2014-01-01

Fixed-Dimensional Stochastic Dynamic Programs: An Approximation Scheme and an Inventory Application

UTD AUTHOR(S): Wei Chen, Milind Dawande and Ganesh Janakiraman

Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Fixed-Dimensional Stochastic Dynamic Programs: An Approximation Scheme and an Inventory Application

Wei Chen, Milind Dawande, Ganesh Janakiraman

To cite this article:
Wei Chen, Milind Dawande, Ganesh Janakiraman (2014) Fixed-Dimensional Stochastic Dynamic Programs: An Approximation Scheme and an Inventory Application. Operations Research 62(1):81-103. http://dx.doi.org/10.1287/opre.2013.1239

OPERATIONS RESEARCH
Vol. 62, No. 1, January–February 2014, pp. 81–103
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.2013.1239

METHODS

Fixed-Dimensional Stochastic Dynamic Programs: An Approximation Scheme and an Inventory Application

Wei Chen, Milind Dawande, Ganesh Janakiraman
Naveen Jindal School of Management, The University of Texas at Dallas,
Richardson, Texas 75080 {wei.chen@utdallas.edu, milind@utdallas.edu, ganesh@utdallas.edu}

We study fixed-dimensional stochastic dynamic programs in a discrete setting over a finite horizon. Under the primary assumption that the cost-to-go functions are discrete $L^\natural$-convex, we propose a pseudo-polynomial time approximation scheme that solves this problem to within an arbitrary prespecified additive error of $\varepsilon > 0$. The proposed approximation algorithm is a generalization of the explicit-enumeration algorithm and offers us full control in the trade-off between accuracy and running time.

The main technique we develop for obtaining our scheme is approximation of a fixed-dimensional $L^\natural$-convex function on a bounded rectangular set, using only a selected number of points in its domain. Furthermore, we prove that the approximation function preserves $L^\natural$-convexity. Finally, to apply the approximate functions in a dynamic program, we bound the error propagation of the approximation. Our approximation scheme is illustrated on a well-known problem in inventory theory, the single-product problem with lost sales and lead times. We demonstrate the practical value of our scheme by implementing our approximation scheme and the explicit-enumeration algorithm on instances of this inventory problem.

Subject classifications: discrete convexity; multidimensional stochastic dynamic programs; approximation algorithms.
Area of review: Optimization.
History: Received December 2011; revisions received September 2012, March 2013, October 2013; accepted October 2013.

1. Introduction

Consider the following finite-horizon dynamic programming (DP) problem. For period $t = 1, 2, \ldots, T$, we have

$$f_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} [G_t(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{w}_t} f_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t))], \quad \text{where } \mathbf{x}_t \in \mathbb{Z}^{k_1}, \mathbf{a}_t \in \mathbb{Z}^{k_2}, \mathbf{w}_t \in \mathbb{Z}^{k_3}. \quad (1)$$

For period t, f_t is the cost-to-go function starting from that period until the end of the horizon, G_t is the single-period cost function, $\mathbf{x}_t$ is the state vector, $\mathbf{a}_t$ is the action vector, and $\mathbf{w}_t$ is an exogenous random vector. We assume that for all t, $\mathbf{x}_t$ and $\mathbf{a}_t$ belong to bounded rectangular sets, $\mathcal{X}_t \subseteq \mathbb{Z}^{k_1}$ and $\mathcal{A}_t(\mathbf{x}_t) \subseteq \mathbb{Z}^{k_2}$, respectively. We let $\mathbf{w}_t \in \mathbb{Z}^{k_3}$, $t = 1, 2, \ldots, T$, be a sequence of independent random vectors. Furthermore, the possible realizations of $\mathbf{w}_t$ and the corresponding probabilities are specified explicitly for each period t. Throughout the paper, we hold k_1, k_2, and k_3 fixed. The transformation function $\mathbb{T}$ transforms $(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)$ to $\mathbf{x}_{t+1}$; i.e., $\mathbf{x}_{t+1} = \mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)$. The objective is to find the minimum expected cost and an optimal policy for the entire horizon, starting with any given initial state vector $\mathbf{x}_1$.

Consider the following backward explicit-enumeration approach to solve the above DP. We start by setting $f_{T+1}(\mathbf{x}_{T+1}) = 0$ for all $\mathbf{x}_{T+1}$. For each $\mathbf{x}_T$, $f_T(\mathbf{x}_T)$ is computed by solving the minimization problem in (1) via straightforward enumeration. By creating a table that stores the values of f_T for every possible value of $\mathbf{x}_T$, we can similarly compute the value of $f_{T-1}(\mathbf{x}_{T-1})$ for every possible value of $\mathbf{x}_{T-1}$. Repeating this enumeration through time, we get $f_1(\mathbf{x}_1)$. Assuming $\mathcal{X}_t = \mathcal{S}$ and $\mathcal{A}_t(\mathbf{x}_t) = \mathcal{A}$ for all $t = 1, 2, \ldots, T$, the running time of this exact algorithm is $O(T|\mathcal{S}||\mathcal{A}|)$. Note that $|\mathcal{S}|$ and $|\mathcal{A}|$ are usually pseudo-polynomial in the binary size of the input.[1]

Halman et al. (2008, 2009) study a special case of this problem when $\mathcal{S}$, $\mathcal{A}$, and w_t are one-dimensional (i.e., $k_1 = k_2 = k_3 = 1$) and the functions f_t and G_t are either convex[2] or monotone. They show that this special case is itself NP-hard. This motivates their consideration of approximation algorithms. They derive an approximation algorithm that can achieve a solution to within an arbitrary specified relative error $\varepsilon > 0$ and runs in polynomial time in the binary size of the input and $1/\varepsilon$; such an approximation is commonly referred to as a Fully Polynomial-Time Approximation Scheme (FPTAS).

The basic idea of the approach in Halman et al. (2008, 2009) is to reduce the amount of information needed by the explicit-enumeration approach and, instead of the exact solution, obtain one within a relative error guarantee of ε.

By doing this, they reduce the running time of the algorithm to a polynomial in the binary size of the input and $1/\varepsilon$. More specifically, in their problem setting, the functions to be minimized in each period, i.e., the functions on the right-hand side of (1), are one-dimensional and convex. This enables them to solve the minimization step using binary search instead of enumeration, yielding a time factor of $O(\log|\mathcal{A}|)$ rather than $O(|\mathcal{A}|)$. Although the values of f_t in period t ($t=1,2,\ldots,T$) are still represented using a table, the number of entries in the table is reduced from $|\mathcal{S}|$ to $O((1/\varepsilon)\log f^{\max})$, where $f^{\max}$ is the maximum of f_t over all t and the corresponding domains; i.e., $f^{\max}=\max_t \max_{\mathbf{x}_t\in\mathcal{X}_t} f_t(\mathbf{x}_t)$. This is achieved by storing the values of f_t for only a selected number of points in the domain such that the function values at these points alone can be used to construct a *relative* error approximation to the function. The idea used is that of approximating a one-dimensional convex function (f_t) using a piecewise-linear function. Their approximation coincides with f_t on a subset of $\mathcal{X}_t$. They refer to this subset as a *K-approximation set*[3] for f_t. The use of binary search for minimization and that of the K-approximation set of f_t results in an approximation algorithm that offers a relative error guarantee of ε and runs in polynomial time in the binary input size and $1/\varepsilon$ (since $O((1/\varepsilon)\log f^{\max})$ is polynomial in the binary input size and $1/\varepsilon$). The assumption that f_t is one-dimensional and convex is crucial for the success of this approach.

In this paper, we attempt a generalization of the results of Halman et al. (2008, 2011, 2009, 2013) for fixed dimensions k_1, k_2, and k_3. By assuming $L^\natural$-convexity of the cost-to-go functions, we obtain *additive* error approximations of these functions by storing their values on subsets of their domains. Furthermore, we prove that these approximations are also $L^\natural$-convex functions. Finally, we bound the error propagation of the approximation to establish the required guarantee.

For a detailed comparison of our algorithm with the explicit enumeration procedure, we refer the readers to §6.4. Our approximation algorithm can be viewed as a flexible algorithmic framework. On the one hand, as the desired additive-error guarantee ε reduces, the algorithm approaches the explicit-enumeration procedure; on the other hand, as ε increases, the running time decreases fast. To illustrate the usefulness of our algorithm, we consider an inventory application: the single-product stochastic inventory control problem with lost sales and lead times; see, e.g., Morton (1969) and Zipkin (2008b). We show that the DP for this problem satisfies the assumptions of our analysis. Consequently, we obtain a pseudo-polynomial additive-approximation scheme for this problem. We demonstrate the practical value of our scheme by implementing our approximation scheme and the explicit-enumeration algorithm on instances of this inventory problem.

The paper is organized as follows: Section 2 reviews the related literature. In §3, we present the definition of a discrete $L^\natural$-convex function and related results. In §4, we describe our approximation of a discrete $L^\natural$-convex function and prove that this approximation is also $L^\natural$-convex. In §5, we present our notation and assumptions, discuss the explicit-enumeration procedure for the DP, and establish the nonexistence of a polynomial-time additive-error approximation algorithm, unless P = NP. In §6, we present our approximation scheme, prove its correctness, analyze its running time, and discuss its properties. Section 7 demonstrates the applicability of our approximation scheme to the above-mentioned inventory problem. Section 8 reports our computational results. We conclude in §9 by discussing an important open question.

2. Literature Review, Our Challenges, and Contributions

We review three fields of literature that are most related to our problem: approximation algorithms for dynamic programs, discrete convexity, and approximation algorithms for stochastic inventory control problems. We end this section with comments regarding some unique challenges we face in approximating fixed-dimensional DPs.

2.1. Approximation Schemes for Dynamic Programs

This is a relatively new direction of research. For deterministic single-dimensional dynamic programs, Woeginger (2000) investigates the conditions under which an FPTAS exists. Halman et al. (2008, 2011) extend this investigation to stochastic single-dimensional dynamic programs. When the possible realizations of the random variables are explicitly specified and are independent in all periods, they propose sufficient conditions to guarantee the existence of an FPTAS. Halman et al. (2013) propose a computationally efficient FPTAS for convex stochastic dynamic programs using techniques in Halman et al. (2008, 2011). The motivation of our work is to obtain an effective (not necessarily polynomial-time) approximation scheme that offers an arbitrarily small *additive* guarantee.

2.2. Discrete Convexity

In the DP (1), we are required to solve a minimization problem in every period. The domain of the objective function in that problem is multidimensional and discrete (more precisely, a subset of $\mathbb{Z}^{k_2}$). Since we seek an efficient solution procedure for the DP, we are interested in identifying a notion of convexity for functions with discrete domains such that the following conditions hold:

1. Positive scaling, addition, translation, uniform variable scaling, and minimization preserve discrete convexity.
2. Efficient minimization algorithms exist.
3. Convex extension over a continuous domain can be efficiently obtained.

Several notions of discrete convexity have been proposed and studied in the literature—for example, Miller's discrete convexity (Miller 1971), integer convexity (Favati and

Tardella 1990), and $L^\natural$-convexity (Murota 2003). For our purpose, we find that the notion of $L^\natural$-convexity satisfies conditions 1, 2, and 3.

For a general introduction to discrete convexity, we refer the reader to Murota (2003, 2007).

2.3. Approximation Algorithms for Stochastic Inventory Problems

As mentioned earlier, we will illustrate our approximation scheme on a well-known inventory problem. The important results for this problem are reviewed in §7.

We proceed to review the results on approximation algorithms for stochastic inventory control problems. The first set of papers that propose such algorithms includes Levi et al. (2007, 2008a, and 2008c). All three papers study single location, single product problems and together encompass capacitated and uncapacitated models with correlated demand processes, and both the possibilities of backordering of excess demand and lost sales. For each of these problems, a 2-approximation algorithm is presented. Given the generality of the models and that demands and inventories are not restricted to be integral, a rigorous complexity analysis of the approximation algorithms is not provided in these papers. However, these algorithms have the attractive computational feature that they are nonrecursive, as opposed to the recursive enumeration approach for solving dynamic programs.

Another stream of work on approximation algorithms for stochastic inventory control problems is that of Halman et al. (2009, 2008, 2011). For a single-product version of the problem with discrete demands, Halman et al. (2009) investigate the possibility of approximation algorithms with arbitrarily close errors. They show that the model with convex cost functions and backlogging but no setup cost is weakly NP-hard and develop an FPTAS for this problem. This result is generalized in Halman et al. (2008, 2011) for single-dimensional stochastic dynamic programs. Our work is an attempt to generalize this stream of work to a fixed number of dimensions. Recall that we have already discussed the main elements of the analysis in Halman et al. (2008, 2009) in §1.

2.4. Challenges in Approximating Fixed-Dimensional DPs and Our Contributions

We now briefly discuss some of the difficulties in approximating fixed-dimensional dynamic programs.

The multidimensionality of the domain of $f_{t]}$ in (1) generates additional requirements (relative to those for a single dimension) on our approximation scheme. First, the notion of discrete convexity needs to be chosen carefully. Second, to approximate a fixed-dimensional function $f_{t]}$ to within an additive error, the K-approximation technique in one dimension needs to be adapted appropriately. Third, to prove that our approximation preserves $L^\natural$-convexity (our choice of the notion of discrete convexity), we need to exploit several properties of $L^\natural$-convex functions and impose additional requirements on our adaptation of the concept of K-approximation sets. Finally, we need some additional developments and assumptions to make our approximation work through time, which is considerably easier for the case of a one-dimensional convex DP.

The main contributions of our work are (i) approximating a fixed-dimensional discrete $L^\natural$-convex function on a bounded rectangular set using only a selected number of points in its domain and (ii) proving that this approximation is also an $L^\natural$-convex function. Furthermore, (iii) we apply this approximation to the DP (1) and develop a pseudo-polynomial time approximation scheme for f_1 with an additive-error guarantee under the primary assumption that all the cost-to-go functions $f_{t]}$ and $G_{t]}$ are $L^\natural$-convex. Our approximation scheme is a generalization of the enumeration algorithm and offers us full control in the trade-off between accuracy and running time; for a more detailed discussion, see §6.4.

Specifically, in our approximation scheme, starting from the last period $T]$, in (1) we replace $f_{T]}$ by its approximation $\check{f}_{T]}$. For period $T]-1$, this results in an approximation $\tilde{f}_{T]-1}$ of $f_{T]-1}$ on the left-hand side of (1). We then prove that $\tilde{f}_{T]-1}$ is also an $L^\natural$-convex function. This enables us to construct the approximation $\check{f}_{T]-1}$ of $\tilde{f}_{T]-1}$ and replace $f_{T]-1}$ by $\check{f}_{T]-1}$ in (1) for period $T]-2$. Repeating this procedure until the first period, we can obtain an approximation of f_1. By bounding the error of the approximation in each period, we achieve the desired additive-error guarantee. The usefulness of our approximation scheme is illustrated by applying it to a well-known stochastic inventory control problem, the single-product problem with lost sales and lead times.

To end this section, it is important to clarify that our approach is fundamentally different from that of approximate dynamic programming; see, e.g., Powell (2007). The aim of that approach is to develop practical solutions for high-dimensional dynamic programs using statistical methods for approximating the cost-to-go functions.

3. Discrete $L^\natural$-Convexity and Related Results

This section is organized as follows. In §3.1, we present three different ways of obtaining continuous extensions[4] of a discrete function—(i) local extension, (ii) global extension, and (iii) convex envelope—and discuss the relationships between them. We show that the convex envelope of a discrete function is the same as its global extension (Lemma 1). In §3.2, we present two different notions of discrete convexity: integer convexity and $L^\natural$-convexity. Several characterizations of integer convexity and $L^\natural$-convexity are also presented. We also present two related definitions: $L^\natural$-convex sets and polyhedral $L^\natural$-convex functions in this section. Finally, in §3.3, we present some properties of discrete $L^\natural$-convex functions that we will use later in our analysis. In particular, we introduce the concept of the Lovász Extension and illustrate its use to obtain a continuous extension

of a discrete $L^\natural$-convex function. A result regarding polyhedral $L^\natural$-convex functions (Lemma 11) is also established in this section.

Throughout the paper, we use $[L,U]_{\mathbb{Z}}$ to denote the set of integers between (and including) two given integers L and U; $-\infty < L \le U < \infty$; i.e., $[L,U]_{\mathbb{Z}} = [L,U] \cap \mathbb{Z}$. Similarly, for a positive integer n, let $[L,U]_{\mathbb{Z}}^n = [L,U]^n \cap \mathbb{Z}^n$. We use $\mathbf{1}$ to denote the vector $(1,1,\ldots,1)$, where its dimensionality should be clear from the context. Also, we use *increasing* to mean nondecreasing and *decreasing* to mean nonincreasing.

