an array of tuples sorted by their first coordinate

vector of random variables $\vec{X}$ and returns a canonical representation of a K-approximation oracle function for $\varphi(z) := \Pr(\vec{c}\cdot\vec{X}\le z)$. The query time of the oracle is $O(\log(\frac{\ell}{\epsilon}\log\frac{1}{\gamma}))$.

We can now assume that we have approximate access to the CDF of the random variables defined by $\vec{\theta}^D\cdot\vec{D}_t$ and $\vec{\sigma}^D\cdot\vec{D}_t$ that appear in the cost functions, as detailed in Condition 2.8. We then need to compute the expectation of a composition of functions using the approximate CDF. If $\vec{\theta}^D\cdot\vec{D}_t$ and $\vec{\sigma}^D\cdot\vec{D}_t$ each have a discrete distribution, i.e., case (ii) of Condition 2.7 holds, we can rely on [12, Proposition 6.8]. The rest of this section details how to efficiently compute expectations involving continuous r.v.s, thereby taking care of case (i) of Condition 2.7.

Our next result shows that we can efficiently compute an expected value of a composition of functions involving a continuous r.v. Essentially, the proposition is the analogue of [12, Proposition 6.8] for continuous r.v.s, but the proof technique has some significant differences, because for continuous r.v.s we do not know how to write an easy-to-evaluate closed-form expression for the expected value.

Proposition 5.2. *Let $\xi:[A,B]\to\mathbb{R}^+$ be a (not necessarily monotone) convex function. Let $K_1,K_2\ge 1$. Let $\psi:[A,B]\to\mathbb{R}^+$ be an increasing piecewise linear convex function that K_1-approximates ξ, with breakpoints $A=a_1<\cdots<a_n<B$ and slopes $0=\Delta_0\le\Delta_1<\cdots<\Delta_n$. Let D be a (not necessarily nonnegative) truncated continuous r.v. with support $[D^{\min},D^{\max}]$ and such that*

$$\min\{\Pr(D=D^{\min}),\Pr(D=D^{\max})\}>0.$$

Suppose D admits a (generalized) PDF F', and let F be the corresponding CDF. Let $\tilde{F}(\cdot)$ be a monotone K_2-approximation of F with breakpoints at $D^{\min}=d_1<\cdots<d_m=D^{\max}$, and let $\tilde{F}(d_0)=0$. Let $f(x,d)=bx+e-d$ for some given $b,e\in\mathbb{R}$, and $m_i(x)=\max\{j\mid d_j\le bx+e-a_i\}$. Then

$$\tilde{\xi}(x)=\psi(A)+\sum_{i=1}^{n}(\Delta_i-\Delta_{i-1})\Bigg((bx+e-d_{m_i(x)}-a_i)\tilde{F}(d_{m_i(x)+1}) + \sum_{k=1}^{m_i(x)-1}(d_{k+1}-d_k)\tilde{F}(d_{k+1})\Bigg) \tag{5.1}$$

is a convex piecewise linear K_1K_2-approximation function of $\mathbb{E}_D(\xi(f(x,D)))$ with $O(mn)$ pieces. Furthermore, one can construct in $O(mn\log n)$ time an oracle for $\tilde{\xi}(\cdot)$ in the form of either a canonical representation or a representation that consists of breakpoints and slopes, each of size $O(mn)$.

Algorithm 5.2 Function CompressExpVal$(\psi, (b, e, c), \tilde{F})$.

1: Let $[A, B]$ be the domain of ψ and x^* be one of its minimizers
2: Define ψ_1 as the restriction of ψ to $[A, x^*]$, ψ_2 as the restriction of ψ to $[x^*, B]$
3: $W_1 \leftarrow$ CompressExpValDec$(\psi_1, (b, e, c), \tilde{F})$
4: $W_2 \leftarrow$ CompressExpValInc$(\psi_2, (b, e, c), \tilde{F})$
5: **return** $W_1 \cup W_2$ as an array of tuples sorted by their first coordinate

Algorithm 5.3 Function CompressExpValInc$(\psi, (b, e, c), \tilde{F})$.

1: Using the approximation set for r.v. D inducing $\tilde{F}$, compute an approximation set $W_{D'}$ for the transformed random variable $D' = -D/c$
2: Let $\tilde{F}'$ be the monotone extension of $W_{D'}$, $d_1, \ldots, d_m$ be the breakpoints of $W_{D'}$
3: Let $a_1, \ldots, a_n$ be the breakpoints of ψ, and $\Delta_0, \ldots, \Delta_n$ its slopes
4: $S \leftarrow \text{sort}(\bigcup_{i=1}^{n} \{d_j - e + a_i : j = 1, \ldots, m\})$
5: Compute the $m-1$ partial sums $\sum_{k=1}^{j} (d_{k+1} - d_k)\tilde{F}'(d_{k+1})$ for $j = 1, \ldots, m-1$
6: $W \leftarrow \{(x, \tilde{\xi}(x)) : x \in S\}$, where $\tilde{\xi}$ is defined in (5.1)
7: Loop over the elements of W to eliminate redundant breakpoints where the slope does not change
8: **return** W as an array of tuples sorted by their first coordinate

The proof technique involves decomposing $\mathbb{E}_{D_t}(\xi(f(x, D)))$ as a finite sum involving $\tilde{F}$ for each piece of the piecewise linear function φ. The error bound is obtained by constructing "easy-to-handle" discrete r.v.s $\hat{D}$, $\check{D}$ such that $\hat{D} \preceq D_t \preceq \check{D}$ in the usual stochastic order. Since the proof is long and technical, it is given online; see [16, Appendix B].

