Fully Polynomial Time (Σ, Π)-Approximation Schemes for Continuous Nonlinear Newsvendor and Continuous Stochastic Dynamic Programs

Nir Halman [*] Giacomo Nannicini [†]

Abstract

We study the nonlinear newsvendor problem concerning goods of a non-discrete nature, and a class of stochastic dynamic programs with several application areas such as supply chain management and economics. The class is characterized by continuous state and action spaces, either convex or monotone cost function, and affine transition function. We establish that these problems cannot be approximated to any degree of either relative or additive error, regardless of the scheme used.

To circumvent these hardness results, we generalize the concept of fully polynomial-time approximation scheme allowing arbitrarily small additive and multiplicative error at the same time, while requiring a polynomial running time in the input size and the error parameters. We develop approximation schemes of this type for the classes of problems mentioned above. In light of our hardness results, such approximation schemes are "best possible". A computational evaluation shows the promise of this approach.

Keywords: Newsvendor problem, stochastic inventory control, hardness of approximation, approximation algorithms, stochastic dynamic programming, K-approximation sets and functions.

[*] Hebrew University of Jerusalem, Israel, E-mail: halman@huji.ac.il

[†] IBM T. J. Watson, Yorktown Heights, NY, E-mail: nannicini@us.ibm.com

1 Overview of the paper

Newsvendor problem (NV). A fundamental single-period problem in stochastic inventory theory is the newsvendor problem. A vendor needs to decide upon the quantity x of an item with short life cycle (e.g. newspapers, fashion items or perishable liquids such as fresh squeezed juice and egg yolk) to order based on the known demand distribution, the ordering cost, the holding cost, and the lost sales cost. Let the cost of ordering quantity x of the item in the beginning of the period be $c(x)$, and the cost of having x unsold amount at the end of the selling period be $h(x)$. Let $b(x)$ be the shortage cost, which is incurred when there is unfulfilled demand quantity of x. All these functions are nonnegative and nondecreasing. We denote the stochastic demand by the random variable D. For the sake of simplicity, we assume from now on that $h(x) = b(x) = 0$ for all $x \leq 0$. Researchers have followed two approaches to solving NV. In the first approach, the expected cost of overestimating and underestimating demand is minimized. In the second approach the expected profit is maximized. Both approaches yield the same optimal solution. In this paper we follow the first approach: at the beginning of the period, the vendor decides upon the amount x to order so as to minimize the total costs, i.e.

$$\min_{x \geq 0} c(x) + \mathbb{E}_D[h(x - D) + b(D - x)]. \tag{1}$$

When all cost functions are linear, the problems collapses to $\min \mathbb{E}_D[o(x - D)^+ + u(D - x)^+]$, where o is an overage cost per item and u is an underage cost per item. [AHM51] showed that in this case the problem permits a simple solution policy: Determine a *base stock* $S := \arg\max \text{Prob}(D \leq S) \leq \frac{u}{u+o}$ and order enough to bring the stock level to S, the so-called *base stock policy*. $\frac{u}{u+o}$ is called the *critical ratio*. See also [Kho99, QWV$^+$11], [SCB14, p. 120] or any standard reference on inventory management. Interest in the NV has greatly increased over the past 60 years. This interest stems from the fact that the problem serves as a building block in many inventory and supply chain models as well as its relevance to practice, e.g., in service industries such as air transportation and hospitality. Moreover, the increase in product variety along with shrinkage of product lifecycles and relative steadiness of production and transportation leadtimes make single-period inventory models more relevant than ever [Kho99].

Nonlinear newsvendor problem (NNV). In a more realistic setting the cost functions of NV are not necessarily linear, e.g., when the ordering cost includes quantity discounts (such as truckload discounts and all units discounts), or setup costs, or increasing unit replenishment costs, see Fig. 1. Unlike its linear counterpart, the objective function (1) in the NNV is not necessarily convex. During the last half century a large body of work has been focused on the *structure* of the optimal policy for NNV as a function of the initial inventory level, under various assumptions on the cost functions (refer to the excellent extensive survey of [Por90] and the references therein for more detail). The assumption of arbitrary monotone nondecreasing cost functions is in general too weak to yield a structured optimal policy, so it is interesting to study approaches to *compute* approximate solutions [HOSL12].

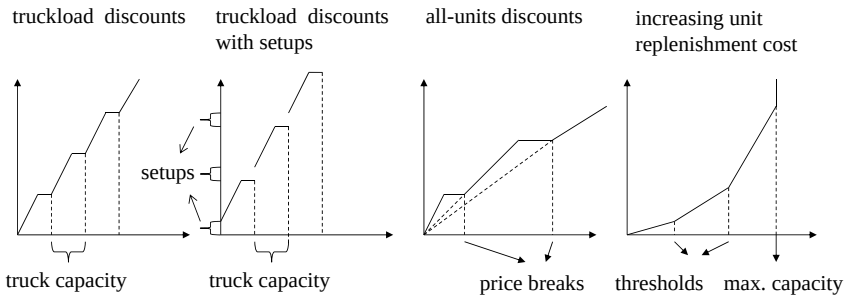


Figure 1: Examples of nonlinear ordering costs for NNV.

Approximation schemes. Given a minimization problem and a bound $\epsilon > 0$ on the relative error, a $(1+\epsilon)$-approximation algorithm finds a solution in time polynomial in the input size, where the value of the objective function is at most $(1+\epsilon)$-times the optimal value, i.e., with *multiplicative error* of at most $1+\epsilon$. A common alternative error measure is additive error. Given a bound $\Sigma > 0$ on the additive error, an additive Σ-approximation algorithm finds a solution in time polynomial in the input size, where the value of the objective function differs from the optimal value by at most Σ. If ϵ (resp. Σ) are parameters given explicitly to the algorithm and the running time of the approximation algorithm is also polynomial in $1/\epsilon$ (resp. $\log(1/\Sigma)$), then we call it a relative (resp. additive) *fully polynomial-time approximation scheme (FPTAS)*. For the sake of brevity, whenever we use the term FPTAS without mentioning the type of error, we refer to relative FPTAS. We implicitly assume that $\frac{1}{\epsilon} \geq 1$ and $\log(1/\Sigma) \geq 0$ without loss of generality.

Continuous vs. discrete inventory systems. The *linear* NV permits a simple solution regardless of whether the item is of discrete or continuous nature: the base stock policy, and the reorder level S admits the simple closed-form formula mentioned above. Unlike its linear counterpart, the problem of calculating the expected optimal cost of the NNV is intractable even for items of discrete nature [Hal16, Prop. A.3], but it does admit an FPTAS [Hal16, Thm. 1.5]. In this paper we show that in contrast to linear NV, the continuous variant of NNV is harder to solve than its discrete counterpart: while the discrete NNV admits an FPTAS, the continuous NNV admits neither relative nor additive approximation in general.

Inventory systems when information is given as an oracle. To the best of our knowledge, all past work about the NV except for [HOSL12] assumed that either all functions are given to the vendor *explicitly* as formulae, or that additional structure about the cost functions is known. This is not always a realistic assumption: in some cases the supplier does not reveal the order cost function $c(\cdot)$ to the vendor, and instead gives quotes $c(x)$ for every query x submitted by the vendor. This scenario applies, for example, when the vendor purchases in the spot market, as well as situations where orders are placed over the Internet. For instance, suppose the vendor is a tourism agency that books a block of seats in a specific flight. It is not realistic to assume that the airline provides the vendor with the function $c(\cdot)$. The aforementioned quotes model is more appropriate in these settings, i.e., the various functions are given to the vendor as "black boxes", or oracle functions.

We note that approximation schemes that only require oracle calls are appealing because the corresponding computer implementation is very general: to apply the approximation scheme, the only requirement for the user is to provide pointers to an implementation of the functions defining the NNV instance. In particular, the approximation scheme does not require knowledge of the exact mathematical formulation of the cost functions in analytical form. Another reason to use oracle functions to represent nonlinear cost functions is that oracle functions do not restrict the nonlinear function to be given in any particular form. Thus, an FPTAS that relies on an oracle function will be an FPTAS for any function that can be computed in polynomial time. This encompasses the case in which evaluating the oracle requires a nontrivial computation, e.g. solving a (polynomial-time) mathematical program. Oracle functions also permit strong negative results, such as proving that an exponential number of steps (i.e. oracle calls) is required to solve a problem.

An alternative inventory control approach. In this paper we show that the optimal expected cost of the continuous NNV cannot be approximated to within any given relative or additive error. This begs the question, what can a firm do to identify effective inventory control policies? We propose to focus on two dimensions when maximizing business performance, combining relative and additive error into a unified and intuitive two-dimensional error measure that can be arbitrarily small. E.g., instead of trying to find a solution which lies 1% off the optimum or a given fixed amount apart (e.g. \$1,000), which as discussed above may not be tractable in general, we will efficiently find a solution which is 1% *plus* \$1,000 off the optimum. More

generally, for a bound $\Pi = 1 + \epsilon > 1$ on the *multiplicative* error and a bound $\Sigma > 0$ on the *additive* error, a (Σ, Π)-approximation algorithm finds a feasible solution with value at most $\Pi v^* + \Sigma$, where $v^* \geq 0$ is the value of an optimal solution. If such a (Σ, Π)-approximation algorithm runs in time polynomial in the input size, we call it a *polynomial-time* (Σ, Π)*-approximation scheme*. If the running time is also polynomial in $1/\epsilon$ and $\log(1/\Sigma)$, we call it a *fully polynomial-time* (Σ, Π)*-approximation scheme*, abbreviated as (Σ, Π)-FPTAS. We argue that because additive and relative approximations are widely accepted measures of error in the business world, the same should be true for (Σ, Π)-approximations.

Dynamic Programming (DP). Our negative and positive results generalize to multi-period models via dynamic programming. Dynamic programming is an algorithmic technique used for solving sequential, or multi-stage, decision problems and is a fundamental tool in combinatorial optimization (see, e.g., [Hoc97], [Vaz01, Ch. 8]). We study a discrete-time finite-horizon stochastic dynamic program (DP, to be distinguished from "dynamic programming" by context), as defined in [Ber05]. The system dynamics are of the form: $I_{t+1} = f(I_t, x_t, D_t)$ for $t = 1, \ldots, T$, 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), $x_t \in \mathcal{A}_t(I_t)$ 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), D_t is a random variable over the sample space $\mathcal{D}_t$, and T is the number of time periods. To reduce the complication of the construction of algorithms, we assume throughout the paper that D is a discrete random variable. (In Section 7 we discuss how to extend our algorithmic results to continuous random variables.) In the context of this paper, D_t represents an exogenous information flow. The cost function $g_t(I_t, x_t, D_t)$ gives the cost of performing action x_t from state I_t at time t for each possible realization of the random variable D_t. In our context, I_t and x_t are one-dimensional, while D_t is a fixed-dimensional vector. The random variables are assumed independent but not necessarily identically distributed. Costs are accumulated over all time periods: the total incurred cost is equal to $\sum_{t=1}^{T} g_t(I_t, x_t, D_t) + g_{T+1}(I_{T+1})$. In this expression, $g_{T+1}(I_{T+1})$ is the cost paid if the system ends in state I_{T+1}, and the sequence of states is defined by the system dynamics. The problem is that of choosing a sequence of actions $x_1, \ldots, x_T$ that minimizes the expectation of the total incurred cost. This problem is called a *stochastic dynamic program*. Formally, we want to determine:

$$z^*(I_1) = \min_{x_1 \in \mathcal{A}_t(I_1), \ldots, x_T \in \mathcal{A}_t(I_t)} E\left[g_1(I_1, x_1, D_1) + \sum_{t=2}^{T} g_t(f_t(I_{t-1}, x_{t-1}, D_{t-1}), x_t, D_t) + g_{T+1}(I_{T+1}) \right], \tag{2}$$

where I_1 is the initial state of the system and the expectation is taken w.r.t. the joint probability distribution of the random variables D_t. It is well known that such a problem can be solved through a recursion.

Theorem 1.1 (Bellman optimality equation [BD62]) *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 recursion:*

$$z_t(I_t) = \min_{x_t \in \mathcal{A}_t(I_t)} \mathbb{E}_{D_t} \left\{ g_t(I_t, x_t, D_t) + z_{t+1}(f_t(I_t, x_t, D_t)) \right\}, \quad t = 1, \ldots, T. \tag{3}$$

The function $z_t(I_t)$ in the recursion above is called the *cost-to-go function* or *value function*.

We distinguish between *discrete* and *continuous* state and action spaces. In the former case, which we call *discrete DP*, the state and action spaces are finite sets of elements, in the latter case, which we call *continuous DP*, they are compact intervals on the real line. Combinatorial optimization problems such as the knapsack problem are naturally cast as discrete DPs. While solving such DPs is intractable, discrete DPs as defined here admit FPTASs [HKL$^+$14, Thm. 3.2]. In other contexts, e.g., inventory systems, economics and finance, the state and action may be best represented as continuous variables, for instance when they represent money; examples are provided in [AC03, Ch. 5], [NP10, Phe62]. This paper deals with continuous

DPs, and we assume that our goal is that of computing $z_1(I)$ for the entire initial state space $\mathcal{S}_1$, as is customary in DP. Several applications of our continuous DP framework are discussed in Section 4, e.g. a continuous single-item multi-period stochastic inventory control model. We note that computing the value function at a single point, i.e. $z^*(I_1)$, is already computationally intractable: Corollary 2.7 of this paper shows that even an *approximation* of $z^*(I_1)$ within any constant ratio is not possible in finite time.

Summary of the main results. We start by stating our results regarding NNV and then proceed to DPs.

Theorem 1.2 *Let $\Sigma, \epsilon > 0$ be given reals. Given an instance of NNV where the demand has only the two realizations 0 and A with equal probabilities, it is not possible in general to find an additive Σ-approximation (respectively, a multiplicative $(1+\epsilon)$-approximation) of the optimal expected cost, or an order quantity that yields such cost, in time polynomial in the instance size and $\log \frac{1}{\Sigma}$ (resp. $\frac{1}{\epsilon}$).*

A common approach to deal with continuous base stock levels is to discretize them in some meaningful way, e.g., with stepsize $\delta > 0$, and then solve (or approximate) the resulting discrete problem. A finer discretization results in smaller accuracy loss. If the multiplicative error introduced by the discretization could be bounded as a function of δ, and if the corresponding discrete problem admits a K-approximation algorithm for some constant $K > 1$, then one could get a KL-approximation for the continuous problem by approximating the discretized problem, for any constant $L > 0$. A similar argument works for additive approximations. In this paper we show that such a scheme may fail:

Theorem 1.3 *There does not exist, in general, a discretization of the base stock levels into a subset of cardinality polynomial in the instance size, with which one can bound either the relative or the absolute error between the value of an optimum of the continuous NNV and that of the discrete NNV.*

Thm. 1.3 follows from the fact that if there were such a polynomial discretization, we could have used the FPTAS for discrete NNV as given in [Hal16, Thm. 1.5] to get an FPTAS for continuous NNV, thereby contradicting to Theorem 1.2. The reason for the preceding result is that the relative error bound fails around zero values, and the additive error is not efficient when values are large.

Of course, a solution methodology based on discretization can be satisfactory in practice, and average-case analysis could explain its performance. But what if one insists on rigorous worst-case bounds? Possible approaches to overcome this hardness result include: changing the objective function in a way that reduces error, imposing additional constraints to simplify the problem, or perturbing the instance. We propose to work directly on the original problem while extending the error measure to the two dimensions Σ and Π as explained above. We introduce (Σ, Π)-FPTASs as efficient approximation algorithms that are natural generalization of "traditional" FPTASs, and show how to construct such an algorithm for continuous NNV. As a consequence of our result that additive or multiplicative approximations alone may not exist, a (Σ, Π)-FPTAS is in some sense "best possible" for continuous NNV.

Theorem 1.4 *The continuous NNV admits a (Σ, Π)-FPTAS that yields an approximation of the optimal cost and a corresponding order quantity.*

In fact, we show a stronger result and construct a (Σ, Π)-FPTAS for the continuous NNV even if we do not have direct access to the exact cost functions, but only to (Σ, Π)-FPTASs for them.

We next state our results about DPs. The first is an extension of Thm. 1.3.

Theorem 1.5 *There does not exist, in general, a discretization of the state and action spaces of a continuous DP to subsets of cardinality polynomial in the instance size, with which one can bound either the relative or the absolute error between the value of an optimum of the continuous DP and that of the discrete DP.*

We study 3 special cases of DPs. In the first case, for every t, the cost function g_t is increasing in I_t and monotone in x_t, while the transition function f_t is increasing in I_t and monotone in x_t. We call this the *increasing case*. The second case, in which the conditions are analogous to the increasing case, is called the *decreasing case*. We refer to these first two cases as the *monotone case*. In the third case, the transition function f_t is linear in I_t and x_t, while the cost function g_t has a convex structure (we give a formal definition in Section 4), and we call this the *convex case*. Monotone and convex DPs find many applications, some of which will be discussed in this paper (see also [HKL$^+$14, HNO15] and the references therein). Our second main result about DPs is a generalization of Thm. 1.4.

Theorem 1.6 *Every stochastic continuous DP satisfying some technical conditions (Conditions 1, 2 and 3 in this paper) admits a fully polynomial-time* (Σ, Π)*-approximation scheme.*

As noted earlier, relative error fails around zero values. But can one design a purely-multiplicative FPTAS when the cost functions are bounded away from zero? The theorem below shows that the answer is in the affirmative. As an example, we give in Section 6.2 an FPTAS for the continuous nonlinear knapsack problem.

Theorem 1.7 *Every stochastic continuous DP in which the cost functions are bounded away from zero and that satisfies some technical conditions (Conditions 1, 3 and 4 in this paper) admits an FPTAS.*

It is important to note that the term FPTAS typically refers to the computation of (approximate) solutions to problems that are NP-hard to compute exactly, but in this paper the approximation is done for problems that are impossible to solve exactly in any given finite time due to information theoretical considerations. For the same reason, the hardness results in this paper may be also termed impossibility results.

From a numerical point of view, we provide a preliminary computational evaluation of our approximation scheme, comparing it to solving or approximating the discretized DP, and to an approximate DP approach taken from the literature. The results show that our method is superior in terms of running time and solution quality, indicating that the methodology proposed in this paper has the potential of being successful in practice on the class of problems for which it applies. In particular, while discretization works quite well as a heuristic, working with the continuous problem directly yields better solutions for comparable computational effort.

Technique used. Our constructive proof methodology is an extension of the technique of K-approximation sets and functions [HKM$^+$09] to continuous functions and (Σ, Π)-approximations. We believe that this technique will be useful for further development of approximation algorithms. The idea behind such approximation sets is to keep a small (i.e. polynomially bounded size) set of points in the domain of a function, ensuring that some type of interpolation guarantees rigorous error bounds.

Relevance to existing literature. Numerous approaches have been devised to circumvent hardness of approximation results. A possibility is to forsake worst-case analysis in favor of average-case or smoothed analysis [ST04]. Alternatives include modifying the objective function or constraints of the instance in some prescribed way, e.g. [CFN77], or provide performance guarantees that degrade according to properties of the optimal solution [FIMN13, KPR04].

This paper circumvents a hardness of approximation result by using a two-dimensional error measure. This idea has been exploited in other contexts, but they all differ from our approach for two fundamental reasons: (a) For the class of problems that we consider, we show that additive or multiplicative approximations alone may not be possible, therefore a two-dimensional error measure is *necessary*; (b) Our approximation scheme allows the error to be *arbitrarily close to zero* (i.e., each of Σ, Π may be arbitrary close to $0, 1$ at the same time), as one would expect from a generalization of the concept of FPTAS.

We provide some examples. [KK82] build a $(1/\epsilon^2, 1+\epsilon)$-approximation scheme for the one-dimensional Bin-Packing problem. In their algorithm relative and additive errors are interdependent. Furthermore, smaller relative errors result in bigger additive errors. [EP04] construct a spanner of a general graph using additive and relative approximation, but in their case Σ cannot be arbitrarily close to 0, and $(0, \Pi)$ and $(\Sigma, 1)$ approximations are possible. [DKM$^+$06] study approximate (ϵ, δ) differential privacy, but $(\epsilon, 0)$ and $(0, \delta)$ approximations are possible. [SS06, SS12] study multi- and two-stage stochastic LPs, giving randomized (Σ, Π)-FPTASs that return a (Σ, Π)-approximated solutions with high probability: not only this is a randomized scheme (FPRAS), but their parameter Σ, which stems from the absolute error in Chernoff bound, can be zero under some assumptions. [FSS13] provide a (Σ, Π)-approximation of the sum of the squared Euclidean distances from the rows of $A \in \mathbb{R}^{n\times d}$ to any compact set spanned by k vectors in $\mathbb{R}^d$, but in their case Σ is a constant that depends only on A (i.e., it is not an input parameter), and while it can be normalized to be arbitrarily small, the running time does not scale up in $\log 1/\Sigma$.

We now discuss our contributions as compared to frameworks for FPTASs for classes of optimization problems, few of which have been reported in the literature. The frameworks in [Woe00, PW07, MS13] deal with deterministic discrete DPs only. The ones in [SS06, SS12] deal with stochastic linear programs (LPs) and are in fact randomized schemes. The one in [HKL$^+$14] deals with stochastic discrete DPs. The one in [HNO15] works under the same DP model, but provides a faster FPTAS from both theoretical and practical standpoints. Our approach draws on the idea of K-approximation sets and functions of [HKM$^+$09, HKL$^+$14, HNO15], but we deal with continuous domains *directly*, rather than discretizing and approximating the discretized problem (which would fail due to Theorem 1.5).

Continuous DP models are frequently employed in several domains, see e.g. [AC03]. Discretization (possibly with refinement) is a standard solution approach. Typically, error bounds with respect to the optimal value function are not provided. For example, the well-known approximate linear programming approach [dFVR03] provides a bound on the approximation error as compared to the best value function approximation that can be obtained as a linear combination of a given choice of basis functions. This type of error bound is clearly different in spirit from the natural concepts of additive and multiplicative error with respect to the optimal solution. We are not aware of any approximate DP (ADP) approach that provides error bounds of the type discussed in this paper in polynomial time; thus, for space reasons instead of surveying the abundant ADP literature we refer to the comprehensive references [Ber05, Pow11]. However, algorithms that asymptotically converge to the optimal value have been proposed in the literature: the most relevant for this paper are SPAR and CAVE, see [GP01, PRT04, NP13]. In particular, [NP13] studies a class of continuous DPs very similar to ours, and proposes an algorithm that is shown to converge to the optimal policy as the number of iterations goes to infinity. It is important to note that the ADP methodologies discussed above require fewer structural properties than our approach and can be applied to a broader set of problems. We argue that (Σ, Π)-FPTASs are interesting despite the more limited applicability because they provide much stronger guarantees on the approximation error and running time than is typically found in the ADP literature.

Finally, some of our results are related to the stream of research in derivative-free optimization that is concerned with lower bounds to the number of function evaluations to attain an optimum, such as [Sha13]. There are however notable differences: [Sha13] studies (only) Σ-approximations of the minimum of a strongly convex or quadratic d-dimensional function, whereas our paper studies Σ-, Π- and (Σ, Π)-approximations of general monotone and convex unidimensional functions over their entire domains. For example, [Sha13, Thm. 7] gives a polynomial lower bound in expectation assuming a possibly randomized querying strategy. Our Proposition 2.2 deals with a deterministic polynomial-time algorithm to Σ-approximate a minimum of a (ordinary) univariate convex function.

Organization of the paper. The paper is organized as follows. In Section 2 we define our model of computation and establish our hardness results. Then, we introduce (Σ, Π)-approximation sets and functions, and algorithms to compute them. We demonstrate the usefulness of these concepts and algorithms by designing in Section 3.2 a (Σ, Π)-FPTAS for NNV. In the rest of the paper we lift our algorithmic results to multi-stage decision making, i.e., to DPs. In Section 4 we state the stochastic continuous DP model to which our (Σ, Π)-approximation schemes apply, and our main result for DPs that shows how to construct the approximation scheme. Section 5 builds the link between DP and (Σ, Π)-approximation sets and functions, and includes the design and analysis of our FPTASs for convex and monotone DPs. We discuss extensions to (purely-multiplicative) FPTASs for special families of continuous stochastic DPs in Section 6. Section 7 concludes the paper discussing extensions and future directions. The Appendix contains a description of two specific problems (machine scheduling and inventory control) that fit in our framework, and the computational evaluation.

Notation. Let $\mathbb{R}, \mathbb{R}^+, \mathbb{Z}, \mathbb{Z}^+, \mathbb{N}$ denote the set of real numbers, nonnegative real numbers, integers, nonnegative integers and positive integers, respectively. For every $a, b \in \mathbb{R}$ with $a < b$, let $[a, b] = \{x \mid a \leq x \leq b\}$. Given a compact set $D \subset \mathbb{R}$, we denote by $D^{\min}$ and $D^{\max}$ the minimum and maximum element in D, respectively. For every $A, B \in \mathbb{Z}$ with $A < B$, let $[A, \dots, B] = \{A, A+1, \dots, B\}$. For simplicity, we assume throughout that $B - A > 1$. 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 a function $\varphi : D \to \mathbb{R}$ over a linearly ordered set D is called either *increasing* or *decreasing*, it is meant in the weak sense. Let $\varphi^{\max} = \sup_{x \in D} \varphi(x)$, $\varphi^{\min} = \inf_{x \in D} \varphi(x)$, and denote by t_φ the time needed to evaluate φ at a single point in its domain. A function is called *Lipschitz continuous with a given Lipschitz constant* κ (κ-Lipschitz continuous, in short), if for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is not greater than κ. Let $D' \subseteq D \subset \mathbb{R}$ where D' is a finite set and D is a compact interval. We define the *piecewise linear extension of φ induced by D'* as the function obtained by making φ linear between successive values on D'. A function φ over D is called *convex over D'* if its piecewise linear extension induced by D' is convex.

2 Hardness results and a new type of approximation

In this paper we face the problem of efficiently constructing a succinct representation for a function $\varphi : D \to \mathbb{R}^+$, having access only to a (zero-order, a.k.a. derivative-free) value oracle that for every point $x \in D$ returns $\varphi(x)$. By a representation of a function we mean a way to compute the function values using information stored on the computer memory, e.g., store its explicit formula. A representation of a function is *succinct* if it occupies a "small" space in the computer memory. A succinct representation is often either not possible or very costly, so an efficient succinct representation for an approximation of φ is of interest. In the context of this paper, examples of functions that we want to approximate efficiently and succinctly are the value functions z_t. To discuss approximation algorithms, we need to define the model of computations that we use.