3.1. Continuous Extensions of a Discrete Function

We start this section by defining the convex hull of a set of points in $\mathbb{R}^n$.

Definition 1. Given a set of m points $H_m = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m\}$ in $\mathbb{R}^n$, the convex hull of H_m, denoted by $\mathrm{Conv}(H_m)$, is defined as follows:

$$\mathrm{Conv}(H_m) = \left\{ \mathbf{p} \in \mathbb{R}^n \text{ such that } \mathbf{p} = \sum_{i=1}^{m} \lambda_i \mathbf{p}_i, \lambda_i \ge 0 \text{ and } \sum_{i=1}^{m} \lambda_i = 1 \right\}.$$

It is easy to see that $\mathrm{Conv}([L,U]_{\mathbb{Z}}^n) = [L,U]^n$. Next, we formally define a piecewise-linear function in n-dimensional real space.

Definition 2 (Piecewise-Linear Functions). Let $D \subseteq \mathbb{R}^n$ be an n-dimensional polyhedron. A function $f(\mathbf{x}): D \to \mathbb{R}$ is referred to as *piecewise-linear* if there exists a set of n-dimensional polyhedra $\mathcal{P} = \{P_1, P_2, \ldots, P_i, \ldots\}$, indexed by $i \in I$, such that

1. $\bigcup_{i \in I} P_i = D$.
2. For all $i, j \in I$, $i \neq j$, the dimension of $P_i \cap P_j$ is strictly less than n.
3. The function f is linear in $\mathbf{x}$ on $P_i \in \mathcal{P}$ for all $i \in I$.

Given a discrete function, there are different ways to obtain a continuous extension. One natural way to do so is via *local* extension: on each unit hypercube in its domain, extend the discrete function to the convex hull of the function graph and finally paste together all these extensions to obtain a continuous function. The next definition, due to Favati and Tardella (1990), formalizes this idea. Given a point $\mathbf{x} \in [L,U]^n$, let

$$N(\mathbf{x}) = \{\mathbf{y} \in [L,U]_{\mathbb{Z}}^n : \|\mathbf{x} - \mathbf{y}\|_\infty < 1\},$$

where $\|\mathbf{x}\|_\infty$ is the L_∞-norm. We refer to $N(\mathbf{x})$ as the discrete neighborhood of $\mathbf{x}$.

Definition 3 (Local Extension of a Discrete Function). For $n \in \mathbb{Z}^+$, given a discrete function $f: [L,U]_{\mathbb{Z}}^n \to \mathbb{R}^+$, its local extension $f^-: [L,U]^n \to \mathbb{R}^+$ is defined as follows:

$$f^-(\mathbf{x}) = \min\left\{ \sum_{i=1}^{|N(\mathbf{x})|} \alpha_i f(\mathbf{z}_i) : \mathbf{z}_i \in N(\mathbf{x}), \sum_{i=1}^{|N(\mathbf{x})|} \alpha_i \mathbf{z}_i = \mathbf{x}, \sum_{i=1}^{|N(\mathbf{x})|} \alpha_i = 1, \alpha_i \ge 0 \right\}.$$

Observe that for an arbitrary discrete function, its local extension is convex in each unit hypercube and piecewise-linear on the entire domain.

Another natural way to obtain a continuous extension of a discrete function on a bounded domain is the following: instead of the *local* extension, we use the *global* extension, that is, the convex hull of the function graph on the entire domain. The following definition formalizes this idea:

Definition 4 (Global Extension of a Discrete Function). For $n \in \mathbb{Z}^+$, given a discrete function $f: [L,U]_{\mathbb{Z}}^n \to \mathbb{R}^+$, its global extension $f^G: [L,U]^n \to \mathbb{R}^+$ is defined as follows:

$$f^G(\mathbf{x}) = \min\left\{ \sum_{i=1}^{k} \alpha_i f(\mathbf{z}_i) : \mathbf{z}_i \in [L,U]_{\mathbb{Z}}^n, \sum_{i=1}^{k} \alpha_i \mathbf{z}_i = \mathbf{x}, \sum_{i=1}^{k} \alpha_i = 1, \alpha_i \ge 0 \right\},$$

where $k = (U-L+1)^n$ is the number of points in the set $[L,U]_{\mathbb{Z}}^n$.

Clearly, the global extension of any discrete function is convex.

A third way to extend a discrete function is via its convex envelope. The following definition is adapted from Falk and Hoffman (1976).

Definition 5 (Convex Envelope of a Discrete Function). The convex envelope of function $f: D \subseteq \mathbb{Z}^n \to \mathbb{R}$ is a continuous function $\bar{f}: \mathrm{Conv}(D) \to \mathbb{R}$ such that

1. $\bar{f}(\mathbf{x})$ is a convex function.
2. For all $\mathbf{x} \in D$, $\bar{f}(\mathbf{x}) \le f(\mathbf{x})$.
3. For any other convex function g defined over $\mathrm{Conv}(D)$, if $g(\mathbf{x}) \le f(\mathbf{x}), \forall\, \mathbf{x} \in D$, then $g(\mathbf{x}) \le \bar{f}(\mathbf{x}), \forall\, \mathbf{x} \in \mathrm{Conv}(D)$.

For a function $f: D \subseteq \mathbb{R}^n \to \mathbb{R}$ and a set $P \subseteq D$, let the *restriction* of f to P be defined as $f|_P: P \to \mathbb{R}$, where $f|_P(\mathbf{x}) = f(\mathbf{x}), \forall\, \mathbf{x} \in P$. As observed in Favati and Tardella (1990), for a discrete function $f: [L,U]_{\mathbb{Z}}^n \to \mathbb{R}$, we have the identity $\overline{f|_{N(\mathbf{x})}}(\mathbf{x}) = f^-(\mathbf{x}), \forall\, \mathbf{x} \in [L,U]^n$.

The next lemma characterizes the relationship between the global extension of a discrete function and its convex envelope on its domain. This lemma is a generalization of Theorem 3 in Falk and Hoffman (1976). The proof is simple and, therefore, omitted.

Lemma 1. *For any discrete function $f: [L,U]_{\mathbb{Z}}^n \to \mathbb{R}^+$, its global extension coincides with its convex envelope defined on its domain. That is, for all $\mathbf{x} \in [L,U]^n$, $f^G(\mathbf{x}) = \bar{f}(\mathbf{x})$.*

3.2. Definitions of Discrete Convexity

We review two notions of discrete convexity that are used in our analysis. We first define submodularity of a discrete function.

Definition 6 (Submodularity). For $n \in \mathbb{Z}^+$, a function $f : L, U \subseteq \mathbb{Z}^n \to \mathbb{R}^+$ is submodular if it satisfies the following condition:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}),$$

for all $\mathbf{x}, \mathbf{y} \in L, U \subseteq \mathbb{Z}^n$, where the vector $\mathbf{x} \vee \mathbf{y}$ (respectively, $\mathbf{x} \wedge \mathbf{y}$) denotes the component-wise maximum (respectively, minimum) of $\mathbf{x}$ and $\mathbf{y}$.

The first notion of discrete convexity we introduce is integer convexity.

Definition 7 (Integer Convexity, Favati and Tardella 1990). For $n \in \mathbb{Z}^+$, a function $f : L, U \subseteq \mathbb{Z}^n \to \mathbb{R}^+$ is called integrally convex if its local extension $\bar{f}(\mathbf{x}) : [L, U]^n \to \mathbb{R}^+$ is convex.

The following characterizations of integer convexity are due to Favati and Tardella (1990).

Lemma 2. *The following conditions are equivalent*:

1. *Function $f : L, U \subseteq \mathbb{Z}^n \to \mathbb{R}^+$ is integrally convex.*
2. *The local extension of f coincides with its global extension*; *i.e.*,

$$\bar{f}(\mathbf{x}) = f^c(\mathbf{x}), \quad \forall \mathbf{x} \in [L, U]^n.$$

In general, addition does not preserve integer convexity; see Favati and Tardella (1990). However, a stronger notion of discrete convexity, namely $L^\natural$-convexity, is preserved under addition.

Definition 8 ($L^\natural$-Convexity, Murota 2003). For $n \in \mathbb{Z}^+$, a function $f : L, U \subseteq \mathbb{Z}^n \to \mathbb{R}^+$ is called $L^\natural$-convex if the function $\psi(\mathbf{x}, \lambda) = f(\mathbf{x} - \lambda\mathbf{1}), \lambda \in \mathbb{Z}, \lambda \geq 0$, is submodular in $(\mathbf{x}, \lambda)$.

The next lemma presents several characterizations of $L^\natural$-convexity. The second and the third characterization in the following lemma are due to Murota (2003, 2007). The fourth characterization is due to Fujishige and Murota (2000).

Lemma 3. *The following conditions are equivalent*:

1. *Function $f : L, U \subseteq \mathbb{Z}^n \to \mathbb{R}^+$ is $L^\natural$-convex.*
2. *Function f satisfies translation submodularity, defined as follows*:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f((\mathbf{x} - \alpha\mathbf{1}) \vee \mathbf{y}) + f(\mathbf{x} \wedge (\mathbf{y} + \alpha\mathbf{1})),$$

for all $\mathbf{x}, \mathbf{y} \in L, U \subseteq \mathbb{Z}^n$, for all $\alpha \in \mathbb{Z}^+$ such that $\mathbf{x} - \alpha\mathbf{1}, \mathbf{y} + \alpha\mathbf{1} \in L, U \subseteq \mathbb{Z}^n$.

3. *Function f satisfies discrete midpoint convexity, defined as follows*:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f\left(\left\lfloor \frac{\mathbf{x}+\mathbf{y}}{2} \right\rfloor\right) + f\left(\left\lceil \frac{\mathbf{x}+\mathbf{y}}{2} \right\rceil\right),$$

for all $\mathbf{x}, \mathbf{y} \in L, U \subseteq \mathbb{Z}^n$, where for all $\mathbf{v} \in \mathbb{R}^n$, $\lfloor \mathbf{v} \rfloor = (\lfloor v_1 \rfloor, \lfloor v_2 \rfloor, \ldots, \lfloor v_n \rfloor)$ and $\lceil \mathbf{v} \rceil = (\lceil v_1 \rceil, \lceil v_2 \rceil, \ldots, \lceil v_n \rceil)$.

4. *Function f is both submodular and integrally convex.*

Next, we present two concepts that will be useful later in our analysis.

Definition 9 ($L^\natural$-Convex Set, Murota 2003). A subset $A \subseteq \mathbb{Z}^n$ is an $L^\natural$-convex set if there exist constants $a_i, b_i, c_{ij} \in \mathbb{Z}$ such that

$$A = \{\mathbf{x} \in \mathbb{Z}^n : a_i \leq x_i \leq b_i, x_i - x_j \leq c_{ij}, \forall i, j \in 1, \ldots, n, i \neq j\}.$$

A continuous function is called polyhedral if its epigraph is polyhedral.

Definition 10 (Polyhedral $L^\natural$-Convexity, Murota and Shioura 2000). Let D be a polyhedron. A function $f : D \to R$ is polyhedral $L^\natural$-convex if

1. f is a polyhedral, convex function and
2. f satisfies translation submodularity, i.e.,

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f((\mathbf{x} - \alpha\mathbf{1}) \vee \mathbf{y}) + f(\mathbf{x} \wedge (\mathbf{y} + \alpha\mathbf{1})),$$

for all $\mathbf{x}, \mathbf{y} \in D$, for all $\alpha \in \mathbb{R}^+$ such that $\mathbf{x} - \alpha\mathbf{1}, \mathbf{y} + \alpha\mathbf{1} \in D$.

3.3. Useful Properties of Discrete $L^\natural$-Convex Functions

As mentioned in §2, discrete $L^\natural$-convex functions have the following useful properties:

1. Positive scaling, addition, translation, uniform variable scaling, and minimization preserve $L^\natural$-convexity.
2. Efficient minimization algorithms exist.
3. Global extension over a bounded domain can be efficiently obtained.

In this section, we present these properties formally. Property 1 corresponds to Lemmas 4–7, Property 2 corresponds to Lemma 8, and Property 3 corresponds to Lemma 10. All these results will be used in our analysis in §§4, 6, and 7.

Lemma 4 (Murota 2003). *Let $f : \mathbb{Z}^n \to \mathbb{R}^+$ be a discrete $L^\natural$-convex function. Then for all $\alpha \in \mathbb{R}^+$, the function $f' : \mathbb{Z}^n \to \mathbb{R}^+$, defined as*

$$f'(\mathbf{x}) = \alpha f(\mathbf{x}),$$

is $L^\natural$-convex.

Lemma 5 (Murota 2003). *Let $f_1 : \mathbb{Z}^n \to \mathbb{R}^+$ and $f_2 : \mathbb{Z}^n \to \mathbb{R}^+$ be two discrete $L^\natural$-convex functions. Then the function $g : \mathbb{Z}^n \to \mathbb{R}^+$, defined as*

$$g(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}),$$

is $L^\natural$-convex.

LEMMA 6 (MUROTA 2003). *Let $f:\mathbb{Z}^n \to \mathbb{R}^+$ be a discrete $L^\natural$-convex function. Then for all $a \in \mathbb{Z}^n$, $b \in \mathbb{Z}\setminus\{0\}$, the function $f':\mathbb{Z}^n \to \mathbb{R}^+$, defined as*

$$f'(\mathbf{x}) = f(a + b\mathbf{x}),$$

is $L^\natural$-convex.

LEMMA 7. *Let $g(\mathbf{x},\mathbf{y}): C \subseteq \mathbb{Z}^{n+m} \to \mathbb{R}$ be an $L^\natural$-convex function, where C is an $L^\natural$-convex set. Define*

$$h(\mathbf{x}) = \min_{\mathbf{y}:(\mathbf{x},\mathbf{y})\in C} g(\mathbf{x},\mathbf{y}).$$

Then $h(\mathbf{x})$ is also $L^\natural$-convex.[5]

PROOF. To prove the $L^\natural$-convexity of $h(\mathbf{x})$, we use translation submodularity. That is, for all $\mathbf{x}_1$, $\mathbf{x}_2$, and $\alpha \in \mathbb{Z}^+$, we prove the following:

$$h(\mathbf{x}_1) + h(\mathbf{x}_2) \geq h((\mathbf{x}_1 - \alpha\mathbf{1}) \vee \mathbf{x}_2) + h(\mathbf{x}_1 \wedge (\mathbf{x}_2 + \alpha\mathbf{1})).$$

Suppose

$$h(\mathbf{x}_1) = g(\mathbf{x}_1,\mathbf{y}_1) \quad \text{and} \quad h(\mathbf{x}_2) = g(\mathbf{x}_2,\mathbf{y}_2).$$

Since $g(\mathbf{x},\mathbf{y})$ is $L^\natural$-convex, it satisfies translation submodularity. We have for $\alpha \in \mathbb{Z}^+$:

$$\begin{aligned}
h(\mathbf{x}_1) + h(\mathbf{x}_2) &= g(\mathbf{x}_1,\mathbf{y}_1) + g(\mathbf{x}_2,\mathbf{y}_2) \\
&\quad g\{[(\mathbf{x}_1,\mathbf{y}_1) - \alpha\mathbf{1} \vee (\mathbf{x}_2,\mathbf{y}_2)\} \\
&\quad + g\{(\mathbf{x}_1,\mathbf{y}_1) \wedge (\mathbf{x}_2,\mathbf{y}_2) + \alpha\mathbf{1}]\} \\
&= g\{(\mathbf{x}_1 - \alpha\mathbf{1}) \vee \mathbf{x}_2, (\mathbf{y}_1 - \alpha\mathbf{1}) \vee \mathbf{y}_2\} \\
&\quad + g\{\mathbf{x}_1 \wedge (\mathbf{x}_2 + \alpha\mathbf{1}), \mathbf{y}_1 \wedge (\mathbf{y}_2 + \alpha\mathbf{1})\} \\
&\quad h((\mathbf{x}_1 - \alpha\mathbf{1}) \vee \mathbf{x}_2) + h(\mathbf{x}_1 \wedge (\mathbf{x}_2 + \alpha\mathbf{1})).
\end{aligned}$$

Note that $g\{[(\mathbf{x}_1,\mathbf{y}_1) - \alpha\mathbf{1} \vee (\mathbf{x}_2,\mathbf{y}_2)\}$ and $g\{(\mathbf{x}_1,\mathbf{y}_1) \wedge (\mathbf{x}_2,\mathbf{y}_2) + \alpha\mathbf{1}]\}$ are well defined since $(\mathbf{x}_1,\mathbf{y}_1) - \alpha\mathbf{1} \vee (\mathbf{x}_2,\mathbf{y}_2), (\mathbf{x}_1,\mathbf{y}_1) \wedge (\mathbf{x}_2,\mathbf{y}_2) + \alpha\mathbf{1} \in C$, which, in turn, follows from the characterization of an $L^\natural$-convex set; see Property ($\mathbf{SBS}^\natural$ **Z**) of Murota (2003, p. 128). Thus, the result follows. □

LEMMA 8. *Given a discrete $L^\natural$-convex function $f: L,U]^k \to \mathbb{R}^+$, it takes time $O(\log^k \lceil U - L \rceil)$ to find a minimizer of f.*

PROOF. We have that

$$\min_{\mathbf{x}\in L,U]^k} f(\mathbf{x}) = \min_{x_1\in L,U]} \min_{x_2\in L,U]} \cdots \min_{x_k\in L,U]} f(\mathbf{x}).$$

Since minimization preserves $L^\natural$-convexity (Lemma 7), we know that for $i = 1, 2, \ldots, k$, the function

$$\min_{x_i\in L,U]} \min_{x_{i+1}\in L,U]} \cdots \min_{x_k\in L,U]} f(\mathbf{x})$$

is again $L^\natural$-convex in $(x_1, x_2, \ldots, x_{i-1})$. Therefore, to find $\arg\min f$, we can use binary search on each dimension recursively. The result follows. □

The following definition of the Lovász Extension (Lovász 1983) is needed to obtain a continuous extension of an $L^\natural$-convex function. We define this extension for an arbitrary discrete function (not necessarily $L^\natural$-convex) on the hypercube $\{L,U\}^n$. Note that $\{L,U\}^n$ refers to the set $\{\mathbf{z} \in \mathbb{Z}^n : z_j \in \{L,U\}, j = 1, 2, \ldots, n\}$. This is the set of vertices of $L,U]^n$. It is easy to see that $\mathrm{Conv}(\{L,U\}^n_{\mathbb{Z}}) = \mathrm{Conv}(\{L,U\}^n) = L,U]^n$.