As in [12], it is easy to see that Proposition 5.2 can be applied to separable linear transition functions $f(x, d) = bx + e + cd$ for any coefficient c, using a transformed r.v. D' with $D' := D/(-c)$, and a similar result holds for convex functions approximated by a decreasing piecewise linear convex function. As a consequence, we obtain a modified version of [12, Corollary 6.9], stated below.

Proposition 5.3. *Let $\xi : [A, B] \to \mathbb{R}^+$ be a convex function. Let $K_1, K_2 \geq 1$. Let $\psi : [A, B] \to \mathbb{R}^+$ be a piecewise linear convex function with n breakpoints that K_1-approximates ξ over $[A, B]$. Let D be a (not necessarily nonnegative) continuous r.v. satisfying the conditions stated in Proposition* 5.2. *Let $\tilde{F}(\cdot)$ be a monotone nondecreasing K_2-approximation of the CDF of D with m breakpoints. Let $f(x, D) = bx + e + cd$, with $b, e, c \in \mathbb{R}$. Then we can construct in $O(mn \log n)$ time a convex piecewise linear K_1K_2-approximation function of $\mathbb{E}_D(\xi(f(x, D)))$ in the form of either a canonical representation or a representation that consists of breakpoints and slopes, each of size $O(mn)$.*

The proof of Proposition 5.3 follows from the application of Proposition 5.2, twice, to the nondecreasing and nonincreasing parts of ψ. In Algorithm 5.2 we define a routine CompressExpVal$(\psi, (b, e, c), \tilde{F})$ that takes as inputs ψ, b, e, c, $\tilde{F}$ as defined in Proposition 5.3, and returns an oracle for a K_1K_2-approximation function of $\mathbb{E}_D(\xi(f(\cdot, D)))$ in the form of a canonical representation. The query time for the oracle is $O(\log mn)$. CompressExpVal uses one subroutine for the increasing part of the convex function ψ, called CompressExpValInc and defined in Algorithm 5.3, and one for the decreasing part, called CompressExpValDec. The pseudocode for CompressExpValDec is the same as CompressExpValInc after "mirroring" the function ψ given as the first argument to transform it from decreasing to increasing.

6. The approximation scheme. We now show how to approximate a DP that satisfies Conditions 2.6–2.8. We give a full proof of the correctness of the algorithm for (the simpler) Condition 2.8(i), according to which the cost functions are known explicitly. The FPTAS under Condition 2.8(ii) is similar, and its proof highlights only the differences from Condition 2.8(i).

6.1. Condition 2.8(i): Explicit functions. Recall that by Condition 2.8(i) we have $g_t(I_t, \vec{x}_t, \vec{D}_t) = g_t^I(I_t, \vec{x}_t) + g_t^D(f_t^g(I_t, \vec{x}_t, \vec{D}_t))$; to avoid additional notation, we will not specify the expression of g_t^I and g_t^D in terms of breakpoints and slopes, but we will exploit their structure to construct an FPTAS. (Hereafter, given any function f we use the notation $f(\cdot)$, where "$\cdot$" stands for the varying argument of f.)

PROPOSITION 6.1. *Suppose the DP formulation* (DP) *satisfies Conditions* 2.6–2.8. *Let* $K_1, K_2, K_3 > 1$. *Let* $\hat{z}_{t+1}$ *be a piecewise linear convex* K_1*-approximation of the value function* z_{t+1}. *For a given* $I_t \in S_t$, *let* $\tilde{Z}_{t+1}(\cdot)$ *be a piecewise linear convex* K_2*-approximation of* $\mathbb{E}[\hat{z}_{t+1}(\cdot + \theta^I I_t + \vec{\theta}^D \cdot \vec{D}_t)]$, *and* $\tilde{G}_t^D(\cdot)$ *be a piecewise linear convex* K_3*-approximation of* $\mathbb{E}[g_t^D(\cdot + \sigma^I I_t + \vec{\sigma}^D \cdot \vec{D}_t)]$. *Define the following mathematical program:*

$$\left.\begin{aligned} \bar{z}_t(I_t) := \min_{\vec{x}_t} \; & g_t^I(I_t, \vec{x}_t) + \tilde{G}_t^D(w) + \tilde{Z}_{t+1}(y), \\ & A_t \vec{x}_t \geq \vec{b}_t + \vec{\delta}_{b_t} I_t, \\ & \vec{\sigma}^x \cdot \vec{x}_t - w = 0, \\ & \vec{\theta}^x \cdot \vec{x}_t - y = 0, \\ & \vec{x}_t \geq 0. \end{aligned}\right\} \tag{6.1}$$

Then the value of (6.1) *is a* $\max\{K_1K_2, K_3\}$*-approximation of the optimal value function* $z_t(I_t)$. *System* (6.1) *can be cast as an LP using the standard minimax reformulation of piecewise linear convex functions.*

Proof. The expression for the optimal value function is

$$z_t(I_t) = \min_{\vec{x}_t \in \mathcal{A}_t(I_t)} \mathbb{E}\{g_t(I_t, \vec{x}_t, \vec{D}_t) + z_{t+1}(f_t(I_t, \vec{x}_t, \vec{D}_t))\}.$$

By the linearity of the expectation and definition of g_t^I, $\tilde{G}_t^D$, $\tilde{Z}_{t+1}$, f_t, f_t^g, and $A_t(I_t)$, it is straightforward to check that (6.1) corresponds exactly to the expression of the value function, with two of the terms replaced by approximations. The fact that the resulting value is a $\max\{K_1K_2, K_3\}$-approximation of $z_t(I_t)$ follows from Proposition 2.5 (summation of approximation and approximation of approximation). ∎

We are now ready to state our main result. The proof consists of an application of the different results already stated in this paper, being careful to keep track of the runtime of the algorithm.