2.1 Model of computation and black-box-oriented methods

We use the real arithmetic model as defined in [BN13, Sec. 5.1.1]. In this model, the computations are carried out by an idealized version of the usual computer that is capable of storing countably many reals and can perform the standard *exact* real arithmetic operations. Consider a problem (P) over a function $\varphi : D \to \mathbb{R}$. For example, in [BN13] (P) is the problem of finding the minimum of φ over D; in our paper, it is the construction of a succinct representation of φ over D. Let us fix a family $\mathcal{F}(D)$ of problems (P) with D common for all problems from the family, so that the family itself is nothing but a certain family of

functions over domain D. We define the information-based complexity of $\mathcal{F}(D)$ as in [BN13, Sec 5.1.1]; here we provide a brief summary, referring to [BN13] for the details. We use the same notation: $\delta > 0$ is an accuracy parameter to which the problem should be solved according to some accuracy measure, and $\mathcal{B}$ is an algorithm that solves (P) by generating a sequence of search points and calling the oracle to get the values of φ on these points. $T_{\mathcal{B}}(\varphi, D, \delta)$ denotes the number of steps in which $\mathcal{B}$ is capable of solving within accuracy δ a problem φ over D. The complexity of $\mathcal{F}(D)$ w.r.t. a solution method $\mathcal{B}$ is the function $\text{Compl}_{\mathcal{B}}(D, \delta) = \max_{\varphi \in \mathcal{F}(D)} T_{\mathcal{B}}(\varphi, D, \delta)$. Finally, the *information-based complexity* of the family $\mathcal{F}(D)$ of problems is defined as $\text{Compl}(D, \delta) = \min_{\mathcal{B}} \text{Compl}_{\mathcal{B}}(D, \delta)$, the minimum being taken over all solution methods.

The real arithmetic model of computation is typically employed in the continuous optimization community. Other communities prefer less powerful models that are more similar to real-world computers, such as finite precision. Our hardness results generalize to the finite precision model (straightforward modifications in the proofs are sometimes necessary), and all our algorithmic results remain valid under such model if we query only rational points and add to one of our assumptions (Condition 2) the natural requirement that the values returned by the oracle of the cost function g_t are polynomially bounded by the (binary) size of the input.

2.2 Representation of random events

Besides the model of computation, it is crucial to establish how the random variables are represented. There are several possibilities. In this paper, we adopt a natural approach in which the random variables are discrete and are explicitly given by listing in the input all values in their support and the corresponding probabilities. That is, for every random variable D_t (in the NNV there is a single random variable, but our DP models allow one random variable per time period) we are given n_t, the number of different values it admits with positive probability, its support $\mathcal{D}_t := \{d_{t,1}, \ldots, d_{t,n_t}\}$, where $d_{t,i} < d_{t,j}$ for $i < j$, and ordered pairs $(d_{t,i}, p_{t,i})$ where $p_{t,i} = \text{Prob}(D_t = d_{t,i}))$, $\quad i = 1, \ldots, n_t$. This type of representation is natural but not necessarily compact. A different model is that in which only the endpoints of the support are given, and the CDF is accessible via a value oracle. Such an implicit model reduces the size of the input, but makes computations more difficult. All our hardness results are written under the explicit representation model and therefore clearly hold under the implicit model as well. We briefly sketch in Section 7 the main ideas to extend our positive results (i.e. the construction of (Σ, Π)-FPTASs) from the explicit model of this paper to the implicit case and to continuous random variables. A rigorous development of such extensions is very involved and is pursued elsewhere [Hal16, HN17].

2.3 Additive and multiplicative approximation functions

We now discuss the limitations of the natural concepts of additive and multiplicative approximation functions in the context of finding succinct approximations of functions over continuous domains. Additive and multiplicative approximations are defined in the usual way (see Section 1). We remark that we always consider approximations from above. We write Σ-approximation for additive approximations ($\Sigma \geq 0$) and Π-approximation for multiplicative approximations ($\Pi \geq 1$), and this notation is used for approximations of both real values and functions.

Definition 2.1 (Succinct approximations) *A Σ-approximation function of $\varphi : [A, B] \to \mathbb{R}^+$ is called* succinct *if it admits a representation in space polynomial in $\log \varphi^{\max} + \log(B - A) + \log(1/\Sigma)$. The same terminology is used for Π-approximations, replacing $\log(1/\Sigma)$ with $1/\epsilon$ where $\Pi = 1 + \epsilon$.*

We are concerned with determining which functions φ over continuous intervals on the real line one can approximate efficiently, and with finding approximations for $\varphi^{\min}$. This is because we want to construct

approximations of the value functions z_t, that cannot be computed exactly in polynomial time. We show that approximating $\varphi^{\min}$ can be done efficiently for additive approximations, but not for multiplicative approximations, and Lipschitz-continuous functions are hard to approximate additively or multiplicatively in general. We summarize our results regarding convex and monotone functions in Table 1.

Table 1: Summary of complexity results about approximating a Lipschitz continuous function φ. ($\checkmark$: solvable in polynomial time, $\nexists$: cannot be constructed in polynomial time, (x): see Prop./Cor. x).

	Σ-apx. $\varphi^{\min}$	succinct Σ-apx. of φ	Π-apx. $\varphi^{\min}$	succinct Π-apx. of φ
φ is convex	$\checkmark$ (2.2)	$\nexists$, even if φ is monotone (2.3)	$\nexists$ (2.4)	$\nexists$ (2.5)
φ is monotone	trivial	$\nexists$ even if φ is convex (2.3)	trivial	$\nexists$ (2.6)

Proposition 2.2 *For every positive real numbers $\Sigma, \kappa > 0$ and a κ-Lipschitz continuous convex function $\varphi : [A, B] \to \mathbb{R}^+$, the function* APXARGMIN *(Alg. 1) finds a point D' such that $\varphi(D')$ is a Σ-approximation of $\varphi^{\min}$ in $O(\log \frac{\kappa(B-A)}{\Sigma})$ queries to φ.*

Proof. Using the convexity of φ, we first note that throughout its iterations the algorithm keeps track of an (ever decreasing in size) interval $[A', B']$, where A', B' are as defined in line 9, in which a minimum of φ is located.

Due to the convexity of φ, its minimum lies either (i) not below the intersection of the half plane $x \geq A'$ and the half plane above the line l_{CB}, or (ii) not below the intersection of the half plane $x \leq B'$ and the half plane above the line l_{AC}. After step 12, the value of κ' is an upper bound to the Lipschitz constant on $[A', B']$ because of convexity. Therefore, the maximum difference between the true minimum and $\varphi(D')$ is at most $\kappa'(B' - A')/2$. If this term is less than Σ, the algorithm exits the while loop and reports D'.

The number of queries done throughout the execution of the algorithm derives from the fact that in each iteration the length of the interval is at most half of the length of its predecessor. □

Remark 1 *The $O(\log(\kappa(B - A)/\Sigma))$ term can be tight when φ is 2-piecewise linear with slopes $-\kappa, \kappa$. However, the actual number of queries performed by* APXARGMIN *may be significantly less when the series of κ' generated by the algorithm is strictly monotone decreasing, e.g., when φ is nearly flat around its minimum.*

1: **Function ApxArgMin**$(\varphi, [A, B], \Sigma)$
2: Let $A' \leftarrow A,\ B' \leftarrow B,\ \kappa' \leftarrow 3\Sigma/(B' - A')$
3: **while** $\kappa'(B' - A') > 2\Sigma$ **do**
4: Divide $[A', B']$ into 4 equal-size subintervals and perform queries to φ on the corresponding 5 endpoints
5: let D' be an argmin of these values
6: **if** φ is not monotone on these endpoints **then** let $C' \leftarrow D'$
7: **elseif** φ is increasing on these points **then** let C' be the neighbor of A'
8: **else** let C' be the neighbor of B'
9: let A' be the neighbor of C' to the left and B' be its neighbor to the right
10: Let l_{AC} be the line that passes through $(A', \varphi(A'))$ and $(C', \varphi(C'))$, and let Δ_{AC} be its slope
11: Let l_{CB} be the line that passes through $(C', \varphi(C'))$ and $(B', \varphi(B'))$, and let Δ_{CB} be its slope
12: $\kappa' \leftarrow \max\{|\Delta_{AC}|, |\Delta_{CB}|\}$
13: **end while**
14: **return** D'

Algorithm 1: FUNCTION APXARGMIN$(\varphi, [A, B], \Sigma)$ returns an argmin of a Σ-approximation for $\varphi^{\max}$ of a convex function φ

Remark 2 *In Proposition 2.2, as well as in all subsequent algorithmic results in this paper, the value of κ need not be given explicitly, even though the running time may depend on it.*

Proposition 2.3 *Let $\Sigma > 0$ be an arbitrary real number, A be an arbitrary positive integer number, and let $\kappa := 2A\Sigma$. It is necessary, in general, to perform at least $A/3$ oracle calls to a κ-Lipschitz continuous monotone convex function $\varphi : [0, A] \to \mathbb{R}^+$ in order to construct for it a succinct Σ-approximation.*

Proof. Let $\varphi^* : [0, A] \to \mathbb{R}^+$ be an A-piecewise linear convex function with breakpoints in every integer number in $[0, A]$, with $\varphi^*(0) = 0, \varphi^*(1) = 4\Sigma$ and such that in every piece the slope of φ^* increases by 4Σ. Hence, $\varphi^{*\max} = \varphi^*(A) = 2A(A+1)\Sigma$, φ^* is κ-Lipschitz continuous, and has an exact representation in $A + 1$ space. For every $i = 1, \ldots, A-1$, let φ_i be an $(A-1)$-piecewise linear convex function that coincides with φ^* everywhere excepts for the interval $(i-1, i+1)$, in where the function is linear increasing from $\varphi^*(i-1)$ to $\varphi^*(i+1)$, so it has an exact representation in A space. Note that the maximum (additive) difference between φ_i and φ^* is in i and is 2Σ. Let $\Phi = \{\varphi^*, \varphi_1, \ldots, \varphi_{A-1}\}$. Thus, Φ is a family of convex monotone functions, in which each pair φ^*, φ_i has a difference of at least 2Σ in exactly one integer point in $[0, A]$, i.e., in i. This implies that every additive Σ-approximation of φ^* and of φ_i must distinguish between φ^* and φ_i. Note that φ^* and φ_i differ in an interval of length 2, whose endpoints are integers, and which we call the critical interval i. Note also that any interval of length 3 contains an interval of length 2 whose endpoints are integers.

We define an adversarial oracle that returns the value $\varphi^*(x)$ for all $x \in [0, A]$. Suppose that the number of queries to the adversarial oracle is bounded by $A/3 - 1$, so the size of the representation is bounded by this number as well. But then there exists an index i' and at least one interval of length at least 3, which contains the critical interval i' in which no queries were made. Therefore, the answers of the adversarial oracle are compatible with both φ^* and $\varphi^{i'}$, so these functions are indistinguishable even though any additive Σ-approximation must distinguish between them. □

Propositions 2.2 and 2.3 imply that constructing a succinct additive approximation function is strictly harder than finding an additive approximation of $\varphi^{\min}$.

Proposition 2.4 *Let $\Pi > 1$ be an arbitrary real number, and $\varphi : [0, 2] \to \mathbb{R}^+$ be a 1-Lipschitz continuous convex function. It is not possible, in general, to find a Π-approximation of $\varphi^{\min}$ in any finite number of queries to φ.*

Proof. Consider the family Φ consisting of all nonnegative convex functions $\varphi : [0, 2] \to [0, 1/2]$ that are 1-Lipschitz continuous with $\varphi(0) = \varphi(2) = 1/2$. We develop a resisting (adversarial) oracle that imitates a function φ in Φ to which it is hard to decide whether it is strictly positive. The oracle starts from the entire set Φ and tries to answer each call of the approximation algorithm in what is, in some sense, the "worst" possible way . Let $\Phi' \subseteq \Phi$ be the set of functions compatible with all answers of the oracle so far. The oracle answers queries so that the property that Φ' contains at least one function that attains value zero, as well as at least one strictly positive function remains invariant throughout the entire process of answering queries.

In order to produce "bad" answers, the oracle keeps record of (i) the set Q consisting of all query points $(x, \varphi(x))$ it answered so far (we let Q_X denote the projection of the set Q on the x-axis), (ii) the largest interval $[B, C]$ on which the value of at least one function in Φ' is always zero, (iii) the rightmost query point L to the left of B, i.e. $L = \max\{x \le B \mid x \in Q_X\}$, and (iv) the leftmost query point R to the right of C, i.e., $R = \min\{x \ge C \mid x \in Q_X\}$. At the beginning of the process Q is set to $\{(0, 1/2);\ (2, 1/2)\}$. Because φ is 1-Lipschitz continuous, we get that $[B, C] = [1/2, 3/2]$. Therefore, $L = 0$ and $R = 2$. We denote by $\varphi_D : [0, L] \to [0, 1/2]$ the piecewise linear function whose breakpoints are $\{(x, y) \in Q \mid x \le L\}$. By the construction of Q as explained below, it is easy to see that φ_D is monotone decreasing. Similarly, we

denote by $\varphi_I : [R, 2] \to [0, 1/2]$ the piecewise linear function whose breakpoints are $\{(x, y) \in Q \mid x \geq R\}$. By the construction of Q as explained below, it is easy to see that φ_I is monotone increasing.

In subsequent stages, when the oracle gets a query point $x \leq L$, it returns the value $\varphi_D(x)$, so the set Φ' does not change. If the oracle gets a query point $x \geq R$, it returns the value $\varphi_I(x)$, so again the set Φ' does not change. Otherwise, suppose that $x \leq (B + C)/2$ (if $x > (B + C)/2$ we proceed in a similar way). The oracle returns then the value v of the linear function between $(L, \varphi(L))$ and $((B + 2C)/3, 0)$ on x. The following "bookkeeping" is done. (i) $Q \leftarrow Q \cup \{(x, v)\}$, (ii) φ_D is updated by adding to it the linear piece between $(L, \varphi(L))$ and $(x, \varphi(x))$, (iii) $B \leftarrow (B + 2C)/3$, (iv) $L \leftarrow x$, (v) Φ' is updated to consist of all nonnegative convex functions $\varphi : [0, 2] \to [0, 1/2]$ that are 1-Lipschitz continuous and compatible with the query points in Q. In this way, the size of the updated interval $[B, C]$ is the third of the size of the old interval. Note that φ_D concatenated with the linear function between $(L, \varphi(L))$ and $(R, \varphi(R))$, and with φ_I is a strictly positive convex function that belongs to Φ', while φ_D concatenated with the linear function between $(L, \varphi(L))$ and $(B, 0)$, and with the zero function over $[B, C]$, and with φ_I is a convex function that attains the value 0 and also belongs to Φ'. Therefore, the invariant that Φ' contains at least one function that attains value zero, as well as at least one strictly positive function remains valid. Consequently, it is not possible to find, at any step of the process, a single Π-approximation value to all the minimum values of the functions in Φ' for any value of Π, and the process must continue forever. □

Remark 3 *Note that the above proof will fail for functions defined over discrete intervals* $\{0, 1, \ldots, 2A\}$. *In this case the process stops when the interval* $[B, C]$ *contains exactly one integer point. This will take at most* $O(\log_3 2A)$*, i.e, polynomial in the (binary) input size. This is no surprise as constructing a succinct* Π*-approximation of a convex function over a discrete interval can be done in polynomial time [HKL*$^+$*14, Prop.4.5-4.6].*

Corollary 2.5 *Let* $\Pi > 1$ *be an arbitrary real number, and* $\varphi : [0, 2] \to \mathbb{R}^+$ *be a 1-Lipschitz continuous convex function. It is not possible, in general, to construct a succinct* Π*-approximation for* φ *in any finite number of queries to* φ*.*

Proposition 2.6 *Let* $\Pi > 1$ *be an arbitrary real number, and* $\varphi : [0, 1] \to [0, 1]$ *be a 2-Lipschitz continuous monotone (and not necessarily convex) function. It is not possible, in general, to construct a succinct* Π*-approximation for* φ *in any finite number of queries to* φ*.*

Proof. Consider the family $\Phi = \{\varphi_i \mid i \in [0, \frac{1}{2})\}$, where

$$\varphi_i(x) = \begin{cases} 0 & \text{if } x < i; \\ 2(x - i) & \text{if } i \leq x < 2i; \\ x & \text{otherwise.} \end{cases}$$

Note that all functions in Φ are monotone 2-Lipschitz continuous. Suppose it is possible to build in finite time a Π-approximation function $\tilde{\varphi}$ for any function in Φ and that x^* is the smallest query point greater than zero generated by the algorithm that builds $\tilde{\varphi}$. Since this algorithm is finite, x^* is well defined. We define an adversarial oracle that returns $\varphi(x) = x$ for all $x \in [0, 1]$. This oracle is compatible with both φ_0 and $\varphi_{x^*/2}$ so $\tilde{\varphi}$ must Π-approximate both these functions at the same time. But since $\varphi_{x^*/2}(x^*/4) = 0$ and $\varphi_0(x^*/4) = x^*/4$ this cannot hold true. The contradiction arises from our assumption that it is possible to build $\tilde{\varphi}$ in finite time. □

Corollary 2.7 *For any fixed multiplicative error $\Pi > 1$, it is not possible, in general, to Π-approximate in finite time a continuous DP, even if the time horizon consists of a single period, the cost function is either monotone or convex, and the cost function is Lipschitz continuous with a fixed Lipschitz constant.*

Using similar arguments to the proofs of Prop. 2.3 and 2.4 we can show the following results establishing hardness of approximation for the NNV.

Proposition 2.8 *Let $\Sigma > 0$ be an arbitrary real number, A be an arbitrary positive integer, and consider a continuous nonlinear newsvendor problem with demand 0 or A with equal probability and cost functions given by a value oracle. It is not possible, in general, to determine in less than $A/3$ oracle calls to the shortage cost function an additive Σ-approximation of the optimal cost, or an order quantity that yields such cost, even if the holding and shortage costs are monotone $4A\Sigma$-Lipschitz continuous functions and the ordering cost is identically zero.*

Proof. Given that the demand is in $\{0, A\}$, only order quantities $x \in [0, A]$ should be considered. Therefore, we want to compute:

$$\arg \min_{x\in[0,A]} \mathbb{E}\left[b(D - x) + h(x - D)\right],$$

and by expanding the expected value and noticing that $(0 - x)$ and $(x - A)$ are always nonpositive, we can rewrite the expression as:

$$\arg \min_{x\in[0,A]} \frac{1}{2}b(A - x) + \frac{1}{2}h(x).$$

Let $\varphi^*(x)$ and $\varphi_i(x)$ be as in the proof of Prop. 2.3, and define $b(A - x) = 4A(A + 1)\Sigma - 2\varphi_i(x)$, $h(x) = 2\varphi^*(x)$. Notice that

$$\frac{1}{2}b(A - x) + \frac{1}{2}h(x) = 2A(A + 1)\Sigma + \varphi^*(x) - \varphi_i(x).$$

Therefore, the objective function value of the newsvendor problem is $2A(A + 1)\Sigma$ for all x except for $i - 1 < x < i + 1$, and its minimum value is $2A(A + 1) - 2\Sigma$ for $x = i$. In order to find an order quantity that Σ-approximates the optimal cost, any algorithm must determine the interval $(i - 1, i + 1)$. Using the same argument as in the proof of Prop. 2.3, this requires at least $A/3$ queries to the cost functions, implying a running time that is not polynomial in $\log A$.

□

Proposition 2.9 *Let $\Pi > 1$ be an arbitrary real number, and consider a continuous nonlinear newsvendor problem with demand 0 or 2 with equal probability and cost functions given by a value oracle. It is not possible, in general, to determine in finite time a multiplicative Π-approximation of the optimal cost, or an order quantity that yields such cost, even if the holding and shortage costs are monotone convex 1-Lipschitz continuous functions, and the ordering cost is identically zero.*

Proof. Consider a continuous NNV problem with possible demand $D \in \{0, 2\}$, each with probability $\frac{1}{2}$. Similarly to the proof of Prop. 2.8, the optimal order quantity is given by:

$$\arg \min_{x\in[0,2]} \frac{1}{2}b(2 - x) + \frac{1}{2}h(x).$$

We now construct an adversarial oracle for the functions b, h. Given a set of query points Q at which the value of the functions b, h has already been returned by oracle, let φ_D and φ_I be as in the proof Prop. 2.4,

and define:

$$b(2-x) = \begin{cases} \varphi_D(x) & \text{if } x \le L \\ 0 & \text{otherwise} \end{cases} \qquad h(x) = \begin{cases} 0 & \text{if } x < R \\ \varphi_I(x) & \text{otherwise.} \end{cases}$$

By the construction given in the proof of Prop. 2.4, these are convex 1-Lipschitz continuous function whose sum is zero in an interval $[B, C]$. The argument of that proof shows that the adversarial oracle ensures that the interval $[B, C]$ in which the optimal cost could be zero shrinks indefinitely, but cannot be determined in finite time by any algorithm. To determine an order quantity that Π-approximates the optimal cost we must find a point inside $[B, C]$, but this is impossible in finite time. □

Thm. 1.2 follows directly from the two propositions above.

2.4 (Σ, Π)-approximation functions and their calculus

Let $\mathcal{F}([A, B])$ be the family of either convex or monotone functions φ over interval $[A, B]$ that are κ-Lipschitz continuous. Let (P) be the problem of the construction of a succinct representation for a δ-approximation of φ. The previous subsection shows that $\text{Compl}([A, B], \delta)$ is not polynomial in $\log(B - A) + \log(1/\delta)$ (resp. $1/\delta$), when the measure of approximation is δ-additive (resp. δ-relative) error. In other words, the notions of efficient additive succinct approximation and efficient multiplicative approximation are too strong to be used for the family $\mathcal{F}([A, B])$. In this section we introduce a natural measure of approximation for which $\text{Compl}([A, B], \delta)$ is polynomial in $\log(B - A) + 1/\delta$, as will be shown in Section 3.

Definition 2.10 ((Σ, Π)-approximation functions) *Let $\Sigma \ge 0,\ \Pi = 1 + \epsilon \ge 1$, and $r, \tilde{r} \ge 0$ be arbitrary real numbers. We say that $\tilde{r}$ is a* (Σ, Π)-approximation *of r if $r \le \tilde{r} \le \Pi r + \Sigma$. Let $\varphi, \tilde{\varphi} : D \to \mathbb{R}^+$ be real-valued functions over continuous domain D. The function $\tilde{\varphi} : D \to \mathbb{R}^+$ is said to be a* (Σ, Π)-approximation *of φ if $\varphi(x) \le \tilde{\varphi}(x) \le \Pi\varphi(x) + \Sigma$ for all $x \in D$. Let $D = [A, B]$ be an interval on the real line. We call a function $\tilde{\varphi}$ a* succinct (Σ, Π)-approximation *of φ if it admits a representation in space polynomial in $\log \varphi^{\max} + \log(B - A) + \log(1/\Sigma) + 1/\epsilon$ and is a (Σ, Π)-approximation of φ.*

(Σ, Π)-approximation generalizes additive and multiplicative approximations in the sense that $(\Sigma, 1)$-approximation is additive Σ-approximation and $(0, \Pi)$-approximation is Π-multiplicative approximation. The next proposition tells us that (Σ, Π)-approximation is still a too strong measure of error when dealing with "ordinary" (as opposed to Lipschitz) continuous functions.

Proposition 2.11 *Let $1 > \Sigma > 0,\ \Pi > 1$ be arbitrary real numbers, and $\varphi : [0, 1] \to [0, 1]$ be a continuous monotone function. It is not possible, in general, to construct a succinct (Σ, Π)-approximation for φ in any finite number of queries to φ.*

Proof. Consider the family Φ consisting of all nonnegative monotone increasing functions $\varphi : [0, 1] \to [0, 1]$ with $\varphi(0) = 0$ and $\varphi(1) = 1$. Suppose it is possible to build in finite time a (Σ, Π)-approximation function $\tilde{\varphi}$ for any function in Φ and that x^* is the smallest query point greater than zero generated by the algorithm that builds $\tilde{\varphi}$. Since this algorithm is finite, x^* is well defined. We define an adversarial oracle that returns $\varphi(x) = 1$ for all $x \in (0, 1]$.

Let φ_0 be a nonnegative monotone increasing function $\varphi_0 : [0, 1] \to [0, 1]$ with $\varphi_0(0) = \varphi_0(x^*/2) = 0$ and $\varphi_0(1) = 1$. Let φ_1 be a nonnegative monotone increasing function $\varphi_1 : [0, 1] \to [0, 1]$ with $\varphi_1(0) = 0$ and $\varphi_0(x^*/2) = \varphi_0(1) = 1$. Clearly, $\varphi_0, \varphi_1 \in \Phi$. Moreover, our oracle is compatible with both φ_0 and φ_1, so $\tilde{\varphi}$ must (Σ, Π)-approximate both these functions at the same time. But since $\varphi_0(x^*/2) = 0$ and $\varphi_1(x^*/2) = 1$ and since $\Sigma < 1$ this cannot hold true. The contradiction arises from our assumption that it is possible to build $\tilde{\varphi}$ in finite time. □

In view of Proposition 2.11, in the rest of the paper we will limit our discussion to Lipschitz continu-

ous functions and show that (Σ, Π)-approximations of such (either convex or monotone) functions can be computed efficiently.

Definition 2.12 (Efficient (Σ, Π)-approximation functions) *Let $\Sigma \geq 0,\ \Pi = 1 + \epsilon \geq 1$, and $\varphi : [A, B] \to \mathbb{R}^+$ be a κ-Lipschitz continuous function. Let $\tilde{\varphi}$ be a succinct (Σ, Π)-approximation of φ. We call $\tilde{\varphi}$* efficient *if it can be constructed in time polynomial in $\log \varphi^{\max} + \log(B - A) + \log(1/\Sigma) + 1/\epsilon + \log \kappa$.*

The following proposition provides a set of general computational rules of (Σ, Π)-approximation functions. Its validity follows directly from the definition of (Σ, Π)-approximation functions. Our approximation scheme relies on such rules, that allow us to perform several operations on (Σ, Π)-approximation functions, while bounding the error.

Proposition 2.13 (Calculus of (Σ, Π)-approximation Functions) *For $i = 1, 2$ let $\Sigma_i \geq 0,\ \Pi_i \geq 1$, let $\varphi_i : D \to \mathbb{R}^+$ be an arbitrary function over continuous domain D, and let $\tilde{\varphi}_i : D \to \mathbb{R}^+$ be a (Σ_i, Π_i)-approximation of φ_i. Let $\psi_1 : D \to D$, and let $\alpha_i \in \mathbb{R}^+$. The following properties hold:*

1. *φ_1 is a (0,1)-approximation of itself,*
2. *(linearity of approximation) $\alpha_1\tilde{\varphi}_1 + \alpha_2$ is a $(\alpha_1\Sigma_1, \Pi_1)$-approximation of $\alpha_1\varphi_1 + \alpha_2$,*
3. *(summation of approximation) $\tilde{\varphi}_1 + \tilde{\varphi}_2$ is a $(\Sigma_1 + \Sigma_2, \max\{\Pi_1, \Pi_2\})$-approximation of $\varphi_1 + \varphi_2$,*
4. *(composition of approximation) $\tilde{\varphi}_1(\psi_1)$ is a (Σ_1, Π_1)-approximation of $\varphi_1(\psi_1)$,*
5. *(minimization of approximation) $\min\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $(\max\{\Sigma_1, \Sigma_2\}, \max\{\Pi_1, \Pi_2\})$-approximation of $\min\{\varphi_1, \varphi_2\}$,*
6. *(maximization of approximation) $\max\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $(\max\{\Sigma_1, \Sigma_2\}, \max\{\Pi_1, \Pi_2\})$-approximation of $\max\{\varphi_1, \varphi_2\}$,*
7. *(approximation of approximation) If $\varphi_2 = \tilde{\varphi}_1$ then $\tilde{\varphi}_2$ is a $(\Sigma_2 + \Pi_2\Sigma_1, \Pi_1\Pi_2)$-approximation of φ_1.*

The following technical proposition is used to bound the Lipschitz constant of a summation of compositions of Lipschitz continuous functions and will be used in order to show that the Lipschitz constants do not get too large while using the Bellman optimality equation (3).