DEFINITION 11 (LOVÁSZ EXTENSION). Given a discrete function $f:\{L,U\}^n \to \mathbb{R}$, for any given point $\mathbf{x} \in L,U]^n$, let σ be a permutation of $\{1, 2, \ldots, n\}$ such that $x_{\sigma(1)} \geq x_{\sigma(2)} \geq \cdots \geq x_{\sigma(n)}$. For $0 \leq i \leq n$, define $\mathbf{y}(i) \in \{L,U\}^n$ such that $y_{\sigma(1)}(i) = \cdots = y_{\sigma(i)}(i) = U$ and $y_{\sigma(i+1)}(i) = y_{\sigma(i+2)}(i) = \cdots = y_{\sigma(n)}(i) = L$. Note that for $0 \leq i \leq n$, the point $\mathbf{y}(i)$ depends on $\mathbf{x}$. Let λ_i be the unique coefficient of $\mathbf{y}(i)$, if we represent $\mathbf{x}$ as a convex combination of $\mathbf{y}(i), i = 0, 1, \ldots, n$; i.e.,

$$\mathbf{x} = \sum_{i=0}^{n} \lambda_i \mathbf{y}(i). \tag{2}$$

The Lovász Extension $f^{\bar{L}}: L,U]^n \to \mathbb{R}$ of function f at point $\mathbf{x}$ is defined as follows:

$$f^{\bar{L}}(\mathbf{x}) = \sum_{i=0}^{n} \lambda_i f(\mathbf{y}(i)).$$

Observe that in Definition 11, for $i = 0, 1, \ldots, n$, λ_i can be obtained by solving Equation (2). In closed form, we have the following: $\lambda_0 = (U - x_{\sigma(1)})/(U - L)$, $\lambda_j = (x_{\sigma(j)} - x_{\sigma(j+1)})/(U - L)$, $j = 1, 2, \ldots, n-1$, and $\lambda_n = (x_{\sigma(n)} - L)/(U - L)$.

To help the readers gain some intuition about the previous definition, we illustrate it using a simple two-dimensional example. Let $f:\{0,1\}^2 \to \mathbb{R}$ be defined as follows: $f(0,0) = 1$, $f(0,1) = 2$, $f(1,0) = 3$, and $f(1,1) = 4$. Consider the point $\mathbf{x} = (0.4, 0.6)$. We have, $x_{\sigma(1)} = 0.6$ and $x_{\sigma(2)} = 0.4$. Observe that in Definition 11, given any $\mathbf{x} \in L,U]^n$, $\mathbf{x}$ can be written as a unique convex combination of $\mathbf{y}(0), \mathbf{y}(1), \ldots, \mathbf{y}(n)$. By definition, $\mathbf{y}(0) = (0,0)$, $\mathbf{y}(1) = (0,1)$, and $\mathbf{y}(2) = (1,1)$. Since

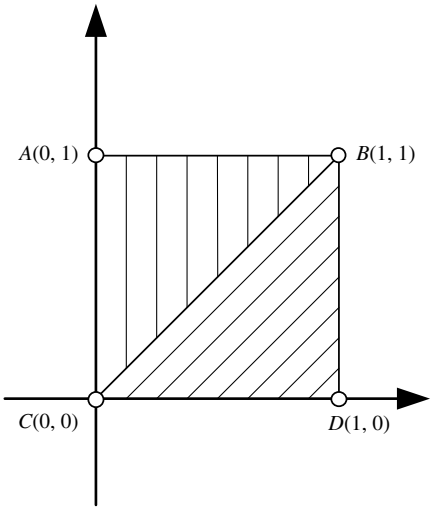
$$\mathbf{x} = 0.4\mathbf{y}(0) + 0.2\mathbf{y}(1) + 0.4\mathbf{y}(2),$$

we conclude that

$$f^{\bar{L}}(\mathbf{x}) = 0.4f(\mathbf{y}(0)) + 0.2f(\mathbf{y}(1)) + 0.4f(\mathbf{y}(2)) = 2.4.$$

It is easy to see that the Lovász Extension of a discrete function on a unit hypercube is piecewise-linear. Furthermore, for an n-dimensional unit hypercube, there are at most $n!$ linear pieces, each corresponding to a domain region such that for any point $\mathbf{x}$ in that region, the order

Figure 1. Illustrating the Lovász Extension of a two-dimensional discrete function defined on the standard unit hypercube.



of its coordinates is fixed. See Figure 1 for an illustration in the two-dimensional case. In this case, the Lovász Extension consists of the two linear pieces on the following triangular regions: $\triangle ABC$ and $\triangle BCD$.

For a discrete submodular function defined on a hypercube $\{L, U\}^n$, its convex envelope is identical to its Lovász Extension. The following result is due to Murota (2003).

Lemma 9. *For a discrete function f defined on a hypercube $\{L, U\}^n$, its convex envelope is identical to its Lovász Extension if and only if f is submodular.*

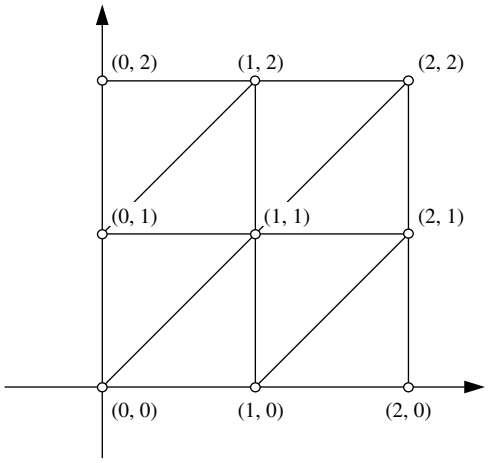
Observe that for a discrete function f, its Lovász Extension is efficiently computable, given oracle access to f. This and the above lemma imply that for a discrete submodular function defined on a hypercube, its convex envelope is efficiently computable. Consequently, for a discrete submodular function on a larger domain, its local extension can be efficiently computed. This is because, as mentioned earlier, for a discrete function f, its local extension restricted to a unit hypercube in its domain is identical to the convex envelope of the function obtained by restricting f to that unit hypercube.

The following result, due to Murota (2003), is about the construction of the global extension of a discrete $L^\natural$-convex function.

Lemma 10. *Given a discrete $L^\natural$-convex function f, its global extension can be obtained as follows: first obtain its Lovász Extension for every unit hypercube in its domain, and then paste all these extensions together.*

To help the readers, we explain the proof of this lemma. By the definition of $L^\natural$-convexity, we know that f is both submodular and integrally convex. From the submodularity of f, by Lemma 9, we know that the local extension of f can be constructed by obtaining its Lovász Extension for every unit hypercube in its domain and then pasting all these extensions together. From the integer convexity of f, by Lemma 2, we know that its local extension is identical to its global extension.

Figure 2. Illustrating the convex extension of a two-dimensional discrete $L^\natural$-convex function defined on $\{0, 2\}^2$.



From now on, we will refer to the global extension of a discrete $L^\natural$-convex function as its *convex extension*. Observe that for any discrete $L^\natural$-convex function f, given oracle access to f, the above lemma gives an efficient algorithm to construct its convex extension. See Figure 2 for an illustration of the convex extension of a two-dimensional discrete $L^\natural$-convex function defined on $\{0, 2\}^2$. The convex extension is linear within each triangular region shown in the figure. Next, we present a result regarding polyhedral $L^\natural$-convex functions that will be useful later in our analysis.

Lemma 11. *Let $f : \mathbb{Z}^n \to \mathbb{R}$ be a discrete $L^\natural$-convex function and f' be its convex extension. Then for $\mathbf{a} \in \mathbb{Z}^n$ and $\beta \in \mathbb{R}$, the function $g : \mathbb{Z}^n \to \mathbb{R}$ defined as $g(\mathbf{x}) = f'(\beta(\mathbf{x} + \mathbf{a}))$ is $L^\natural$-convex.*

Proof.[6] By Theorem 7.3(1) in Murota (2007),[7] f' is a polyhedral $L^\natural$-convex function. For any integral $\mathbf{a}$ and positive β, $h(\mathbf{x}) = f'(\beta(\mathbf{x} + \mathbf{a}))$, defined for a real vector $\mathbf{x}$, is a polyhedral $L^\natural$-convex function; see Theorem 7.32(1) in Murota (2003). Finally, $g(\mathbf{x}) = h(\mathbf{x})$, defined for an integral vector $\mathbf{x}$, is a discrete $L^\natural$-convex function; see Theorem 7.1(1) in Murota (2007). □

4. Approximating $L^\natural$-Convex Functions

Our goal in this section is to approximate an $L^\natural$-convex function by considering its values on only a subset of its domain. To this end, §4.1 defines the notions of an approximation and an ε-approximation set. In §4.2, we present our approximation of an $L^\natural$-convex function and show that this approximation is also an $L^\natural$-convex function. These results will then be used in §6 to develop our approximation algorithm.

4.1. ε-Approximation Sets and Functions

In this subsection, we modify the notion of a K-approximation set in Halman et al. (2009, 2013) for a relative-error approximation to obtain analogous results for an additive-error approximation. We need the following definitions.

Definition 12 (Additive-Error Approximation). Let $\varepsilon>0$, $n\in\mathbb{Z}^+$, $D\subseteq\mathbb{R}^n$, and $f\colon D\to\mathbb{R}^+$. We say that $\hat{f}\colon D\to\mathbb{R}^+$ is an ε-approximation of f, if for all $x\in D$ we have

$$f(x)\le \hat{f}(x)\le f(x)+\varepsilon.$$

In the following lemma, we discuss some operations that preserve ε-approximation. The proof is straightforward and, therefore, omitted.

Lemma 12. *Given $\varepsilon_1,\varepsilon_2,\varepsilon_3>0$, $m,n\in\mathbb{Z}^+$, $D_1\subseteq\mathbb{R}^m$, $D_2\subseteq\mathbb{R}^n$, $f_1\colon D_1\to\mathbb{R}^+$, $f_2\colon D_1\to\mathbb{R}^+$, and $g\colon D_1\times D_2\to\mathbb{R}^+$. Let $\hat{f}_1$ be an ε_1-approximation of f_1, $\hat{f}_2\colon D_1\to\mathbb{R}^+$ be an ε_2-approximation of f_2 and $\hat{g}\colon D_1\times D_2\to\mathbb{R}^+$ be an ε_3-approximation of g. Then the following statements hold*:

1. *For $\alpha>0$, $\alpha\hat{f}_1$ is an $\alpha\varepsilon_1$-approximation of αf_1.*
2. *The function $\hat{f}_1+\hat{f}_2$ is an $(\varepsilon_1+\varepsilon_2)$-approximation of f_1+f_2.*
3. *The function $\min_{\mathbf{y}}\hat{g}(\mathbf{x},\mathbf{y})$ is an ε_3-approximation of $\min_{\mathbf{y}} g(\mathbf{x},\mathbf{y})$.*

Definition 13 (ε-Approximation Set for a Convex Function). Let $\varepsilon>0$ and let $g\colon L,U\cap\mathbb{Z}\to\mathbb{R}^+$ be a convex function. An ε-approximation set of g is an ordered set of integers $S=\{i_1=L<i_2<\cdots<i_r=U\}\subseteq L,U\cap\mathbb{Z}$ satisfying the following properties:
For each $k\in\{1,2,\ldots,r-1\}$ such that $i_{k+1}>i_k+1$, the following holds:

1. If g is monotonically increasing in $i_k,i_{k+1}\cap\mathbb{Z}$, then

$$g(i_{k+1})-g(i_k)-(g(i_k+1)-g(i_k))(i_{k+1}-i_k)\le\varepsilon.$$

2. If g is monotonically decreasing in $i_k,i_{k+1}\cap\mathbb{Z}$, then

$$g(i_k)-g(i_{k+1})-(g(i_{k+1}-1)-g(i_{k+1}))(i_{k+1}-i_k)\le\varepsilon.$$

3. If g first decreases and then increases in $i_k,i_{k+1}\cap\mathbb{Z}$, let $x^*=\arg\min_{x\in\{i_k\},i_{k+1}\cap\mathbb{Z}} g(x)$. Then

$$g(i_k)-g(x^*)\le\varepsilon \quad\text{and}\quad g(i_{k+1})-g(x^*)\le\varepsilon.$$

Note that (i) an ε-approximation set of a convex function always exists, and in the worst case, includes all the points in $L,U\cap\mathbb{Z}$ and (ii) the minimizer $x^*=\arg\min_{x\in L,U\cap\mathbb{Z}} g(x)$ is not necessarily included in an ε-approximation set of g. We now show how an ε-approximation set of a convex function can be used to construct an ε-approximation of that function.

Lemma 13. *Let $\varepsilon>0$ and let $g\colon L,U\cap\mathbb{Z}\to\mathbb{R}^+$ be a convex function. Let $S=\{i_1=L<i_2<\cdots<i_r=U\}\subseteq L,U\cap\mathbb{Z}$ be an ε-approximation set of g. Then the function $\breve{g}\colon L,U\cap\mathbb{Z}\to\mathbb{R}^+$, defined as follows, is an ε-approximation of g:*

$$\breve{g}(x)=\begin{cases} g(x), & \text{if } x\in\{i_1,i_2,\ldots,i_r\},\\ \dfrac{g(i_{k+1})-g(i_k)}{i_{k+1}-i_k}x+\dfrac{g(i_k)i_{k+1}-g(i_{k+1})i_k}{i_{k+1}-i_k}, & \\ & \text{if } i_k<x<i_{k+1}, \text{for some } k\in\{1,2,\ldots,r-1\}.\end{cases}$$

In words, the function $\breve{g}$ is constructed by linearly interpolating the function values defined on any two consecutive points in S. The proof of the lemma follows easily from the convexity of g and is, therefore, omitted.

Note that an ε-approximation set may not be unique. Next, to suit our purpose of approximating a discrete $L^\natural$-convex function, we define the canonical *uniform-interval* ε-approximation set.