THEOREM 6.2. *Given a stochastic DP satisfying Conditions* 2.6–2.8(i), *the approximation scheme* APXSCHEME1(ϵ) *(Algorithm* 6.1*) computes a* $(1+\epsilon)$*-approximation of the optimal value function* z_1, *and runs in polynomial time in the binary input size and* $1/\epsilon$.

Proof. The proof proceeds by induction for $t = T+1, \ldots, 1$, showing that at stage t we obtain a piecewise linear convex $K^{2(T+1-t)}$-approximation $\hat{z}_t$ of the value function z_t via an approximation set of cardinality

$$O\left(\log_K \frac{(T+2-t)U_g}{g_{T+1}^{\min}}\right) = O\left(\frac{T}{\epsilon} \log \frac{(T+2-t)U_g}{g_{T+1}^{\min}}\right)$$

Algorithm 6.1 Procedure ApxScheme1(ϵ) for Condition 2.8(i)

1: $K \leftarrow \sqrt[2T]{1+\epsilon}$, $\hat{z}_{T+1} \leftarrow g_{T+1}$
2: **for** $t := T$ **downto** 1 **do**
3: $\quad \tilde{F}_g \leftarrow \text{CompressConvolution}(\vec{D}_t, \vec{\sigma}^D, K)$
4: $\quad \tilde{F}_z \leftarrow \text{CompressConvolution}(\vec{D}_t, \vec{\theta}^D, K)$
5: $\quad$ for fixed I_t, define $\tilde{G}_t^D \leftarrow \text{CompressExpVal}(g_t^D, (1, \sigma^I I_t, 1), \tilde{F}_g)$
$\quad\quad \{\tilde{G}_t^D(\cdot)$ is an oracle for a K-approximation of $\mathbb{E}[g_t^D(\cdot + \sigma^I I_t + \vec{\sigma}^D \cdot \vec{D}_t)]\}$
6: $\quad$ for fixed I_t, define $\tilde{Z}_{t+1} \leftarrow \text{CompressExpVal}(\hat{z}_{t+1}, (1, \theta^I I_t, 1), \tilde{F}_z)$
$\quad\quad \{\tilde{Z}_{t+1}(\cdot)$ is an oracle for a K-approximation of $\mathbb{E}[\hat{z}_{t+1}(\cdot + \theta^I I_t + \vec{\theta}^D \cdot \vec{D}_t)]\}$
7: $\quad \hat{z}_t \leftarrow \text{ScaledCompressConv}(\bar{z}_t, [\mathcal{S}_t^{\min}, \mathcal{S}_t^{\max}], K)$, where $\bar{z}_t$ is defined as in (6.1)
8: **return** $\hat{z}_1$

since $K = \sqrt[2T]{1+\epsilon}$. The argument of the log follows from the fact that the largest value of the value function at stage t is bounded above by $(T+2-t)U_g$, and its smallest value is bounded below by $g_{T+1}^{\min}$. The step $t = T+1$ is trivial, as we have $\hat{z}_{T+1} = g_{T+1}$, which is exactly the value function at the terminal stage, and we are given its piecewise linear convex description in the input. We now show the induction step.

In step 3, using the routine CompressConvolution, we compute a K-approximation $\tilde{F}_g$ of the CDF of $\vec{\sigma}^D \cdot \vec{D}_t$ in the form of a canonical representation. By Proposition 5.1, the cardinality of the canonical representation is $O(\frac{T\ell}{\epsilon} \log \frac{1}{\gamma})$, and this step requires $O(\frac{t_F T \ell^3}{\epsilon^2} \log^2 \frac{1}{\gamma} \log(\kappa \ell n^* U_f))$ time. Step 4 is similar and computes a K-approximation $\tilde{F}_z$ of the CDF of $\vec{\theta}^D \cdot \vec{D}_t$.

In step 5, for a fixed value of I_t we define a K-approximation $\tilde{G}_t^D(\cdot)$ of

$$\mathbb{E}_{\vec{D}_t}[g_t^D(\cdot + \sigma^I I_t + \vec{\sigma}^D \cdot \vec{D}_t)],$$

using Proposition 5.3 with parameters set to $\xi = \psi = g_t^D$, $n = O(m_t)$, $m = O(\frac{T\ell}{\epsilon} \log \frac{1}{\gamma})$, $K_1 = 1$, $K_2 = K$. This is possible because by Condition 2.8(i) we have a description of g_t^D in terms of breakpoints and slopes, which is equivalent to a 1-approximation set of g_t^D with $O(m_t)$ points. We remark that no actual computation is involved in this step because we have not fixed a value of I_t yet (it will be determined later, in step 7), but including step 5 in the algorithm helps us for the analysis. By Proposition 5.3, given a value for I_t, computing $\tilde{G}_t^D$ in the form of a canonical representation of size $O(\frac{m_t T \ell}{\epsilon} \log \frac{1}{\gamma})$, requires $O(\frac{m_t T \ell}{\epsilon} \log \frac{1}{\gamma} \log m_t)$ time.

In step 6, for a fixed value of I_t we define a K-approximation $\tilde{Z}_{t+1}(\cdot)$ of

$$\mathbb{E}[\hat{z}_{t+1}(\cdot + \theta^I I_t + \vec{\theta}^D \cdot \vec{D}_t)],$$

using Proposition 5.3 with parameters set to $\xi = \psi = \hat{z}_{t+1}$,

$$n = O\left(\frac{T}{\epsilon} \log \frac{(T+1-t)U_g}{g_{T+1}^{\min}}\right), \quad m = O\left(\frac{T\ell}{\epsilon} \log \frac{1}{\gamma}\right), \quad K_1 = 1, \quad K_2 = K.$$

Given a value for I_t, computing $\tilde{Z}_{t+1}$ in the form of a canonical representation of size $O(\frac{T^2\ell}{\epsilon^2} \log \frac{(T+1-t)U_g}{g_{T+1}^{\min}} \log \frac{1}{\gamma})$ requires $O(\frac{T^2\ell}{\epsilon^2} \log \frac{(T+1-t)U_g}{g_{T+1}^{\min}} \log \frac{1}{\gamma} \log \frac{T}{\epsilon} \log\log \frac{(T+1-t)U_g}{g_{T+1}^{\min}})$ time.