Proposition 2.14 (Lipschitz Continuity Invariant) *Let $[A, B]$ be a compact interval, and let $C(x) = [C^{\min}(x), C^{\max}(x)]$ be a compact interval for every $x \in [A, B]$, where $C^{\min}(\cdot), C^{\max}(\cdot)$ are each κ_C-Lipschitz continuous functions. Let $\varphi_i : [A, B] \to \mathbb{R}^+$ be κ_φ-Lipschitz continuous functions over $[A, B]$, and let $\psi_i : [A, B] \otimes C \to [A, B]$ be κ_ψ-Lipschitz continuous functions for $i = 1, \ldots, n$. Define function $\varphi : [A, B] \to \mathbb{R}^+$ as*

$$\varphi(x) = \min_{y \in C(x)} \left\{ \sum_{i=1}^{n} \varphi_i(\psi_i(x, y)) \right\}. \tag{4}$$

Then $\varphi(\cdot)$ is an $n\kappa_\varphi\kappa_\psi(1 + \kappa_C)$-Lipschitz continuous function.

Proof. We show that φ is a Lipschitz continuous function with the claimed Lipschitz constant. Let

$$\alpha(x, y) := \sum_{i=1}^{n} \varphi_i(\psi_i(x, y)).$$

Note that by composition of Lipschitz continuous functions we get that $\varphi_i(\psi_i(x, y))$ is a $\kappa_\varphi\kappa_\psi$-Lipschitz continuous function, $i = 1, \ldots, n$. Note also that due to summation of Lipschitz continuous functions we have that $\alpha(x, y)$ is a $n\kappa_\varphi\kappa_\psi$-Lipschitz continuous function.

Let $x', x'' \in [A, B]$. We need to show that $|\varphi(x') - \varphi(x'')| \leq \kappa|x' - x''|$ for $\kappa = n\kappa_\varphi\kappa_\psi(1 + \kappa_C)$. Let $y' \in C(x')$ and $y'' \in C(x'')$ be argmins of $\varphi(x'), \varphi(x'')$, respectively, i.e., $\varphi(x') = \alpha(x', y'),\ \varphi(x'') = \alpha(x'', y'')$.

Suppose first that $y' \in C(x'')$. Then we have

$$\varphi(x'') - \varphi(x') = \alpha(x'', y'') - \alpha(x', y') \leq \alpha(x'', y') - \alpha(x', y') \leq n\kappa_\varphi\kappa_\psi|x'' - x'|, \tag{5}$$

where the first inequality is due to the optimality of y'' coupled with the fact that $y' \in C(x'')$, and the second inequality is due to α being $n\kappa_\varphi\kappa_\psi$-Lipschitz continuous.

Suppose now that $y' \notin C(x'')$, say $y' < C^{\min}(x'')$ (we treat the case $y' > C^{\max}(x'')$ in a similar way). Let y''' be the point in $C(x'')$ nearest to y', i.e., $y''' = C^{\min}(x'')$. We get that

$$|y''' - y'| = |C^{\min}(x'') - y'| \leq |C^{\min}(x'') - C^{\min}(x')| \leq \kappa_C|x'' - x'|, \tag{6}$$

where the first inequality is due to the fact that $y' < C^{\min}(x'')$ (and therefore $C^{\min}(x') \leq y' < C^{\min}(x'')$) and the second inequality follows from $C^{\min}(\cdot)$ being κ_C-Lipschitz continuous. We conclude that

$$\alpha(x'', y'') \leq \alpha(x'', y''') \leq \alpha(x'', y') + n\kappa_\varphi\kappa_\psi|y''' - y'| \leq \alpha(x'', y') + n\kappa_\varphi\kappa_\psi\kappa_C|x'' - x'|, \tag{7}$$

where the first inequality is due to the optimality of y'' coupled with the fact that $y''' \in C(x'')$, the second inequality is because of α being $n\kappa_\varphi\kappa_\psi$-Lipschitz continuous, and the last inequality is due to (6). Plugging (7) in the first inequality of (5) we get that the Lipschitz constant of $\varphi(\cdot)$ in this case is at most $n\kappa_\varphi\kappa_\psi(1 + \kappa_C)$, as needed. □

Computational rules for (Σ, Π)-approximations specialized to *convex* functions can be constructed as well.

Proposition 2.15 (Approximation of summation of composition of convex functions) *Let $\Sigma_i \geq 0,\ \Pi_i \geq 1$ for $i = 1, \ldots, n$, and let $[A, B]$ be a compact interval. Let $C(x)$ be a compact interval for every $x \in [A, B]$, and suppose that $[A, B] \otimes C$ is a convex set. Let $\varphi_i : [A, B] \to \mathbb{R}^+$ be a convex function over $[A, B]$, $\breve{\varphi}_i$ be a convex (Σ_i, Π_i)-approximation of φ_i, and $\psi_i : [A, B] \otimes C \to [A, B]$ be an affine function, $i = 1, \ldots, n$. Define functions $\varphi, \bar{\varphi} : D \to \mathbb{R}^+$ as*

$$\varphi(x) = \min_{y \in C(x)} \left\{ \sum_{i=1}^{n} \varphi_i(\psi_i(x, y)) \right\} \quad \text{and} \quad \bar{\varphi}(x) = \min_{y \in C(x)} \left\{ \sum_{i=1}^{n} \breve{\varphi}_i(\psi_i(x, y)) \right\}. \tag{8}$$

Then, φ is convex over $[A, B]$ and $\bar{\varphi}$ is a convex $(\Sigma_1 + \ldots + \Sigma_n, \max\{\Pi_1, \ldots, \Pi_n\})$-approximation of φ.

Proof. Let us fix $x \in D$. By summation of approximation and composition of approximation (i.e., properties 3 and 4 of Proposition 2.13), $\sum_{i=1}^{n} \breve{\varphi}_i(\psi_i)(x, \cdot)$ is a $(\Sigma_1 + \ldots + \Sigma_n, \max\{\Pi_1, \ldots, \Pi_n\})$-approximation function of $\sum_{i=1}^{n} \varphi_i(\psi_i)(x, \cdot)$ for every fixed x.

We now prove the convexity of $\varphi(\cdot)$. The proof of convexity of $\bar{\varphi}(\cdot)$ is similar. It suffices to prove that for every $m, M \in D$ $(m < M)$ we have $2\varphi(\frac{m+M}{2}) \leq \varphi(m) + \varphi(M)$. Let $y_m \in C(m)$ be such that $\varphi(m) = \sum_{i=1}^{n} \varphi_i(\psi_i(m, y_m))$. Let $y_M \in C(M)$ be such that $\varphi(M) = \sum_{i=1}^{n} \varphi_i(\psi_i(M, y_M))$. For every $i = 1, \ldots, n$ let a_i, b_i, c_i be real numbers such that $\psi_i(x, y) = a_i x + b_i y + c_i$. For every $x \in D$ let $f_{i,x}(\cdot) = \varphi_i(\psi_i(x, \cdot))$. Note that due to the convexity of φ_i and the fact that ψ_i is affine in its variables we get that $f_{i,x}(\cdot)$ is convex in $C(x)$. Note also that

$$\begin{aligned} f_{i,m}(y_m) &= \varphi_i(a_i m + b_i y_m + c_i) \\ &= \varphi_i\left(a_i(\tfrac{m+M}{2}) - a_i(\tfrac{M-m}{2}) + b_i y_m + c_i\right) \\ &= \varphi_i\left(a_i(\tfrac{m+M}{2}) + b_i(y_m - \tfrac{a_i(M-m)}{2b_i}) + c_i\right) \\ &= f_{i,\frac{m+M}{2}}\left(y_m - \tfrac{a_i(M-m)}{2b_i}\right). \end{aligned} \tag{9}$$

Similarly, we have that

$$f_{i,M}(y_M) = f_{i,\frac{m+M}{2}}\left(y_M + \frac{a_i(M-m)}{2b_i}\right). \tag{10}$$

Using (9)-(10) and summing over i we get that

$$\begin{aligned}\varphi(m)+\varphi(M) &= \textstyle\sum_{i=1}^{n} f_{i,\frac{m+M}{2}}\left(y_m - \frac{a_i(M-m)}{2b_i}\right) + f_{i,\frac{m+M}{2}}\left(y_M + \frac{a_i(M-m)}{2b_i}\right)\\ &\geq 2\textstyle\sum_{i=1}^{n} f_{i,\frac{m+M}{2}}\left(\frac{y_m+y_M}{2}\right)\\ &\geq 2\varphi\left(\frac{m+M}{2}\right),\end{aligned}$$

where the first inequality is due to the convexity of $f_{i,x}(\cdot)$ and the second inequality holds true because the convexity of $D \otimes C$ implies that $\frac{y_m+y_M}{2} \in C(\frac{m+M}{2})$. This completes the proof of convexity of $\varphi(\cdot)$ over D. □

Remark 4 *If the set $C(x)$ were a (fixed) compact interval that does not depend on the value of x then we could get a much shorter proof using the following well-known result.*

Proposition 2.16 *Let $a, b, c : A \to \mathbb{R}$ be functions over a convex set A, let b, c be convex over A, and let $f(\cdot,\cdot)$ be an affine function over its variables. For all $y \in A$ let $a(y) = \min_{x\in A}\{b(x) + c(f(x,y))\}$. Then a is a convex function over A.*

Note that the $\breve{\varphi}_i(\psi_i(x,\cdot))$ are convex and the minimization in (8) is taken over the entire compact interval $C(x)$. Therefore, by applying function ApxArgMin we can compute efficiently an additive approximation for $\bar{\varphi}(x)$, for any $x \in [A, B]$. In Proposition 3.4 of the next section we will deal with a similar equation but with $\breve{\varphi}_i$ monotone rather than convex. We will perform the minimization over a certain finite subset of $C(x)$ of cardinality $O(\log_\Pi \frac{\epsilon\varphi^{\max}}{\Sigma})$ that we call a (Σ,Π)-approximation set.

3 Computing (Σ,Π)-approximation functions via (Σ,Π)-approximation sets

The basic idea underlying succinct (Σ,Π)-approximation functions is to keep only a number of points in the domain that is logarithmic (with base Π) in $\frac{\varphi^{\max}}{\Sigma}$, in what is called a (Σ,Π)-approximation set, and then interpolate between these points to approximate the original function φ. If φ is a convex (resp. monotone) function, we use a piecewise linear function (resp. a step function) for the interpolation. We give some definitions first (recall that the definition of piecewise linear extension is given at the end of Section 1).

Let $\varphi : D \to R$ be a function over a linearly ordered set D and let $D' \subseteq D$ be a finite set containing $D^{\min}, D^{\max}$. The *convex extension of φ induced by D'* is the function defined as the lower envelope of the convex hull of $\{(x,\varphi(x)) \mid x \in D'\}$, which is piecewise linear. (Note that if φ is convex over D then its convex extension induced by D' is simply its piecewise linear extension induced by D'.) Given $x \in [D^{\min}, D^{\max}]$, for $x < D^{\max}$ let $\text{next}(x, D') := \min\{y \in D' \mid y > x\}$, and for $x > D^{\min}$ let $\text{prev}(x, D') := \max\{y \in D' \mid y < x\}$. The *monotone extension of φ induced by D'* is

$$\hat{\varphi}(x) = \begin{cases} \varphi(x) & \text{if } x \in D';\\ \max\{\varphi(\text{prev}(x,D')), \varphi(\text{next}(x,D'))\} & \text{otherwise.}\end{cases}$$

We are now ready to give a formal definition of (Σ,Π)-approximation sets.

Definition 3.1 ((Σ,Π)-approximation sets) *Let $\Sigma > 0$ and $\Pi \geq 1$, let $\varphi : [A, B] \to \mathbb{R}^+$ be a unimodal function, and let $W \subseteq [A, B]$ be a finite set containing the endpoints A, B. Denote by $\hat{\varphi}$ the piecewise linear*

1: **Function FuncSearchInc**($\varphi, [A, B], \text{low}(\cdot), \text{high}(\cdot)$)
2: **if** $\varphi(B) \leq \text{high}(B)$ **then return** B
3: **while** $\varphi((A+B)/2) > \text{high}((A+B)/2)$ or $\varphi((A+B)/2) < \text{low}((A+B)/2)$ **do**
4: **if** $\varphi((A+B)/2) > \text{high}((A+B)/2)$ **then** $B \leftarrow (A+B)/2$ **else** $A \leftarrow (A+B)/2$
5: **end while**
6: **return** $(A+B)/2$

Algorithm 2: FUNCTION FUNCSEARCHINC($\varphi, [A, B], \text{low}(\cdot), \text{high}(\cdot)$) returns a point $x \in [A, B]$ that satisfies $\text{low}(x) \leq \varphi(x) \leq \text{high}(x)$, if such a point exists, and point B otherwise.

(monotone) extension of φ induced by W in case φ is convex (monotone), respectively. We say that W is a (Σ, Π)-approximation set of φ *if $\hat{\varphi}$ is a (Σ, Π)-approximation function of φ. If the set $\{(x, \varphi(x)) \mid x \in W\}$ is sorted by x, we call it the* canonical representation of $\hat{\varphi}$.

Remark 5 *By definition, W is a $(0, 1)$-approximation set of $\hat{\varphi}$.*

Remark 6 *By definition, if φ is a κ-Lipschitz continuous convex function, then so is $\hat{\varphi}$.*

The basic routine that we employ to construct (Σ, Π)-approximation sets is essentially a modified binary search, called FUNCSEARCHINC and given in Algorithm 2. This routine takes as input either a convex or a monotone increasing Lipschitz continuous function $\varphi : [A, B] \to \mathbb{R}^+$, and returns a point that is "sandwiched" between two given functions $\text{low}(\cdot), \text{high}(\cdot)$. We define function FUNCSEARCHDEC similarly for monotone decreasing functions.

Proposition 3.2 *Given a κ-Lipschitz continuous function φ over the interval $[A, B]$ and functions low$(\cdot)$ and high$(\cdot)$ such that low$(\cdot) <$ high$(\cdot)$ and $\varphi(A) <$ low(A), suppose that one of the following holds:*

(i) φ is convex over $[A, B]$ and low$(\cdot)$, high$(\cdot)$ are linear functions;
(ii) φ is monotone increasing and low$(\cdot)$, high$(\cdot)$ are constants;
(iii) φ is a not necessarily monotone (Σ, Π)-approximation of a monotone increasing function ψ.

Then function FUNCSEARCHINC *(Alg. 2) returns in $O\left((1 + t_\varphi)(\log \frac{(B-A)\kappa}{\textit{high}(B) - \textit{low}(A)})\right)$ time a point x that satisfies low$(x) \leq \varphi(x) \leq$ high(x) in cases (i) and (ii), low$(x) \leq \psi(x) \leq$ high(x) in case (iii), if such a point exists, and the point B otherwise.*

Proof. Suppose first that the condition on line 2 is satisfied. If $\varphi(B) \geq \text{low}(B)$, we found a point satisfying the desired condition and we are done. Otherwise, $\varphi(A) < \text{low}(A), \varphi(B) < \text{low}(B)$, and we handle cases (i)-(iii) as follows. Regarding cases (i)-(ii), either the convexity or the monotonicity of φ implies that no point x satisfying $\text{low}(x) \leq \varphi(x) \leq \text{high}(x)$ exists, and we are done returning B as the failure condition. Regarding case (iii), we have that $\text{low}(B) > \varphi(B) \geq \psi(B)$ so the monotonicity of ψ implies that no point x satisfying $\text{low}(x) \leq \psi(x) \leq \text{high}(x)$ exists, and again we are done by returning B as the failure condition.

If the condition on line 2 is not satisfied, the algorithm keeps the inequalities $\varphi(B) > \text{high}(B)$ and $\varphi(A) < \text{low}(A)$ invariant throughout the execution of the while loop. Because φ is κ-Lipschitz continuous, the minimum size of an interval where such invariants hold is $\frac{\text{high}(B) - \text{low}(A)}{\kappa}$. Therefore, running time is $O(\log \frac{(B-A)\kappa}{\text{high}(B) - \text{low}(A)})$. Note that φ need not be monotone. □

In the next subsections we design algorithms to construct (Σ, Π)-approximation sets that are specialized to either monotone or convex functions. Our approach will recursively build such sets using an approximation for the DP recursion (3) as an approximate oracle for the next-stage value function. We start our discussion with monotone functions because of their direct connection to the NNV problem.

3.1 Monotone functions

The construction of (Σ, Π)-approximation sets relies on special structure of the functions to be approximated. In the monotone case, the approximate oracle obtained from the approximate DP recursion may not preserve the structure. In other words, even if we know that the underlying value function is monotone, we may obtain a non-monotone (Σ, Π)-approximation oracle for it, see Prop. 5.2. We first study the case in which we have direct access to the original structured function, then proceed to the more complicated case in which we only have access to an unstructured approximation oracle of the original structured function.

3.1.1 Direct access to φ

The first building block for our approximation scheme is a procedure that constructs a (Σ, Π)-approximation set for a monotone increasing function φ. We call it APXSETINC, see Alg. 3.

1: **Function ApxSetInc**$(\varphi, [A, B], \Sigma, \Pi)$
2: $x \leftarrow A$, $W \leftarrow \{A, B\}$
3: **while** $x < B$ **do**
4: $\quad x \leftarrow$ FUNCSEARCHINC$(\varphi, [x, B], \frac{1}{2}(\Pi + 1)\varphi(x) + \Sigma/2, \Pi\varphi(x) + \Sigma)$
5: $\quad W \leftarrow W \cup \{x\}$
6: **end while**
7: **return** W

Algorithm 3: FUNCTION APXSETINC$(\varphi, [A, B], \Sigma, \Pi)$

We define function APXSETDEC similarly for monotone decreasing functions, and define function APXSETMON to equal APXSETINC whenever φ is increasing and equal APXSETDEC otherwise.

Sometimes, direct access to a monotone increasing function $\varphi : [A, B] \to \mathbb{R}^+$ is time-consuming, therefore it is beneficial to construct an explicit succinct approximation of that function. We define function COMPRESSINC for this purpose.

1: **Function CompressInc**$(\varphi, [A, B], \Sigma, \Pi)$
2: $W \leftarrow$ APXSETINC$(\varphi, [A, B], \Sigma, \Pi)$
3: **return** $\{(x, \varphi(x)) \mid x \in W\}$ as a sorted array

Algorithm 4: FUNCTION COMPRESSINC returns a canonical representation of a step (Σ, Π)-approximation of an increasing function φ.

We similarly define function COMPRESSDEC for decreasing functions and define function COMPRESSMON to be COMPRESSINC when φ is increasing and to be COMPRESSDEC when φ is decreasing. We are now ready to state our next result. (Recall that the definition of canonical representations is given in Definition 3.1.)

Proposition 3.3 (Approximation of a monotone function with direct access) *Let $\varphi : [A, B] \to \mathbb{R}^+$ be a κ-Lipschitz continuous monotone function. Then for every constants $\Sigma > 0$, $\Pi = 1 + \epsilon > 1$ the following holds. (i)* APXSETMON$(\varphi, [A, B], \Sigma, \Pi)$ *and* COMPRESSMON$(\varphi, [A, B], \Sigma, \Pi)$ *construct a (Σ, Π)-approximation set W for φ of cardinality $O\left(\frac{1}{\epsilon} \log \frac{\epsilon\varphi^{\max}}{\Sigma}\right)$ in $O\left((1 + t_\varphi)(\frac{1}{\epsilon} \log \frac{\epsilon\varphi^{\max}}{\Sigma}) \log \frac{(B-A)\kappa}{\Sigma}\right)$ time. (ii)* COMPRESSMON$(\varphi, [A, B], \Sigma, \Pi)$ *returns the canonical representation of the monotone extension of φ induced by W, $\hat{\varphi}$, which is a monotone step (Σ, Π)-approximation function of φ. Moreover, the value of $\hat{\varphi}(\cdot)$ can be determined in $O(\log |W|)$ time at any point in $[A, B]$.*

Proof. Let $\epsilon = \Pi - 1$. We consider first the cardinality of W. By step 4 of the algorithm, the difference between the values of $\varphi(\cdot)$ on successive values of x is at least $\Sigma/2$, so after $O(\frac{1}{\epsilon})$ iterations we get that

$\varphi(x) > \Sigma/\epsilon$. By the same step of the algorithm, the ratio between the values of $\varphi(\cdot)$ on successive values of x is at least $1 + \epsilon/2$, so after $O(\log_\Pi \frac{\epsilon\varphi^{\max}}{\Sigma})$ more iterations we get that the value of $\varphi(x)$ must reach $\varphi^{\max}$. The bound on the cardinality of W follows from the fact that $O(\log_\Pi v) = O(\frac{\log v}{\epsilon})$ holds true for all $0 < \epsilon < 1$ and $v > 0$.

We next prove that W is a (Σ, Π)-approximation set of φ. We assume w.l.o.g. that φ is increasing. Clearly, $A, B \in W$. Let $x \in [A, B]$. If $x \in W$, then from the definition of the monotone extension of φ induced by W we get $\hat{\varphi}(x) = \varphi(x)$. Otherwise, $\hat{\varphi}(x) = \max\{\varphi(\text{prev}(x, W)), \varphi(\text{next}(x, W))\} = \varphi(\text{next}(x, W))$. By the construction of W we get that $\varphi(\text{next}(x, W)) \le \Pi\varphi(\text{prev}(x, W))+\Sigma$. By the monotonicity of φ we have $\varphi(\text{prev}(x, W)) \le \varphi(x) \le \varphi(\text{next}(x, W))$, so $\hat{\varphi}$ is indeed a (Σ, Π)-approximation function of φ.

The running time of the algorithm follows from the fact that each call to FUNCSEARCH takes $O((1 + t_\varphi) \log \frac{(B-A)\kappa}{\Sigma})$ time.

If $\hat{\varphi}$ is stored in a sorted array $\{(x, \varphi(x)) \mid x \in W\}$, then for every $x \in [A, B]$, we can determine the value of $\hat{\varphi}(x)$ in $O(\log |W|)$ time using binary search. □

In the next section we will show how to use functions APXSETMON and COMPRESSMON with a not necessarily monotone function ψ that approximates a monotone function φ. The key building block is still function FUNCSEARCHINC from Section 3, that also works on such functions ψ.

3.1.2 Approximate (indirect) access to φ

A difficulty that arises when working with continuous DPs is that the minimization over the action space in (3) may be too time-consuming, i.e. not doable in finite time because of the infinite action space. A further, more subtle difficulty is that we do not know how to efficiently construct an approximate *monotone* oracle for the value function. The next proposition deals with these two difficulties by constructing a not necessarily monotone oracle, and performing the minimization via an exhaustive search over a polylogarithmic-size subset of the action space, constructed as the union of a small number of (Σ, Π)-approximation sets.

Proposition 3.4 *For $i = 1, \dots, n$, let $\Sigma_i, \Sigma'_i \ge 0$, $\Pi_i, \Pi'_i \ge 1$, let $\varphi_i : [A, B] \to \mathbb{R}^+$, let $\tilde{\varphi}_i$ be a (Σ_i, Π_i)-approximation of φ_i, and let $\psi_i : [A, B] \otimes C \to [A, B]$ be a function such that for any fixed $x \in [A, B]$, $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is monotone over a compact interval $C(x) = [C^{\min}(x), C^{\max}(x)]$. Let m be an integer such that $1 \le m \le n$. For any $i = 1, \dots, m$ let $W_i(x) \subseteq C(x)$ be a (Σ'_i, Π'_i)-approximation set of $\tilde{\varphi}_i(\psi_i(x, \cdot))$ over $C(x)$. Let $\varphi, \tilde{\varphi} : [A, B] \to \mathbb{R}^+$ be such that*

$$\varphi(x) = \min_{y \in C(x)} \left\{ \sum_{i=1}^{n} \varphi_i(\psi_i(x, y)) \right\} \quad \textit{and} \quad \tilde{\varphi}(x) = \min_{y \in \bigcup_{i=1}^{m} W_i(x)} \left\{ \sum_{i=1}^{n} \tilde{\varphi}_i(\psi_i(x, y)) \right\}.$$

Suppose that function $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is monotone in one direction (e.g., increasing) for $i = 1, \dots, m$ and is monotone in the other direction (e.g., decreasing) for $i = m + 1, \dots, n$ and any fixed $x \in [A, B]$. Then $\tilde{\varphi}$ is a $\big(\sum_{i=1}^{m}(\Sigma'_i + \Pi'_i\Sigma_i) + \sum_{i=m+1}^{n} \Sigma_i, \max\{\Pi_1\Pi'_1, \dots, \Pi_m\Pi'_m, \Pi_{m+1}, \dots, \Pi_n\}\big)$-approximation of φ.

Proof. We prove that $\tilde{\varphi}$ is a $(\sum_{i=1}^{m} (\Sigma'_i+\Pi'_i\Sigma_i)+\sum_{i=m+1}^{n} \Sigma_i, \max\{\Pi'_1\Pi_1, \dots, \Pi'_m\Pi_m, \Pi_{m+1}, \Pi_{m+1}, \dots, \Pi_n\})$-approximation of φ.

Consider any fixed $x \in [A, B]$. Let $y^{@} \in \bigcup_{i=1}^{m} W_i(x)$ be a realizer of $\tilde{\varphi}(x)$; that is, $\tilde{\varphi}(x) = \sum_{i=1}^{n} \tilde{\varphi}_i(\psi_i(x, y^{@}))$. By composition of approximation (Proposition 2.13), $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is a (Σ_i, Π_i)-approximation of $\varphi_i(\psi_i(x, \cdot))$

for $i = 1, \ldots, n$, and therefore,

$$\varphi(x) = \min_{y \in C(x)} \left\{ \sum_{i=1}^{n} \varphi_i(\psi_i(x, y)) \right\} \leq \sum_{i=1}^{n} \varphi_i(\psi_i(x, y^{@})) \leq \sum_{i=1}^{n} \tilde{\varphi}_i(\psi_i(x, y^{@})) = \tilde{\varphi}(x).$$

Let y^* be the smallest realizer of $\varphi(x)$.