Definition 14 (Uniform-Interval ε-Approximation Set). A uniform-interval ε-approximation set for a convex function $g\colon L,U\cap\mathbb{Z}\to\mathbb{R}^+$ is an ε-approximation set for g such that the intervals between two consecutive points in the set have the same length; i.e.,

$$i_r-i_{r-1}=i_{r-1}-i_{r-2}=\cdots=i_2-i_1.$$

The *canonical* uniform-interval ε-approximation set for g is a uniform-interval ε-approximation set such that the length of the interval is the maximum possible power of 2.

Figure 3 illustrates the above definition. For the function $f\colon 0,14\cap\mathbb{Z}\to\mathbb{R}^+$ shown in the figure, $\{0,2,4\}$ is the canonical uniform-interval 3-approximation set.

The following lemma bounds the size of the canonical uniform-interval ε-approximation set for a monotone convex function g and the time required for its construction.

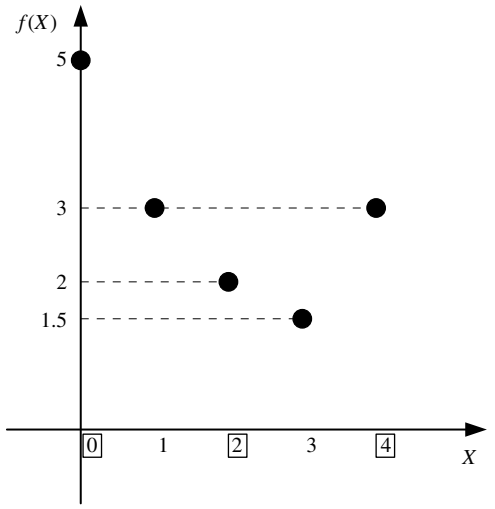
Lemma 14. *Let $g\colon L,U\cap\mathbb{Z}\to\mathbb{R}^+$ be a monotone convex function and let $s=\max\{|g(L)-g(L+1)|,|g(U)-g(U-1)|\}$. For any $\varepsilon>0$, the canonical uniform-interval ε-approximation set S of g has cardinality $O((U-L)\min\{1,s/\varepsilon\})$. Furthermore, it takes time $O((1+t_g)(U-L)\min\{1,s/\varepsilon\})$ to construct this set, where t_g is the time needed to evaluate g at a point.*

Proof. We prove the statement for the case when g is increasing. The proof for the case when g is decreasing is similar and, therefore, omitted.

For a given $\varepsilon>0$, let t denote the length of the interval in the canonical uniform-interval ε-approximation set S of g. Since $2t$ is not the interval length, then by definition, there must exist some $x\in\{L,L+2t,\ldots,U-2t\}$ such that

$$g(x+2t)-g(x)-(g(x+1)-g(x))2t>\varepsilon.$$

Figure 3. Illustrating the definition of the canonical uniform-interval ε-approximation set for a convex function f.



Then we have

$$\begin{aligned} & g(x+2t)-g(x)-(g(x+1)-g(x))2t \\ & \quad g(U)-g(U-2t)-(g(x+1)-g(x))2t \\ & \qquad\qquad \text{(convexity of } g\text{)} \\ & \quad g(U)-g(U-2t)] \quad (g \text{ is increasing}) \\ & \quad 2ts \quad \text{(convexity of } g \text{ and the defnition of } s\text{)} \end{aligned}$$

Thus,

$$2ts > \varepsilon. \tag{3}$$

Therefore, the cardinality of S, given by $1+(U-L)/t$, is bounded by $O((U-L)(s/\varepsilon))$. Additionally, since the cardinality is also trivially bounded by $O(U-L)$, we conclude that the cardinality is $O((U-L)\min\{1,s/\varepsilon\})$. As a result, the determination of the interval length t and the construction of the ε-approximation set S can be done in time $O(t_g(U-L)\min\{1,s/\varepsilon\})$, by simply checking interval lengths in the sequence of $2^l, 2^{l-1}, \ldots, 1$ (assuming $U-L=2^l$). □

Note that the upper bound $O((U-L)\min\{1,s/\varepsilon\})$ is not tight because of the strict inequality in (3).

We now generalize the lemma above to convex functions. The proof is similar and, therefore, omitted.

Lemma 15. *Let $g: L,U]_{\mathbb{Z}^k} \to \mathbb{R}^+$ be a convex function, and let $s=\max\{|g(L)-g(L+1)|, |g(U)-g(U-1)|\}$. For every $\varepsilon>0$, the canonical uniform-interval ε-approximation set S of g has cardinality $O((U-L)/\min\{1,s/\varepsilon\})$ and can be constructed in time $O((1+t_g)(U-L)\min\{1,s/\varepsilon\})$, where t_g is the time needed to evaluate g at a point.*

4.2. Approximating a Discrete $L^\natural$-Convex Function

In this subsection, our purpose is to construct an additive-error approximation of a discrete $L^\natural$-convex function $f: L,U]_{\mathbb{Z}^k} \to \mathbb{R}^+$. Our approximation only uses the values of the original function on a *subset* of its domain, and is a generalization of the single-dimensional approximation of §4.1. Hereafter, we refer to the domain of f, i.e., $L,U]_{\mathbb{Z}^k}$ as the *domain cube*.

Let t be a factor of $U-L$ (i.e., t divides $U-L$ exactly). Let $a_i \in \{L, L+t, \ldots, U\}$, $i=1,2,\ldots,k$. Then the induced *gridlines* of the hypercube $\{L, L+1, \ldots, U\}^k$ corresponding to a_i, $i=1,2,\ldots,k$, are as follows:

$$\begin{aligned} & \{(x, a_2, a_3, \ldots, a_k) \mid x \in \{L, L+1, \ldots, U\}\}, \\ & \{(a_1, x, a_3, \ldots, a_k) \mid x \in \{L, L+1, \ldots, U\}\}, \\ & \vdots \\ & \{(a_1, a_2, a_3, \ldots, x) \mid x \in \{L, L+1, \ldots, U\}\}. \end{aligned}$$

The complete set of induced grid-lines of $\{L, L+1, \ldots, U\}^k$ is obtained by considering all possible choices of $a_1, a_2, \ldots, a_k \in \{L, L+1, \ldots, U\}$. The following definition generalizes to multiple dimensions our notion of canonical uniform-interval ε-approximation set for a single-dimensional function (Definition 14).

Definition 15 (ε-Coarse Cube). Let $\varepsilon>0$ and t be a factor of $U-L$. Let $f: L,U]_{\mathbb{Z}^k} \to \mathbb{R}^+$ be an $L^\natural$-convex function. We refer to the set $\{L, L+1, \ldots, U\}^k$ as the ε-coarse cube of f and t as the length of the ε-coarse cube if the following two conditions hold:

1. For each induced gridline E of $\{L, L+1, \ldots, U\}^k$, the set $\{L, L+1, \ldots, U\}$ is an ε-approximation set of $f|_E$, where $f|_E$ is the restriction of f to E.
2. The length t is the maximum power of 2 such that condition 1 above holds.

Note that this definition can be easily generalized to the case when the domain is a rectangular set instead of a hypercube.

We use values of f only at points on its ε-coarse cube to construct its approximation $\breve{f}$.

Definition 16 (Convex ε-Approximation).[8] Given a discrete $L^\natural$-convex function $f: L,U]_{\mathbb{Z}^k} \to \mathbb{R}^+$, $\varepsilon>0$, and the ε-coarse cube C of f, we define $\breve{f}: L,U]_{\mathbb{Z}^k} \to \mathbb{R}^+$ as follows:

$$\breve{f}(\mathbf{x}) = \{\min z \mid (\mathbf{x}, z) \in \text{Conv}(H)\},$$

where H is the set of all points $(\mathbf{p}, f(\mathbf{p}))$ such that $\mathbf{p} \in C$. Thus, $\breve{f}(\mathbf{x}) = \overline{f|_C}(\mathbf{x}), \forall\, \mathbf{x} \in L,U]_{\mathbb{Z}^k}$.

Our next result shows that $\breve{f}$ constructed above closely approximates f and preserves $L^\natural$-convexity.

LEMMA 16. *Given a discrete $L^\natural$-convex function $f: \{L, L+1, \ldots, U\}^k \to \mathbb{R}^+$, $\varepsilon > 0$ and the ε-coarse cube C of f, the convex ε-approximation function $\breve{f}$ of f (obtained as in Definition 16) is its $k\varepsilon$-approximation in the additive sense. Furthermore, $\breve{f}$ is also $L^\natural$-convex.*

PROOF. We first prove the $L^\natural$-convexity result and then prove the approximation result.

Let t be the length of the ε-coarse cube C of f. Let $f_1(\mathbf{x}) = f(\mathbf{x} + L\mathbf{1})$. By Lemma 6, f_1 is a discrete $L^\natural$-convex function defined on $\{0, 1, \ldots, U-L\}^k$. Let $f_2(\mathbf{x}) = f_1(t\mathbf{x})$. Again, by Lemma 6, f_2 is a discrete $L^\natural$-convex function defined on $\{0, 1, \ldots, (U-L)/t\}^k$. Let f_2' be the convex extension of f_2; i.e., for all $\mathbf{x} \in [0, (U-L)/t]^k$, we have

$$f_2'(\mathbf{x}) = \min\left\{\sum_{i=1}^{n} \alpha_i f_2(\mathbf{z}_i) : \mathbf{z}_i \in \left\{0, \frac{U-L}{t}\right\}^k, \sum_{i=1}^{n} \alpha_i \mathbf{z}_i = \mathbf{x}, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0\right\}, \tag{4}$$

where $n = (\lfloor (U-L)/t \rfloor + 1)^k$ is the number of points in the set $\{0, 1, \ldots, \lfloor (U-L)/t \rfloor\}^k$.

Let function $g: [L, U]^k \to \mathbb{R}^+$ be defined as

$$g(\mathbf{x}) = f_2'\left(\frac{1}{t}(\mathbf{x} - L\mathbf{1})\right).$$

Using the above definitions of g, f_2', f_2, and f_1, we obtain that for all $\mathbf{x} \in [L, U]^k$,

$$g(\mathbf{x}) = \min\left\{\sum_{i=1}^{n} \alpha_i f(\mathbf{z}_i) : \mathbf{z}_i \in C, \sum_{i=1}^{n} \alpha_i \mathbf{z}_i = \mathbf{x}, \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0\right\},$$

where C is the ε-coarse cube of f and $n = |C| = (\lfloor (U-L)/t \rfloor + 1)^k$.

Noticing that $\breve{f}(\mathbf{x})$ is the restriction of $g(\mathbf{x})$ to the discrete domain $\{L, L+1, \ldots, U\}^k$ from Lemma 11, we conclude that $\breve{f}(\mathbf{x})$ is discrete $L^\natural$-convex.

Next, we prove that $f(\mathbf{x}) \geq \breve{f}(\mathbf{x}) \geq f(\mathbf{x}) + k\varepsilon$ for all $\mathbf{x} \in [L, U]^k$. Pick any point $\mathbf{x} \in [L, U]^k$. From the convexity of f, we have $f(\mathbf{x}) \geq \breve{f}(\mathbf{x})$. To prove that $\breve{f}(\mathbf{x}) \geq f(\mathbf{x}) + k\varepsilon$, consider any hypercube $\mathcal{H} \subseteq C$ such that $\mathbf{x} \in \text{conv}(\mathcal{H})$ and each edge of $\mathcal{H}$ is of length t. By construction, we have

$$\breve{f}(\mathbf{x}) = \sum_{i=0}^{k} \lambda_i f(\mathbf{y}(i)),$$

where $\mathbf{y}(i)$, $i = 0, 1, \ldots, k$, are as defined in Definition 11 and $\mathbf{x} = \sum_{i=0}^{k} \lambda_i \mathbf{y}(i)$. Notice that for $i = 0, 1, \ldots, k-1$, we have $\mathbf{y}(i) \vee \mathbf{y}(i+1) = \mathbf{y}(i+1)$ and $\mathbf{y}(i) \wedge \mathbf{y}(i+1) = \mathbf{y}(i)$. To proceed, we need to prove

$$\sum_{i=0}^{k} \lambda_i f(\mathbf{y}(i)) - f(\mathbf{x}) \geq k\varepsilon.$$

To this end, observe that

$$\begin{aligned}
&\sum_{i=1}^{k-1} f(\mathbf{y}(i)) + f(\mathbf{x}) \\
&\quad \geq f(\mathbf{x} \wedge \mathbf{y}(1)) + f(\mathbf{x} \vee \mathbf{y}(1)) \wedge \mathbf{y}(2)) + f(\mathbf{x} \vee \mathbf{y}(2)) \wedge \mathbf{y}(3)) \\
&\qquad + \cdots + f(\mathbf{x} \vee \mathbf{y}(k-1)) \wedge \mathbf{y}(k)). \qquad (5)
\end{aligned}$$

The above inequality is obtained by adding the following inequalities, which all hold since f is submodular:

$$\begin{aligned}
&f(\mathbf{x}) + f(\mathbf{y}(1)) \geq f(\mathbf{x} \wedge \mathbf{y}(1)) + f(\mathbf{x} \vee \mathbf{y}(1)), \\
&f(\mathbf{x} \vee \mathbf{y}(1)) + f(\mathbf{y}(2)) \geq f(\mathbf{x} \vee \mathbf{y}(1)) \wedge \mathbf{y}(2)) + f(\mathbf{x} \vee \mathbf{y}(2)) \\
&\qquad\qquad \text{(since } \mathbf{y}(1) \vee \mathbf{y}(2) = \mathbf{y}(2)), \\
&\vdots \\
&f(\mathbf{x} \vee \mathbf{y}(k-1)) + f(\mathbf{y}(k)) \geq f(\mathbf{x} \vee \mathbf{y}(k-1)) \wedge \mathbf{y}(k)) + f(\mathbf{y}(k)) \\
&\qquad\qquad \text{(since } \mathbf{x} \vee \mathbf{y}(k) = \mathbf{y}(k)).
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\sum_{i=0}^{k} \lambda_i f(\mathbf{y}(i)) - f(\mathbf{x}) \\
&\quad = \left(\sum_{i=0}^{k} \lambda_i f(\mathbf{y}(i)) + \sum_{i=1}^{k-1} f(\mathbf{y}(i))\right) - \left[\sum_{i=1}^{k-1} f(\mathbf{y}(i)) + f(\mathbf{x})\right] \\
&\quad \geq \left(\sum_{i=0}^{k} \lambda_i f(\mathbf{y}(i)) + \sum_{i=1}^{k-1} f(\mathbf{y}(i))\right) - f(\mathbf{x} \wedge \mathbf{y}(1)) \\
&\qquad - f(\mathbf{x} \vee \mathbf{y}(1)) \wedge \mathbf{y}(2)) - f(\mathbf{x} \vee \mathbf{y}(2)) \wedge \mathbf{y}(3)) - \cdots / \\
&\qquad - f(\mathbf{x} \vee \mathbf{y}(k-1)) \wedge \mathbf{y}(k))] \quad \text{(from (5))} \\
&\quad = \breve{f}(\mathbf{x} \wedge \mathbf{y}(1)) - f(\mathbf{x} \wedge \mathbf{y}(1)) + \breve{f}(\mathbf{x} \vee \mathbf{y}(1)) \wedge \mathbf{y}(2)) \\
&\qquad - f(\mathbf{x} \vee \mathbf{y}(1)) \wedge \mathbf{y}(2)) + \breve{f}(\mathbf{x} \vee \mathbf{y}(2)) \wedge \mathbf{y}(3)) \\
&\qquad - f(\mathbf{x} \vee \mathbf{y}(2)) \wedge \mathbf{y}(3)) + \cdots + \breve{f}(\mathbf{x} \vee \mathbf{y}(k-1)) \wedge \mathbf{y}(k)) \\
&\qquad - f(\mathbf{x} \vee \mathbf{y}(k-1)) \wedge \mathbf{y}(k)) \\
&\quad \geq k\varepsilon \quad \text{(Definition 15 and Lemma 13).}
\end{aligned}$$

The second to last inequality above holds since

$$\begin{aligned}
&\mathbf{x} = \left(\sum_{i=0}^{k} \lambda_i \mathbf{y}(i)\right), \\
&\mathbf{x} \wedge \mathbf{y}(1) = \left(\sum_{i=0}^{k} \lambda_i \mathbf{y}(i)\right) \wedge \mathbf{y}(1) = \lambda_0 \mathbf{y}(0) + \left(\sum_{i=1}^{k} \lambda_i\right) \mathbf{y}(1), \\
&\mathbf{x} \vee \mathbf{y}(1) \wedge \mathbf{y}(2) = \left(\sum_{i=0}^{1} \lambda_i\right) \mathbf{y}(1) + \left(\sum_{i=2}^{k} \lambda_i\right) \mathbf{y}(2), \\
&\vdots \\
&\mathbf{x} \vee \mathbf{y}(k-1) \wedge \mathbf{y}(k) = \left(\sum_{i=0}^{k-1} \lambda_i\right) \mathbf{y}(k-1) + \lambda_k \mathbf{y}(k),
\end{aligned}$$

Figure 4. Original discrete function f (proof of Lemma 16).