The main workhorse of the FPTAS is step 7, which computes an approximation $\hat{z}_t$ for the value function z_t using the function $\bar{z}_t$ as defined in (6.1). Note that, by Proposition 6.1 with parameters set to $K_1 = K^{2(T-t)}$ and $K_2 = K_3 = K$, $\bar{z}_t$

is a $K^{2(T-t)+1}$-approximation of z_t. The routine ScaledCompressConv can be used because a minimum of $\bar{z}_t$ can be found exactly with the solution of a single LP. Applying Theorem 4.2 to $\bar{z}_t$, coupled with Definition 2.4 and approximation of approximation (Proposition 2.5(7)), we obtain a $K^{2(T+1-t)}$-approximation $\hat{z}_t$ of z_t in the form of a canonical representation of size $O(\frac{T}{\epsilon}\log\frac{(T+2-t)U_g}{g^{\min}_{T+1}})$. This concludes the induction claim. It remains to show that the runtime of this step is polynomial. By Theorem 4.2, the runtime of step 7 is

$$O\bigg(t_{\bar{z}}\log\frac{\kappa_{\bar{z}_t}U_S}{\epsilon g^{\min}_{T+1}}\log_K\frac{(T+2-t)U_g}{g^{\min}_{T+1}}\bigg).$$

Note that by (6.1) the Lipschitz constant $\kappa_{\bar{z}_t}$ is bounded above by $(3U_fU_A^2)^{(T+1-t)}\kappa$. We now proceed to bound $t_{\bar{z}}$, i.e., the time to construct and compute (6.1) at a given value of I_t.

We first analyze the size of the LP (6.1). The variables y, w used in (6.1) can be substituted out. Thus, (6.1) has $p+3$ variables: p variables in the description of $\mathcal{A}_t(I_t)$, plus one variable for the minimax formulation of each of the convex functions $g_t^I(I_t,\vec{\cdot})$, $\tilde{G}_t^D(\cdot)$, and $\tilde{Z}_{t+1}(\cdot)$. The LP has

$$O\bigg(m+q_t+\frac{m_tT\ell}{\epsilon}\log\frac{1}{\gamma}+\frac{T^2\ell}{\epsilon^2}\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}}\log\frac{1}{\gamma}\bigg)$$

constraints plus p nonnegativity constraints: m constraints come from $\mathcal{A}_t(I_t)$, q_t come from the minimax formulation of the piecewise linear convex cost g_t^I, and the remaining constraints come from the minmax formulation of $\tilde{G}_t^D$ and $\tilde{Z}_{t+1}$, whose number of pieces is discussed above. The size of the numbers in the LP is bounded by

$$O\bigg(\max\bigg\{U_A,U_f,\log\frac{1}{\epsilon}+\log\frac{(T+2-t)U_g}{g^{\min}_{T+1}}\bigg\}\bigg).$$

An LP with these characteristics is solved in

$$O\bigg(\bigg(m+q_t+\frac{m_tT\ell}{\epsilon}\log\frac{1}{\gamma}+\frac{T^2\ell}{\epsilon^2}\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}}\log\frac{1}{\gamma}+p\bigg)^{1.5}\times p^2\max\bigg\{U_A,U_f,\log\frac{1}{\epsilon}+\log\frac{(T+2-t)U_g}{g^{\min}_{T+1}}\bigg\}\bigg)$$

arithmetic operations, following [26]. In addition, the piecewise linear representation for the functions $\tilde{G}_t^D$ and $\tilde{Z}_{t+1}$ is computed for each fixed I_t, which requires additional $O(\frac{m_tT\ell}{\epsilon}\log\frac{1}{\gamma}\log m_t+\frac{T^2\ell}{\epsilon^2}\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}}\log\frac{1}{\gamma}\log\frac{T}{\epsilon}\log\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}})$ time.

Hence, the total runtime to compute a single value of $\bar{z}_t$ is

(6.2)

$$\begin{aligned}t_{\bar{z}_t}=O\bigg(\bigg(\bigg(m+q_t+\frac{m_tT\ell}{\epsilon}\log\frac{1}{\gamma}+\frac{T^2\ell}{\epsilon^2}\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}}\log\frac{1}{\gamma}+p\bigg)^{1.5}&\\ \times p^2\max\bigg\{U_A,U_f,\log\frac{1}{\epsilon}+\log\frac{(T+2-t)U_g}{g^{\min}_{T+1}}\bigg\}\bigg)&\\ +\frac{m_tT\ell}{\epsilon}\log\frac{1}{\gamma}\log m_t&\\ +\frac{T^2\ell}{\epsilon^2}\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}}\log\frac{1}{\gamma}\log\frac{T}{\epsilon}\log\log\frac{(T+1-t)U_g}{g^{\min}_{T+1}}\bigg).&\end{aligned}$$