We first consider the scenario where $m \neq n$ and divide the analysis into two cases.
Case 1: $\tilde{\varphi}_1(\psi_1(x, \cdot))$ is increasing, and $\tilde{\varphi}_n(\psi_n(x, \cdot))$ is decreasing. In this case, for $i = 1, \ldots, m$, define $y'_i = y^*$ if $y^* \in W_i(x)$, and define $y'_i = \text{next}(y^*, W_i(x))$ if $y^* \notin W_i(x)$. Because $W_i(x)$ is a (Σ'_i, Π'_i)-approximation set of $\tilde{\varphi}_i(\psi_i(x, \cdot))$, we have $\tilde{\varphi}_i(\psi_i(x, y'_i)) \leq \Sigma'_i + \Pi'_i \tilde{\varphi}_i(\psi_i(x, y^*))$ for $i = 1, \ldots, m$. Let $y' = \min_{i=1,\ldots,m}\{y'_i\}$. For $i = 1, \ldots, m$, since $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is increasing, we have

$$\tilde{\varphi}_i(\psi_i(x, y')) \leq \tilde{\varphi}_i(\psi_i(x, y'_i)) \leq \Sigma'_i + \Pi'_i \tilde{\varphi}_i(\psi_i(x, y^*)) \leq \Sigma'_i + \Pi'_i(\Sigma_i + \Pi_i \varphi_i(\psi_i(x, y^*))), \tag{11}$$

where the last inequality follows from the fact that $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is a (Σ_i, Π'_i)-approximation of $\varphi_i(\psi_i(x, \cdot))$. For $i = m+1, \ldots, n$, since $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is decreasing and $y' \geq y^*$, we have $\tilde{\varphi}_i(\psi_i(x, y')) \leq \tilde{\varphi}_i(\psi_i(x, y^*))$. This, together with the fact that $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is a (Σ_i, Π_i)-approximation of $\varphi_i(\psi_i(x, \cdot))$, we have

$$\tilde{\varphi}_i(\psi_i(x, y')) \leq \Sigma'_i + \Pi_i \varphi_i(\psi_i(x, y^*)) \tag{12}$$

for $i = m+1, \ldots, n$. Because $y' \in \bigcup_{i=1}^{m} W_i(x)$, we have

$$\tilde{\varphi}(x) \leq \sum_{i=1}^{n} \tilde{\varphi}_i(\psi_i(x, y')). \tag{13}$$

From (11), (12), and (13), we have

$$\begin{aligned}
\tilde{\varphi}(x) &\leq \sum_{i=1}^{m} \tilde{\varphi}_i(\psi_i(x, y')) + \sum_{i=m+1}^{n} \tilde{\varphi}_i(\psi_i(x, y')) \\
&\leq \sum_{i=1}^{m} (\Sigma'_i + \Pi'_i(\Sigma_i + \Pi_i \varphi_i(\psi_i(x, y^*)))) + \sum_{i=m+1}^{n} (\Sigma_i + \Pi_i \varphi_i(\psi_i(x, y^*))) \\
&\leq \sum_{i=1}^{m} (\Sigma'_i + \Pi'_i \Sigma_i) + \sum_{i=m+1}^{n} \Sigma_i \ + \max\{\Pi'_1 \Pi_1, \ldots, \Pi'_m \Pi_m, \Pi_{m+1}, \Pi_{m+1}, \ldots, \Pi_n\} \varphi(x).
\end{aligned} \tag{14}$$

Case 2: $\tilde{\varphi}_1(\psi_1(x, \cdot))$ is decreasing, and $\tilde{\varphi}_n(\psi_n(x, \cdot))$ is increasing. In this case, for $i = 1, \ldots, m$, define $y'_i = y^*$ if $y^* \in W_i(x)$, and define $y'_i = \text{prev}(y^*, W_i(x))$ if $y^* \notin W_i(x)$. Letting $y' = \max_{i=1,\ldots,m}\{y'_i\} \leq y^*$ and following the same argument as in Case 1, it is easy to verify that inequality (14) holds.

Next, we consider the scenario where $m = n$. If $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is increasing for $i = 1, \ldots, n$, then inequality (11) holds for $i = 1, \ldots, n$, and therefore (14) remains valid. On the other hand, if $\tilde{\varphi}_i(\psi_i(x, \cdot))$ is decreasing for $i = 1, \ldots, n$, then inequality (12) holds for $i = 1, \ldots, n$, which implies that $\tilde{\varphi}(x) \leq \sum_{i=1}^{n} \tilde{\varphi}_i(\psi_i(x, y')) \leq \sum_{i=1}^{n} (\Sigma_i + \Pi_i \varphi_i(\psi_i(x, y^*))) \leq \sum_{i=1}^{n} \Sigma_i + \max\{\Pi_1, \ldots, \Pi_n\} \varphi_i(x) \leq \sum_{i=1}^{m} (\Sigma'_i + \Pi'_i \Sigma_i) + \sum_{i=m+1}^{n} \Sigma_i \ + \max\{\Pi'_1 \Pi_1, \ldots, \Pi'_m \Pi_m, \Pi_{m+1}, \Pi_{m+1}, \ldots, \Pi_n\} \varphi(x)$. This completes the proof of the proposition. □

Note that $\tilde{\varphi}$ in Prop. 3.4 is not ensured to be monotone. (Σ, Π)-approximation sets for functions that are neither monotone nor convex are not well defined. However, if the function φ that $\tilde{\varphi}$ approximates is a monotone function, we can still build for φ a (Σ, Π)-approximation set having access to $\tilde{\varphi}$ only, as the

following proposition shows. (See [HKL$^+$14, Sec. 4.2] for a detailed discussion on approximate indirect access to monotone *discrete* functions.)

Proposition 3.5 (Approximation of a monotone function via (Σ', Π')-approximation oracle) *Let $\varphi : [A, B] \to \mathbb{R}^+$ be a κ-Lipschitz continuous monotone function. Let $\bar{\varphi}$ be a not necessarily monotone (Σ', Π')-approximation function of φ. Then for every constants $\Sigma > 0, \Pi = 1+\epsilon > 1$ the following holds (i)* ApxSetMon$(\bar{\varphi}, [A, B], \Sigma, \Pi)$ *and* CompressMon$(\bar{\varphi}, [A, B], \Sigma, \Pi)$ *construct a set W of cardinality $O\left(\frac{1}{\epsilon} \log \frac{\epsilon \varphi^{\max}}{\Sigma}\right)$ in $O\left((1 + t_{\bar{\varphi}})(\frac{1}{\epsilon} \log \frac{\epsilon \varphi^{\max}}{\Sigma}) \log \frac{(B-A)\kappa}{\Sigma}\right)$ time. (ii)* CompressMon$(\bar{\varphi}, [A, B], \Sigma, \Pi)$ *returns the canonical representation of the monotone extension of $\bar{\varphi}$ induced by W, $\hat{\bar{\varphi}}$, which is a monotone step $(\Sigma + \Pi\Sigma', \Pi'\Pi)$-approximation function of φ. Moreover, $\hat{\bar{\varphi}}(\cdot)$ can be determined in $O(\log |W|)$ time at any point in $[A, B]$.*

Proof. Let $\epsilon = \Pi - 1$. Note that while $\bar{\varphi}$ is not necessarily monotone over $[A, B]$, by the construction of W it is monotone over W. The proofs of the bound on the cardinality of W and on the running time of the algorithm are identical to the ones given in Proposition 3.3.

We next prove that the monotone extension of $\bar{\varphi}$ induced by W, $\hat{\bar{\varphi}}$ is a $(\Sigma + \Pi\Sigma', \Pi'\Pi)$-approximation function of φ. Note that $A, B \in W$. Let $x \in [A, B]$. We assume w.l.o.g. that φ is increasing. If $x \in W$, then from the definition of the monotone extension of $\bar{\varphi}$ induced by W we get $\hat{\bar{\varphi}}(x) = \bar{\varphi}(x) \leq \Sigma' + \Pi'\varphi(x)$. Otherwise,

$$\hat{\bar{\varphi}}(x) = \max\{\bar{\varphi}(\text{prev}(x, W)), \bar{\varphi}(\text{next}(x, W))\} = \bar{\varphi}(\text{next}(x, W)). \tag{15}$$

By the construction of W we get that

$$\varphi(\text{next}(x, W)) \leq \Pi\varphi(\text{prev}(x, W)) + \Sigma. \tag{16}$$

We therefore have

$$\begin{aligned}\varphi(x) &\leq \varphi(\text{next}(x, W)) \leq \hat{\bar{\varphi}}(x) = \bar{\varphi}(\text{next}(x, W)) \\ &\leq \Pi\bar{\varphi}(\text{prev}(x, W)) + \Sigma \leq \Pi(\Sigma' + \Pi'\varphi(\text{prev}(x, W))) + \Sigma \leq \Pi(\Sigma' + \Pi'\varphi(x)) + \Sigma,\end{aligned}$$

where the equality is due to (15), the first and fifth inequalities are due to the monotonicity of φ, the second and forth inequalities are because $\bar{\varphi}$ is a (Σ', Π')-approximation of φ, and the third inequality is due to (16). We conclude that $\hat{\bar{\varphi}}$ is indeed a $(\Sigma + \Pi\Sigma', \Pi'\Pi)$-approximation function of φ.

If $\hat{\bar{\varphi}}$ is stored in a sorted array $\{(x, \bar{\varphi}(x)) \mid x \in W\}$, then for every $x \in [A, B]$, we can determine the value of $\hat{\bar{\varphi}}(x)$ in $O(\log |W|)$ time using binary search. □

3.2 A (Σ, Π)-FPTAS for the nonlinear newsvendor

The examples of cost functions given in Fig. 1 are not always Lipschitz continuous or even continuous, due to presence of setup costs. To deal with certain types of discontinuities, we relax requirements on the cost functions to *piecewise* Lipschitz continuity with explicitly-given discontinuity points. We extend the definition of (Σ, Π)-approximation sets to monotone functions over non-compact intervals, i.e., for φ increasing over $(A, B]$ and for φ decreasing over $[A, B)$, in the most natural way. We modify the routine ApxSetInc for functions over non-compact intervals $(A, B]$ by changing step 2 of Algorithm 3 to $x \leftarrow A$, $W \leftarrow \{B\}$.

Suppose now that φ is monotone increasing and α-piecewise Lipschitz continuous over the interval $[A, B]$, with explicitly-given discontinuity points $A_1, \ldots, A_{\alpha-1}$ and the property that it is Lipschitz continuous over the subintervals $[A, A_1]$; $(A_1, A_2]$; $\ldots (A_{\alpha-1}, B]$. (For ease of notation, we assume $A_\alpha = B$,

and if φ is Lipschitz continuous over $[A, B]$ we say that it is 1-piecewise Lipschitz continuous.) Using function APXSETINC we build (Σ, Π)-approximation sets $W_1, \ldots, W_\alpha$ of φ over these subintervals. It follows directly from the definition of (Σ, Π)-approximation sets that $W := \cup_{i=1}^{\alpha} W_i$ is a (Σ, Π)-approximation set of φ over the entire interval $[A, B]$.

Similarly, suppose φ is monotone decreasing, α-piecewise Lipschitz continuous over the interval $[A, B]$, and Lipschitz continuous over the subintervals $[A, A_1)$; $[A_1, A_2)$; $\ldots [A_{\alpha-1}, B]$. If $W_1, \ldots, W_\alpha$ are (Σ, Π)-approximation sets of φ over these subintervals, it is easy to verify that $W := \cup_{i=1}^{\alpha} W_i$ is a (Σ, Π)-approximation set of φ over $[A, B]$.

Remark 7 *Under the oracle model, the assumption that the discontinuity points are given explicitly cannot be dropped: if $\varphi : [0, 1] \to \mathbb{R}^+$ is such that $\varphi(x) = 0$ for $x \leq a, a \in [0, 1]$, and $\varphi(x) = 2$ for $x \in (a, 1]$, constructing a $(1, 2)$-approximation of φ may require an infinite number of queries to φ in general.*

The results presented in the preceding subsection can be directly applied to construct a (Σ, Π)-FPTAS for the continuous NNV, as is shown next. The following theorem is a generalization of Thm. 1.4.

Theorem 3.6 *Consider a continuous newsvendor problem with explicitly given discrete demand D, and monotone ordering, shortage and holding cost functions accessible via a monotone (Σ, Π)-FPTAS. Suppose that either the shortage cost or both the ordering and holding costs are piecewise κ-Lipschitz continuous with explicitly-given discontinuity points. Then for any $0 < \epsilon < 1$ and $\delta > 0$, one can compute a $(\delta, 1 + \epsilon)$-approximation of the optimal cost and a corresponding order quantity in time polynomial in $\frac{1}{\epsilon}, \log \frac{1}{\delta}, \log \kappa$ and the binary input size.*

Proof. Let the demand D take values $d_1 < \cdots < d_\ell$ with probability $p_1, \ldots, p_\ell$. The total cost can be written as:

$$\min_x \mathbb{E}\left[b(D - x) + h(x - D)\right] + c(x).$$

This can be expanded as:

$$\min_x \sum_{i=1}^{\ell} p_i b(d_i - x) + \sum_{i=1}^{\ell} p_i h(x - d_i) + c(x).$$

Notice that $b(d_i - x)$ is monotone decreasing in x, whereas both $h(x - d_i)$ and $c(\cdot)$ are monotone increasing. Without loss of generality we assume that $b(\cdot)$ is α-piecewise κ-Lipschitz continuous over $[d_1, \ldots, d_\ell]$ with discontinuity points $a_1, \ldots, a_{\alpha-1}$. (The case where both $h(\cdot)$ and $c(\cdot)$ are piecewise κ-Lipschitz continuous can be treated similarly.)

We denote by $\tilde{b}(\cdot), \tilde{h}(\cdot), \tilde{c}(\cdot)$ the monotone $(\frac{\delta}{6(1+\epsilon/4)}, 1 + \epsilon/4)$-approximations of $b(\cdot), h(\cdot), c(\cdot)$ that are realized by calling the (Σ, Π)-FPTASs for $b(\cdot), h(\cdot), c(\cdot)$ with parameters set to $\Sigma = \frac{\delta}{6(1+\epsilon/4)}, \Pi = 1 + \epsilon/4$. Using Prop. 3.3, we compute $(\frac{\delta}{2}, 1 + \epsilon/4)$-approximation sets $S_1^b, \ldots, S_\alpha^b$ of size $O(\frac{1}{\epsilon} \log \frac{\epsilon b^{\max}}{\delta})$ for $\tilde{b}(x)$ over the intervals $[d_1, a_1); \ldots; [a_{\alpha-2}, a_{\alpha-1})$; $[a_{\alpha-1}, d_\ell]$ in $O(\frac{\alpha}{\epsilon} \log \frac{\epsilon b^{\max}}{\delta} \log \frac{\kappa(d_\ell - d_1)}{\delta})$ total time. Note that $W^b := \cup_{i=1}^{\alpha} S_i^b$ is a $(\frac{\delta}{2}, 1 + \epsilon/4)$-approximation set of $\tilde{b}(\cdot)$ over $[d_1, d_\ell]$. By Prop. 2.13 (linearity of approximation and composition of approximation) we obtain a $(\frac{p_i \delta}{2}, 1 + \epsilon/4)$-approximation set W_i^b for $p_i \tilde{b}(d_i - x)$ for all $i = 1, \ldots, \ell$, by shifting the points in the approximation set.

We now apply Prop. 3.4 with $n = 2\ell + 1$, $m = \ell$; for $i = 1, \ldots, \ell$ we set $\varphi_i(\cdot) = p_i b(\cdot)$, $\tilde{\varphi}_i(\psi_i(x, y)) = p_i \tilde{b}(d_i - y)$, $\Sigma_i = \frac{p_i \delta}{6(1+\epsilon/4)}$, $\Pi_i = 1 + \epsilon/4$, $W_i(x) = W_i^b$, $\Sigma_i' = \frac{p_i \delta}{2}$, $\Pi_i' = 1 + \epsilon/4$; for $i = \ell + 1, \ldots, 2\ell$ we set $\varphi_i(\cdot) = p_{i-\ell} h(\cdot)$, $\tilde{\varphi}_i(\psi_i(x, y)) = p_{i-\ell} \tilde{h}(y - d_{i-\ell})$, $\Sigma_i = \frac{p_{i-\ell} \delta}{6(1+\epsilon/4)}$, $\Pi_i = 1 + \epsilon/4$. We set $\varphi_{2\ell+1}(\cdot) = c(\cdot)$, $\tilde{\varphi}_{2\ell+1}(\psi_{2\ell+1}(x, y)) = \tilde{c}(y)$, $\Sigma_{2\ell+1} = \frac{\delta}{6(1+\epsilon/4)}$ and $\Pi_{2\ell+1} = 1 + \epsilon/4$. We obtain that $\tilde{\varphi}(0)$ as defined in

1: **Function ApxSetConvInc**($\varphi, [A, B], \Sigma, \Pi$)
2: $x \leftarrow$ FUNCSEARCHINC($\varphi, [A, B], \varphi(A) + \Sigma/3, \varphi(A) + 2\Sigma/3$)
3: $\Delta_{\dot{x}} \leftarrow x - A$, $\dot{\sigma}_\varphi(x) \leftarrow \frac{\varphi(x)-\varphi(A)}{\Delta_{\dot{x}}}$, $W \leftarrow \{A, x, B\}$
4: **while** $x < B$ **do**
5: $\quad x \leftarrow$ FUNCSEARCHINC($\varphi, [x, B], \frac{1}{2}(\Pi + 1)(\varphi(x) + \dot{\sigma}_\varphi(x)(\cdot - x)) + \Sigma/2, \Pi(\varphi(x) + \dot{\sigma}_\varphi(x)(\cdot - x)) + \Sigma$)
6: $\quad W \leftarrow W \cup \{x\}$, $\dot{\sigma}_\varphi(x) \leftarrow \frac{\varphi(x)-\varphi(x-\Delta_{\dot{x}})}{\Delta_{\dot{x}}}$
7: **end while**
8: **return** W

Algorithm 5: FUNCTION APXSETCONVINC($\varphi, [A, B], \Sigma, \Pi$)

Prop. 3.4 is a $(\delta, 1 + \epsilon)$-approximation of:

$$\varphi(0) = \min_{y \in [d_1, d_\ell]} \sum_{i=1}^{\ell} p_i b(d_i - y) + \sum_{i=1}^{\ell} p_i h(y - d_i) + c(y).$$

(In the computation of the approximation factor, remember that $1 + \epsilon/4 \leq \sqrt{1 + \epsilon}$ for all $\epsilon \leq 1$.) But this is exactly the expression of the optimal cost for the newsvendor problem. The y that attains the minimum is the corresponding order quantity. To compute such minimum we need to build and scan all the sets W_i^b, which brings the overall time requirement to $O(\frac{\alpha}{\epsilon} \log \frac{\epsilon b^{\max}}{\delta} \log \frac{\kappa(d_\ell - d_1)}{\delta} + \frac{\alpha \ell}{\epsilon} \log \frac{\epsilon b^{\max}}{\delta})$. □

We note that Theorem 3.6 can be generalized to the case in which the cost functions are accessed via not necessarily monotone (Σ, Π)-FPTASs. In this case we will use Prop. 3.5 instead of Prop. 3.3.

3.3 Convex functions

The difficulties stated in Sect. 3.1.2 can be easily overcome in the convex case because we can use APXARGMIN to approximately minimize over the action space, and the APXARGMIN operator applied to a convex value function preserves convexity.

Algorithm 5 (called APXSETCONVINC) constructs a (Σ, Π)-approximation set for a convex function φ that decreases by at most $\Sigma/3$ before being monotone increasing. We briefly overview the algorithm. Step 2 finds a point x such that φ is ensured to be monotone increasing over $[x, B]$. We then iteratively apply FUNCSEARCHINC to find additional points to be added to the approximation set. The values of the parameters low$(\cdot)$ and high$(\cdot)$ in the call to FUNCSEARCHINC in step 5 ensure the following properties at each iteration: (i) the value of φ increases by at least $\Sigma/2$ and at most Σ; (ii) the value of φ increases by an additional multiplicative factor of at least $1 + \epsilon/2$ and at most $1 + \epsilon$; (iii) the slope of φ increases by a multiplicative factor of at least $1+\epsilon/2$ and at most $1+\epsilon$. Properties (i)-(ii) ensure that we construct a (Σ, Π)-approximation set of φ, property (iii) accelerates the algorithm. In the first iterations of the while loop, if the slope and the value of φ are both very close to 0, the additive increase of $\Sigma/2$ is the main workhorse. After at most $2/\epsilon$ iterations, the value of φ is at least Σ/ϵ, so the multiplicative increase (ii) becomes dominant over the additive increase (i).

We define function APXSETCONVDEC similarly for convex decreasing functions that increase by at most $\Sigma/3$ after their $\arg\min$, and define function APXSETCONV that constructs a (Σ, Π)-approximation sets for convex functions, see Algorithm 6. Similar to the monotone case, we define a function COMPRESSCONV to construct an explicit succinct approximation of a function that is too expensive to access directly.

Let $\tilde{x}$ be the value computed in step 2 of Algorithm 6. Let $\dot{x}$ (resp. $\ddot{x}$) be the value of x computed in step 2 of Algorithm 5 (resp. of function APXSETCONVDEC) when called by step 3 of Algorithm 6. Let

1: **Function ApxSetConv**($\varphi, [A, B], \Sigma, \Pi$)
2: $\tilde{x} \leftarrow$ APXARGMIN($\varphi, [A, B], \Sigma/3$)
3: **return** APXSETCONVDEC($\varphi, [A, \tilde{x}], \Sigma, \Pi$) $\cup$ APXSETCONVINC($\varphi, [\tilde{x}, B], \Sigma, \Pi$)

Algorithm 6: FUNCTION APXSETCONV($\varphi, [A, B], \Sigma, \Pi$)

1: **Function CompressConv**($\varphi, [A, B], \Sigma, \Pi$)
2: $W \leftarrow$ APXSETCONV($\varphi, [A, B], \Sigma, \Pi$)
3: **return** $\{(x, \varphi(x)) \mid x \in W\}$ as a sorted array

Algorithm 7: FUNCTION COMPRESSCOV returns a canonical representation of a piecewise linear convex (Σ, Π)-approximation function of a convex function φ.

$\Delta_{\dot{x}} := \dot{x} - \tilde{x}$ and $\Delta_{\ddot{x}} := \tilde{x} - \ddot{x}$. Let $\dot{\sigma}_\varphi(x) := \frac{\varphi(x)-\varphi(x-\Delta_{\dot{x}})}{\Delta_{\dot{x}}}$ be a lower bound on the slope of φ at x for any $x \geq \dot{x}$. Let $\ddot{\sigma}_\varphi(x) := \frac{\varphi(x)-\varphi(x+\Delta_{\ddot{x}})}{\Delta_{\ddot{x}}}$ be a lower bound on the absolute value of the slope of φ at x for any $x \leq \ddot{x}$. Let $\dot{x}'$ be $\dot{x}$ if $\varphi(\dot{x}) > \Sigma/\epsilon$ and otherwise be $\arg\min_{x>\dot{x}} \varphi(x) = \Sigma/\epsilon$. Similarly, let $\ddot{x}'$ be $\ddot{x}$ if $\varphi(\ddot{x}) > \Sigma/\epsilon$ and otherwise be $\arg\max_{x<\ddot{x}} \varphi(x) = \Sigma/\epsilon$. (Thus, note that $A \leq \ddot{x}' \leq \ddot{x} < \tilde{x} < \dot{x} \leq \dot{x}' \leq B$ holds.) Let $\sigma_\varphi^{\max} := \max\{\ddot{\sigma}_\varphi(A), \dot{\sigma}_\varphi(B)\}$. Let $\sigma_\varphi^{\min} := \min\{\ddot{\sigma}_\varphi(\ddot{x}'), \dot{\sigma}_\varphi(\dot{x}')\}$ be a lower bound on the absolute value of σ_φ over $[A, \ddot{x}'] \cup [\dot{x}', B]$. We are now ready to state our next result. (Recall that the definition of canonical representations is given in Definition 3.1.)

Proposition 3.7 (Approximation of a convex function with direct access) *Let $\varphi : [A, B] \to \mathbb{R}^+$ be a κ-Lipschitz continuous convex function. Then for every constants $\Sigma > 0$ and $\Pi = 1 + \epsilon > 1$ the following holds. (i)* APXSETCONV*$(\varphi, [A, B], \Sigma, \Pi)$ and* COMPRESSCONV*$(\varphi, [A, B], \Sigma, \Pi)$ construct a (Σ, Π)-approximation set W for φ of cardinality $O\left(\frac{1}{\epsilon} \log \min\{\frac{\sigma_\varphi^{\max}}{\sigma_\varphi^{\min}}, \frac{\epsilon\varphi^{\max}}{\Sigma}\}\right)$ in $O\left((1 + t_\varphi)(\frac{1}{\epsilon} \log \min\{\frac{\sigma_\varphi^{\max}}{\sigma_\varphi^{\min}}, \frac{\epsilon\varphi^{\max}}{\Sigma}\}) \log \frac{\kappa(B-A)}{\Sigma}\right)$ time. (ii)* COMPRESSCONV*$(\varphi, [A, B], \Sigma, \Pi)$ returns the canonical representation of the piecewise linear extension of φ induced by W, $\hat{\varphi}$, which is a convex piecewise linear κ-Lipschitz (Σ, Π)-approximation of φ. Moreover, the value of $\hat{\varphi}(\cdot)$ can be determined in $O(\log |W|)$ time at any point in $[A, B]$.*

Proof. Let $\tilde{x}, \dot{x}, \dot{x}', \ddot{x}, \ddot{x}'$ be as defined in Section 3.3. By Proposition 2.2, $\varphi(\tilde{x})$ is an additive $\Sigma/3$-approximation of $\varphi^{\min}$. Therefore, $\varphi : [A, \tilde{x}] \to \mathbb{R}^+$ increases by at most $\Sigma/3$ after its true argmin over $[A, \tilde{x}]$, and $\varphi : [\tilde{x}, B] \to \mathbb{R}^+$ decreases by at most $\Sigma/3$ between $\tilde{x}$ and its true argmin. Thus, the calls to APXSETCONVINC and APXSETCONVDEC are well defined. We next prove (i) and (ii) for the restriction of φ over $[\tilde{x}, B]$. The proof for the restriction of φ over $[A, \tilde{x}]$ is similar. Considering APXSETCONVINC, after the first call to FUNCSEARCHINC (step 2), φ is ensured to be monotone increasing and convex over $[\dot{x}, B]$. Moreover, $\varphi(\dot{x}) - \min\{\varphi(x) \mid \tilde{x} \leq x \leq \dot{x}\} \leq \Sigma$, so $\{\tilde{x}, \dot{x}\}$ is a $(\Sigma, 1)$-approximation set of φ over $[\tilde{x}, \dot{x}]$. If $\dot{x} \neq \dot{x}'$ (i.e., if $\varphi(\dot{x}) < \Sigma/\epsilon$), due to the parameters with which FUNCSEARCHINC is called in step 5, it takes $O(1/\epsilon)$ iterations of the while loop until finding an x with $\varphi(x) \geq \Sigma/\epsilon$. It takes $O(\log_\Pi \min\{\frac{\sigma_\varphi^{\max}}{\sigma_\varphi^{\min}}, \frac{\epsilon\varphi^{\max}}{\Sigma}\})$ more iterations until the while loop is exhausted. The proof that the while-loop computes a (Σ, Π)-approximation set of φ over $[\dot{x}, B]$ follows from [HNO15, Thm. 3.2] coupled with the fact that $\varphi(x) \geq \Sigma/\epsilon$ over this interval. The bound on the running time follows from the fact that $O(\log_\Pi v) = O(\frac{\log v}{\epsilon})$ holds true for all $0 < \epsilon < 1$ and $v > 0$.

Regarding the Lipschitz constant of $\hat{\varphi}$, i.e., the slope of its rightmost linear piece, we note that due to the convexity of φ, it is bounded by the slope of φ at B. □

Whenever we access φ indirectly via a convex approximation $\breve{\varphi}$ of it, Proposition 2.13(7) coupled with Proposition 3.7 yields the following result.