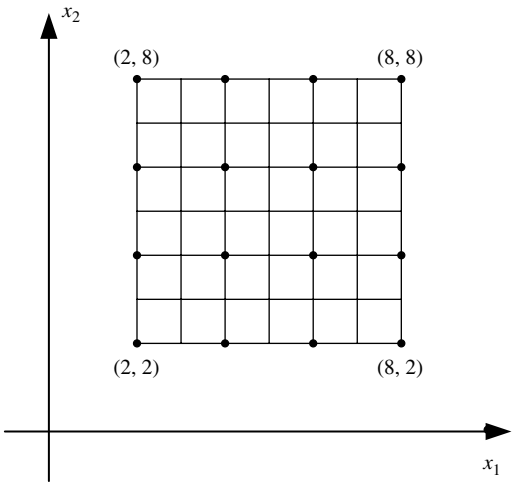
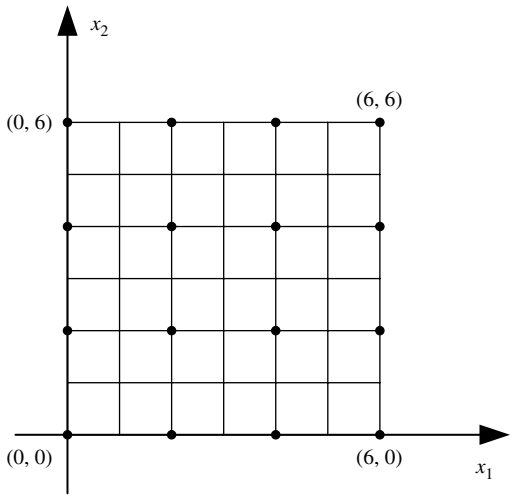


Figure 5. Discrete function f_1 (proof of Lemma 16).



and

$$\breve{f}(\mathbf{x}\wedge\mathbf{y}(1)) = \lambda_0 f(\mathbf{y}(0)) + \left(\sum_{i=1}^{k}\lambda_i\right) f(\mathbf{y}(1)),$$

$$\breve{f}((\mathbf{x}\vee\mathbf{y}(1))\wedge\mathbf{y}(2)) = \left(\sum_{i=0}^{1}\lambda_i\right) f(\mathbf{y}(1)) + \left(\sum_{i=2}^{k}\lambda_i\right) f(\mathbf{y}(2)),$$

$$\vdots$$

$$\breve{f}((\mathbf{x}\vee\mathbf{y}(k-1))\wedge\mathbf{y}(k)) = \left(\sum_{i=0}^{k-1}\lambda_i\right) f(\mathbf{y}(k-1)) + \lambda_k f(\mathbf{y}(k)).$$

The result follows. □

Figures 4 through 9 illustrate the proof of Lemma 16. In the figures, the original function f is defined on $\{2, 8\} \times \frac{2}{\mathbb{Z}}$. The length of its ε-coarse cube is 2. The domain points on the ε-coarse cube are depicted using black dots.

Figure 6. Discrete function f_2 (proof of Lemma 16).

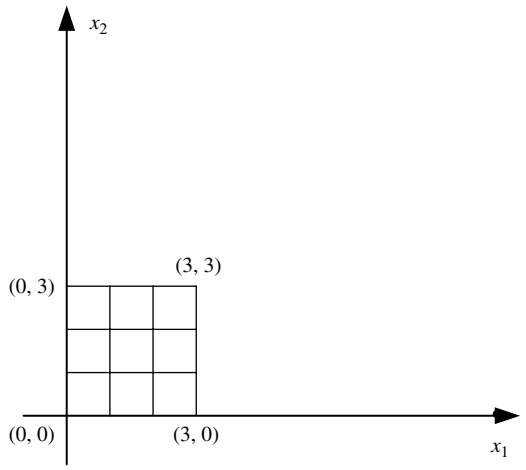


Figure 7. Function $f_2'^{\prime}$ as the convex extension of f_2 (proof of Lemma 16).

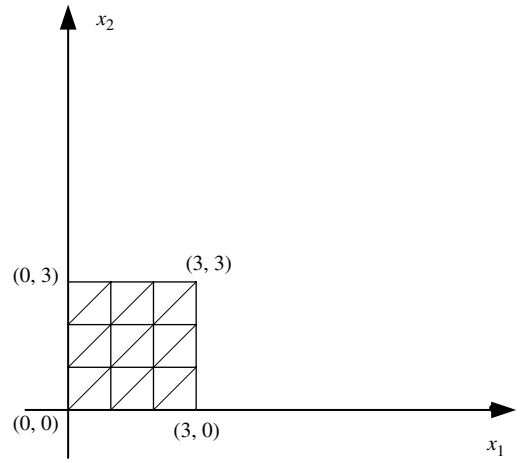


Figure 8. Function g (proof of Lemma 16).

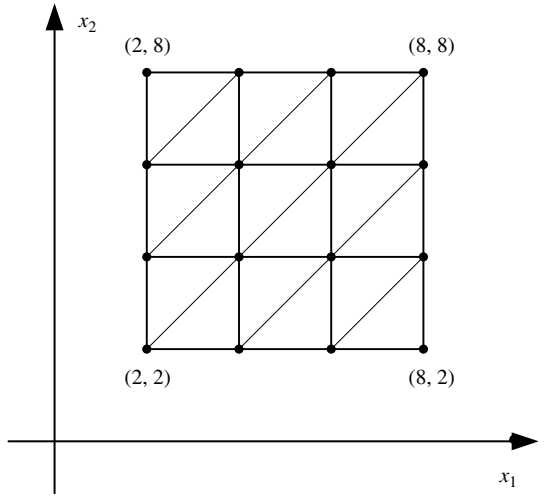
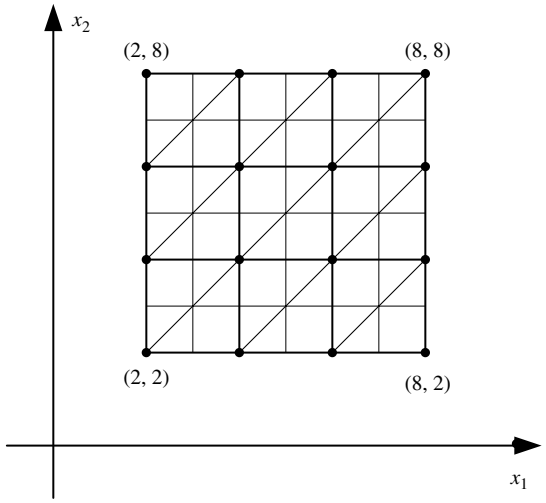


Figure 9. Discrete function $\breve{f}$ (proof of Lemma 16).



We now generalize Lemma 15 to higher dimensions. The proof is similar and is, therefore, omitted.

LEMMA 17. *Let $f: [L,U]^k_{\mathbb{Z}} \to \mathbb{R}^+$ be a discrete $L^\natural$-convex function, and let s be an upper bound on the absolute value of the slope of the function $f|_E$ for all induced gridline E of $[L,U]^k_{\mathbb{Z}}$. For every $\varepsilon > 0$, the ε-coarse cube of f has cardinality $O((U-L)^k \min\{1, s/\varepsilon\}^k)$ and can be constructed in time $O((1+t_f)(U-L)^k \min\{1, s/\varepsilon\}^k)$, where t_f is the time needed to evaluate f at a point.*

Given an $L^\natural$-convex function $f: [L,U]^k_{\mathbb{Z}} \to \mathbb{R}^+$ and its ε-coarse cube C, the proof of Lemma 16 suggests an alternative way to construct $\breve{f}$ as an approximation of f. First, we obtain function g defined in the proof using the following method: On each smallest hypercube within C, obtain the Lovász Extension of f and then paste these extensions together. Then our desired approximation $\breve{f}$ can be obtained by restricting g to the discrete domain. In other words, *for any given $\mathbf{x} \in [L,U]^k_{\mathbb{Z}}$, $\breve{f}(\mathbf{x})$ is the Lovász Extension of f at point $\mathbf{x}$ in the smallest hypercube in C that contains $\mathbf{x}$.*

Recall that to obtain the Lovász Extension of f for a given point $\mathbf{x}$, by Definition 11, we need to access the neighboring points of $\mathbf{x}$ on the ε-coarse cube and the corresponding values of f. Later, in our approximation algorithm, for a given $\varepsilon > 0$ and a discrete $L^\natural$-convex function f, we will store the points of its ε-coarse cube and the function values at these points and use this information to support future queries of $\breve{f}$. The next result characterizes the query time, given this information. The *query time* of $\breve{f}(\mathbf{x})$ is the time taken to return the value of $\breve{f}(\mathbf{x})$ for any $\mathbf{x}$ in the domain.

LEMMA 18. *For a given $L^\natural$-convex function $f: [L,U]^k_{\mathbb{Z}} \to \mathbb{Z}^+$ and its ε-coarse cube C, if all points in C and the function values at these points are stored in a sorted list, then for any $\mathbf{x} \in [L,U]^k_{\mathbb{Z}}$, the query time of $\breve{f}(\mathbf{x})$ is $O((k+1)(\log(U-L)+b)+b+1)$, where 2^l is the length of C and b is the amount of space required to store $f(\mathbf{x})$ for a given $\mathbf{x}$ in the domain.*

PROOF. As discussed earlier, for any $\mathbf{x} \in [L,U]^k_{\mathbb{Z}}$, $\breve{f}(\mathbf{x})$ is the Lovász Extension of f at point $\mathbf{x}$ in the smallest hypercube in C that contains $\mathbf{x}$. To compute this, we first determine the smallest hypercube in C that contains $\mathbf{x}$ and then use Definition 11. Recall that in Definition 11, for any given $\mathbf{x}$, we find the $k+1$ points $\mathbf{y}(0), \mathbf{y}(1), \ldots, \mathbf{y}(k)$ and interpolate the corresponding function values $f(\mathbf{y}(0)), f(\mathbf{y}(1)), \ldots, f(\mathbf{y}(k))$. This requires time $O((k+1)(\log(U-L)+b))$. Since 2^l is the length of C and f is constructed using linear interpolation, $2^l \breve{f}(\mathbf{x})$ is integer-valued for all $\mathbf{x} \in [L,U]^k$. Thus, outputting the exact value of $\breve{f}(\mathbf{x})$ takes time $O(b+1)$. The result follows. □

We will use this lemma in §6 to analyze the running time of our approximation algorithm.

5. Assumptions on the Dynamic Program and Analysis of the Explicit-Enumeration Algorithm

We first define the notation, state the assumptions for our problem, and specify the input in §5.1. Next, in §5.2, we present an explicit-enumeration dynamic programming algorithm and analyze its running time in the binary size of the input. At the end of this section, we show that no polynomial-time approximation algorithm with a fixed additive-error guarantee can exist for the problem unless P = NP.

5.1. Notation and Assumptions

Let T denote the length of the finite horizon. In each period $t = 1, 2, \ldots, T$, the sequence of events is as follows: At the beginning of the period, $\mathbf{x}_t$ is observed. Then an action $\mathbf{a}_t$ is taken. At the end of the period, the random variable $\mathbf{w}_t$ is realized. For each period, the action $\mathbf{a}_t$ belongs to a constrained action space $\mathcal{A}_t(\mathbf{x}_t)$. The state transition equation is $\mathbf{x}_{t+1} = \mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)$, where $\mathbb{T}: \mathbb{Z}^{k_1+k_2+k_3} \to \mathbb{Z}^{k_1}$ is a transformation function. Given $\mathbf{x}_t, \mathbf{a}_t$, and $\mathbf{w}_t$, the cost incurred in period t is $r_t(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)$. Let $G_t(\mathbf{x}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{w}_t}[r_t(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)]$ denote the expected cost incurred in period t for a given $\mathbf{x}_t$ and $\mathbf{a}_t$. Starting from period t with state $\mathbf{x}_t$, the expected total cost incurred until the end of the horizon is denoted by $f_t(\mathbf{x}_t)$. For convenience, the notation is summarized below.

$\mathbf{x}_t$:	beginning state vector for period t;
$\mathbf{a}_t$:	action vector for period t;
$\mathbf{w}_t$:	random vector, realized after the action is decided in period t;
$\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)$:	ending state vector for period t, which is also $\mathbf{x}_{t+1}$;
$G_t(\mathbf{x}_t, \mathbf{a}_t)$:	the expected cost for period t, when the starting state is $\mathbf{x}_t$ and the action decided is $\mathbf{a}_t$;

$f_t(\mathbf{x}_t)$: the expected total cost incurred, starting from period t with state $\mathbf{x}_t$, until the end of the horizon.

We assume that we are given an oracle, as part of the input, which computes functions G_t for $t = 1, 2, \ldots, T$. Note that to encode an oracle that outputs a positive integer-valued function f, we need at least $\Omega(\log f^{\max})$ space since this is the minimum space required to output the value $f^{\max}$, where $f^{\max}$ is the maximum value of f on its domain. The possible values of the random vector $\mathbf{w}_t$ and their corresponding rational probabilities are explicitly specified. In period t, the possible values of $\mathbf{w}_t$ are $\mathbf{w}_{t,1}, \mathbf{w}_{t,2}, \ldots, \mathbf{w}_{t,n_t}$. The corresponding probabilities are specified via positive integer numbers $q_{t,1}, q_{t,2}, \ldots, q_{t,n_t}$, with $\text{Prob}\{\mathbf{w}_t = \mathbf{w}_{t,i}\} = q_{t,i}/(\sum_{j=1}^{n_t} q_{t,j})$, for $i = 1, 2, \ldots, n_t$. We assume that the sequence of random variables $\{\mathbf{w}_t\}_{t=1}^{T}$ is independent. We define the following values (when appropriate, for every $t = 1, 2, \ldots, T$ and $i = 1, 2, \ldots, n_t$):

$p_{t,i} = \text{Prob}\{\mathbf{w}_t = \mathbf{w}_{t,i}\}$: Probability that $\mathbf{w}_t$ is realized as $\mathbf{w}_{t,i}$;

$w^{\max} = \max_t \max_i \|\mathbf{w}_{t,i}\|_\infty$: Maximum L_∞-norm of possible realizations of the random vector $\mathbf{w}_t$ over the entire horizon;

$n^{\max} = \max_t n_t$: Maximum value of n_t over the entire horizon;

$Q_t = \sum_{j=1}^{n_t} q_{t,j}$: A common denominator of all the probabilities in period t;

$M_t = \prod_{j=t}^{T} Q_j$: A common denominator of all the probabilities over periods $t, t+1, \ldots, T$;

$M_{T+1} = 1$.

The objective is to find the minimum expected total cost over the entire horizon, starting from any given initial state vector $\mathbf{x}_1$. We recall the DP recursion (1): For period $t = 1, 2, \ldots, T$, we have

$$f_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} \left[G_t(\mathbf{x}_t, \mathbf{a}_t) + \sum_{\mathbf{w}_t} f_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)) \right].$$

Our objective is to solve this problem. For all $t = 1, 2, \ldots, T$, define $\mathcal{D}_t \subseteq \mathbb{Z}^{k_1+k_2}$ as

$$\mathcal{D}_t = \{(\mathbf{x}_t, \mathbf{a}_t) : \mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)\}.$$

We make the following assumptions.

Assumption 1. *In* (1), *for all* $t = 1, 2, \ldots, T$, f_t *and* G_t *are* $L^\natural$*-convex and* $\mathcal{D}_t$ *is an* $L^\natural$*-convex set.*

Assumption 2. *The transformation* $\mathbb{T}$ *satisfies the following condition*: *If* $f(\mathbf{x})$ *is* $L^\natural$*-convex, then* $f(\mathbb{T}(\mathbf{x}, \mathbf{a}, \mathbf{w}))$ *is also* $L^\natural$*-convex in* $(\mathbf{x}, \mathbf{a})$ *for every integral* $\mathbf{w}$.