Step 7 has to be repeated T times. Adding the time to compute $\tilde{F}_z$ and $\tilde{F}_g$, we obtain the total runtime of the approximation scheme (Algorithm 6.1):

$$O\left(t_{\bar{z}_1}\frac{T^2}{\epsilon}\log\frac{\kappa_{\bar{z}_1}U_S}{\epsilon g^{\min}_{T+1}}\log\frac{(T+1)U_g}{g^{\min}_{T+1}}+\frac{t_F T^2\ell^3}{\epsilon^2}\log^2\frac{1}{\gamma}\log(\kappa\ell n^* U_f)\right),$$

where $t_{\bar{z}_0}$ is as in (9). If we perform the substitution, the dependency on ϵ is at most $\frac{1}{\epsilon^5}\log^2\frac{1}{\epsilon}$, the dependency on T is at most $T^6\log^3 T$, the dependency on ℓ is at most ℓ^3, and the dependency on p is at most $p^{3.5}$. If κ, U_A, and U_f are small enough so that the Lipschitz constant $\kappa_{\bar{z}_1}$ is bounded by a constant independent of the problem instance, then the dependency on T decreases to at most $T^5\log^3 T$. ∎

6.2. Condition 2.8(ii): Oracle model. We provide an FPTAS via a reduction to the FPTAS for Condition 2.8(i). To adapt the FPTAS to the case in which g_{T+1}, g_t^D are Lipschitz continuous univariate convex functions accessed by value oracles, we want to construct piecewise linear convex approximations of these functions with bounded error. In general, relative approximations for Lipschitz continuous univariate convex functions that are described via value oracles may not be computable in finite time (cf. [15, Corollary 2.5]). However, approximations that attain small multiplicative plus additive error (i.e., with both additive and relative error) can be found in polynomial time [15, Proposition 3.7]. The optimum of (DP) has a value of at least $g^{\min}_{T+1} > 0$ by assumption. Therefore, a small additive error in each of the T backward recursion steps can be absorbed without forsaking the approximation guarantee. In particular, we slightly reduce the relative approximation factor at each stage while allowing an additive error of the order of $\epsilon g^{\min}_{T+1}/2(2T+1)$.

We now give more details. We start by formally defining (Σ,Π)-approximation functions, as introduced in [15, Definition 2.10]. Σ denotes an additive approximation error, and $\Pi = 1+\epsilon$ a multiplicative approximation error; Π has the same role as K in this paper, and after the main definition we revert to our usual notation using K for the relative error.

DEFINITION 6.3. *Let $\epsilon > 0$, $\Sigma > 0$, $\Pi = 1+\epsilon$, and let $\varphi : [A,B] \to \mathbb{R}^+$. We say that $\tilde{\varphi} : [A,B] \to \mathbb{R}^+$ is a* (Σ,Π)-approximation function *of φ (or, more briefly, a (Σ,Π)-approximation of φ) if for all $x \in [A,B]$ we have $\varphi(x) \le \tilde{\varphi}(x) \le \Pi\varphi(x) + \Sigma$.*

It is possible to efficiently construct piecewise linear convex (Σ,Π)-approximation functions for Lipschitz continuous univariate convex functions that are described via value oracles. For space reasons, we do not give full details here, but we refer the reader to [15] or our online appendix [16, Appendix C]. The next three results are generalizations of Propositions 5.2, 5.3, and 6.1; the proofs are almost identical, but we use [16, Proposition C.2] for the error bounds.

PROPOSITION 6.4. *Let $\xi : [A,B] \to \mathbb{R}^+$ be a (not necessarily monotone) convex function. Let $K_1, K_2 \ge 1$ and $\Sigma > 0$. Let $\psi : [A,B] \to \mathbb{R}^+$ be an increasing piecewise linear convex function that (Σ,K_1)-approximates ξ, with breakpoints $A = a_1 < \cdots < a_n < B$ and slopes $0 = \Delta_0 \le \Delta_1 < \cdots < \Delta_n$. Let D be a (not necessarily nonnegative) truncated continuous r.v. with support $[D^{\min}, D^{\max}]$ and such that $\min\{\Pr(D = D^{\min}), \Pr(D = D^{\max})\} > 0$. Suppose D admits a (generalized) PDF F', and let F be the corresponding CDF. Let $\tilde{F}(\cdot)$ be a monotone K_2-approximation of F with breakpoints at $D^{\min} = d_1 < \cdots < d_m = D^{\max}$, and let $\tilde{F}(d_0) = 0$. Let $f(x,d) = bx + e - d$ for some given $b, e \in \mathbb{R}$, and $m_i(x) = \max\{j \mid d_j \le bx + e - a_i\}$.*

Then

$$\tilde{\xi}(x) = \psi(A) + \sum_{i=1}^{n} (\Delta_i - \Delta_{i-1}) \Bigg((bx + e - d_{m_i(x)} - a_i)\tilde{F}(d_{m_i(x)+1}) + \sum_{k=1}^{m_i(x)-1} (d_{k+1} - d_k)\tilde{F}(d_{k+1}) \Bigg)$$

is a convex piecewise linear $(\Sigma K_2, K_1 K_2)$*-approximation function of* $\mathbb{E}_D(\xi(f(x, D)))$ *with* $O(mn)$ *pieces. Furthermore, one can construct in* $O(mn \log n)$ *time an oracle for* $\tilde{\xi}(\cdot)$ *in the form of either a canonical representation or a representation that consists of breakpoints and slopes, each of size* $O(mn)$*.*

PROPOSITION 6.5. *Let* $\xi : [A, B] \to \mathbb{R}^+$ *be a convex function. Let* $K_1, K_2 \geq 1$ *and* $\Sigma > 0$*. Let* $\psi : [A, B] \to \mathbb{R}^+$ *be a piecewise linear convex function with* n *breakpoints that* (Σ, K_1)*-approximates* ξ *over* $[A, B]$*. Let* D *be a (not necessarily nonnegative) continuous r.v. satisfying the conditions stated in Proposition* 5.2*. Let* $\tilde{F}(\cdot)$ *be a monotone nondecreasing* K_2*-approximation of the CDF of* D *with* m *breakpoints. Let* $f(x, D) = bx + e + cd$*, with* $b, e, c \in \mathbb{R}$*. Then we can construct in* $O(mn \log n)$ *time a convex piecewise linear* $(K_2\Sigma, K_1K_2)$*-approximation function of* $\mathbb{E}_D(\xi(f(x, D)))$ *in the form of either a canonical representation, or a representation that consists of breakpoints and slopes, each of size* $O(mn)$*.*