Proposition 3.8 *Let $\Pi, \Pi' \geq 1,\ \Sigma, \Sigma' \geq 0$ be real numbers, and let $\varphi : [A, B]$ be a convex function over $[A, B]$. Let $\breve{\varphi}$ be a κ-Lipschitz continuous convex (Σ', Π')-approximation function of φ. Then function* COMPRESSCONV*$(\breve{\varphi}, [A, B], \Sigma, \Pi)$ returns in $O\left((1 + t_{\breve{\varphi}})(\frac{1}{\epsilon}\log\min\{\frac{\sigma^{\max}_{\breve{\varphi}}}{\sigma^{\min}_{\breve{\varphi}}}, \frac{\epsilon\breve{\varphi}^{\max}}{\Sigma}\})\log\frac{\kappa(B-A)}{\Sigma}\right)$ time a (Σ, Π)-approximation set W of $\breve{\varphi}$, together with the canonical representation of the piecewise linear extension of $\breve{\varphi}$ induced by W, $\hat{\breve{\varphi}}$. Moreover, $\hat{\breve{\varphi}}$ is a κ-Lipschitz continuous convex $(\Sigma + \Pi\Sigma', \Pi'\Pi)$-approximation of φ.*

Proof. By property (ii) of Proposition 3.7, the piecewise linear extension of $\breve{\varphi}$ induced by W is a convex (Σ, Π)-approximation of $\breve{\varphi}$, and by approximation of approximation (Proposition 2.13), it is a $(\Sigma + \Pi\Sigma', \Pi'\Pi)$-approximation of φ. The rest of the proof follows from Proposition 3.7 □

4 DP model statement

We now describe the DP models that we consider in this paper, and in subsequent sections we will construct approximation algorithms for them using the machinery developed so far. We consider a discrete-time finite-horizon stochastic DP with continuous state and action spaces as described in Section 1. Then the expression in (3) can be rewritten as:

$$\mathbb{E}_{D_t}\left\{g_t(I_t, x_t, D_t) + z_{t+1}(f_t(I_t, x_t, D_t))\right\} = \sum_{j=1}^{n_t} p_{t,j}\Big[g_t(I_t, x_t, d_{t,j}) + z_{t+1}(f_t(I_t, x_t, d_{t,j}))\Big]. \tag{17}$$

In our analysis, the following notation will be used.

$n_t = |\mathcal{D}_t|$: maximum number of different values that D_t can take;

$n^* = \max_t n_t$: maximum number of different values that D_t can take over the entire time horizon;

$U_{\mathcal{S}} = \max_{t=1,\ldots,T+1}(\mathcal{S}_t^{\max} - \mathcal{S}_t^{\min})$: maximum length of the state space;

$U_{\mathcal{A}} = \max_t \max_{I_t \in \mathcal{S}_t}(\mathcal{A}_t^{\max}(I_t) - \mathcal{A}_t^{\min}(I_t))$: maximum length of the action space;

$g_t^{\max} = \max_{I \in \mathcal{S}_t,\, x \in \mathcal{A}_t(I),\, d \in \mathcal{D}_t} g_t(I, x, d)$: maximum cost value in time period $t,\ t = 1, \ldots, T$;

$g_{T+1}^{\max} = \max_{I \in \mathcal{S}_{T+1}} g_{T+1}(I)$;

$U_g = \max_{t=1,\ldots,T+1} g_t^{\max}$: maximum cost value in a time period;

κ : largest Lipschitz constant among all the functions mentioned in Condition 2 below.

To derive a (Σ, Π)-FPTAS for the DP, the following conditions are needed (recall that the definition of the operator $\otimes$ is given at the end of Section 1):

Condition 1 *$\mathcal{S}_{T+1}, \mathcal{S}_t, \mathcal{A}_t(I_t)$ are compact intervals on the real line, for $I_t \in \mathcal{S}_t$ and $t = 1, \ldots, T$. For every $t = 1, \ldots, T$, the number of different values the random variable D_t admits with positive probability is a given integer n_t, and its probability distribution function is given as n_t ordered pairs $(d_{t,i}, p_{t_i})$, where $p_{t,i} = Prob(D_t = d_{t,i})$ for $i = 1, \ldots, n_t$.*

Condition 2 *For every $t = 1, \ldots, T+1$, the functions f_t, g_t are either given explicitly (i.e., as explicit formulae) or accessed via value oracles; f_t, g_t and the lower and upper envelopes of $\cup_{I_t \in \mathcal{S}_t}\mathcal{A}_t(I_t)$ (i.e., functions $\mathcal{A}_t^{\min}(\cdot), \mathcal{A}_t^{\max}(\cdot)$) are all Lipschitz continuous functions; g_t is nonnegative.*

Condition 3 *At least one of the following properties holds:*

(i) **(Increasing DP)** *Function g_{T+1} is increasing. For $t = 1, \ldots, T$, function f_t is increasing in its first variable and monotone in its second variable, and g_t is monotone in its second variable. For each $t = 1, \ldots, T$, either z_t is increasing, or g_t is increasing in its first variable and $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$ for all $I, I' \in \mathcal{S}_t$ with $I \geq I'$.*

(ii) **(Decreasing DP)** *Function g_{T+1} is decreasing. For $t = 1, \ldots, T$, function f_t is increasing in its first variable and monotone in its second variable, and g_t is monotone in its second variable. For each $t = 1, \ldots, T$, either z_t is decreasing, or g_t is decreasing in its first variable and $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$ for all $I, I' \in \mathcal{S}_t$ with $I \leq I'$.*

(iii) **(Convex DP)** *The function g_{T+1} is convex over $\mathcal{S}_{T+1}$. For $t = 1, \ldots, T$, the set $\mathcal{S}_t \otimes \mathcal{A}_t$ is convex. The function g_t can be expressed as $g_t(I, x, d) = g_t^I(I, d) + g_t^x(x, d) + g_t^u(f_t(I, x, d))$, where $g_t^I(\cdot, d), g_t^x(\cdot, d), g_t^u(\cdot)$ are univariate nonnegative Lipschitz continuous convex functions and f_t is linear and separable in its variables.*

The input data of the problem includes the number of time periods T, the initial state I_1, the constants U_g, U_S, U_A and the explicit description of the random variables as described in Condition 1. Our algorithms run in time polynomial in the (binary enconding of the) input size. While our algorithms do not require the values of the Lipschitz constants to be given, the running time depends polylogarithmically on them, therefore we think of the Lipschitz constants as part of the input. We call a DP formulation (3) *monotone* whenever it satisfies either Condition 3(i) or Condition 3(ii), and *convex* whenever it satisfies Condition 3(iii).

We briefly highlight the main differences between the stochastic continuous DP model discussed in this paper and the stochastic discrete DP model of [HKL$^+$14, HNO15], besides the model of computation as discussed in Sec. 2.1. Due to Proposition 2.11, in this paper we require that the functions f_t and g_t be Lipschitz continuous. Furthermore, we include the maximum length of the state and action spaces as part of the input data of the problem; in the discrete model this is not necessary, because [HKL$^+$14, Condition 1] implies that these values are polynomially bounded by the input size. Finally, the convex case of the discrete model requires the coefficients of x in the transition functions to be in $\{-1, 0, 1\}$, see [HKL$^+$14, Thm. 9.2]. In the present paper, this can be relaxed while preserving convexity of the value functions: we require instead that the transition functions are linear functions that are separable in their variables.

Our main result (Theorem 1.6) is that every stochastic DP satisfying Conditions 1, 2 and 3 admits a (Σ, Π)-FPTAS. Moreover, the approximation scheme succinctly (Σ, Π)-approximates z_1 and returns an approximate policy. [HKL$^+$14] presented seven stochastic problems that fit into their FPTAS framework. Six of these problems (single-item inventory control, single-item batch dispatch, single-resource revenue management, growth models, lifetime consumption of risky capital and cash management) seem to be more naturally cast in a framework with continuous state and action spaces. It is easy to verify that these problems, as described in [HKL$^+$14], satisfy Conditions 1, 2 and 3, and therefore admit a (Σ, Π)-FPTAS as a consequence of Theorem 1.6. In this sense we argue that the technical conditions 1-3 are not more restrictive than the ones required for the FPTAS for the discrete DP model. In the Appendix we provide specific examples of problems fitting in our framework: in Appendix B we discuss a continuous time-cost trade-off machine scheduling problem and formulate it as a monotone DP that satisfies Conditions 1, 2 and 3. In Appendix C we discuss an inventory control problem and formulate it as a convex DP that satisfies Conditions 1, 2 and 3. In Appendix D we perform a computational evaluation of the proposed approximation scheme on instances of the inventory control problem.

We prove Theorem 1.6 by efficiently constructing succinct (Σ, Π)-approximations of the value functions z_t by backward induction. Proposition 2.3 tells us that it is not possible in general to build a succinct

additive Σ-approximation to z_t. Corollary 2.5 tells us that it is not possible in general to build a succinct multiplicative Π-approximation for z_t. This implies that the proof approach of Theorem 1.6 cannot yield additive or multiplicative approximations to stochastic continuous DPs. In fact, such approximations cannot be obtained in general, as is stated below. The proof of the next result also implies Theorem 1.5.

Theorem 4.1 *A stochastic continuous DP satisfying Conditions 1, 2 and 3 does not necessarily admit an additive or a multiplicative approximation, regardless of the scheme used to build the approximation.*

Proof. Consider first multiplicative approximations with the following deterministic convex single-period DP. Let A be an arbitrary positive integer number, $T = 1,\ \mathcal{S}_1 = \mathcal{S}_2 = [0, 2],\ \mathcal{A}_1(I) = [0, 2],\ \forall I \in \mathcal{S}_1,\ D_1 \equiv 0,\ f_1(I, x, D) \equiv 0,\ g_2(I) \equiv 0$ and $g_1(I, x, d) = \varphi(x)$, where $\varphi \in \Phi$ is a family of 1-Lipschitz continuous convex functions over $[0, 2]$ as defined in the proof of Proposition 2.4. It is easy to verify that Conditions 1 and 2 are satisfied. By defining $g_1^I = g_1^u \equiv 0$ and $g_1^x(x) = \varphi(x)$ we get that Condition 3(iii) is also satisfied, so we get indeed a continuous DP which fits into our model. Note that $z_1(I) = 0$ if and only if $\varphi^{\min} = 0$ for all $I \in [0, 2]$. Since any multiplicative approximation of $z_1(I)$ is 0 if and only if $z_1(I) = 0$, this cannot be computed in finite time due to Proposition 2.4. As each call to the oracle described in the proof of Proposition 2.4 can be identified with a discretization point, Theorem 1.5 follows with relative errors.

We next consider additive approximations with the following deterministic monotone DP. Let A be an arbitrary positive integer number, $T = 1,\ \mathcal{S}_1 = \mathcal{S}_2 = [0, A],\ \mathcal{A}_1(I) = [0, A],\ \forall I \in \mathcal{S}_1,\ D_1 \equiv 0,\ f_1(I, x, D) = x,\ g_2(I) = A + \varphi^*(I)$ and $g_1(I, x, d) = A - \varphi(x)$, where φ^* and $\varphi \in \Phi$ are Lipschitz continuous nonnegative monotone functions as defined in the proof of Proposition 2.3. It is easy to verify that Conditions 1 and 2 are satisfied. Moreover, g_1 is monotone decreasing in its second variable and g_2 is monotone increasing so Condition 3(i) is also satisfied, so we get indeed a continuous increasing DP. Note that $z_1(I) = 2A$ if $\varphi \equiv \varphi^*$ and $z_1(I) = 2A - 2\Sigma$ if $\varphi = \varphi_i$ as defined in the proof of Proposition 2.3 by choosing action $x = i$. Since any additive Σ-approximation value of $z_1(I)$ is at most $2A - \Sigma$ if $z_1(I) = 2A - 2\Sigma$, and is at least $2A$ if $z_1(I) = 2A$ we get a separation between these two cases. But due to Proposition 2.3 this cannot be computed in time polynomial in $\log A$. As each point in the representation described in Proposition 2.3 can be identified with a discretization point, Theorem 1.5 follows with additive errors.

□

5 From (Σ, Π)-approximation sets and functions to DP

The remainder of this paper concerns the construction of approximation schemes for DPs using the concept of (Σ, Π)-approximation sets. Because the convex case (Condition 3(iii)) is considerably easier than the two monotone cases, we always discuss it first to give an intuition of the main ideas.

Our first step is to state two propositions that link the notions of (Σ, Π)-approximation sets and functions to DP. The first proposition deals with convex DPs and is applied when both g_t and z_{t+1} are known to be convex. Note that the minimization in equation (18), which is taken over the entire action space $\mathcal{A}_t(I_t)$, can be performed efficiently by exploiting convexity of $\mathbb{E}_{D_t}\{\tilde{g}_t(I_t, \cdot, D_t)\}$ and $\mathbb{E}_{D_t}\{\tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))\}$.

Proposition 5.1 *Suppose the DP formulation* (3) *is convex, i.e. Condition 3(iii) is satisfied. Let (Σ^z, Π^z), (Σ^g, Π^g), t, and I_t be fixed values, where $\Sigma^z, \Sigma^g \geq 0$, $\Pi^z \geq \Pi^g \geq 1$, $I_t \in \mathcal{S}_t$, and $t \in [1, \ldots, T]$. Let $\breve{g}_t(I_t, \cdot, d_{t,i})$ be a κ_g-Lipschitz continuous convex (Σ^g, Π^g)-approximation function of $g_t(I_t, \cdot, d_{t,i})$ for every $i = 1, \ldots, n_t$. Let $\breve{z}_{t+1}$ be a κ_z-Lipschitz continuous convex (Σ^z, Π^z)-approximation function of z_{t+1}. Let κ_f be a bound*

on the Lipschitz constant of f_t, and let κ_A be a bound on the Lipschitz constants of $\mathcal{A}_t^{\min}(\cdot), \mathcal{A}_t^{\max}(\cdot)$. Let

$$\bar{z}_t(I_t) = \min_{x_t \in \mathcal{A}_t(I_t)} \mathbb{E}_{D_t}\big\{\breve{g}_t(I_t, x_t, D_t) + \breve{z}_{t+1}(f_t(I_t, x_t, D_t))\big\}. \tag{18}$$

Then, $\bar{z}_t(\cdot)$ is a convex $(\Sigma^g + \Sigma^z, \Pi^z)$-approximation function of $z_t(\cdot)$ over $\mathcal{S}_t$. Moreover, for every $I_t \in \mathcal{S}_t$, $\Sigma > 0$ an additive Σ-approximation value of $\bar{z}_t(I_t)$, which is a $(\Sigma + \Sigma^g + \Sigma^z, \Pi^z)$-approximation value of $z_t(I_t)$, can be determined in $O\left(n_t(t_{\breve{g}_t} + t_{f_t} + t_{\breve{z}_{t+1}}) \log \frac{(\kappa_g + \kappa_f\kappa_z)(1+\kappa_A)(\mathcal{A}_t^{\max}(I_t) - \mathcal{A}_t^{\min}(I_t))}{\Sigma}\right)$ time.

Proof. We apply Proposition 2.15 with the following parameter setting: Let $[A, B] = \mathcal{S}_t$, $C(\cdot) = \mathcal{A}_t(\cdot)$, $n = 2n_t$, $x = I_t$, $y = x_t$, and $\varphi(\cdot) = z_t(\cdot)$. For $i = 1, \ldots, n_t$, let $\varphi_i(\cdot) = p_{t,i} g_t(I_t, \cdot, d_{t,i})$, $\breve{\varphi}_i(\cdot) = p_{t,i}\breve{g}_t(I_t, \cdot, d_{t,i})$, $\psi_i(x, y) = y$, and $(\Sigma_i, \Pi_i) = (p_{t,i}\Sigma^g, \Pi^g)$. For $i = n_t + 1, \ldots, 2n_t$, let $\varphi_i(\cdot) = p_{t,i-n_t} z_{t+1}(\cdot)$, $\breve{\varphi}_i(\cdot) = p_{t,i-n_t}\breve{z}_{t+1}(\cdot)$, $\psi_i(x, y) = f_t(x, y, d_{t,i-n_t})$, and $(\Sigma_i, \Pi_i) = (p_{t,i-n_t}\Sigma^z, \Pi^z)$. Because $\breve{g}_t(I_t, \cdot, d_{t,i})$ is a (Σ^g, Π^g)-approximation of $g_t(I_t, \cdot, d_{t,i})$, by linearity of approximation (Proposition 2.13), $\breve{\varphi}_i(\cdot)$ is a $(p_{t,i}\Sigma^g, \Pi^g)$-approximation (i.e., (Σ_i, Π_i)-approximation) of $\varphi_i(\cdot)$ for $i = 1, \ldots, n_t$. Similarly, $\breve{\varphi}_i(\cdot)$ is a $(p_{t,i-n_t}\Sigma^z, \Pi^z)$-approximation (i.e., (Σ_i, Π_i)-approximation) of $\varphi_i(\cdot)$ for $i = n_t + 1, \ldots, 2n_t$. Hence, by Proposition 2.15, $\breve{\varphi}$ is a convex $(\Sigma^g + \Sigma^z, \max\{\Pi^g, \Pi^z\})$-approximation of φ; that is, $\bar{z}_t$ is a $(\Sigma^g + \Sigma^z, \Pi^z)$-approximation of z_t.

We next turn to analyzing the running time. We note first that by Proposition 2.14 the Lipschitz constant of $\bar{z}_t$ is bounded by $(\kappa_f\kappa_z)(1 + \kappa_A)$. We also note that for any fixed d_t, function $f_t(I_t, \cdot, d_t)$ is linear. Thus, $\breve{z}_{t+1}(f_t(I_t, \cdot, d_t))$ is a convex function over $\mathcal{A}(I_t)$. Because a conical combination (i.e., linear combination with nonnegative coefficients) of convex functions is convex, $\bar{z}_t(I_t, \cdot) := \mathbb{E}_{D_t}\{\breve{g}_t(I_t, \cdot, D_t) + \breve{z}_{t+1}(f_t(I_t, \cdot, D_t))\}$ is a convex function over $\mathcal{A}_t(I_t)$, and therefore by Proposition 2.2 we can compute an additive Σ-approximated minimum $\tilde{\bar{z}}_t(I_t)$ in $O(\log((\kappa_f\kappa_z)(1+\kappa_A)|\mathcal{A}_t(I_t)|/\Sigma))$ steps by performing binary search over the interval $\mathcal{A}_t(I_t)$. As each of these steps involves n_t queries to $\breve{g}_t$, f_t, and $\breve{z}_{t+1}$, the claimed running time follows.

We conclude that $\tilde{\bar{z}}_t(I_t)$ is a $(\Sigma + \Sigma^g + \Sigma^z, \Pi^z)$-approximation value of $\bar{z}_t(I_t)$ due to approximation of approximation (i.e., property 7 of Proposition 2.13 applied with parameters set to $n = 2$, $\varphi_1(\cdot) = \bar{z}_t(\cdot)$, $\varphi_2(\cdot) = \tilde{\bar{z}}_t(\cdot)$, $(\Sigma_1, \Pi_1) = (\Sigma^g + \Sigma^z, \Pi^z)$; $(\Sigma_2, \Pi_2) = (\Sigma, 1)$). □

The second proposition deals with monotone DPs and is applied when function z_{t+1} is guaranteed to be monotone. In equation (19) below, the function $\mathbb{E}_{D_t}\{\tilde{g}_t(I_t, \cdot, D_t)\} + \mathbb{E}_{D_t}\{\tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))\}$ is not necessarily convex, and therefore we cannot use binary search to determine its minimum point. To find an efficient approximation, the minimization in equation (19) is over an approximation set. The proposition is split into two cases. The first case is somewhat weaker, because it assumes that an inverse of the transition function is available, which is actually the case in all of the applications in [HKL$^+$14]. This allows us to perform the minimization faster. The second case requires approximation sets for $\tilde{g}_t(I_t, \cdot, D_{t,i})$, $i = 1, \ldots, n_t$, which results in a slower running time, but on the other hand does not require to have oracle access to the inverse of the transition function $f_t(I, \cdot, D_t)$.

Proposition 5.2 *Suppose the DP formulation* (3) *is monotone, i.e. Condition 3(i) or Condition 3(ii) are satisfied. Let $(\Sigma^z, \Pi^z), (\Sigma^g, \Pi^g), (\Sigma^W, \Pi^W)$, t, and I_t be fixed values, where $\Sigma^z, \Sigma^g, \Sigma^W \geq 0$, $\Pi^z, \Pi^W, \Pi^g \geq 1$, $I_t \in \mathcal{S}_t$, $t \in [1, \ldots, T]$, and $\Pi^W\Pi^g \leq \Pi^z$. Let g_t be as stated in Conditions 3(i) and 3(ii). Let $\tilde{z}_{t+1}$ be a monotone (Σ^z, Π^z)-approximation of z_{t+1}, and let $\tilde{g}_t(I_t, \cdot, D_t)$ be a monotone (Σ^g, Π^g)-approximation of $g_t(I_t, \cdot, D_t)$.*

(i) For every fixed I_t and D, assume that we are given a value oracle to calculate the inverse of f_t in its second variable, $f_t^{-1}(I_t, w, D) = \{\arg\min_{x \in \mathcal{A}_t(I_t)} f_t(I_t, x, D) = w,\ \arg\max_{x \in \mathcal{A}_t(I_t)} f_t(I_t, x, D) = w\}$. Let W be a (Σ^W, Π^W)-approximation set of $\tilde{z}_{t+1}$, let $f_{t,i}^{-1}(I_t, W) = \bigcup_{w \in W} f_t^{-1}(I_t, w, d_{t,i})$ and

let $W^{-1}(I_t) = \big(\bigcup_{i=1}^{n_t} f_{t,i}^{-1}(I_t, W)\big) \cup \{\mathcal{A}_t^{\min}(I_t), \mathcal{A}_t^{\max}(I_t)\}$. *Let*

$$\bar{z}_t(I_t) = \min_{x_t \in W^{-1}(I_t)} \mathbb{E}_{D_t}\big\{\tilde{g}_t(I_t, x_t, D_t) + \tilde{z}_{t+1}(f_t(I_t, x_t, D_t))\big\}. \tag{19}$$

Then $\bar{z}_t(I_t)$ *is a* $(\Sigma^W + \Pi^W\Sigma^z + \Sigma^g, \Pi^z\Pi^W)$*-approximation value of* $z_t(I_t)$.

(ii) Let W_i *be a* (Σ^W, Π^W)*-approximation set of* $\tilde{g}_t(I_t, \cdot, d_{t,i})$ *over* $\mathcal{A}_t(I_t)$*, and let* $W^{-1}(I_t) = \bigcup_{i=1}^{n_t} W_i$. *Then* $\bar{z}_t(I_t)$ *as defined in* (19) *is a* $(\Sigma^W + \Pi^W\Sigma^g + \Sigma^z, \Pi^z)$*-approximation value of* $z_t(I_t)$.

In both cases, $\bar{z}_t(\cdot)$ *is not necessarily monotone and its value on any* I_t *can be determined in* $O(n_t(t_{\tilde{g}_t} + t_{f_t} + t_{\tilde{z}_{t+1}})|W^{-1}(I_t)|)$ *time if the elements of* $W^{-1}(I_t)$ *are given.*

Proof. Since f_t is monotone in its second variable and $\tilde{z}_{t+1}$ is monotone, the function $\mathbb{E}_{D_t}\tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))$ is monotone. Also, since $\tilde{g}_t$ is monotone in its second variable, the function $\mathbb{E}_{D_t}\tilde{g}_t(I_t, \cdot, D_t)$ is monotone. We consider two different cases.

Case 1: $\mathbb{E}_{D_t}\tilde{g}_t(I_t, \cdot, D_t)$ and $\mathbb{E}_{D_t}\tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))$ are monotone in the same direction. We consider the situation where these two functions are increasing (the analysis for the decreasing case follows from a similar argument). Under this situation, the minimum of the expression $\mathbb{E}_{D_t}\{\tilde{g}_t(I_t, \cdot, D_t) + \tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))\}$ is attained when x_t is the smallest element in $\mathcal{A}_t(I_t)$ (which is also an element of $W^{-1}(I_t)$). By composition of approximation (Proposition 2.13), $\tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))$ is a (Σ^z, Π^z)-approximation of $z_{t+1}(f_t(I_t, \cdot, D_t))$. By linearity of approximation and summation of approximation (Proposition 2.13), $\mathbb{E}_{D_t}\{\tilde{g}_t(I_t, \cdot, D_t) + \tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))\}$ is a $(\Sigma^z + \Sigma^g, \max\{\Pi^z, \Pi^g\})$-approximation of $\mathbb{E}_{D_t}\{g_t(I_t, \cdot, D_t) + z_{t+1}(f_t(I_t, \cdot, D_t))\}$. Hence, due to the inequality $\Pi^W\Pi^g \le \Pi^z$, we get that $\bar{z}_t(I_t)$ is a $(\Sigma^z + \Sigma^g, \Pi^z)$-approximation of $z_t(I_t)$. In both cases, $\bar{z}_t(I_t)$ can be determined in $O(n_t(t_{\tilde{g}_t} + t_{f_t} + t_{\tilde{z}_{t+1}}))$ time.

Case 2: $\mathbb{E}_{D_t}\tilde{g}_t(I_t, \cdot, D_t)$ and $\mathbb{E}_{D_t}\tilde{z}_{t+1}(f_t(I_t, \cdot, D_t))$ are monotone in opposite directions. We split this case according to cases (i) and (ii) as stated in the proposition assertion.

Case 2(i): We apply Proposition 3.4 with the following parameter setting (similarly to the proof of Proposition 5.1, we use here linearity of approximation (Proposition 2.13)): Let $[A, B] = \mathcal{S}_t$, $C(\cdot) = \mathcal{A}_t(\cdot)$, $n = 2n_t$, $m = n_t$, $x = I_t$, $y = x_t$, $\varphi(\cdot) = z_t(\cdot)$, and $\tilde{\varphi}(\cdot) = \bar{z}_t(\cdot)$. For $i = 1, \ldots, n_t$, let $\varphi_i(\cdot) = p_{t,i}z_{t+1}(\cdot)$, $\tilde{\varphi}_i(\cdot) = p_{t,i}\tilde{z}_{t+1}(\cdot)$, $\psi_i(x, y) = f_t(x, y, d_{t,i})$, $\Sigma_i' = p_{t,i}\Sigma^W$, $\Pi_i' = \Pi^W$, $\Sigma_i = p_{t,i}\Sigma^z$, $\Pi_i = \Pi^z$, and $W_i(x) = f_{t,i}^{-1}(x, W)$. For $i = n_t + 1, \ldots, 2n_t$, let $\varphi_i(\cdot) = p_{t,i-n_t}g_t(I_t, \cdot, d_{t,i-n_t})$, $\tilde{\varphi}_i(\cdot) = p_{t,i-n_t}\tilde{g}_t(I_t, \cdot, d_{t,i-n_t})$, $\psi_i(x, y) = y$, and $\Sigma_i = p_{t,i}\Sigma^g$, $\Pi_i = \Pi^g$.