Let $g_t(\mathbf{x}_t, \mathbf{a}_t) = G_t(\mathbf{x}_t, \mathbf{a}_t) + \sum_{\mathbf{w}_t} f_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t))$. By Assumptions 1 and 2, we have that for all $t = 1, 2, \ldots, T$, g_t is $L^\natural$-convex. This is because addition and positive scaling preserve $L^\natural$-convexity; see Lemmas 4 and 5.

Assumption 3.[9] *For fixed positive integers* k_1, k_2, *and* k_3, $\mathbf{x}_t \in \mathbb{Z}_1, \mathbb{Z}_1^{k_1}, \mathbf{a}_t \in \mathbb{Z}_2, \mathbb{Z}_2^{k_2}, \mathbf{w}_t \in \mathbb{Z}_3, \mathbb{Z}_3^{k_3}$.

Assumption 4. *For* $t = 1, 2, \ldots, T$, *the value of* G_t *at any point in its domain is a positive rational number and can be evaluated in polynomial time in the size of the input using the corresponding oracle* (*which implies that* $\log G_t^{\max}$ *is polynomially bounded in the size of the input*).

Let t_G denote the time it takes for a single call of the oracle.

Next, we determine the minimum space required to specify the input of our problem. Let $\mathcal{A} = \{\mathcal{A}_u\}$, where $u = \arg\max_i |\mathcal{A}_i|$. To specify the constrained action set $\mathcal{A}_t(\mathbf{x}_t)$ for all periods, we need $\Omega(T \log |\mathcal{A}|)$ space. To specify the possible realizations of $\mathbf{w}_t$ over $t = 1, 2, \ldots, T$, we need $O(T n^* k_3 \log w^{\max})$ space. We assume that $O(1)$ space is needed to specify the transformation function $\mathbb{T}$. To specify the oracle that computes G_t for all $t = 1, 2, \ldots, T$, we require space $\Omega(T \log G^{\max})$, where $G^{\max} = \max_t G_t^{\max}$. Therefore, the overall input size is bounded *below* by

$$\Omega(T + \log |\mathcal{A}| + n^* + \log w^{\max} + \log G^{\max}).$$

We now present an explicit-enumeration algorithm, which provides an exact solution to the problem. Then we analyze its time complexity.

5.2. An Explicit-Enumeration Dynamic Programming Algorithm

Let $F_t(\mathbf{x}_t) = M_t f_t(\mathbf{x}_t)$ be the integer version of f_t. Algorithm 1 below is an explicit-enumeration dynamic program. Note that all the function values encountered in the algorithm are integers. Let $\mathcal{S} = \{\mathcal{X}_v\}$, where $v = \arg\max_t |\mathcal{X}_t|$. Consider the minimization in Step 4. From Assumption 1, we know that the function to be minimized is $L^\natural$-convex. Thus, the minimization step can be performed using recursive binary search (see Lemma 8). Consequently, Algorithm 1 has time complexity $O(t_G(\log(M_1 T G^{\max}) + \log |\mathcal{S}|) T |\mathcal{S}| \log |\mathcal{A}|)$ and space complexity $O((\log(M_1 T G^{\max}) + \log |\mathcal{S}|)|\mathcal{S}|)$, where $G^{\max} = \max_t G_t^{\max}$ is the maximum value of G_t over all feasible points and over all periods $t = 1, 2, \ldots, T$. Note that $O(\log(M_1 T G^{\max}))$ is the time required to store a function value in Step 3 and $O(\log |\mathcal{S}|)$ is the time required to store a point in $\mathcal{S}$. Observe that $|\mathcal{S}|$ typically depends on the input in a super-log manner. Thus, the algorithm is, both in time and space, pseudo-polynomial in the input size.

Algorithm 1 (An Explicit-Enumeration Dynamic Program)

1 Let $F_{T+1} = 0$;
2 **for** $t = T$ **to** 1 **do**
3 | For all points $\mathbf{x}_t$ in the domain, calculate and store
4 | $F_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} [M_t G_t(\mathbf{x}_t, \mathbf{a}_t) + Q_t \sum_{i=1}^{n_t} p_{t,i} \cdot F_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_{t,i}))]$;
5 | Delete the stored values $F_{t+1}(\mathbf{x}_{t+1})$ for all $\mathbf{x}_{t+1}$;
6 **end**
7 Return $F_1(\mathbf{x}_1)/M_1$.

It is easy to see that the problem studied in Halman et al. (2009)—the single-item stochastic inventory control problem with discrete demands—is a special case of our problem. They show that this problem is NP-hard, which implies that our problem is NP-hard as well. Also, the above pseudo-polynomial time algorithm implies that our problem is weakly NP-hard.

Note that, unless P = NP, no polynomial-time approximation algorithm exists with a fixed additive-error guarantee for our problem. Following an argument similar to the proof of Theorem 3.1 of Ausiello et al. (1999, p. 89), this statement can be proved by contradiction. Suppose that there exists a polynomial-time approximation algorithm A with an additive error δ for our problem. Then we can construct an exact algorithm that runs in polynomial time in the following manner: For our problem, for any instance, we can scale up all function values to obtain a new instance where all function values are multiples of $\lceil \delta \rceil + 1$. For the new instance, Algorithm A is an exact algorithm with polynomial running time. However, this implies membership in class P, which contradicts the NP-hardness of the problem unless P = NP.

In the next section, we will present our approximation scheme with an arbitrary additive error $\varepsilon > 0$, which runs in time pseudo-polynomial in the size of the input and polynomial in $1/\varepsilon$.

6. An Additive-Error Approximation Scheme

In this section, we propose and analyze an approximation scheme for the dynamic program (1). Section 6.1 provides an overview of this algorithm and an additional result that we will use in our analysis. Then §6.2 presents the details of the algorithm and its analysis. Subsequently, we discuss the corresponding near-optimal policy in §6.3. Finally, in §6.4, we summarize the highlights of our analysis.

6.1. Overview of the Scheme and Additional Developments

Recall the dynamic programming recursion (1): For period $t = 1, 2, \ldots, T$, we have

$$f_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} G_t(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{w}_t} f_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)),$$

where $\mathbf{x}_t \in L_1 \cap U_1 \cap \mathbb{Z}^{k_1}$, $\mathbf{a}_t \in L_2 \cap U_2 \cap \mathbb{Z}^{k_2}$, and $\mathbf{w}_t \in L_3 \cap U_3 \cap \mathbb{Z}^{k_3}$. The procedure we use is a modification of the explicit-enumeration algorithm. Specifically, we start from the last period T and, assuming $f_{T+1} = 0$, obtain f_T via (1). For period $T-1$, note that the calculation of f_{T-1} requires the value of f_T. Rather than storing f_T for all $\mathbf{x}_t$ (as in explicit enumeration), we approximate f_T using the convex ε-approximation $\check{f}_T$ (Definition 16). This results in the following approximation of f_{T-1}:

$$\tilde{f}_{T-1}(\mathbf{x}_{T-1}) = \min_{\mathbf{a}_{T-1} \in \mathcal{A}_{T-1}(\mathbf{x}_{T-1})} G_{T-1}(\mathbf{x}_{T-1}, \mathbf{a}_{T-1}) + \mathbb{E}_{\mathbf{w}_{T-1}} \check{f}_T(\mathbb{T}(\mathbf{x}_{T-1}, \mathbf{a}_{T-1}, \mathbf{w}_{T-1}))].$$

To continue using this approximation method for period $T-2$, it is desirable for $\tilde{f}_{T-1}$ to have the following two properties: (i) $\tilde{f}_{T-1}$ is $L^\natural$-convex (this will enable us to use our results in §4.2 to obtain $\check{f}_{T-1}$ as an approximation to $\tilde{f}^{10}_{T-1}$) and (ii) $\tilde{f}_{T-1}$ closely approximates f_{T-1}. Furthermore, if we can establish these properties for $\tilde{f}_t$, $t = 1, 2, \ldots, T-1$, then we can repeatedly approximate f_t for all $t = 1, 2, \ldots, T-1$ using the following recursion:

$$\tilde{f}_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} G_t(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{w}_t} \check{f}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t))], \quad (6)$$

where $\check{f}_t$ is the convex ε-approximation of $\tilde{f}_t$. Our next result establishes these properties.

Lemma 19. *For $t \in \{1, 2, \ldots, T-1\}$, consider functions f_t and f_{t+1} in (1). Suppose that $\check{f}_{t+1} : L \cap U \cap \mathbb{Z}^k \to \mathbb{R}^+$ is an ε-approximation of f_{t+1} and is an $L^\natural$-convex function. Then $\tilde{f}_t$ (defined as in (6) above) is also an $L^\natural$-convex function and is an ε-approximation of f_t.*

Proof. Using Lemma 12, it is easy to show that $\tilde{f}_t$ is an ε-approximation of f_t. Next, we prove that $\tilde{f}_t$ is $L^\natural$-convex. For notational convenience, denote

$$\check{g}_t(\mathbf{x}_t, \mathbf{a}_t) = G_t(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{w}_t} \check{f}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)). \quad (7)$$

We now have

$$\tilde{f}_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} \check{g}_t(\mathbf{x}_t, \mathbf{a}_t)]. \quad (8)$$

We know that $\check{f}_{t+1}(\mathbf{x}_{t+1})$ is $L^\natural$-convex. Next, we prove that $\check{g}_t(\mathbf{x}_t, \mathbf{a}_t)$ is $L^\natural$-convex. We can write $\mathbb{E}_{\mathbf{w}_t} \check{f}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t)) = \sum_{i=1}^{n_t} p_{t,i} \check{f}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_{t,i}))$. By Assumption 2, $\check{f}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_{t,i}))$ is $L^\natural$-convex in $(\mathbf{x}_t, \mathbf{a}_t)$ for all $\mathbf{w}_{t,i}$. Observe from (7) that $\check{g}_t(\mathbf{x}_t, \mathbf{a}_t)$ is a positive linear combination of $L^\natural$-convex functions and is, therefore, $L^\natural$-convex; see Lemmas 4 and 5. By Lemma 7, minimization over an $L^\natural$-convex set[11] preserves $L^\natural$-convexity. Therefore, we conclude that $\tilde{f}_t(\mathbf{x}_t)$ is again $L^\natural$-convex. The result follows. □

Next, we present our approximation algorithm formally.

6.2. Our Approximation Scheme

Our approximation algorithm is a modification of Algorithm 1. As in Algorithm 1, we work with the integer versions of the cost-to-go functions: $F_t(\mathbf{x}_t) = M_t f_t(\mathbf{x}_t)$. In each period $t = 1, 2, \ldots, T$, instead of storing F_t for every possible value of $\mathbf{x}_t$ in the domain, we construct $\check{F}_t$ as an approximation of F_t using less space.[12] Rewriting (6) by multiplying both sides of the equation by M_t, we have for $t = 1, 2, \ldots, T$,

$$\tilde{F}_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} M_t G_t(\mathbf{x}_t, \mathbf{a}_t) + \mathcal{Q}_t \mathbb{E}_{\mathbf{w}_t} \check{F}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_t))]. \quad (9)$$

We now present our approximation algorithm. Given an initial state vector $\mathbf{x}_1$, Algorithm 2 below approximates $f_1(\mathbf{x}_1)$ to within an arbitrary prespecified additive error $\varepsilon > 0$. Note that Step 3 in Algorithm 2 is only a formal step; i.e., it merely defines the function $\tilde{F}_t(\mathbf{x}_t)$ instead of computes it for all $\mathbf{x}_t$. The following result establishes the approximation guarantee of the algorithm and its time complexity. The corresponding policy is discussed in §6.3.

Algorithm 2 (Our DP-Based Approximation Scheme)

1 Let $\varepsilon' = \varepsilon/(k_1 T)$ and $\check{F}_{T+1}(\cdot) = 0$.
2 **for** $t = T$ **to** 1 **do**
3 Let $\tilde{F}_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in \mathcal{A}_t(\mathbf{x}_t)} [M_t G_t(\mathbf{x}_t, \mathbf{a}_t) + Q_t \sum_{i=1}^{n_t} p_{t,i} \cdot \check{F}_{t+1}(\mathbb{T}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{w}_{t,i}))]$.
4 Find and store the $\varepsilon' M_t$-coarse cube (see §4.2), denoted by D_t, of $\tilde{F}_t(\cdot)$.
5 Store all values of $\tilde{F}_t(\cdot)$ on D_t.
6 Let $\check{F}_t(\cdot)$ be the convex $\varepsilon' M_t$-approximation of $\tilde{F}_t(\cdot)$ (see Definition 16).
7 **end**
8 Return $\check{f}_1(\mathbf{x}_1) = \check{F}_1(\mathbf{x}_1)/M_1$.

Theorem 1 (Approximation Guarantee of Algorithm 2 and Its Running Time). *Given $\varepsilon > 0$ and an initial state $\mathbf{x}_1$, under Assumptions 1–4, the output of Algorithm 2, $\check{f}_1(\mathbf{x}_1)$, is an ε-approximation of $f_1(\mathbf{x}_1)$ in the additive sense. The worst-case running time of Algorithm 2 is pseudo-polynomial in the input size and polynomial in $1/\varepsilon$.*

Proof. We will first show that $\check{f}_1$ is a ε-approximation of f_1 in the additive sense and then analyze the running time of Algorithm 2. In the argument below, F_t, $t = 1, 2, \ldots, T$, is as defined in Algorithm 1 (§5.2).

• For the last period $t = T$, we have $\tilde{F}_T = F_T$. From Lemma 16, $\check{F}_T$ is a $k_1 \varepsilon' M_T$-approximation of F_t in the additive sense and is also $L^\natural$-convex.

• Next, consider period $T - 1$. Since $\check{F}_T$ is a $k_1 \varepsilon' M_T$-approximation of F_T in the additive sense and also $L^\natural$-convex, we conclude that $\check{f}_T$ is a $k_1 \varepsilon'$-approximation of f_T in the additive sense and $L^\natural$-convex. This is because by Lemma 4, scaling preserves $L^\natural$-convexity. From Lemma 19, we know that $\tilde{f}_{T-1}$ is a $k_1 \varepsilon'$-approximation of f_{T-1} in the additive sense and $L^\natural$-convex. This implies that $\tilde{F}_{T-1}$ is a $k_1 \varepsilon' M_{T-1}$-approximation of F_{T-1} in the additive sense and $L^\natural$-convex. Therefore, by Lemma 16, $\check{F}_{T-1}$ is a $2k_1 \varepsilon' M_{T-1}$-approximation of F_{T-1} and $L^\natural$-convex.

• Repeating this procedure until we get $\check{F}_1$, we reach the conclusion that $\check{F}_1$ is a $k_1 T \varepsilon' M_1$-approximation of F_1 in the additive sense. Since $\varepsilon' = \varepsilon/(k_1 T)$, we have $k_1 T \varepsilon' M_1 = \varepsilon M_1$. Therefore, $\check{F}_1$ is an εM_1-approximation of F_1 in the additive sense. Consequently, $\check{f}_1$ is an ε-approximation of f_1 in the additive sense. This concludes the proof of the guarantee offered by Algorithm 2.

We now analyze the running time of the algorithm. Following Halman et al. (2009), we denote the query time for a single point in the domain of a function g by t_g. Consider an arbitrary period t. In Step 3, we define the functions $\tilde{F}_t$. Note that (i) in Step 4, we need to query the value of $\check{F}_{t+1}$ and (ii) each query to $\tilde{F}_t$ is done by solving the DP Equation (9). Therefore, we now decide the time needed to solve the minimization problem (9). To solve this minimization problem, we use binary search on each dimension of $\mathbf{a}_t$ (see Lemma 8). Therefore, we need to calculate the right-hand side of (9) $O(k_2 \log |A|)$ times, with each calculation requiring time $O(t_G + n_t t_{\check{F}_{t+1}})$, where $t_{\check{F}_{t+1}}$ is the query time of $\check{F}_{t+1}$.

By Lemma 18, $t_{\check{F}_{t+1}} = O((k_1 + 1)(\log(U_1 - L_1)) + b)) + l$, where 2^l is the length of the ε-coarse cube D_{t+1} of $\tilde{F}_{t+1}$, b is the amount of storage space required to store a function value of $\tilde{F}_{t+1}$, and $b + l$ is the amount of storage space required to store a function value of $\check{F}_{t+1}$ (see the proof of Lemma 18). It is easy to see that l is bounded by $O(\log(U_1 - L_1))$ since the length of an ε-coarse cube is at most $U_1 - L_1$. The following claim bounds the amount of storage required for a function value of $\tilde{F}_t$, $t = 1, 2, \ldots, T$. Let $F^{\max}$ denote the maximum possible value of F_t over its domain over all possible $t = 1, 2, \ldots, T$.