PROPOSITION 6.6. *Suppose the DP formulation* (DP) *satisfies Conditions* 2.6–2.8*. Let* $K_1, K_2, K_3 > 1$*,* $\Sigma > 0$*. Let* $\hat{z}_{t+1}$ *be a piecewise linear convex* K_1*-approximation of the value function* z_{t+1}*. For a given* $I_t \in S_t$*, let* $\tilde{Z}_{t+1}(\cdot)$ *be a piecewise linear convex* K_2*-approximation of* $\mathbb{E}[\hat{z}_{t+1}(\cdot + \theta^I I_t + \vec{\theta}^D \cdot \vec{D}_t)]$*, and* $\tilde{G}_t^D(\cdot)$ *be a piecewise linear convex* (Σ, K_3)*-approximation of* $\mathbb{E}[g_t^D(\cdot + \sigma^I I_t + \vec{\sigma}^D \cdot \vec{D}_t)]$*. Define the following mathematical program:*

$$\left.\begin{aligned} \bar{z}_t(I_t) := \min_{\vec{x}_t} \; & g_t^I(I_t, \vec{x}_t) + \tilde{G}_t^D(w) + \tilde{Z}_{t+1}(y), \\ & A_t \vec{x}_t \geq \vec{b}_t + \vec{\delta}_{b_t} I_t, \\ & \vec{\sigma}^x \cdot \vec{x}_t - w = 0, \\ & \vec{\theta}^x \cdot \vec{x}_t - y = 0, \\ & \vec{x}_t \geq 0. \end{aligned}\right\} \tag{6.3}$$

Then the value of (6.3) *is a* $(\Sigma, \max\{K_1K_2, K_3\})$*-approximation of the optimal value function* $z_t(I_t)$*. System* (6.3) *can be cast as an LP using the standard minimax reformulation of piecewise linear convex functions.*

Equipped with these concepts, we analyze the FPTAS for the case of Condition 2.8(ii).

THEOREM 6.7. *Given a stochastic DP satisfying Conditions* 2.6*,* 2.7*, and* 2.8(ii)*, the approximation scheme* APXSCHEME2(ϵ) *(Algorithm* 6.2*) computes a* $(1 + \epsilon)$*-approximation of the optimal value function* z_1*, and runs in polynomial time in the binary input size and* $1/\epsilon$*.*

Proof. We outline here only the differences from the proof of Theorem 6.2, focusing on the computation of the approximation factor. The runtime analysis is very similar to that of Theorem 6.2, with one main modification: g_t^D is not described by

Algorithm 6.2 Procedure ApxScheme2(ϵ) for Condition 2.8(ii).

$\bar{\epsilon} \leftarrow \frac{\epsilon}{2T+1}, K \leftarrow 1+\bar{\epsilon}, \bar{K} \leftarrow 1+\frac{\bar{\epsilon}}{2}$
Using [16, Proposition C.1], let $\hat{z}_{T+1}$ be a $(\frac{g^{\min}_{T+1}\bar{\epsilon}}{2}, \bar{K})$-approximation of g_{T+1}
for $t := T$ **downto** 1 **do**
  compute a $(\frac{g^{\min}_{T+1}\bar{\epsilon}}{2}, K)$-approximation of g^D_t, called $\tilde{g}^D_t$, using [16, Proposition C.1]
  $\tilde{F}_g \leftarrow$ CompressConvolution$(\vec{D}_t, \vec{\sigma}^D, K)$
  $\tilde{F}_z \leftarrow$ CompressConvolution$(\vec{D}_t, \vec{\theta}^D, \bar{K})$
  for fixed I_t, define $\tilde{G}^D_t \leftarrow$ CompressExpVal$(\tilde{g}^D_t, (1, \sigma^I I_t, 1), \tilde{F}_g)$
    $\{\tilde{G}^D_t(\cdot)$ is an oracle for a $(\frac{g^{\min}_{T+1}\bar{\epsilon}K}{2}, K^2)$-approximation of $\mathbb{E}[g^D_t(\cdot + \sigma^I I_t + \vec{\sigma}^D \cdot \vec{D}_t)]\}$
  for fixed I_t, define $\tilde{Z}_{t+1} \leftarrow$ CompressExpVal$(\hat{z}_{t+1}, (1, \theta^I I_t, 1), \tilde{F}_z)$
    $\{\tilde{Z}_{t+1}(\cdot)$ is an oracle for a $\bar{K}$-approximation of $\mathbb{E}[\hat{z}_{t+1}(\cdot + \theta^I I_t + \vec{\theta}^D \cdot \vec{D}_t)]\}$
  $\hat{z}_t \leftarrow$ ScaledCompressConv$(\bar{z}_t, [\mathcal{S}^{\min}_t, \mathcal{S}^{\max}_t], K)$, where $\bar{z}_t$ is defined as in (6.3)
return $\hat{z}_1$

m_t breakpoints and slopes, but by an approximation set of size $O(\frac{T}{\epsilon}\log\frac{U_g}{g^{\min}_{T+1}\epsilon})$ computed in the course of the algorithm. This has repercussions through the analysis; however, it is easy to verify that it yields only a polynomial difference with respect to the runtime of Theorem 6.2.