Because $\tilde{g}_t(I_t, \cdot, d_{t,i})$, $\tilde{z}_{t+1}$, and $\psi_i(I_t, \cdot)$ are monotone, the function $\tilde{\varphi}_i(\psi_i(I_t, \cdot))$ is monotone for $i = 1, \ldots, 2n_t$. In addition, $\tilde{\varphi}_i(\psi_i(I_t, \cdot))$ is monotone in one direction for $i = 1, \ldots, n_t$ and is monotone in the other direction for $i = n_t + 1, \ldots, 2n_t$. Because $f_t(I_t, x_t, d_{t,i})$ is monotone in x_t, and because W is a (Σ^W, Π^W)-approximation set of $\tilde{z}_{t+1}$, $f_{t,i}^{-1}(I_t, W) \cup \{\mathcal{A}_t^{\min}(I_t), \mathcal{A}_t^{\max}(I_t)\}$ is a (Σ^W, Π^W)-approximation set of $\tilde{z}_{t+1}(f_t(I_t, \cdot, d_{t,i}))$ over $\mathcal{A}_t(I_t)$, for $i = 1, \ldots, n_t$, and $f_{t,i}^{-1}(I_t, W) \cup \{\mathcal{A}_t^{\min}(I_t), \mathcal{A}_t^{\max}(I_t)\}$ is a (Σ^W, Π^W)-approximation set of $\tilde{\varphi}_i(\psi_i(I_t, \cdot))$ over $\mathcal{A}_t(I_t)$, for $i = 1, \ldots, n_t$. Thus, by Proposition 3.4, $\tilde{\varphi}$ is a $(\sum_{i=1}^m(\Sigma_i' + \Pi_i'\Sigma_i) + \sum_{i=m+1}^n \Sigma_i, \max\{\Pi_1\Pi_1', \ldots, \Pi_m\Pi_m', \Pi_{m+1}, \ldots, \Pi_n\})$-approximation of φ. Hence, $\bar{z}_t$ is a $(\Sigma^W + \Pi^W\Sigma^z + \Sigma^g, \max\{\Pi^W\Pi^z, \Pi^g\})$-approximation of z_t (i.e., a $(\Sigma^W + \Pi^W\Sigma^z + \Sigma^g, \Pi^W\Pi^z)$-approximation of z_t).

Case 2(ii): We apply Proposition 3.4 with the following parameter setting: Let $[A, B] = \mathcal{S}_t$, $C(\cdot) = \mathcal{A}_t(\cdot)$, $n = 2n_t$, $m = n_t$, $x = I_t$, $y = x_t$, $\varphi(\cdot) = z_t(\cdot)$, and $\tilde{\varphi}(\cdot) = \bar{z}_t(\cdot)$. For $i = 1, \ldots, n_t$, let $\varphi_i(\cdot) = p_{t,i}g_t(I_t, \cdot, d_{t,i})$, $\tilde{\varphi}_i(\cdot) = p_{t,i}\tilde{g}_t(I_t, \cdot, d_{t,i})$, $\psi_i(x, y) = y$, $\Sigma_i' = p_{t,i}\Sigma^W$, $\Pi_i' = \Pi^W$, $\Sigma_i = p_{t,i}\Sigma^g$, $\Pi_i = \Pi^g$, and $W_i(x) = W_i(I_t)$. For $i = n_t + 1, \ldots, 2n_t$, let $\varphi_i(\cdot) = p_{t,i-n_t}z_{t+1}(\cdot)$, $\tilde{\varphi}_i(\cdot) = p_{t,i-n_t}\tilde{z}_{t+1}(\cdot)$, $\psi_i(x, y) = f_t(x, y, d_{t,i-n_t})$, and $\Sigma_i = p_{t,i-n_t}\Sigma^z$, $\Pi_i = \Pi^z$.

Because $\tilde{g}_t(I_t, \cdot, d_{t,i})$, $\tilde{z}_{t+1}$, and $\psi_i(I_t, \cdot)$ are monotone, the function $\tilde{\varphi}_i(\psi_i(I_t, \cdot))$ is monotone for $i = 1, \ldots, 2n_t$. In addition, $\tilde{\varphi}_i(\psi_i(I_t, \cdot))$ is monotone in one direction for $i = 1, \ldots, n_t$ and is monotone in the other direction for $i = n_t + 1, \ldots, 2n_t$. Thus, by Proposition 3.4, $\tilde{\varphi}$ is a $(\sum_{i=1}^m(\Sigma_i' + \Pi_i'\Sigma_i) +$

$\sum_{i=m+1}^{n} \Sigma_i, \max\{\Pi_1\Pi'_1, \ldots, \Pi_m\Pi'_m, \Pi_{m+1}, \ldots, \Pi_n\}$)-approximation of φ. Hence, $\bar{z}_t$ is a $(\Sigma^W + \Pi^W\Sigma^g + \Sigma^z, \max\{\Pi^W\Pi^g, \Pi^z\})$-approximation of z_t (i.e., a $(\Sigma^W + \Pi^W\Sigma^g + \Sigma^z, \Pi^z)$-approximation of z_t).

The running time analysis for both cases 2(i) and 2(ii) is identical: In equation (19), the minimum of the function can be obtained in $|W^{-1}(I_t)|$ steps by scanning all the elements of $W^{-1}(I_t)$. Each of these steps involves n_t queries to $\tilde{g}_t$, f_t, and $\tilde{z}_{t+1}$, and requires $O(n_t(t_{\tilde{g}_t} + t_{f_t} + t_{\tilde{z}_{t+1}}))$ time. □

Remark 8 *As will be shown in Proposition 5.4, function z_t in the DP formulation* (3) *is guaranteed to be monotone. But $\bar{z}_t$ is not necessarily so. In Section 5.2 we will build from $\bar{z}_t$ a monotone approximation for z_t, by using Proposition 3.5*

Remark 9 *From Propositions 5.1 and 5.2, we can see that approximating the stochastic DP recursion* (3) *is essentially as hard as approximating the deterministic counterpart of the problem (i.e., when the random variable is constant with probability 1), except for an additional complexity factor of n_t (i.e., the size of the support of the random variable). This situation is substantially different from the determination of an exact solution for the problem. For example, the single-item stochastic inventory control problem with discrete demand is #P-hard [HKM+09, Thm. 4.2], but it is known that the deterministic counterpart of this problem can be solved in polynomial time as a minimum convex cost network flow problem or as a linear program [FLR80, Sec. 4].*

5.1 A (Σ, Π)-FPTAS for convex DP

We now develop an FPTAS for convex DPs. Our FPTAS is stated as Algorithm 8. The main result of this section is stated in the next theorem.

1: **Procedure FPTASConvDP**(δ, ϵ)
2: $\Sigma \leftarrow \frac{\delta}{2(T+1)(1+\epsilon)}$, $\Pi \leftarrow \sqrt[T+1]{1+\epsilon}$, $(W_{T+1}, \breve{z}_{T+1}) \leftarrow$ CompressConv$(g_{T+1}, \mathcal{S}_{T+1}, \Sigma, \Pi)$
3: **for** $t := T$ **downto** 1 **do**
4: $\quad (W_t, \breve{z}_t) \leftarrow$ CompressConv$(\bar{z}_t, \mathcal{S}_t, \Sigma, \Pi)$ /* *$\bar{z}_t$ is as defined in* (18)*; see details in the text* */
5: **end for**

Algorithm 8: FPTAS for convex DP

Theorem 5.3 (FPTAS for convex DP) *Consider a DP that satisfies Conditions 1, 2, and 3(iii). For any $0 < \epsilon < 1$ and $\delta > 0$, $\breve{z}_1(\cdot)$ that is generated by step 4 in the last iteration of Algorithm 8 is a convex $(\delta, 1+\epsilon)$-approximation function of the optimal cost $z^*(\cdot)$. Moreover, Algorithm 8 runs in time polynomial in $\frac{1}{\epsilon}$, $\log\frac{1}{\delta}$ and the (binary) input size.*

Proof. Note first that Proposition 5.1 ensures that the $\bar{z}_t$'s are all convex functions. Hence, all calls to function CompressConv in step 4 are valid.

Next, we prove that Algorithm 8 returns a $(\delta, 1+\epsilon)$-approximation solution. To do so, we first show by induction that $\breve{z}_t$ is a convex $(-\Sigma + 2\sum_{i=1}^{T+2-t} \Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t that is given explicitly. By Proposition 3.7, in step 2 we compute a set W_{T+1} so that $\breve{z}_{T+1}$ is the piecewise linear extension of z_{T+1} induced by W_{T+1} and is a convex (Σ, Π)-approximation of z_{T+1}. Thus, the base case of $t = T+1$ is valid. (Note that by Remark 5, W_{T+1} is a $(0,1)$-approximation set of $\breve{z}_{T+1}$.) The induction hypothesis is that function CompressConv computes a set W_{t+1} so that $\breve{z}_{t+1}$ is the piecewise linear extension of $\bar{z}_{t+1}$ induced by W_{t+1} which is a convex $(-\Sigma + 2\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma, \Pi^{T+1-t})$-approximation function of z_{t+1}. Again, by Remark 5, W_{t+1} is a $(0,1)$-approximation set of $\breve{z}_{t+1}$. We will show that $\breve{z}_t$ is a convex $(-\Sigma + 2\sum_{i=1}^{T+2-t} \Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t.

We apply Proposition 5.1 with parameters set to $\breve{g}_t = g_t$, $\Sigma^g = 0, \Pi^g = 1$ (since $\breve{g}_t \equiv g_t$), $\Sigma^z = -\Sigma + 2\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma$ and $\Pi^z = \Pi^{T+1-t}$ (since $\breve{z}_{t+1}$ is a convex $(-\Sigma + 2\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma, \Pi^{T+1-t})$-approximation of z_{t+1}). It follows that $\bar{z}_t$ is a convex $(2\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma, \Pi^{T+1-t})$-approximation function of z_t. Applying Proposition 3.8 with $\varphi = z_t$, $\breve{\varphi} = \bar{z}_t$, $\Sigma' = 2\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma$, $\Pi' = \Pi^{T+1-t}$, and $[A, B] = \mathcal{S}_t$, we obtain a set W_t so that $\breve{z}_t$ is the piecewise linear extension of $\bar{z}_t$ induced by W_t and is a convex $(-\Sigma + 2\sum_{i=1}^{T+2-t} \Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t. This completes the proof by induction, and the result implies that $\tilde{z}_1$ is a $(-\Sigma + 2\sum_{i=1}^{T+1} \Pi^{i-1}\Sigma, \Pi^{T+1})$-approximation of z_1. By the choice of Σ, Π, it follows that $\tilde{z}_1$ is a $(\delta, 1+\epsilon)$-approximation of z_1 as claimed.

It remains to prove that the running time of Algorithm 8 is polynomial in both the input size and $\frac{1}{\epsilon}\log\frac{1}{\delta}$. Recall that $\log U_{\mathcal{S}}$, $\log U_{\mathcal{A}}$, and $\log U_g$ are all part of the input size and that κ is the largest Lipschitz constant among all the functions mentioned in Condition 2. For ease of exposition, we assume that (i) the values of $U_{\mathcal{S}}$, $U_{\mathcal{A}}$, and U_g are at least 2 (so that their logarithmic values are at least 1); and (ii) the error parameters satisfy $\epsilon < 1$ (multiplicative error less than 2) and $\delta < TU_g$ (additive error less than maximal possible value of a feasible solution); and (iii) in step 2 we replace Π by an even smaller parameter Π^*, i.e. $\Pi^* \leftarrow 1 + \frac{\epsilon}{2(T+1)}$. (Because the inequality $(1+\frac{x}{n})^n \leq 1 + 2x$ holds for every $0 \leq x \leq 1$ and $n \in \mathbb{N}$, we have indeed that $\Pi^* = 1 + \frac{\epsilon}{2(T+1)} \leq \sqrt[T+1]{1+\epsilon} = \Pi$). This, of course, decreases the approximation factor and increases the running time.

The running time of Algorithm 8 is dominated by the for-loop, which has T iterations. Each iteration consists of a single call to function CompressConv. By Proposition 3.8, each execution of function CompressConv takes $O(t_{\bar{z}_t}|W_t|\log\frac{\kappa_{\bar{z}_t}U_{\mathcal{S}}}{\Sigma})$ time, where $|W_t| = O(\frac{T}{\epsilon}\log\min\{\frac{\sigma^{\max}_{\bar{z}_t}}{\sigma^{\min}_{\bar{z}_t}}, \frac{\epsilon U_g}{\Sigma}\})$. (Note that (i) here, and anywhere in the rest of the proof, we use the modified parameter Π^*; and (ii) the maximum possible value of $\bar{z}_t$ is bounded from above by $\Pi^{T+2-t}(T+2-t)U_g + (T+2-t)\Sigma \sim O(TU_g)$, as $\Pi^{T+1} \leq 1+\epsilon < 2$ and $\delta < TU_g$.)

By Proposition 5.1, evaluating $\bar{z}_t$ takes $O\left(n_t(t_{g_t} + t_{f_t} + t_{\breve{z}_{t+1}})\log\frac{\kappa_{\bar{z}_t}\kappa U_{\mathcal{A}}}{\Sigma}\right)$ time. Note also that $\breve{z}_{t+1}$ (which is obtained from the previous iteration of the for-loop) is stored succinctly in a sorted array of size $|W_{t+1}|$. Hence, $t_{\breve{z}_{t+1}} = O(\log|W_{t+1}|)$. It follows that

$$t_{\bar{z}_t} = O\left(n_t\big(t_{g_t} + t_{f_t} + \log(\frac{T}{\epsilon}\log\min\{\frac{\sigma^{\max}_{\bar{z}_{t+1}}}{\sigma^{\min}_{\bar{z}_{t+1}}}, \frac{\epsilon U_g}{\Sigma}\})\big)\log\frac{\kappa_{\bar{z}_t}\kappa U_{\mathcal{A}}}{\Sigma})\right).$$

This in turn implies that each execution of function CompressConv takes

$$O\left(n_t\big(t_{g_t} + t_{f_t} + \log(\frac{T}{\epsilon}\log\min\{\frac{\sigma^{\max}_{\bar{z}_{t+1}}}{\sigma^{\min}_{\bar{z}_{t+1}}}, \frac{\epsilon U_g}{\Sigma}\})\big)\frac{T}{\epsilon}\log\min\{\frac{\sigma^{\max}_{\bar{z}_t}}{\sigma^{\min}_{\bar{z}_t}}, \frac{\epsilon U_g}{\Sigma}\}\log\frac{\kappa_{\bar{z}_t}U_{\mathcal{S}}}{\Sigma}\log\frac{\kappa_{\bar{z}_t}\kappa U_{\mathcal{A}}}{\Sigma}\right)$$

time. Therefore, substituting Σ with $O(\frac{\delta}{T})$, we conclude that the running time of the entire algorithm is

$$O\left(\frac{T^2 n^*}{\epsilon}\left[t_{g_t} + t_{f_t} + \log(\frac{T}{\epsilon}\log\min\{\frac{\sigma^{\max}_{\bar{z}}}{\sigma^{\min}_{\bar{z}}}, \frac{\epsilon TU_g}{\delta}\})\right]\log\min\{\frac{\sigma^{\max}_{\bar{z}}}{\sigma^{\min}_{\bar{z}}}, \frac{\epsilon TU_g}{\delta}\}\log\frac{\kappa_{z_1}TU_{\mathcal{S}}}{\delta}\log\frac{\kappa_{z_1}\kappa TU_{\mathcal{A}}}{\delta}\right), \tag{20}$$

where $\frac{\sigma^{\max}_{\bar{z}}}{\sigma^{\min}_{\bar{z}}} := \max_{t=1,\ldots,T+1}\frac{\sigma^{\max}_{\bar{z}}}{\sigma^{\min}_{\bar{z}}}$. We conclude by noting that the term in (20) is polynomial in $\frac{1}{\epsilon}\log\frac{1}{\delta}$ and the input size. □

Remark 10 *The dependency of the running time of the algorithm on T is at most $T^2\log^4 T\log\log T$, on ϵ is at most $\frac{1}{\epsilon}\log\frac{1}{\epsilon}$, and on δ is at most $\log^3\frac{1}{\delta}\log\log\frac{1}{\delta}$.*

5.2 A (Σ, Π)-FPTAS for monotone DP

It is now possible to develop an FPTAS for monotone DPs. Our FPTAS is summarized in Algorithm 9.

1: **Procedure FPTASMonDP**(δ, ϵ)
2: $\Sigma \leftarrow \frac{\delta}{(T+1)(1+\epsilon)}$, $\Pi \leftarrow \sqrt[T+1]{1+\epsilon}$, $(W_{T+1}, \tilde{z}_{T+1}) \leftarrow$ CompressMon$(g_{T+1}, \mathcal{S}_{T+1}, \Sigma, \Pi)$
3: **for** $t := T$ **downto** 1 **do**
4: $\quad (W_t, \tilde{z}_t) \leftarrow$ CompressMon$(\bar{z}_t, \mathcal{S}_t, \Sigma, \Pi)$ /* *$\bar{z}_t$ is as defined in* (19)*; see details in the text* */
5: **end for**

Algorithm 9: FPTAS for monotone DP

For brevity, in the remaining of this section we deal only with increasing DPs. Performing an analysis for decreasing DPs is similar. To prove that Algorithm 9 is indeed a (Σ, Π)-FPTAS, we look for invariant properties throughout the execution of the algorithm. Such a property is stated in the next proposition.

Proposition 5.4 (Monotone Invariant) *If Condition 3(i) is satisfied, then for every $t = 1, \ldots, T+1$, function z_t in the DP formulation* (3) *is increasing over $\mathcal{S}_t$.*

Prop. 5.4 can be shown by backward induction, and the proof follows the same steps as the proof of the Monotone Invariant for discrete monotone DPs, see [HKL$^+$14, Prop. 8.1]. For the sake of completeness we give a proof in the Appendix. The main result of this section is stated in the next theorem.

Theorem 5.5 (FPTAS for monotone DP) *Consider a DP that satisfies Conditions 1, 2, and 3(i). For any $0 < \epsilon < 1$ and $\delta > 0$, $\tilde{z}_1(\cdot)$ that is generated by step 4 in the last iteration of the for-loop in Algorithm 9 is a $(\delta, 1+\epsilon)$-approximation of the optimal cost $z^*(\cdot)$. Moreover, Algorithm 9 runs in time polynomial in $\frac{1}{\epsilon}$, $\log \frac{1}{\delta}$ and the (binary) input size.*

Proof. We prove first that Algorithm 9 returns a $(\delta, 1+\epsilon)$-approximation solution. To do so, we first show by induction that $\tilde{z}_t$ is an increasing $(\sum_{i=1}^{T+2-t} \Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t that is given explicitly. By Proposition 3.3, in step 2 we get a set W_{T+1} so that $\tilde{z}_{T+1}$ is the monotone extension of z_{T+1} induced by W_{T+1}, and is therefore an increasing (Σ, Π)-approximation of z_{T+1}. Thus, the base case of $t = T+1$ is valid. (Note that by Remark 5, W_{T+1} is a $(0,1)$-approximation set of $\tilde{z}_{T+1}$.) The induction hypothesis is that function CompressMon computes a set W_{t+1} so that $\tilde{z}_{t+1}$ is the monotone extension of $\bar{z}_{t+1}$ induced by W_{t+1} and is therefore an increasing $(\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma, \Pi^{T+1-t})$-approximation function of z_{t+1}. Again, by Remark 5, W_{t+1} is a $(0,1)$-approximation set of $\tilde{z}_{t+1}$. We will show that $\tilde{z}_t$ is an increasing $(\sum_{i=1}^{T+2-t} \Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t.

We first consider the case where oracles for f_t^{-1} are given (i.e., in Proposition 5.2, case (i) holds). In the for-loop we iteratively call function CompressMon where the value of $\bar{z}_t$ is calculated via (19) with the exact value of g_t and with the set W_{t+1} that was already computed in previous iterations. Specifically, the evaluation of $\bar{z}_t$ in step 4 of Algorithm 9 is performed by applying Proposition 5.2(i) with parameters set to $W = W_{t+1}$, $\tilde{g}_t = g_t$, $\Sigma^g = \Sigma^W = 0$, $\Pi^g = \Pi^W = 1$, $\Sigma^z = \sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma$, and $\Pi^z = \Pi^{T+1-t}$. This way we get that $\bar{z}_t$ is a not necessarily monotone $(\sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma, \Pi^{T+1-t})$-approximation of z_t

Note that by the Monotone Invariant, z_t is a monotone increasing function. Therefore, by applying Proposition 3.5 with $\varphi = z_t$, $\bar{\varphi} = \bar{z}_t$, $\Sigma' = \sum_{i=1}^{T+1-t} \Pi^{i-1}\Sigma$, $\Pi' = \Pi^{T+1-t}$, and $[A, B] = \mathcal{S}_t$, we have that in step 4 we get a set W_t so that $\tilde{z}_t$ is the monotone extension of $\bar{z}_t$ induced by W_t and is an increasing $(\sum_{i=1}^{T+2-t} \Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t. This completes the proof by induction, and the result implies that $\tilde{z}_1$ is a $(\sum_{i=1}^{T+1} \Pi^{i-1}\Sigma, \Pi^{T+1})$-approximation of z_1. By the choice of Σ, Π we get that $\tilde{z}_1$ is a $(\delta, 1+\epsilon)$-approximation of z_1 as claimed.

In the case where oracles for f_t^{-1} are not given (i.e., in Proposition 5.2, case (ii) holds), we slightly change the algorithm so that the value of the parameter Σ is reduced by half to $\frac{\delta}{2(T+1)(1+\epsilon)}$, and where

in each iteration we also calculate $(\Sigma/\Pi, \Pi^{T+1-t})$-approximation sets V_i for $g_t(I_t, \cdot, d_{t,i}),\ i = 1, \ldots, n_t$. In this case the induction hypothesis is that function COMPRESSMON computes a set W_{t+1} so that $\tilde{z}_{t+1}$ is the monotone extension of $\bar{z}_{t+1}$ induced by W_{t+1} and is an increasing $(2\sum_{i=1}^{T+1-t}\Pi^{i-1}\Sigma, \Pi^{T+1-t})$-approximation function of z_{t+1}. We will show that $\tilde{z}_t$ is a an increasing $((1+\Pi(1/\Pi+2\sum_{i=1}^{T+1-t}\Pi^{i-1}))\Sigma, \Pi^{T+2-t})$-approximation function of z_t, i.e., $\tilde{z}_t$ is a an increasing $(2\sum_{i=1}^{T+2-t}\Pi^{i-1}\Sigma, \Pi^{T+2-t})$-approximation function of z_t.

In the for-loop we iteratively call function COMPRESSMON where the value of $\bar{z}_t$ is calculated via (19) with the exact value of g_t and the sets V_i that were calculated in the current iteration. Specifically, the evaluation of $\bar{z}_t$ in step 4 of Algorithm 9 is performed by applying Proposition 5.2(ii) with parameters set to $\Sigma^g = \Sigma/\Pi,\ W_i = V_i,\ \Pi^W = 1, \Sigma^W = 0,\ \Pi^g = \Pi^{T+1-t}, \Sigma^z = 2\sum_{i=1}^{T+1-t}\Pi^{i-1}\Sigma$, and $\Pi^z = \Pi^{T+1-t}$. This way we get that $\bar{z}_t$ is a not necessarily monotone $((1/\Pi + 2\sum_{i=1}^{T+1-t}\Pi^{i-1})\Sigma, \Pi^{T+1-t})$-approximation of z_t

Note that by the Monotone Invariant, z_t is a monotone increasing function. Therefore, by applying Proposition 3.5 with $\varphi = z_t$, $\bar{\varphi} = \bar{z}_t$, $\Sigma' = (1/\Pi + 2\sum_{i=1}^{T+1-t}\Pi^{i-1})\Sigma$, $\Pi' = \Pi^{T+1-t}$, and $[A, B] = \mathcal{S}_t$, we have that in step 4 we get a set W_t so that $\tilde{z}_t$ is the monotone extension of $\bar{z}_t$ induced by W_t and is an increasing $((1 + \Pi(1/\Pi + 2\sum_{i=1}^{T+1-t}\Pi^{i-1}))\Sigma, \Pi^{T+2-t})$-approximation function of z_t. This completes the proof by induction, and the result implies that $\tilde{z}_1$ is a $(2\sum_{i=1}^{T+1}\Pi^{i-1}\Sigma, \Pi^{T+1})$-approximation of z_1. By the choice of Σ, Π we get that $\tilde{z}_1$ is a $(\delta, 1+\epsilon)$-approximation of z_1 as claimed.

It remains to prove that the running time of Algorithm 9 is polynomial in both the input size and $\frac{1}{\epsilon}\log\frac{1}{\delta}$. Recall that $\log U_{\mathcal{S}}$, $\log U_{\mathcal{A}}$, and $\log U_g$ are all part of the input size. For ease of exposition, we assume that (i) the values of $U_{\mathcal{S}}$, $U_{\mathcal{A}}$, and U_g are at least 2 (so that their logarithmic values are at least 1); and (ii) the error parameters make sense, e.g., $\epsilon < 1$ (multiplicative error less than 2) and $\delta < TU_g$ (additive error less than maximal possible value of a feasible solution); and (iii) in step 2 we set Π to be even smaller, i.e. $\Pi \leftarrow 1 + \frac{\epsilon}{2(T+1)}$. (Because the inequality $(1+\frac{x}{n})^n \leq 1+2x$ holds for every $0 \leq x \leq 1$ and $n \in \mathbb{N}$, we have indeed that $1 + \frac{\epsilon}{2(T+1)} \leq \sqrt[T+1]{1+\epsilon}$). This, of course, decreases the approximation factor and increases the running time. We will first consider the case where oracles for f_t^{-1} are given (so that in Proposition 5.2, case (i) holds). We then explain in short why essentially the same bound holds for the other case.