Claim 1. *For period $t = 1, 2, \ldots, T$, the amount of storage space required to store a function value of $\tilde{F}_t$ is $O((T - t)(\log(U_1 - L_1)) + \log F^{\max})$.*

Proof. We start from the last period T. Since the function $\tilde{F}_T$ is integer valued, it takes space $O(\log F^{\max})$ to store a function value. For the period $T - 1$, the function $\tilde{F}_{T-1}$, in general, is not integer-valued anymore. This is because $\check{F}_T$ is constructed using linear interpolation. In other words, the amount of additional space required to store a function value of $\tilde{F}_{T-1}$ is the same as the amount of additional space that is required to store a function value of $\check{F}_T$ over that for $\tilde{F}_T$. From our discussion above, we know that the latter is bounded by $O(\log(U_1 - L_1))$. Therefore, we conclude that it takes space $O(\log(U_1 - L_1)) + \log F^{\max})$ to store a function value of $\tilde{F}_{T-1}$. For each additional period forward, it takes an additional $O(\log(U_1 - L_1))$ space to store the corresponding function value. □

Consequently, we have that for all $t = 1, 2, \ldots, T$, $t_{\check{F}_t} = O((k_1 + 2)(T \log(U_1 - L_1)) + \log F^{\max}))$.

Combining the arguments above, each query to $\tilde{F}_t$ takes time (omitting the constant term $O(k_1 + 2)$)

$$t_{\tilde{F}_t} = O(k_2 \log |\mathcal{A}|(t_G + n_t T \log(U_1 - L_1)) \log F^{\max}).$$

Next, we investigate the complexity of Steps 4 and 5. In Step 4, the time required to construct the $\varepsilon' M_t$-coarse cube (see Lemma 17) is

$$O\left((1 + 1/\varepsilon_{\tilde{F}_t})(U_1 - L_1)^{k_1} \left[\min\left\{1, \frac{s}{\varepsilon'}\right\}\right]^{k_1}\right).$$

Here, s satisfies the following condition: $s \times M_t$ is an upper bound on the absolute values of the slopes of the approximate cost-to-go functions $\tilde{F}_1|_E, \tilde{F}_2|_E, \ldots, \tilde{F}_T|_E$ for

all induced gridlines E of the domain cube $L_1, U_1^{k_1}$. The time to store the $\varepsilon' M_t$-coarse cube in Step 4 and the corresponding function values in Step 5 is

$$O\left(t_{\tilde{F}_t}(U_1-L_1)^{k_1}\left[\min\left\{1,\frac{s}{\varepsilon'}\right\}\right]^{k_1}(T\log(U_1-L_1)\log F^{\max})\right).$$

Note that the term $O(T\log(U_1-L_1)\log F^{\max})$ accrues from the need to store the k_1 coordinates of a point on the ε-cube and store the corresponding function value (to be precise, this term is $O(k_1\log(U_1-L_1)+T\log(U_1-L_1)\log F^{\max})$, but we only use the dominant factor).

Thus, for any period t, the running time of Algorithm 2 (omitting the constant terms) is

$$O\left(\log|\mathcal{A}|n^3T^2\log^2(U_1-L_1)\log^2 F^{\max}(U_1-L_1)^{k_1}\cdot\left[\min\left\{1,\frac{s}{\varepsilon'}\right\}\right]^{k_1}\right).$$

Observe that $O(\log F^{\max}) = O(\log(M_1TG^{\max}))$. The overall running time is

$$O\left(\log|\mathcal{A}|n^3T^2\log^2(U_1-L_1)\log^2(M_1TG^{\max})(U_1-L_1)^{k_1}\cdot\left[\min\left\{1,\frac{s}{\varepsilon'}\right\}\right]^{k_1}\right).$$

Since k_1 is fixed and $\varepsilon' = \varepsilon/(k_1T)$, the only nonpolynomial factors in the above expression, with respect to the binary size of the input and $1/\varepsilon$, are $O((U_1-L_1)^{k_1}\cdot/\min\{1,(sk_1T)/\varepsilon\}^{k_1})$. The result follows. □

6.3. A Corresponding Near-Optimal Policy

Theorem 1 only guarantees a near-optimal *value* for our problem. It is also important for our approximation to generate a corresponding near-optimal *policy*. We first define a policy and its expected cost and then present our result in Theorem 2. For our problem, a policy π is defined by a set of functions

$$\{a_t^{\pi}(\mathbf{x}):\mathbb{Z}^{k_1}\to\mathcal{A}_t\},\quad t=1,2,\ldots,T.$$

The expected cost of policy π is given by $f_1^{\pi}(\mathbf{x}_1)$, where

$$f_t^{\pi}(\mathbf{x}_t)=G_t(\mathbf{x}_t,a_t^{\pi}(\mathbf{x}_t))+\mathbb{E}_{\mathbf{w}_t}f_{t+1}^{\pi}(\mathbb{T}(\mathbf{x}_t,a_t^{\pi}(\mathbf{x}_t),\mathbf{w}_t))$$

for $t=1,2,\ldots,T$ and $f_{T+1}^{\pi}(\mathbf{x})=0, \forall\,\mathbf{x}\in L_1,U_1^{k_1}$.

Given $\varepsilon>0$, let π^{ε} denote the policy defined as follows:

$$\forall(t,\mathbf{x}_t),\ a_t^{\pi^{\varepsilon}}(\mathbf{x}_t)=\arg\min_{\mathbf{a}_t\in\mathcal{A}_t(\mathbf{x}_t)}\left[M_tG_t(\mathbf{x}_t,\mathbf{a}_t)+\mathbb{E}_{\mathbf{w}_t}\sum_{i=1}^{n_t}p_{ti}\cdot\check{F}_{t+1}(\mathbb{T}(\mathbf{x}_t,\mathbf{a}_t,\mathbf{w}_{ti}))\right],$$

where $\check{F}_t(\cdot)$, $t=1,2,\ldots,T+1$, are as defined in Algorithm 2. We have the following result. The proof is straightforward and, therefore, omitted.

Theorem 2 (ε-Approximate Policy). *Under Assumptions 1 through 4, the policy π^{ε} defined above is an ε-approximate policy for our problem. That is, for any initial state vector $\mathbf{x}_1$, $f_1^{\pi^{\varepsilon}}(\mathbf{x}_1)-f_1(\mathbf{x}_1)\le\varepsilon$.*

Note that by the definition of π^{ε}, we need to obtain $a_t^{\pi^{\varepsilon}}(\mathbf{x}_t)$ for every t and $\mathbf{x}_t$. If we choose to store $a_t^{\pi^{\varepsilon}}(\mathbf{x}_t)$ for all t and $\mathbf{x}_t$, then the space required is linear in $O(|\mathcal{S}|)$, which may be undesirable. Alternatively, for any given t and $\mathbf{x}_t$, we can query the corresponding $a_t^{\pi^{\varepsilon}}(\mathbf{x}_t)$ in the following way: After executing Algorithm 2, we can obtain $a_t^{\pi^{\varepsilon}}(\mathbf{x}_t)$ by solving

$$a_t^{\pi^{\varepsilon}}(\mathbf{x}_t)=\arg\min_{\mathbf{a}_t\in\mathcal{A}_t(\mathbf{x}_t)}\left[M_tG_t(\mathbf{x}_t,\mathbf{a}_t)+\mathbb{E}_{\mathbf{w}_t}\sum_{i=1}^{n_t}p_{ti}\cdot\check{F}_{t+1}(\mathbb{T}(\mathbf{x}_t,\mathbf{a}_t,\mathbf{w}_{ti}))\right]$$

in time $O(\log|\mathcal{A}|(t_G+n_t)T\log(U_1-L_1))\log(T\log(M_1G^{\max}))$.

6.4. Highlights of Our Analysis

Generalization of the approximation technique used in Halman et al., (2009, 2008, 2011, 2013)

Although the aim in Halman et al. (2009, 2008, 2011, 2013) is a relative-error approximation and our goal is an additive-error approximation, our approach can be viewed as a generalization of theirs, in the following sense:

1. Recall that in Halman et al. (2009, 2008, 2011) and (2013), the authors use the following idea to approximate a single-dimensional discrete convex function $f: L, U_{\mathbb{Z}}\to\mathbb{R}^+$:

(a) Select only a subset S of $L, U_{\mathbb{Z}}$ by using the concept of a K-approximation set.

(b) Let g be the piecewise-linear extension[13] of f, induced by S.

(c) Restrict g to $L, U_{\mathbb{Z}}$ to approximate[14] f.

2. Our approach to approximate a fixed-dimensional discrete $L^{\natural}$-convex function $f: L, U_{\mathbb{Z}}^{k_1}\to\mathbb{R}$ generalizes the above idea:

(a) Select only a subset S of $L, U_{\mathbb{Z}}^{k_1}$ by using the concept of an ε-coarse cube.

(b) Let g be the global extension (see Definition 4) of the function obtained by restricting f to S.

(c) Restrict g to $L, U_{\mathbb{Z}}^{k_1}$ to approximate f.

6.4.1. Properties of Algorithm 2 and Comparison with Algorithm 1

1. Algorithm 2 preserves the structure of the problem. That is, in each period, our approximation of the cost-to-go function is also $L^{\natural}$-convex, regardless of the choice of the error-guarantee ε.

2. Algorithm 2 can be viewed as a domain-reduction technique. Specifically, in each period, given the desired additive error ε, let 2^l be the length of the corresponding ε-coarse cube. Then, the total number of domain points is $(1+U-L)^{k_1}$, whereas the number of points on the ε-coarse cube is only $(1+(U-L)/2^l)^{k_1}$.

3. To achieve this reduction, the overhead that our algorithm incurs is the increase in the space required to store the function values of $\tilde{F}_t$ on the points of the ε-coarse cube (see Claim 1 on page 31). However, this increase is only a logarithmic factor. For example, if the lengths of the ε-coarse cubes for period $1, 2, \ldots, T$ are respectively $2^{l_1}, 2^{l_2}, \ldots, 2^{l_T}$, then to store a function value of $\tilde{F}_t$ for $t = 1, 2, \ldots, T$, it takes $\sum_{i=t+1}^{T} l_i$ additional bits compared to the space required by Algorithm 1.

4. The running time of Algorithm 2 is decided by items 2 and 3 above. For each period, on the one hand, item 2 allows us to reduce the number of points that we store and hence decreases the running time. On the other hand, for each point, item 3 increases the storage space required and hence increases the running time. However, the increase is logarithmic compared to the decrease.

6.4.2. Practical Algorithmic Framework. From the discussion above, Algorithm 2 can be viewed as a practical algorithmic framework that offers us full control in the trade-off between accuracy and running time. To this end, a possible use of Algorithm 2 may take the following steps:

1. For a given problem, we start with a relatively large *additive* error ε so that Algorithm 2 provides a quick solution. For the justification of considering a relatively large additive error ε, we refer the readers to item 3 below.

2. For any initial state $\mathbf{x}_1$, Algorithm 2 outputs $\check{f}_1(\mathbf{x}_1)$ such that $f_1(\mathbf{x}_1) \le \check{f}_1(\mathbf{x}_1) \le f_1(\mathbf{x}_1) + \varepsilon$, which implies $\check{f}_1(\mathbf{x}_1) - \varepsilon \le f_1(\mathbf{x}_1) \le \check{f}_1(\mathbf{x}_1)$. Thus, Algorithm 2 provides us both a upper bound of $\check{f}_1$ and a lower bound of $\check{f}_1 - \varepsilon$ of the original function value f_1.

3. The bounds in item 2 above imply that the relative error is $(\check{f}_1(\mathbf{x}_1) - f_1(\mathbf{x}_1))/f_1(\mathbf{x}_1) \le \varepsilon/(\check{f}_1(\mathbf{x}_1) - \varepsilon)$ for any initial state $\mathbf{x}_1$. In words, an additive-error guarantee of ε implies a relative-error guarantee of $\varepsilon/(\check{f}_1(\mathbf{x}_1) - \varepsilon)$. Therefore, choosing a large value of ε can still give us an acceptable relative error (assuming $\check{f}_1(\mathbf{x}_1)$ is large enough). This motivates the choice of a large additive error ε.

4. If the obtained bounds are unsatisfactory, then the additive error ε can be reduced. The running time will increase accordingly. This step can be repeated multiple times.

7. Application to an Inventory Problem

Our purpose in this section is to illustrate the use of our approximation scheme for the discrete version of the following well-known problem: the single-product stochastic inventory control problem with lost sales and lead times, studied, among others, in Morton (1969) and Zipkin (2008b). Hereafter, we will refer to this problem as INV-LS.

For INV-LS, Morton (1969) proves certain monotonicity properties of the optimal policy with respect to the state vector. Morton's proof involves an inductive analysis of the second-order derivatives of the cost-to-go functions. Recently, Zipkin (2008b) rederives Morton's results using the following technique: First, he transforms the state and action space to create a new dynamic program. He then shows that the cost-to-go functions of this new DP are $L^\natural$-convex. This implies certain monotonicity properties of the optimal policy in the new DP that, in turn, imply the desired properties of the optimal policy in the original DP. We will use Zipkin's results to show that INV-LS satisfies the assumptions of our analysis. Thus, Algorithm 2 can be used to solve this problem approximately to within a positive additive-error bound of ε (i.e., Theorems 1 and 2 are applicable).

In §7.1, we describe INV-LS. Then in §7.2, we show that the DP formulation corresponding to this problem satisfies our assumptions of §5.

7.1. Problem INV-LS

The objective is to minimize the expected sum of costs for a single-product inventory system over a finite horizon. Next, we describe the sequence of events in period $t = 1, 2, \ldots, T$:

1. The order that is due in period t arrives.

2. A new order (if any) is placed at a purchase cost of c per unit. This order will arrive after a constant lead time of $L \ge 2$ periods.

3. Demand is realized. Any demand that cannot be immediately satisfied is lost. If there is inventory left over at the end of the period, a unit holding cost of h is incurred. Any lost sales result in a penalty cost of p per unit.

For ease of exposition, we assume that the discount factor is 1. We assume that demands across periods are independent. We also assume that *all inventory levels, order quantities, cost parameters* (c, h, *and* p), *and all possible demand realizations are nonnegative integers*.

We now describe the formulation using our DP (1). Let q_t denote the order quantity in period t. Let y_t denote the inventory level after the order that is due in period t is delivered. The ordering cost incurred in period t is cq_t. Let $v_{0t} = y_t$, $v_{1t} = y_{t+1} - d_t, \ldots, v_{L-1,t} = y_{t-1} - d_{t-L+1} - \cdots - d_t$. Then the state vector in period t is $\mathbf{v}_t = (v_{0t}, v_{1t}, \ldots, v_{L-1,t})$. Zipkin (2008b) transforms the state and action space as follows: Let $\mathbf{x}_t = (x_{0t}, x_{1t}, \ldots, x_{L-1,t})$, where $x_{lt} = \sum_{k=l}^{L-1} v_{kt}$ for $0 \le l \le L-1$. Let w_t denote the random demand in period t for the single product. The possible realizations of w_t and the corresponding rational probabilities are explicitly specified in the same way as described in §5. Let $a_t = y_t - q_t$. It is easy to see that since the cost parameters are stationary, it is strictly suboptimal to order more than $w^\dagger$ in any period, where $w^\dagger = \max_{1 \le t \le T} \max_{1 \le i \le n_t} w_{t,i}$. Thus, we let the feasible set of a_t be $\mathcal{A}_t = \{-w^\dagger, \ldots, 0\} \subset \mathbb{Z}$. The transformation function $\mathbb{T}$ from period t to $t+1$ is given by

$$\mathbf{x}_{t+1} = (x_{0t} - x_{1t} - w_t)^+ + (x_{1t} - d_t, x_{2t} - d_t, \ldots, x_{L-1,t} - d_t, -a_t).$$

For our purpose in this paper, we use the same DP formulation as that in the proof of Theorem 4 in Zipkin

(2008b). Specifically, for period t, we use two additional integer decision variables s_t and r_t that denote the selling decision amount and the remaining inventory after sale, respectively. Note that the decisions s_t and r_t are contingent upon the realization of demand w_t.