The proof proceeds by induction for $t = T+1, \dots, 1$, showing that at stage t we obtain a piecewise linear convex $K^{2(T+1-t)+1}$-approximation $\hat{z}_t$ of the value function z_t via an approximation set of cardinality

$$O\bigg(\log_K \frac{(T+2-t)U_g}{g^{\min}_{T+1}\epsilon}\bigg) = O\bigg(\frac{T}{\epsilon}\log\frac{(T+2-t)U_g}{g^{\min}_{T+1}\epsilon}\bigg).$$

The last equality is due to the fact that $K = 1+\frac{\epsilon}{(2T+1)}$. The error bound $K^{2T+1} \le 1+\epsilon$ is a consequence of the inequality $(1+\frac{x}{n})^n \le 1+2x$, which holds for every $0 \le x \le 1$ and $n \in \mathbb{N}$.

The step $t = T+1$ is trivial, because $\hat{z}_{T+1}$ is a $(\frac{g^{\min}_{T+1}\bar{\epsilon}}{2}, \bar{K})$-approximation of g_{T+1}. By the definition of (Σ, Π)-approximation functions, $\hat{z}_{T+1}(I_{T+1}) \ge z_{T+1}(I_{T+1})$ for all I_{T+1}. Furthermore,

$$\begin{aligned}\hat{z}_{T+1}(I_{T+1}) &\le \frac{g^{\min}_{T+1}\bar{\epsilon}}{2} + \bigg(1+\frac{\bar{\epsilon}}{2}\bigg) z_{T+1}(I_{T+1}) \\ &= \frac{g^{\min}_{T+1}\epsilon}{2(2T+1)} + \bigg(1+\frac{\epsilon}{2(2T+1)}\bigg) z_{T+1}(I_{T+1}) \\ &\le \bigg(1+\frac{\epsilon}{2T+1}\bigg) z_{T+1}(I_{T+1}) \le K z_{T+1}(I_{T+1}),\end{aligned}$$

where the second inequality follows from the fact that $\tilde{g}^{\min}_{T+1}$ is a lower bound for g_{T+1} and consequently for z_{T+1}. By [16, Proposition C.1], we obtain a piecewise linear convex explicit description of $\hat{g}_{T+1}$ via an approximation set of the stated cardinality. We now show the induction step.

In step 7, for a fixed value of I_t we define a $(\frac{g^{\min}_{T+1}\bar{\epsilon}K}{2}, K^2)$-approximation $\tilde{G}^D_t(\cdot)$ of $\mathbb{E}_{\vec{D}_t}[g^D_t(\cdot + \sigma^I I_t + \vec{\sigma}^D \cdot \vec{D}_t)]$, using Proposition 6.5 with parameters set to $\xi = g^D_t$, $\psi = \tilde{g}^D_t$,

$$n = O\bigg(\frac{T}{\epsilon}\log\frac{U_g}{g^{\min}_{T+1}\epsilon}\bigg), \quad m = O\bigg(\frac{T\ell}{\epsilon}\log\frac{1}{\gamma}\bigg), \quad K_1 = K, \quad \Sigma = \frac{g^{\min}_{T+1}\bar{\epsilon}}{2}, \quad K_2 = K.$$

In step 8, for a fixed value of I_t we define a $\bar{K}$-approximation $\tilde{Z}_{t+1}(\cdot)$ of

$$\mathbb{E}[\hat{z}_{t+1}(\cdot + \theta^I I_t + \vec{\theta}^D \cdot \vec{D}_t)]$$

using Proposition 5.3 with parameters set to $\xi = \psi = \hat{z}_{t+1}$,

$$n = O\left(\frac{T}{\epsilon} \log \frac{(T+1-t)U_g}{g^{\min}_{T+1}\epsilon}\right), \quad m = O\left(\frac{T\ell}{\epsilon} \log \frac{1}{\gamma}\right), \quad K_1 = 1, \quad K_2 = \bar{K}.$$

Given a value for I_t, we compute $\tilde{Z}_{t+1}$ in the form of a canonical representation of size $O(\frac{T^2\ell}{\epsilon^2} \log \frac{(T+1-t)U_g}{g^{\min}_{T+1}\epsilon} \log \frac{1}{\gamma})$.

In step 9 we compute an approximation $\hat{z}_t$ for the value function z_t, using the function $\bar{z}_t$ as defined in (6.3). By Proposition 6.6 with parameters set to

$$K_1 = K^{2(T-t)+1}, \quad K_2 = \bar{K}, \quad K_3 = K^2, \quad \Sigma = \frac{g^{\min}_{T+1}\bar{\epsilon}K}{2},$$

we have that $\bar{z}_t$ is a $(\frac{g^{\min}_{T+1}\bar{\epsilon}K}{2}, K^{2(T-t)+1}\bar{K})$-approximation of z_t. This is also a $K^{2(T+1-t)}$-approximation of it, because

$$\begin{aligned}
\hat{z}_t(I_t) &\le \frac{g^{\min}_{T+1}\bar{\epsilon}K}{2} + K^{2(T-t)+1}\left(1 + \frac{\bar{\epsilon}}{2}\right) z_t(I_t) \\
&\le K^{2(T-t)+1}\left(\frac{\bar{\epsilon}}{2K^{2(T-t)}} + 1 + \frac{\bar{\epsilon}}{2}\right) z_t(I_t) \\
&\le K^{2(T-t)+1}(1+\bar{\epsilon}) z_t(I_t) \\
&= K^{2(T-t)+2} z_t(I_t),
\end{aligned}$$

where we have used the facts that $g^{\min}_{T+1} \le z_t(I_t)$ for all I_t, and $K^{2(T-t)} \ge 1$ since $t \le T$. The rest of the proof is identical to Theorem 6.2, with the runtime updated as described at the beginning of this proof. ∎