The running time of Algorithm 9 is dominated by the for-loop, which has T iterations. Each iteration consists of a single call to function COMPRESSMON. By Proposition 3.5, each execution of function COMPRESSMON takes $O(t_{\bar{z}_t}\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma}\log\frac{\kappa_{\bar{z}_t}U_{\mathcal{S}}}{\Sigma})$ time. (Note that (i) here, and anywhere in the rest of the proof, we use the modified relative ratio of $\frac{\epsilon}{2T}$; and (ii) the maximum possible value of $\bar{z}_t$ is bounded from above by $\Pi^{T+2-t}(T+2-t)U_g + (T+2-t)\Sigma \sim O(TU_g)$, as $\Pi^{T+1} \leq 1+\epsilon < 2$ and $\delta < TU_g$.) By Proposition 5.2(i), evaluating $\bar{z}_t$ takes $O(n_t(t_{g_t} + t_{f_t} + t_{\tilde{z}_{t+1}})|W^{-1}(I_t)|)$ time once $W^{-1}(I_t)$ is given. The time needed to construct $W^{-1}(I_t)$ is $O(n_t|W|t_{f_t^{-1}})$, once W is given. Thus, $t_{\bar{z}_t} = O\big(n_t(t_{g_t} + t_{f_t} + t_{f_t^{-1}} + t_{\tilde{z}_{t+1}})|W^{-1}(I_t)|\big)$. By Proposition 3.5, $|W| = O(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma})$. Note also that $\tilde{z}_{t+1}$ (which is obtained from the previous iteration of the for-loop) is stored succinctly in a sorted array of size $|W_{t+1}|$. Hence, $t_{\tilde{z}_{t+1}} = O(\log|W_{t+1}|) = O(\log(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma}))$. This implies that (we use here $|W^{-1}(I_t)| = O(n_t|W|)$)

$$t_{\bar{z}_t} = O\big(n_t^2(t_{g_t} + t_{f_t} + t_{f_t^{-1}} + \log(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma}))(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma}))\big).$$

This in turn implies that each execution of function COMPRESSMON takes

$$O\big(n_t^2(t_{g_t} + t_{f_t} + t_{f_t^{-1}} + \log(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma}))(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma})(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma}\log\frac{\kappa_{\bar{z}_t}U_S}{\Sigma})\big)$$

time. Therefore, the running time of the entire algorithm is

$$O\left(\frac{T^3(n^*)^2}{\epsilon^2}\left[t_g + t_f + t_{f^{-1}} + \log(\frac{T}{\epsilon}\log\frac{\epsilon U_g}{\Sigma})\right]\log^2\frac{\epsilon U_g}{\Sigma}\log\frac{\kappa_{z_1}U_{\mathcal{S}}}{\Sigma}\right).$$

Substituting Σ with $O(\frac{\delta}{T})$, we conclude that the running time of the entire algorithm is

$$O\left(\frac{T^3(n^*)^2}{\epsilon^2}\left[t_g + t_f + t_{f^{-1}} + \log(\frac{T}{\epsilon}\log\frac{\epsilon T U_g}{\delta})\right]\log^2\frac{\epsilon T U_g}{\delta}\log\frac{\kappa_{z_1}T U_{\mathcal{S}}}{\delta}\right). \tag{21}$$

which is polynomial in $\frac{1}{\epsilon}\log\frac{1}{\delta}$ and the input size.

It remains to consider the case where oracles for f_t^{-1} are not given (so that in Proposition 5.2, case (ii) holds). In this case for every iteration of the for-loop an extra computational cost of computing the various n_t approximation sets of $g_t(I_t,\cdot,d_{t,i})$ is added, which by Proposition 3.3 is $O\left(n_t(1+t_{g_t})\frac{T}{\epsilon}\log\frac{\epsilon T U_g}{\delta}\log\frac{\kappa T U_{\mathcal{A}}}{\delta}\right)$ (recall that κ is the largest Lipschitz constant among all the functions mentioned in Condition 2). □

Remark 11 *The dependency of the running time of the algorithm on T is at most $T^3\log^4 T\log\log T$, on ϵ is at most $\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}$, and on δ is at most $\log^3\frac{1}{\delta}\log\log\frac{1}{\delta}$.*

6 Purely-multiplicative FPTASs

In this section we prove Theorem 1.7. We begin by stating more precisely a condition under which a continuous DP admits a (purely multiplicative) FPTAS, assuming that Conditions 1 and 3 are also satisfied.

Condition 4 *For every $t = 1,\ldots,T+1$, the functions g_t, f_t are either given explicitly (i.e., as explicit formulae) or accessed via value oracles. Moreover, f_t, g_t and the lower and upper envelopes of $\cup_{I_t\in\mathcal{S}_t}\mathcal{A}_t(I_t)$ (i.e., functions $\mathcal{A}_t^{\min}(\cdot),\mathcal{A}_t^{\max}(\cdot)$) are all Lipschitz continuous functions and g_t is nonnegative. Moreover, function g_{T+1} is bounded away from zero and a lower bound $\underline{z}^{\min} > 0$ on the value of z_{T+1} is given.*

Remark. We distinguish between the actual minimum value of a function φ, denoted by $\varphi^{\min}$, and a given lower bound to such a value, denoted by $\underline{\varphi}^{\min}$.

Next, we show what modifications are needed in the algorithms stated in Sections 2.3 and 3 in order to construct an FPTAS for convex DPs. The modifications needed to build an FPTAS for monotone DPs are similar and are therefore omitted.

6.1 An FPTAS for convex DP

To obtain an FPTAS for convex DPs, we modify function APXARGMIN to find an approximate $\arg\min$ for a given function $\varphi : [A,B]\to\mathbb{R}^+$ that has a multiplicative error of at most Π as follows. The parameters given to the modified algorithm are $\varphi, A, B, \underline{\varphi}^{\min}, \Pi$. We change steps 2 and 3 to:
Let $A' \leftarrow A,\ B' \leftarrow B, \kappa' \leftarrow 3(\Pi-1)\underline{\varphi}^{\min}/(B'-A')$
while $\kappa'(B'-A') > 2(\Pi-1)\underline{\varphi}^{\min}$ **do**

Following the analysis in the proof Proposition 2.2, we obtain that the modified function finds a point x' for which $\varphi(x')$ is a (multiplicative) Π-approximation of $\varphi^{\min}$ in $O(\log(\kappa(B-A)/\underline{\varphi}^{\min}))$ queries to φ.

We modify function APXSETCONVINC as follows. The parameters given to the modified algorithm are the same as the modified APXARGMIN. We change step 2 to:
$x \leftarrow$ FUNCSEARCHINC$(\varphi,[x,B],\frac{1}{2}(\Pi+1)\varphi(A),\Pi\varphi(A)),\ \ W\leftarrow\{A,x,B\}$

We change step 5 to:
$x \leftarrow \text{FUNCSEARCHINC}(\varphi, [x, B], \frac{1}{2}(\Pi+1)(\varphi(x)+\dot{\sigma}_\varphi(x)(\cdot - x)), \Pi(\varphi(x)+\dot{\sigma}_\varphi(x)(\cdot - x)))$

Following the analysis in the proof of Proposition 3.7, the modified APXSETCONVINC constructs a Π-approximation set W for φ in $O\left((1+t_\varphi)(\frac{1}{\epsilon}\log\min\{\frac{\sigma_\varphi^{\max}}{\sigma_\varphi^{\min}}, \frac{\varphi^{\max}}{\varphi^{\min}}\})\log\frac{\kappa(B-A)}{\underline{\varphi}^{\min}}\right)$ time. The cardinality of W is $O\left(\frac{1}{\epsilon}\log\min\{\frac{\sigma_\varphi^{\max}}{\sigma_\varphi^{\min}}, \frac{\varphi^{\max}}{\varphi^{\min}}\}\right)$. We now state the modified version of the remaining algorithms, leading to a purely multiplicative FPTAS.

1: **Function ApxSetConv**($\varphi, [A, B], \underline{\varphi}^{\min}, \Pi$)
2: $\tilde{x} \leftarrow \text{APXARGMIN}(\varphi, [A, B], \underline{\varphi}^{\min}, (\Pi+2)/3)$
3: **return** $\text{APXSETCONVDEC}(\varphi, [A, \tilde{x}], \underline{\varphi}^{\min}, \Pi) \cup \text{APXSETCONVINC}(\varphi, [\tilde{x}, B], \underline{\varphi}^{\min}, \Pi)$

Algorithm 10: A modified function APXSETCONV used for purely-multiplicative FPTASs

1: **Function CompressConv**($\varphi, [A, B], \underline{\varphi}^{\min}, \Pi$)
2: $W \leftarrow \text{APXSETCONV}(\varphi, [A, B], \underline{\varphi}^{\min}, \Pi)$
3: **return** $\{(x, \varphi(x)) \mid x \in W\}$ as a sorted array

Algorithm 11: A modified function COMPRESSCOV used for purely-multiplicative FPTASs

We are ready to present our modified FPTAS. It is easy to verify that the running time of the modified FPTAS is $O(\frac{T^2}{\epsilon})$, up to log terms.

1: **Procedure FPTASConvDP**($\delta, \epsilon, \underline{z}^{\min}$)
2: $\Pi \leftarrow \sqrt[T+1]{1+\epsilon}$, $(W_{T+1}, \breve{z}_{T+1}) \leftarrow \text{COMPRESSCONV}(g_{T+1}, \mathcal{S}_{T+1}, \underline{z}^{\min}, \Pi)$
3: **for** $t := T$ **downto** 1 **do**
4: $\quad (W_t, \breve{z}_t) \leftarrow \text{COMPRESSCONV}(\bar{z}_t, \mathcal{S}_t, \underline{z}^{\min}, \Pi)$ /* $\bar{z}_t$ *is as defined in* (18) */
5: **end for**

Algorithm 12: A purely-multiplicative FPTAS for convex DP

6.2 A purely-multiplicative FPTAS for nonlinear knapsack

In this section we demonstrate the use of our technique to improve the FPTAS of [CER+05] for a minimization version of the continuous nonlinear knapsack problem (called "supply scheduling" in that paper, but we find "nonlinear knapsack" to be a more recognizable name). The problem is formulated as:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n c_i(x_i) \\ \text{subject to} & \sum_{i=1}^n x_i \geq B \\ & x_i \in \{0\} \cup [\ell_i, u_i], \quad i = 1, \ldots, n \end{array} \tag{22}$$

The values B, ℓ_i, u_i, for $i = 1, \ldots, n$ are given positive integers. We assume without loss of generality that $\sum_i u_i \geq B$, so that there always exists a feasible solution. The decision variables x_i are semi-continuous. The cost functions $c_i(\cdot)$ are assumed strictly positive and (not necessarily Lipschitz) continuous increasing over $[\ell_i, u_i]$, and they are accessed via oracles as described below. We assume w.l.o.g. that $c_i(0) = 0$. (Otherwise, we can define a modified problem with cost function $\tilde{c}_i(\cdot) = c_i(\cdot) - c_i(0)$. The value of the optimal solution of the modified problem is reduced by $c_i(0)$ relatively to the original problem. Therefore, a $(1+\epsilon)$-approximation for the modified problem is a $(1+\epsilon)$-approximation for the original problem.)

[CER+05, Def. 1.1] assumes that every cost function $c_t(\cdot)$ can be accessed via an oracle that answers the following queries in constant time:

Q1: For a given real y, evaluate $c_i(y)$.

Q2: For a given real t, determine the supremum y_t of all numbers y in $[\ell_i, u_i]$ with $c_i(y) \leq t$.

[CER$^+$05, Sect. 3] shows that the availability of an FPTAS for Q2 queries, instead of exact supremums, is already sufficient. We next show that one can tradeoff the assumption about the availability of (an FPTAS for) Q2 queries with the assumption that the cost functions are Lipschitz continuous (as opposed to ordinary continuous). In fact, the only place where the FPTAS of [CER$^+$05] uses Q2 queries is for the construction of Π-approximation sets (called threshold sets in their paper) for the cost functions [CER$^+$05, Lem. 2.1]. We outline a different approach that uses Q1 queries only.

Suppose $\varphi : [A, B] \to \mathbb{R}^+$ has a single discontinuity jump at $x^{@}$, so that $\varphi(x) \equiv 0$ for $x < x^{@}$. We first slightly change the definition of the monotone extension of φ induced by a set W, provided that the smallest two elements in W are $A, x^{@}$:

$$\hat{\varphi}(x) = \begin{cases} \varphi(x) & \text{if } x \in W; \\ 0 & \text{if } x < x^{@}; \\ \max\{\varphi(\text{prev}(x, W)), \varphi(\text{next}(x, W))\} & \text{otherwise.} \end{cases} \tag{23}$$

Second, we change the definition of $\varphi^{\min}$ to be the minimum strictly positive value of φ. We can then construct a Π-approximation set for φ by modifying function ApxSetInc as follows: in addition to getting an interval $[A, B]$ as an input parameter, the algorithm gets the discontinuity point $x^{@}$ and the minimum strictly positive value of the function.

1: **Function ApxSetInc**$(\varphi, [A, x^{@}, B], \varphi^{\min}, \Pi)$
2: $x \leftarrow x^{@}$, $W \leftarrow \{A, x^{@}, B\}$
3: **if** $\varphi(x^{@}) = 0$ **then** $x \leftarrow$ FuncSearchInc$(\varphi, [x, B], \frac{1}{2}(\Pi + 1)\varphi^{\min}, \Pi\varphi^{\min})$; $W \leftarrow W \cup \{x\}$
4: **while** $x < B$ **do**
5: $\quad x \leftarrow$ FuncSearchInc$(\varphi, [x, B], \frac{1}{2}(\Pi + 1)\varphi(x), \Pi\varphi(x))$
6: $\quad W \leftarrow W \cup \{x\}$
7: **end while**
8: **return** W

Algorithm 13: A modified function ApxSetInc used for purely-multiplicative FPTASs

We briefly explain the differences between Algorithm 3 and Algorithm 13. Clearly, there is no need for parameter Σ when dealing with purely multiplicative approximations. We require instead the discontinuity point $x^{@}$ and $\varphi^{\min}$, the minimum strictly positive value of φ. Because we deal with multiplicative approximation, for step 5 to be well defined we must have a point x such that $\varphi(x) > 0$ (otherwise $\frac{1}{2}(\Pi + 1)\varphi(x) = \Pi\varphi(x) = 0$). Step 3 finds such a point x. (Note that if $\varphi(x^{@}) = 0$ then the function φ is Lipschitz continuous in the non-compact interval $(x^{@}, B]$.) Due to the definition of $\varphi^{\min}$, the value of φ over $(x^{@}, B]$ is never below $\varphi^{\min}$, so adding to W the point x as defined in step 3, as well as the points as defined in step 5, ensures that the monotone extension of φ induced by W, as defined in (23) gives a $(0, \Pi)$-approximation of φ. Following the proof of Proposition 3.3, we get that the modified function ApxSetMon (Algorithm 13) constructs a $(0, \Pi)$-approximation set W for φ of cardinality $O(\frac{1}{\epsilon} \log \frac{\varphi^{\max}}{\varphi^{\min}})$ in $O\big((1 + t_\varphi)(\frac{1}{\epsilon} \log \frac{\varphi^{\max}}{\varphi^{\min}}) \log \frac{(x^{@} - A)\kappa}{\epsilon\varphi^{\min}}\big)$ time. We conclude that the approximation set can be computed in time polynomially bounded in the size of the problem and in $\frac{1}{\epsilon}$, and using Q1 queries only. To compute approximation sets for the original cost functions c_i, we note that we can easily ensure that they all have a discontinuity jump at $x^{@} = 0$ by extending the definition of c_i over the entire interval $[0, u_i]$, setting $c_i(x) = c_i(\ell_i)$ for $0 < x < \ell_i$. Once the approximation sets for the cost functions are computed applying ApxSetMon as described above, we proceed with the algorithm of [CER$^+$05] and obtain an FPTAS for problem (22).

[CER+05, Sec. 3] states that the requirement for Q2 queries "is not so natural", but is justified by an example in which it "seems hard to do without" such queries. As discussed above, our approach does not need such queries and in fact deals with the example in [CER+05, Sec. 3] with a single call of FUNCSEARCHINC (more specifically, it is easy to check that FUNCSEARCHINC$(c_1, [1, 2], 2, 4)$ finds a point in what is called "hidden interval"). The key reason for our ability to do so is the fact that the example in [CER+05, Sec. 3] consists of an infinite family of Lipschitz continuous functions, and our routine FUNCSEARCHINC can deal with each one of them, even if the Lipschitz constant is not known a priori. We can strengthen their example by providing a non-Lipschitz cost function that cannot be dealt with in finite time by any algorithm without Q2 queries: such a function is the adversarial oracle given in the proof of Prop. 2.11.

To conclude, whenever the cost functions are known to be Lipschitz continuous, using our technique one can have an FPTAS to the problem while assuming the availability of Q1 queries only.

7 Extensions and concluding remarks

Our framework for designing FPTASs for stochastic DPs can be extended well beyond the results stated in Theorems 1.4, 1.6 and 1.7. However, there is a tradeoff between the level of generalization of a framework for designing FPTASs and the complication of its construction and analysis. The goal of this paper, in this respect, is to develop a "reasonable" sufficient set of conditions that guarantees the existence of an FPTAS, and to provide such an FPTAS. The conditions are satisfied by some basic problems in logistics, operations management, economics, and finance, as we demonstrate in Appendices B and C. In this section we briefly discuss an extension of the framework that requires numerous additional analytical results, and is therefore not presented in full in this paper.

Until now we have assumed that $D_1, \ldots, D_T$ are explicitly described as a lists of ordered pairs $(d_{t,i}, \text{Prob}(D_t = d_{t,i}))$. The limitation of explicitly listing the support of the random variables as problem input can be quite severe in practice and does not allow to effectively handle problems with a very rich uncertainty model, e.g., problems for which computing expectations by looping over the support is impractical.

An alternative model is that in which we have access to an oracle for the CDF of D_t. This model was studied in the context of approximation algorithms for DPs in [HOSL12, Hal16]. Using tools from these two papers, we can extend the algorithmic results of the current paper to DPs (with continuous state and action spaces) where the random variables are weighted sums of discrete or continuous random variables defined implicitly, i.e., via an oracle to their CDFs, over a bounded domain. We refer the reader to [HN17] for details. Intuitively, the idea is to define a finite sum (over a K-approximation set of the CDF) that approximates the composition of functions involving a discrete or continuous random variable.

To conclude, this paper shows hardness of the continuous version of the nonlinear newsvendor problem under several settings, but provides an efficient approximate solution approach by introducing the concept of (Σ, Π)-FPTAS. Such an approximation scheme allows both absolute and relative approximation at the same time. We show that it can be constructed for single-period nonlinear newsvendor problems, as well as for three classes of multi-period stochastic dynamic programs with continuous state and action spaces, under some conditions. Our scheme is "best possible" in the sense that under our setting, impossibility results shown in this paper prevent the construction of approximation schemes that rely on either absolute or relative approximation only. The paper therefore provides a constructive way to overcome hardness of approximation for an important class of problems that finds several applications in operations management.

References

[AC03] Jerome Adda and Russel W. Cooper. *Dynamic Economics: Quantitative Methods and Applications*. The MIT Press, Boston, MA, 2003.

[AHM51] Kenneth J. Arrow, Theodore Harris, and Jacob Marschak. Optimal inventory policy. *Econometrica*, 19:250–272, 1951.

[BD62] Richard Bellman and Stuart Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, NJ, 1962.

[Ber05] Dimitri Bertsekas. *Dynamic Programming and Optimal Control, Volume I*. Athena Scientific, Belmont, MA, 2005.

[BN13] Aharon Ben-Tal and Arkadi Nemirovski. Lectures on convex optimization, 2013.

[CCLL98] T.C.E. Cheng, Z.-L. Chen, C.-L. Li, and B.M.-T. Lin. Scheduling to minimize the total compression and late costs. *Naval Research Logistics*, 45:67–82, 1998.

[CER+05] Satyaveer S. Chauhan, Anton V. Eremeev, Anna A. Romanova, Vladimir V. Servakh, and Gerhard J. Woeginger. Approximation of the supply scheduling problem. *Operations Research Letters*, 33:249–254, 2005.

[CFN77] Gérard Cornuéjols, Marshall L Fisher, and George L Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management science*, 23(8):789–810, 1977.

[CKP06] Sergey Chubanov, Mikhail Y. Kovalyov, and Erwin Pesch. An FPTAS for a single-item capacitated economic lot-sizing problem with monotone cost structure. *Mathematical Programming*, 106(2):453–466, 2006.

[dFVR03] Daniela Pucci de Farias and Benjamin Van Roy. The Linear Programming approach to approximate Dynamic Programming. *Operations Research*, 51(6):850–865, 2003.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006*, pages 486–503. 2006.

[DM02] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[EP04] Michael Elkin and David Peleg. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004.

[FIMN13] Uriel Feige, Nicole Immorlica, Vahab S Mirrokni, and Hamid Nazerzadeh. Pass approximation: A framework for analyzing and designing heuristics. *Algorithmica*, 66(2):450–478, 2013.

[FLR80] Michael Florian, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan. Deterministic production planning: Algorithms and complexity. *Management Science*, 26(7):669–679, 1980.

[FSS13] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453, 2013.

[GP01] Gregory A. Godfrey and Warren B. Powell. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science*, 47(8):1101–1112, 2001.

[Hal16] Nir Halman. Provably near-optimal approximation schemes for implicit stochastic and for sample-based dynamic programs. Technical Report 4952, Optimization Online, January 2016.

[HKL+14] Nir Halman, Diego Klabjan, Chung-Lun Li, Jim Orlin, and David Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *SIAM Journal on Discrete Mathematics*, 28(4):1725–1796, 2014.

[HKM+09] Nir Halman, Diego Klabjan, Mohamad Mostagir, Jim Orlin, and David Simchi-Levi. A fully polynomial time approximation scheme for single-item inventory control with discrete demand. *Mathematics of Operations Research*, 34(3):674–685, 2009.

[HN17] Nir Halman and Giacomo Nannicini. Toward breaking the curse of dimensionality: an FPTAS for stochastic dynamic programs with multidimensional action and scalar state. Technical Report 6127, Optimization Online, July 2017.

[HNO15] Nir Halman, Giacomo Nannicini, and James Orlin. A computationally efficient FPTAS for convex stochastic dynamic programs. *SIAM Journal on Optimization*, 25(1):317–350, 2015.

[Hoc97] Dorit Hochbaum. Various notions of approximations: Good, better, best, and more. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, Boston, MA, 1997.

[HOSL12] Nir Halman, James B. Orlin, and David Simchi-Levi. Approximating the nonlinear newsvendor and single-item stochastic lot-sizing problems when data is given by an oracle. *Operations Research*, 60(2):429–446, 2012.

[Kho99] Moutaz Khouja. The single-period (news-vendor) problem: literature review and suggestions for future research. *Omega*, 27(5):537–553, 1999.

[KK82] Narendra Karmarkar and Richard M Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 312–320, 1982.

[KPR04] Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. Segmentation problems. *Journal of the ACM*, 51:263–280, 2004.

[LLKS93] Eugene L Lawler, Jan Karel Lenstra, Alexander HG Rinnooy Kan, and David B Shmoys. Sequencing and scheduling: algorithms and complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors, *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*, pages 445–522. North-Holland, Amsterdam, 1993.

[MS13] Shashi Mittal and Andreas S. Schultz. A general framework for designing approximation schemes for combinatorial optimization problems with many objectives combined into one. *Operations Research*, 61(2):386–397, 2013.

[NP10] Juliana M. Nascimento and Warren B. Powell. Dynamic programming models and algorithms for the mutual fund cash balance problem. *Management Science*, 56(5):801–815, 2010.

[NP13] Juliana M. Nascimento and Warren B. Powell. An optimal approximate dynamic programming algorithm for concave, scalar storage problems with vector-valued controls. *IEEE Transactions on Automatic Control*, 58(12):2995–3010, 2013.

[Phe62] Edmund S. Phelps. The accumulation of risky capital: A sequential utility analysis. *Econometrica*, 30:729–743, 1962.

[Por90] Evan L. Porteus. Stochastic inventory theory. In D.P. Heyman and M.J. Sobel, editors, *Handbook in OR & MS*, volume 2. Elsevier Science Publishers B.V., North-Holland, 1990.

[Pow11] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition*. Wiley, 2011.

[PRT04] Warren B. Powell, Andrzej Ruszczyński, and Huseyin Topaloglu. Learning algorithms for separable approximations of discrete stochastic optimization problems. *Mathematics of Operations Research*, 29(4):814–836, 2004.

[PW07] Kirk Pruhs and Gerhard J. Woeginger. Approximation schemes for a class of subset selection problems. *Theoretical Computer Science*, 382(2):151–156, 2007.

[QWV+11] Yan Qin, Ruoxuan Wang, Asoo J Vakharia, Yuwen Chen, and Michelle MH Seref. The newsvendor problem: Review and directions for future research. *European Journal of Operational Research*, 213(2):361–374, 2011.

[SCB14] David Simchi-Levi, Xin Chen, and Julien Bramel. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*. Springer-Verlag, NY, NY, third edition, 2014.

[Sha13] Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *Proceedings of the Conference on Learning Theory*, pages 3–24, 2013.

[SS06] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM*, 53(6):978–1012, 2006.

[SS12] Chaitanya Swamy and David B. Shmoys. Sampling-based approximation algorithms for multistage stochastic optimization. *SIAM Journal on Computing*, 41(4):975–1004, 2012.

[ST04] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[Van01] Albert P. M. Van Hoesel, Constantinus P. M. Wagelmans. Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Mathematics of Operations Research*, 26(2):339–357, 2001.

[Vaz01] Vijay Vazirani. *Approximation Algorithms*. Springer, Berlin, 2001.

[Woe00] Gerhard J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.

[WW58] Harvey M. Wagner and Thomson M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.

A Proof of Proposition 5.4

(The proof is taken from [HKL$^+$14, Prop. 8.1])

Proof. We use backward induction. We first consider the base case of $t = T+1$. Because $z_{T+1} \equiv g_{T+1}$ and g_{T+1} is increasing, z_{T+1} is increasing. Now, consider any $t = 1, \ldots, T$, and assume that z_{t+1} is increasing. By condition 3(*i*), either z_t is increasing or $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$ for all $I, I' \in \mathcal{S}_t$ with $I \geq I'$. Thus, it suffices to show that if $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$ for all $I, I' \in \mathcal{S}_t$ with $I \geq I'$, then z_t is increasing. Because $f_t(\cdot, x_t, D_t)$ is increasing, so is the composition function $z_{t+1}(f_t(\cdot, x_t, D_t))$. Because $g_t(\cdot, x_t, D_t)$ is increasing, so is the sum $g_t(\cdot, x_t, D_t) + z_{t+1}(f_t(\cdot, x_t, D_t))$. This implies that $\mathbb{E}_{D_t}\{g_t(\cdot, x_t, D_t) + z_{t+1}(f_t(\cdot, x_t, D_t))\}$ is increasing. If $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$ for all $I, I' \in \mathcal{S}_t$ with $I \geq I'$, then $\mathcal{A}_t(\cdot)$ is decreasing (by set containment) over $\mathcal{S}_t$, which implies that the minimization $\min_{x_t \in \mathcal{A}_t(\cdot)} \mathbb{E}_{D_t}\{g_t(\cdot, x_t, D_t) + z_{t+1}(f_t(\cdot, x_t, D_t))\}$ is increasing over $\mathcal{S}_t$. Hence, z_t is increasing. □

B An example: Time-cost trade-off machine scheduling

In this section we give an example for a specific deterministic DP that illustrates the additional complexity incurred when dealing with continuous state and action spaces, and show how this problem fits into our framework.

Consider the following machine scheduling problem: There are one single machine and n jobs $J_1, \ldots, J_n$. Job J_j has a given due date $d_j \in \mathbb{R}^+$, a late penalty $w_j \in \mathbb{R}^+$, a "normal" processing time $\bar{p}_j \in \mathbb{R}^+$, and a decreasing resource consumption function $\rho_j : \mathbb{R}^+ \to \mathbb{R}^+$ with $\rho_j(x) = 0$ for any $x \geq \bar{p}_j$. The processing time of J_j, denoted as x_j, is a nonnegative decision variable, and a cost of $\rho_j(x_j)$ is incurred if x_j is chosen to be less than $\bar{p}_j$. All jobs are available for processing at time 0, and job preemption is not allowed. The objective is to determine the job processing times and to schedule the jobs on the machine so that the total cost, $\sum_{j=1}^{n}[w_j \delta_{C_j > d_j} + \rho_j(x_j)]$, is minimized, where C_j is the completion time of processing of J_j. Here, $\delta_{C_j > d_j} = 1$ if J_j is a late job, and $\delta_{C_j > d_j} = 0$ if J_j is an on-time job. Note that in reality a job processing time x_j cannot be smaller than some lower limit $\underline{p}_j > 0$, no matter the amount of resources we allocate to the job. In such a case, we define $\rho_j(x_j) = M$ for $x_j < \underline{p}_j$, where M is a large number. The input data consists of (i) the number of jobs n, (ii) the parameters d_j, w_j, and $\bar{p}_j$ for every job $j = 1, \ldots, n$, and (iii) an oracle that computes function ρ_j (for each job j).