7. Concluding remarks. This paper presents an FPTAS for stochastic DPs with continuous scalar state spaces and polyhedral action spaces. To construct an approximation algorithm, we introduce several tools within the framework of K-approximation sets and functions. More specifically, we show how to compute the approximate convolution of a finite number of continuous random variables, the expectation of a convex function applied to a linear transformation of a continuous random variable, and how to bound the size of the numbers of a K-approximation set. Combining these tools, we obtain an FPTAS for a general class of DP models. These models can be seen as multistage stochastic LPs with one variable linking the stages. The most important open question is whether our results can be extended to the case of two (or more) variables linking the stages, or, in other words, two-dimensional state spaces for the DP. Under an oracle model for the cost functions, this paper shows that the problem is intractable even for the two-dimensional case. If the cost functions are known explicitly, the tools developed here do not suffice to settle the approximability status of the problem. This is left for future research.

REFERENCES

[1] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[2] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, NJ, 1961.

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

[4] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer Science & Business Media, Berlin, 2011.

[5] W. Chen, M. Dawande, and G. Janakiraman, *Fixed-dimensional stochastic dynamic programs: An approximation scheme and an inventory application*, Oper. Res., 62 (2014), pp. 81–103.

[6] Y. Chen and D. Goldberg, *Beating the Curse of Dimensionality in Options Pricing and Optimal Stopping*, preprint, https://arxiv.org/abs/1807.02227, 2018.

[7] D. P. de Farias and B. Van Roy, *The linear programming approach to approximate dynamic programming*, Oper. Res., 51 (2003), pp. 850–865.

[8] P. Favati and F. Tardella, *Convexity in nonlinear integer programming*, Ric. Oper., 53 (1990), pp. 3–44.

[9] P. Gopalan, A. Klivans, R. Meka, D. Štefankovic, S. Vempala, and E. Vigoda, *An FPTAS for # knapsack and related counting problems*, in Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Press, Piscataway, NJ, 2011, pp. 817–826.

[10] N. Halman, *Approximating convex functions via non-convex oracles under the relative noise model*, Discrete Optim., 16 (2015), pp. 1–16.

[11] N. Halman, *A deterministic fully polynomial time approximation scheme for counting integer knapsack solutions made easy*, Theoret. Comput. Sci., 645 (2016), pp. 41–47.

[12] N. Halman, *Provably Near-Optimal Approximation Schemes for Implicit Stochastic and for Sample-Based Dynamic Programs*, Technical report 4952, http://www.optimization-online.org/DB_HTML/2015/06/4952.html, 2017.

[13] N. Halman, D. Klabjan, C.-L. Li, J. Orlin, and D. Simchi-Levi, *Fully polynomial time approximation schemes for stochastic dynamic programs*, SIAM J. Discrete Math., 28 (2014), pp. 1725–1796.

[14] N. Halman, D. Klabjan, M. Mostagir, J. Orlin, and D. Simchi-Levi, *A fully polynomial time approximation scheme for single-item inventory control with discrete demand*, Math. Oper. Res., 34 (2009), pp. 674–685.

[15] N. Halman and G. Nannicini, *Fully Polynomial-Time* (Σ, Π)*-Approximation Schemes for Continuous Nonlinear Newsvendor and Continuous Stochastic Dynamic Programs*, Technical report 5726, http://www.optimization-online.org/DB_HTML/2016/11/5726.html, 2016.

[16] N. Halman and G. Nannicini, *Toward Breaking the Curse of Dimensionality: An FPTAS for Stochastic Dynamic Programs with Multidimensional Actions and Scalar States*, preprint, https://arxiv.org/abs/1811.11680, 2018.

[17] N. Halman, G. Nannicini, and J. Orlin, *A computationally efficient FPTAS for convex stochastic dynamic programs*, SIAM J. Optim., 25 (2015), pp. 317–350.

[18] N. Halman, G. Nannicini, and J. Orlin, *On the complexity of energy storage problems*, Discrete Optim., 28 (2018), pp. 31–53.

[19] J. Kleinberg, Y. Rabani, and E. Tardos, *Allocating bandwidth for bursty connections*, in Proceedings of the 29th ACM Symposium on the Theory of Computing, Association for Computing Machinery, New York, 1997, pp. 664–673.

[20] J. Li and T. Shi, *A fully polynomial-time approximation scheme for approximating a sum of random variables*, Oper. Res. Lett., 42 (2014), pp. 197–202, https://doi.org/http://dx.doi.org/10.1016/j.orl.2014.02.004.

[21] B. L. Miller, *On minimizing nonseparable functions defined on the integers with an inventory application*, SIAM J. Appl. Math., 21 (1971), pp. 166–185.

[22] K. Murota, *Discrete Convex Analysis*, SIAM, Philadelphia, PA, 2003.

[23] J. M. Nascimento and W. B. Powell, *An optimal approximate dynamic programming algorithm for concave, scalar storage problems with vector-valued controls*, IEEE Trans. Autom. Control, 58 (2013), pp. 2995–3010.

[24] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed., Wiley, New York, 2011.

[25] W. B. Powell, A. George, H. S. ao, W. Scott, A. Lamont, and J. Stewart, *SMART: A stochastic multiscale model for the analysis of energy resources, technology, and policy*,

INFORMS J. Comput., 24 (2012), pp. 665–682.
[26] J. Renegar, *A polynomial-time algorithm, based on Newton's method, for linear programming*, Math. Program., 40 (1988), pp. 59–93.
[27] C. Swamy and D. B. Shmoys, *Sampling-based approximation algorithms for multistage stochastic optimization*, SIAM J. Comput., 41 (2012), pp. 975–1004, https://doi.org/10.1137/100789269.