Note that the special case in which all job compressions are prohibitively expensive (denoted as $1||\sum w_j U_j$ in the machine scheduling literature) and due dates and late penalties are nonnegative integer numbers is already NP-hard [LLKS93]. Thus, our problem is also NP-hard. [CCLL98] considers a special case of this problem in which due dates, late penalties and processing times are nonnegative integers and ρ_j is a linear function: this special case is converted into a profit maximization problem, and an FPTAS for it is developed. However, the existence of an FPTAS for the profit maximization problem does not imply the existence of an FPTAS for the original cost minimization problem. [HKL$^+$14] presents an FPTAS for the original minimization problem and considers a general increasing integer-valued resource consumption function. For every $\epsilon > 0$, the FPTAS constructs a succinct $(1+\epsilon)$-approximation function of the value function $z_1(I)$, where I is an upper bound on the total processing time of the on-time jobs.

We next show that we can substitute the integrality assumption with the assumption that the resource consumption functions are κ-Lipschitz continuous and get a fully polynomial time (Σ, Π)-approximation scheme. We will later show that this is best possible in the sense that achieving an FPTAS in this setting is not possible. We use the same formulation of our problem as a pseudo-polynomial time DP given in [HKL$^+$14]. For the sake of completeness we give below this formulation. First, we permute the jobs so that $d_1 \geq d_2 \geq \cdots \geq d_n$. Note that there exists an optimal schedule in which all on-time jobs are arranged in

increasing order of due dates and all late jobs are scheduled behind the on-time jobs. Hence, it suffices to consider the job list $J_n, \ldots, J_1$, decide which jobs in this list are designated as late jobs (removing them from the list), and decide the amount of resources are allocated to the on-time jobs (retaining them in the list).

Let $z_t(I_t)$ be the minimum total cost of a partial schedule containing $J_t, \ldots, J_n$, given that the total processing time of the on-time jobs in this partial schedule is no greater than I_t. For notational convenience, we denote $d_{n+1} = 0$. The recurrence relation is

$$z_t(I_t) = \min \left\{ \min_{x_t \in [0, I_t]} \Big\{ z_{t+1}(\min\{I_t - x_t,\, d_{t+1}\}) + \rho_t(x_t) \Big\}, z_{t+1}(\min\{I_t, d_{t+1}\}) + w_t \right\}, \tag{24}$$

for $t = 1, \ldots, n$ and $I_t \in [0, d_t]$. Here, "$z_{t+1}(\min\{I_t - x_t,\, d_{t+1}\}) + \rho_t(x_t)$" is the cost of the partial schedule if J_t is made on-time and is assigned x_t units of processing time, while "$z_{t+1}(\min\{I_t, d_{t+1}\}) + w_t$" is the cost of the partial schedule if J_t is selected to be a late job. The boundary condition is $z_{n+1}(0) = 0$. The optimal solution value is $z_1(d_1)$. It is easy to see that equation (24) can be rewritten as

$$z_t(I_t) = \min_{x_t \in [0, I_t]} \Big\{ z_{t+1}(\min\{I_t - x_t,\, d_{t+1}\}) + \min\{\rho_t(x_t), w_t\} \Big\}. \tag{25}$$

Next, we show that the above DP with recurrence relation (25) is a increasing DP that fits into our framework. For simplicity, we assume that $w_{\max} > 0$, where $w_{\max} = \max\{w_1, \ldots, w_n\}$ (otherwise there is an optimal solution where all jobs are late). Define $T = n$, $g_{T+1} \equiv 0$, and $\mathcal{S}_{T+1} = \{0\}$. For $t = 1, \ldots, T$, we define $\mathcal{S}_t = [0, d_t]$, $\mathcal{A}_t(I_t) = [0, I_t]$, $D_t = d_{t+1}$ with probability 1, $g_t(I_t, x_t, D_t) = \min\{\rho_t(x_t), w_t\}$, and $f_t(I_t, x_t, D_t) = \min\{I_t - x_t,\, D_t\}$. Note that $\mathcal{S}_{T+1}$, $\mathcal{S}_t$, and $\mathcal{A}_t(I_t)$ are all compact intervals on the real line for any $I_t \in \mathcal{S}_t$ and $t = 1, \ldots, T$. Thus, Condition 1 holds. Obviously, Condition 2 also holds. As for Condition 3(ii), we notice that for $t = 1, \ldots, T$, function f_t is increasing in I_t and decreasing in x_t, and function g_t is decreasing in I_t and x_t. Furthermore, $\mathcal{A}_t(I) \subseteq \mathcal{A}_t(I')$ for all $I, I' \in \mathcal{S}_t$ with $I \leq I'$. Therefore, Condition 3(ii) is also satisfied and by Theorem 1.6 we get a (Σ, Π)-FPTAS for the value function $z_1(I)$.

We conclude this section by observing that the time-cost trade-off machine scheduling problem with continuous time does not admit an FPTAS of the value function.

Proposition B.1 *The time-cost trade-off machine scheduling problem with continuous time and costs does not admit a succinct constant-factor approximation.*

Proof. There is $n = 1$ job with due date $d_1 = 0$, late penalty $w_1 = 2$, and normal processing time $\bar{p}_1 = 1$. The resource consumption function is $\rho_1(x) := \varphi_i(1 - x)$, where φ_i belong to the family Φ as in the proof of Proposition 2.6. Because the late penalty is always greater than the resource consumption function, it follows that it is always optimal to consume the resource, i.e., $z_1(I) = \varphi_i(1 - I)$. We conclude the proof by noting that the proof of Proposition 2.6 implies that $\varphi_i(1 - I)$ does not admit a succinct constant-factor approximation. □

C A continuous single-item inventory control problem

In this section we formally define a classical single-item inventory control problem and formulate it as a continuous DP satisfying Conditions 1-3. Our exposition is mostly based upon [HKL$^+$14, Sec. A.5]. In the next section we provide a preliminary computational evaluation of the proposed approximation scheme on randomly generated instances for this problem.

Consider the following single-item stochastic inventory control problem with time-varying demand [HKM+09]: The planning horizon is divided into T time periods. At the beginning of a time period t, the inventory level I_t is observed, and then a replenishment decision is made. Let $x_t \geq 0$ denote the replenishment quantity in period t. We assume that the replenishment lead time is zero; that is, the x_t units will arrive in period t. After that, a demand D_t of the item occurs, where D_t is a nonnegative random variable. The ending inventory level of period t equals $I_{t+1} = I_t + x_t - D_t$. Backlogging is allowed, which implies that the inventory level I_{t+1} can be negative. If $I_{t+1} > 0$, then a holding cost is charged; if $I_{t+1} < 0$, then a backlogging cost is incurred. For ease of discussion, we refer to both components as the holding cost. Thus, our objective is

$$\min_{x_1,\ldots,x_T} E \sum_{t=1}^{T} \Big[c_t(x_t) + h_t(I_t + x_t - D_t) \Big],$$

subject to the system dynamics $I_{t+1} = I_t + x_t - D_t$ for $t = 1, \ldots, T$, where $c_t(x_t)$ is the procurement cost in period t when the order size is x_t, $h_t(y)$ is the holding cost in period t when the inventory level at the end of the period is y (y can be positive, zero, or negative), and the expectation is taken with respect to the joint distribution of the random variables involved. We assume that $I_1 = 0$ (i.e., we start with zero inventory). We assume that the procurement and holding costs are convex for all time periods. The input data for the problem consists of (i) the number of time periods T, and (ii) an oracle that computes functions c_t and h_t and an explicit representation of the demand distribution D_t as order pairs of a value and the probability that the demand realizes as this value (for each period t).

We consider first the discrete version of this problem, where all demand, procurement, and inventory levels are integral and functions c_t and h_t are nonnegative integer-valued, for every $t = 1, \ldots, T$. When the demand is deterministic and the cost functions are linear, the problem is reduced to the classical Wagner-Whitin model, which is solvable in polynomial time [WW58]. However, with general cost functions, the problem becomes computationally intractable (see, e.g., [FLR80]). A number of authors have developed FPTASs for various NP-hard deterministic inventory control problems with time-varying demand [Van01, CKP06]. In a previous work of ours [HKM+09], we show that the single-item stochastic inventory control problem with discrete demand is #P-hard, and we give an FPTAS for it. The FPTAS presented in [HKM+09] is an ad-hoc algorithm tailored to the specific problem studied. In [HKL+14] we showed that this problem can be cast into the FPTAS framework for convex DP developed in that paper.

We now consider a continuous version of this problem where the cost functions are Lipschitz continuous, and the random variables are with finite support contained in an interval on the real line as described in Section 4. The continuous version of the problem is interesting in cases where we are managing, e.g., gasoline, liquid nitrogen, or other liquid goods. Let D^* denote the maximal total demand accumulated over the entire time horizon, i.e., $D^* = \sum_{t=1}^{T} D_t^{\max}$. To formulate the problem as a convex continuous DP, we define $\mathcal{S}_{T+1} = [-D^*, D^*]$ and $g_{T+1} \equiv 0$. For $t = 1, \ldots, T$, we define $\mathcal{S}_t = [-D^*, D^*]$, $\mathcal{A}_t(I_t) = [0, \min\{D^* - I_t,\, D^*\}]$, $g_t(I_t, x_t, D_t) = c_t(x_t) + h_t(I_t + x_t - D_t)$, and $f_t(I_t, x_t, D_t) = I_t + x_t - D_t$. Then, our problem can be solved as a DP as presented in Theorem 1.1. Note that $\mathcal{S}_{T+1}$, $\mathcal{S}_t$, and $\mathcal{A}_t(I_t)$ are all intervals on the real line for any $I_t \in \mathcal{S}_t$ and $t = 1, \ldots, T$. Thus, Condition 1 holds. Obviously, Condition 2 also holds. As for Condition 3(*iii*), we note that $\mathcal{S}_t \otimes \mathcal{A}_t$ is a trapezoid with extreme points $(-D^*, 0);\ (-D^*, D^*);\ (0, D^*);\ (D^*, 0)$ which is a convex set. Let $g_t^I(I, d) \equiv 0$, $g_t^x(x, d) = c_t(x)$, and $g_t^u(\cdot) = h_t(\cdot)$. Then, function g_t can be expressed as $g_t(I, x, d) = g_t^I(I, d) + g_t^x(x, d) + g_t^u(f_t(I, x, d))$. We note that $g_t^I(\cdot, d)$, $g_t^x(\cdot, d)$ and $g_t^u(\cdot)$ are all univariate convex functions and f_t is linear and separable in its variables. Therefore, Condition 3(*iii*) is also satisfied.

It is easy to see that the three variants of the single-item stochastic inventory control problem with discrete demand stated in [HKM+09], namely the capacitated version, the discounted version, and the version which allows disposal at a cost, can be formulated as convex DPs in a similar fashion.

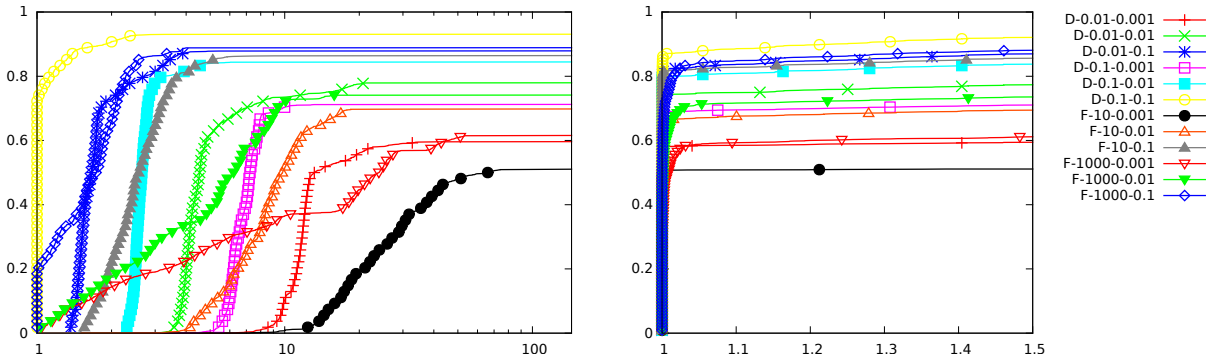
D Computational experiments

We provide a computational evaluation of the proposed approximation scheme, as compared to three other algorithms: an EXACT DP solution to the discretized problem, the discrete FPTAS (called DFPTAS from now on) of [HNO15] applied to the discretized problem, and to our implementation of the CAVE algorithm of [NP13]. Recall by Theorem 1.5 that discretization could in theory induce a large relative or absolute error, though in practice this is typically not the case. We remark that the convergence proof of [NP13] relies on knowing the minimum distance between breakpoints of the value function; in our case, this distance is unknown, and we resort to applying CAVE approximately, using a small value for the minimum distance between breakpoints. This forsakes convergence in theory, but it is a commonly used approach in practice. Despite being a value iteration algorithm, CAVE converges to the optimal policy, but not necessarily to the optimal objective value: it computes the value functions up to a constant, and therefore it does not provide an approximation of the objective value at termination. We use the number of iterations without improvement as a stopping criterion; notice that if this number is set too small, it is possible that sample paths with low probability are never explored and the algorithm exits with a relatively low quality policy. Finally, we remark that CAVE is more general than our algorithm as it can be applied to problems with large-dimensional action spaces, as long as some linearity conditions are satisfied. CAVE converges asymptotically but does not provide an approximation guarantee in finite time. We implemented CAVE as described in [NP13], without additional fine-tuning.

Our comparison takes into account the total CPU time and the value of the policies computed by the algorithms, i.e. the cost of the sequence of actions implicitly defined by the value functions z_t for $t = 1, \dots, T$ via the $\arg\min$ operator, rather than the approximation of the objective function value $z^*(I_1)$ of equation (2). This is because CAVE tries to approximate the value functions up to a constant (i.e. it tries to approximate their slopes only, not their values), therefore it does not provide any approximation of $z^*(I_1)$, but it yields a policy. Note that to compute the value of a policy, we would have to explore all sample paths and determine a sequence of actions for each of them. If this is computationally infeasible, i.e. there are more than 10000 possible realizations of the vector of random variables $(D_1, \dots, D_T)$, we estimate the value of a policy through Monte Carlo simulation over 10000 sample paths (we use the same samples for all algorithms, to reduce variance).

All algorithms are implemented in Python 3 using SciPy when appropriate, and tested on a cluster of identical machines equipped with Intel Xeon E5410 clocked at 2.33Ghz and 32GB of RAM. The time limit is set to 1000 seconds per instance, and if an algorithm does not converge before the time limit, that particular run is considered a failure; however, since CAVE returns a policy even when it does not converge, we consider the corresponding policy when comparing policy values.

We test the algorithms on randomly generated instances of a continuous single-item inventory control problem, as described in Section C. The instances are generated similarly to [HNO15], but the inventory is assumed to take continuous values and the demand is allowed to be fractional. An instance of the stochastic single-item inventory control problem is defined by the number of time periods T, the maximum demand in each period M, and the type of cost functions. At each time period t, the set $\mathcal{D}_t$ of the N demand values is generated as follows: first we draw the maximum demand w_t at stage t uniformly at random in $[\lfloor M/2 \rfloor, M]$, then we draw the remaining $N-1$ values in $[1, w_t]$. This method ensures that none of the instances we generate is too easy and that the difficulty scales with M. The probabilities with which demands occur are randomly assigned by generating N integers $q_i, i = 1, \dots, N$ in $[1, \dots, 10]$, and computing the probability of the k-th value of the demands as $q_k / (\sum_{i=1}^{N} q_i)$. We must also define the cost functions, namely: procurement, storage and backlogging costs. For each of these functions, we select a coefficient c uniformly at random in $[1, \dots, 20]$ for unit procurement cost, in $[1, \dots, 10]$ for storage costs, in $[10, \dots, 50]$ for backlogging costs. Then we consider three different types of cost functions: linear ($f(x) = cx$), quadratic ($f(x) = cx^2$) and cubic ($f(x) = cx^3/1000$).



(a) CPU time.

(b) Policy value.

Figure 2: Performance profile of the running time (left) and policy value (right) for different algorithms. The x-axis of the left graph is a on a log scale. The algorithms are labeled according to the scheme D-δ-ϵ, F-Σ-ϵ, where δ is the discretization step, ϵ is the relative error ($\Pi = 1 + \epsilon$), and Σ is the absolute error.

In our experiments, the number of stages is $T \in \{5, 10, 20\}$, the maximum demand is $M \in \{100, 1000, 10000\}$ (the size of the state and action spaces is proportional to M), the number of possible demand values is $N \in \{10, 100, 1000\}$, and the cost functions are convex mononomials of degree $d \in \{1, 2, 3\}$ as described above. We generate 20 instances for each combination of the parameters T, M, N, d, for a total of 1620 instances.

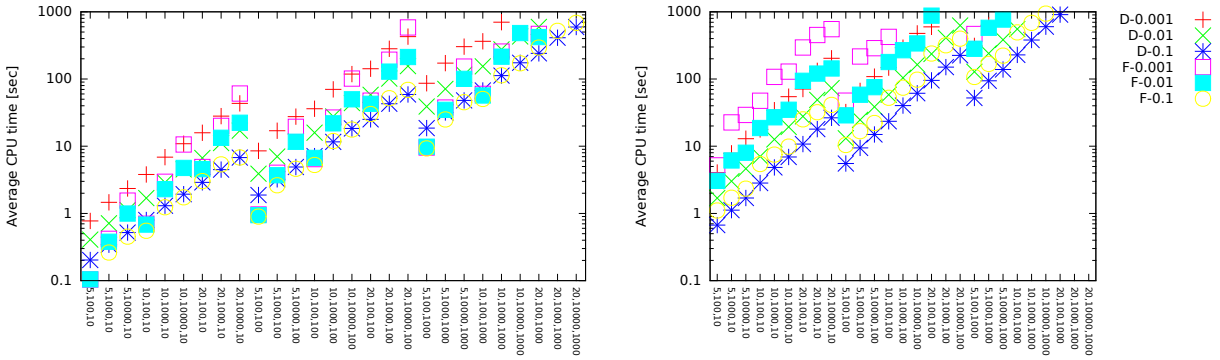
D.1 Discrete versus continuous

We first compare the performance of DFPTAS (label "D") as described in [HNO15] to the (Σ, Π)-FPTAS (label "F") proposed in this paper. We test several values for Σ, Π and the discretization step δ.

We compare the algorithms in terms of CPU time and quality of the computed policies using performance profiles [DM02], see Fig. 2. A performance profile reports the fraction (y axis) of instances for which an algorithm is at most k times (x axis) worse than the best performing algorithm on that instance. Fig. 2 indicates that the fastest algorithm is generally "D-0.1-0.1", i.e. DFPTAS with discretization step $\delta = 0.1$ and multiplicative error $\Pi = 1.1$' this particular parametrization also achieves excellent performance in terms of policy values, being very close to optimal on the vast majority on the instances. We remark that DFPTAS does not give guarantee the relative error ϵ because it is applied on the discretized problem, see Thm. 1.5. Comparable performance is obtained by "F-1000.0-0.1", which is up to a factor 5 slower in some cases, but solves almost the same number of instances yielding near optimal policies. The quality of the policies appears to be hardly affected by the approximation factor: while smaller approximation factors sometimes achieve better policies, the difference is so small that it barely shows in the graphs. Fig. 2 shows that the speed of the FPTAS is affected by both Σ and Π, but the relative approximation factor Π has much larger impact: the performance of algorithm parametrizations that differ only in Σ is quite close, whereas changing Π has a dramatic effect on the computing time.

From our discussion, we conclude that the best parametrizations of each algorithm are "F-1000" for the (Σ, Π)-FPTAS, whereas the best version of DFPTAS is "D-0.1"; we will only report these in the following.

We take a closer look at how the algorithms perform on different instance classes. To this end, in Fig. 3 we give a plot of the CPU times for different classes of instances, grouped by the degree of the cost function. We do not report the graph for $d = 3$ because it is similar to $d = 2$. Fig. 3 shows that "F-0.1" is faster than any other algorithm on instances with linear costs, whereas "D-0.1" is the fastest algorithm on instances with quadratic or cubic costs. This can be explained in light of the fact that when the nonlinearities become more pronounced, building a precise (Σ, Π)-approximation of the continuous problem may require more points



(a) CPU time for instances with $d = 1$.

(b) CPU time for instances with $d = 2$.

Figure 3: Figures (a)-(c) report the running time of different algorithms for instances with $d \in \{1, 2\}$; the x-axis indicates the number of time periods T, the size of the state space M, and the size of the support of the random variables N. Each point on the graph represents the average solution time for solved instances over a group of 20 instances; if more than 10 instances out of 20 hit the time limit, it is considered a failure and the corresponding point is not reported. The algorithms are labeled according to the scheme D-ϵ, F-ϵ, where ϵ is the relative error ($\Pi = 1 + \epsilon$). In these graphs, the discretization step δ for "D" is fixed at 0.1, and the absolute error Σ for "F" is fixed at 1000.

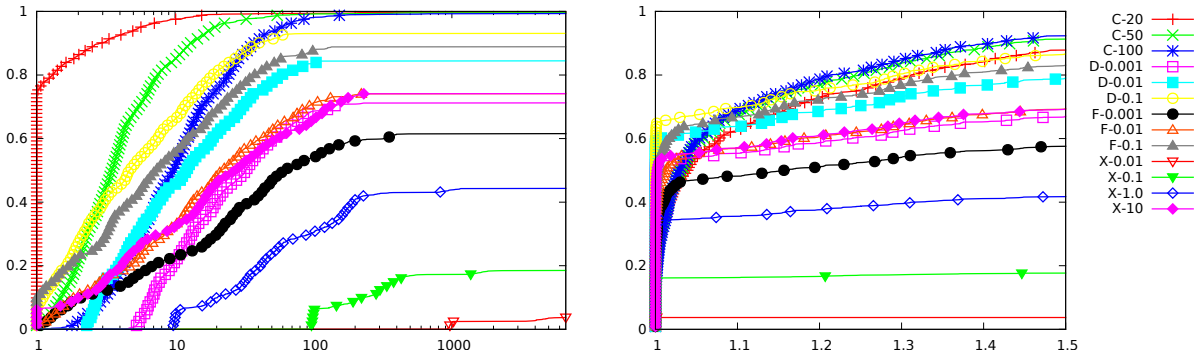
in the approximation set; at the same time, DFTPAS works on a discretized problem and necessarily builds a coarse approximation, so it is not as negatively affected by the nonlinearities. On small linear instances ($T = 5, d = 1$) even "F-0.001" is faster than "D-0.1", but the situation changes as the instance size grows. The running times for both "F" and "D" scale essentially linearly with the size of the support of the random variables N, consistently with the observations in [HNO15].

D.2 Comparison with other algorithms

We now evaluate the performance of the implicit FPTAS and DFPTAS as compared to EXACT (label "X") and CAVE (label "C"). As in the previous section, in Fig. 4 we report performance profiles of the computation time and policy value, and in Fig. 5 we give a plot of the CPU times for different classes of instances, grouped by the size of the support of the random variables.

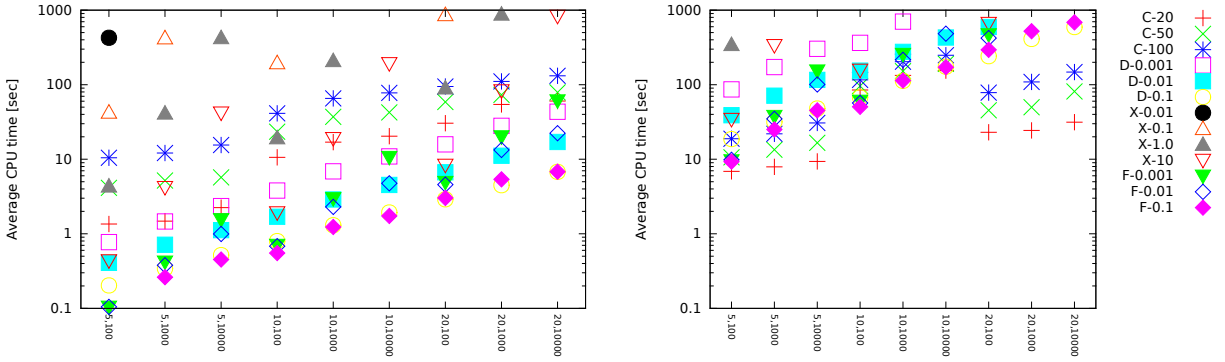
Fig. 4 indicates that "C-20" is the fastest algorithm, by a large margin, followed by "C-50". However the quality of the policies computed by these two algorithms is visibly inferior to the other algorithms: the policy returned by CAVE has cost comparable to EXACT only on less than 20% of the instances. As noted in the previous section, "D-0.1" is slightly faster than "F-0.1" on average and these two algorithms find policies of comparable quality, with a small advantage for "D-0.1". The EXACT algorithm applied to the discretized problem is clearly too slow on these problems: even with a very coarse discretization step of $\delta = 10$, only 75% of the instances can be solved within the time limit, whereas the three parametrizations of CAVE solve all of them.

Fig. 5 explains the advantage of CAVE in terms of speed: Fig. 5 (a) shows that for problem instances with small support of the random variables, the (Σ, Π)-FPTAS "F-0.1" is actually the fastest algorithm and we know from Fig. 4 that it finds optimal or nearly optimal policies in essentially all cases. However, when the size of the support of the random variables becomes large ($N = 1000$), CAVE has a tremendous speed advantage, because it does not compute expectations explicitly looping over the support of the random variable, but relies on samples instead to approximate the computations. We remark once again that CAVE does not offer an approximation guarantee in finite time. "D-0.1" shows comparable performance to "F-0.1" irrespective of N.



(a) CPU time. (b) Policy value.

Figure 4: Performance profile of the running time (left) and policy value (right) for different algorithms. The x-axis of the left graph is a on a log scale. The algorithms are labeled according to the scheme C-η, D-ϵ, X-δ, I-ϵ, where η is the number of iterations without improvement before the algorithm is stopped, δ is the discretization step, and ϵ is the relative error $(\Pi = 1 + \epsilon)$. The discretization step for "D" is fixed at 0.1, and the value of Σ for "F" is fixed at 1000.



(a) CPU time for instances with $N = 10$. (b) CPU time for instances with $N = 1000$.

Figure 5: Figures (a)-(b) report the running time of different algorithms for instances with linear costs; the x-axis indicats the number of time periods T and the size of the state space M. Each point on the graph represents the average solution time for solved instances over a group of 20 instances; if more than 10 instances out of 20 hit the time limit, it is considered a failure and the corresponding point is not reported.