Assume that the last period in which ordering is allowed is T; i.e., $\mathcal{A}_{T+1} = \mathcal{A}_{T+2} = \cdots = \mathcal{A}_{T+L} = \{0\}$. Also, assume that holding and penalty costs continue to accrue until period $T+L$. By convention, let $f_{T+L+1}(\cdot) = 0$. The expected cost $f_t(\mathbf{x}_t)$, incurred from period $t = 1, 2, \ldots, T+L$ until the end of the horizon $T+L$, is given by the following DP:

$$f_t(\mathbf{x}_t) = \min_{a_t \in \mathcal{A}_t} \{ ca_t + \mathbb{E}_{w_t}[u_t(\mathbf{x}_t, a_t, w_t)] \} \tag{10}$$

where

$$u_t(\mathbf{x}_t, a_t, w_t) = \min_{\substack{s_t, r_t \in \mathbb{Z} \\ s_t + r_t = x_{0t} - x_{1t} \\ 0 \le s_t \le w_t,\ r_t \ge 0}} \{ hr_t + p(w_t - s_t) + f_{t+1}(x_{1t} + r_t, x_{2t}, \ldots, x_{L-1,t}, D) \} - ca_t \mathbf{1}_{\{a_t > 0\}}.$$

Using $s_t + r_t = x_{0t} - x_{1t}$ to remove s_t, we obtain

$$u_t(\mathbf{x}_t, a_t, w_t) = \min_{\substack{r_t \in \mathbb{Z}, r_t \ge 0, \\ 0 \le x_{0t} - x_{1t} - r_t \le w_t}} \{ hr_t + p(w_t + r_t + x_{1t} - x_{0t}) + f_{t+1}(x_{1t} + r_t, x_{2t}, \ldots, x_{L-1,t}, D) \} - ca_t \mathbf{1}_{\{a_t > 0\}}.$$

Letting $x_{+t} = x_{1t} + r_t$, we have

$$u_t(\mathbf{x}_t, a_t, w_t) = \min_{\substack{x_{+t} \in \mathbb{Z}, x_{+t} \ge x_{1t} \\ 0 \le x_{0t} - x_{+t} \le w_t}} \{ h(x_{+t} - x_{1t}) + p(w_t + x_{+t} - x_{0t}) + f_{t+1}(x_{+t}, x_{2t}, \ldots, x_{L-1,t}, 0) \} - ca_t \mathbf{1}_{\{a_t > 0\}}.$$

Observe that in the above equation, the feasible set of x_{+t} is *dependent* on the demand w_t. However, recall that in our DP (1), the feasible set of actions is *independent* of the demand. Thus, to move toward the form (1), we need some additional notation.

For a possible realization $w_{t,j}$ of w_t, let x^i_{+t} denote the corresponding x_{+t}, $i = 1, 2, \ldots, n_t$. Let the feasible set of x^i_{+t} be

$$\mathcal{C}^i(\mathbf{x}_t) = \{x_{+t} \in \mathbb{Z} : x_{+t} \ge x_{1t},\ x_{+t} \le x_{0t} - s_t,\ s_t \le w_{t,j}\}.$$

Define $\tilde{\mathbf{a}}_t = (a_t, x^1_{+t}, x^2_{+t}, \ldots, x^{n_t}_{+t})$. Then the feasible set of $\tilde{\mathbf{a}}_t$ is

$$\tilde{\mathcal{A}}_t(\mathbf{x}_t) = \mathcal{A}_t \times \mathcal{C}^1(\mathbf{x}_t) \times \mathcal{C}^2(\mathbf{x}_t) \times \cdots \times \mathcal{C}^{n_t}(\mathbf{x}_t).$$

Let

$$G_t(\mathbf{x}_t, \tilde{\mathbf{a}}_t) = -ca_t + \sum_{i=1}^{n_t} p_{t,j} \{ h(x^i_{+t} - x_{1t}) + p(w_{t,j} + x^i_{+t} - x_{0t}) \},$$

where $p_{t,j}$ is the probability that the random variable w_t realizes as $w_{t,j}$, $i = 1, 2, \ldots, n_t$, as specified in §5.

Now (10) can be rewritten as

$$f_t(\mathbf{x}_t) = \min_{\tilde{\mathbf{a}}_t \in \tilde{\mathcal{A}}_t(\mathbf{x}_t)} \{ G_t(\mathbf{x}_t, \tilde{\mathbf{a}}_t) + \mathbb{E}_{w_t} f_{t+1}(\mathbb{T}(\mathbf{x}_t, \tilde{\mathbf{a}}_t, w_t)) \}, \tag{11}$$

where $\mathbb{T}(\mathbf{x}_t, \tilde{\mathbf{a}}_t, w_t) = (x^i_{+t}, x_{2t}, \ldots, x_{L-1,t}, D) - ca_t \mathbf{1}_{\{a_t > 0\}}$ if $w_t = w_{t,j}$ for $i = 1, 2, \ldots, n_t$.

7.2. Application of the Approximation Algorithm to INV-LS

To establish the applicability of our developments in §§4 and 5, we show that the results of Theorems 1 and 2, which characterize the running time and accuracy of Algorithm 2, hold for INV-LS. To this end, Theorem 3 below verifies that INV-LS satisfies Assumptions 1, 2, and 4, which were used to prove Theorems 1 and 2. Although Assumption 3 is not directly satisfied, we will discuss a simple modification in Algorithm 2 to make it applicable to INV-LS and ensure that the results of Theorems 1 and 2 hold.

THEOREM 3. *The DP formulation* (11) *of INV-LS satisfies Assumptions* 1, 2, *and* 4.

PROOF. First, we will show that Assumption 1 is satisfied. That is, for all t, $f_t(\mathbf{x}_t)$ and $G_t(\mathbf{x}_t, \tilde{\mathbf{a}}_t)$ are $L^\natural$-convex and the set $\tilde{\mathcal{A}}_t = \{(\mathbf{x}_t, \tilde{\mathbf{a}}_t) : \tilde{\mathbf{a}}_t \in \tilde{\mathcal{A}}_t(\mathbf{x}_t)\}$ is $L^\natural$-convex. Note that Zipkin (2008b) has already shown that when demand, states, and actions are all integer-valued, the corresponding cost-to-go function f_t is $L^\natural$-convex; see (Zipkin 2008b, p. 940). Also, since $G_t(\mathbf{x}_t, \tilde{\mathbf{a}}_t)$ is linear, it is also $L^\natural$-convex. From Definition 9, it is easy to see that $\tilde{\mathcal{A}}_t$ is $L^\natural$-convex. We conclude that INV-LS satisfies Assumption 1.

Second, consider Assumption 2; i.e., the transformation $\mathbb{T}$ satisfies the following condition: if $f(\mathbf{x}_t)$ is $L^\natural$-convex, then $f(\mathbb{T}(\mathbf{x}_t, \tilde{\mathbf{a}}_t, w_t))$ is also $L^\natural$-convex in $(\mathbf{x}_t, \tilde{\mathbf{a}}_t)$, for every integral w_t. For $i = 1, 2, \ldots, n_t$, if $w_t = w_{t,j}$, we have that $\mathbb{T}(\mathbf{x}_t, \tilde{\mathbf{a}}_t, w_t) = (x^i_{+t}, x_{2t}, \ldots, x_{L-1,t}, D) - ca_t \mathbf{1}_{\{a_t > 0\}}$. Therefore, it is sufficient to prove the following claim. The proof of this claim is essentially due to Zipkin (2008b), with a few additional clarifications.

CLAIM 2. *If* $f(\mathbf{x}_t)$ *is* $L^\natural$*-convex, then for every* $i = 1, 2, \ldots, n_t$, $f(x^i_{+t}, x_{2t}, \ldots, x_{L-1,t}, D) - ca_t \mathbf{1}_{\{a_t > 0\}}$ *is* $L^\natural$*-convex in* $(\mathbf{x}_t, \tilde{\mathbf{a}}_t)$.

PROOF OF CLAIM 2. Since f is $L^\natural$-convex, by Definition 8, we know that the function $\psi(x^i_{+t}, x_{2t}, x_{3t}, \ldots, x_{L-1,t}, u, a_t) = f(x^i_{+t}, x_{2t}, x_{3t}, \ldots, x_{L-1,t}, u) - ca_t \mathbf{1}_{\{a_t > 0\}}$ (u is a dummy variable) is submodular in $(x^i_{+t}, x_{2t}, x_{3t}, \ldots, x_{L-1,t}, u)$. Thus, $f(x^i_{+t}, x_{2t}, x_{3t}, \ldots, x_{L-1,t}, u) - ca_t \mathbf{1}_{\{a_t > 0\}}$ is submodular in $(\mathbf{x}_t, \tilde{\mathbf{a}}_t, u)$. Consequently, by Lemma 1 of Zipkin (2008b), we know that $f(x^i_{+t}, x_{2t}, \ldots, x_{L-1,t}, u) - ca_t \mathbf{1}_{\{a_t > 0\}}$ is $L^\natural$-convex in $(\mathbf{x}_t, \tilde{\mathbf{a}}_t, u)$. By restricting $u = 0$, from item 5 of Theorem 6.19 of Murota (2007), we conclude that $f(x^i_{+t}, x_{2t}, \ldots, x_{L-1,t}, D) - ca_t \mathbf{1}_{\{a_t > 0\}}$ is $L^\natural$-convex in $(\mathbf{x}_t, \tilde{\mathbf{a}}_t)$. □

Finally, consider the closed-form expression of G_t:

$$G_t(\mathbf{x}_t, \tilde{\mathbf{a}}_t) = -ca_t + \sum_{i=1}^{n_t} p_{t,j} \{ h(x^i_{+t} - x_{1t}) + p(w_{t,j} + x^i_{+t} - x_{0t}) \}.$$

Since the parameters are either nonnegative integers or rational numbers, we conclude that $G_t(\mathbf{x}_t, \tilde{\mathbf{a}}_t)$ is a rational number. Assumption 4 is satisfied after proper scaling of the DP recursion (11). □

In Assumption 3 of §5, we have assumed, for ease of exposition, that (i) the dimensionality of $\mathbf{a}_t$ is fixed and (ii) $\mathbf{x}_t$ belongs to some hypercube $[L_1, U_1]^{k_1} \cap \mathbb{Z}^{k_1}$. However, for INV-LS, these assumptions are too restrictive. Specifically, (i) observe that the dimensionality of $\tilde{\mathbf{a}}_t$ in DP (11) depends on the number of different realizations of w_t, which is part of the input. Since this dependence is only a linear dependence, it does not affect the running time of our approximation algorithm. (ii) The feasible set of $\mathbf{x}_t$ in (11) is not a rectangular set. We now discuss simple modifications to Algorithm 2 so that it can be applied to INV-LS.

Recall that $w^{\max}$ denotes the maximum possible demand over all periods. We assume that the starting vector $\mathbf{v}_1 = (v_{10}, v_{11}, \ldots, v_{1,L-1}) \in \mathbb{Z}^L$ satisfies

$$\begin{array}{lll} 0 & v_{10} & 2^{\lceil \log(w^{\max}) \rceil} \\ 0 & v_{11} & 2^{\lceil \log(w^{\max}) \rceil} \\ \vdots & & \\ 0 & v_{1,L-1} & 2^{\lceil \log(w^{\max}) \rceil}. \end{array}$$

Recall that the order quantity $q_t = a_t \le w^{\max}$. This, together with the above assumption on $\mathbf{v}_1$, implies that the untransformed state vector $\mathbf{v}_t = (y_t, q_{t+1-L}, \ldots, q_{t-1}) \in \mathbb{Z}^L$ satisfies the following:

$$\begin{array}{lll} 0 & y_t & t2^{\lceil \log(w^{\max}) \rceil} \\ 0 & q_{t+1-L} & 2^{\lceil \log(w^{\max}) \rceil} \\ \vdots & & \\ 0 & q_{t-1} & 2^{\lceil \log(w^{\max}) \rceil}. \end{array}$$

Denote the set of all possible $\mathbf{v}_t$ that satisfy the above inequalities by V_t.

In terms of the transformed state $\mathbf{x}_t = (x_{0t}, x_{1t}, \ldots, x_{L-1,t}) \in \mathbb{Z}^L$, these constraints become

$$\begin{array}{lll} 0 & x_{0t} - x_{1t} & t2^{\lceil \log(w^{\max}) \rceil} \\ 0 & x_{1t} - x_{2t} & 2^{\lceil \log(w^{\max}) \rceil} \\ \vdots & & \\ 0 & x_{L-1,t} & 2^{\lceil \log(w^{\max}) \rceil}. \end{array}$$

Denote the set of all possible $\mathbf{x}_t$ that satisfy the above inequalities by X_t. Note that there is a linear correspondence between X_t and V_t, represented by

$$X_t = \tau \cdot V_t$$

and

$$V_t = \tau^{-1} \cdot X_t,$$

where τ is the following $L \times L$ matrix:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 0 & 1 & \cdots & 1 & 1 \\ 0 & 0 & \ddots & 1 & 1 \\ 0 & \cdots & 0 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

The main obstacle in applying Algorithm 2 to INV-LS is the following: since we are working with the transformed state $\mathbf{x}_t$, its domain X_t is no longer a rectangular set. Therefore, we cannot construct the ε-coarse cube of the cost-to-go functions directly using the definition in §4.2. This difficulty is resolved as follows. We construct the ε-coarse cube on the untransformed state (we can do this since the corresponding domain is a rectangular set) and then use the transformation τ to transform the ε-coarse cube.

Formally, for all $t = 1, 2, \ldots, T$, let $\ddot{F}_t(\mathbf{v}_t) = F_t(\tau^{-1}\mathbf{x}_t)$, where F_t is the integer version of f_t in (11). We construct the ε-coarse cube of $\ddot{F}_t$: $V_t \to \mathbb{R}^+$ as follows. Let $E_1, E_2, \ldots, E_{L2^{L-1}}$ denote the set of points on the edges of V_t. Let the restriction of $\ddot{F}_t$ to $E_1, E_2, \ldots, E_{L2^{L-1}}$ be $\ddot{F}_{t,E_1}, \ddot{F}_{t,E_2}, \ldots, \ddot{F}_{t,E_{L2^{L-1}}}$, respectively. Since the function F_t is $L^\natural$-convex and the transformation τ is linear, it is easy to see that the single-dimensional functions $\ddot{F}_{t,E_1}, \ddot{F}_{t,E_2}, \ldots, \ddot{F}_{t,E_{L2^{L-1}}}$ are convex.

Next, we construct the canonical uniform-interval ε-approximation sets of the functions $\ddot{F}_{t,E_1}, \ddot{F}_{t,E_2}, \ldots, \ddot{F}_{t,E_{L2^{L-1}}}$. Let $l_1, l_2, \ldots, l_{L2^{L-1}}$ be the lengths of the intervals in the canonical uniform-interval ε-approximation sets of the functions $\ddot{F}_{t,E_1}, \ddot{F}_{t,E_2}, \ldots, \ddot{F}_{t,E_{L2^{L-1}}}$, respectively. Let $l = \min\{l_1, l_2, \ldots, l_{L2^{L-1}}\}$. Let $S_{1t} = \{0, l, 2l, \ldots, t2^{\lceil \log(w^{\max}) \rceil}\}$ and $S_{2t} = \{0, l, 2l, \ldots, 2^{\lceil \log(w^{\max}) \rceil}\}$. For a given ε, we refer to the set $D_t = S_{1t} \times S_{2t}^{L-1}$ as the ε-coarse cube of $\ddot{F}_t$. Then τD_t is defined to be the ε-coarse cube for $F_t(\mathbf{x}_t)$. With this definition of the ε-coarse cube, it is not hard to check that our approximation results in §4.2 continue to hold.

We can now apply Algorithm 2 to INV-LS. The following result notes the additive-error guarantee and the running time of Algorithm 2 for INV-LS.

Corollary 1. *Given $\varepsilon > 0$, under Assumptions 1 through 4, Algorithm 2 solves INV-LS to within an additive error of ε. The corresponding running time is pseudo-polynomial in the binary size of the input and polynomial in $1/\varepsilon$. Furthermore, the corresponding ε-approximate policy can also be obtained (see §6.3).*

8. Computational Experiments

In this section, we report computational results on instances of the INV-LS problem to demonstrate the practical value of Algorithm 2. Section 8.1 describes the generation of the instances and §8.2 reports the results.

8.1. The Test Bed

Our test bed consists of instances of Problem INV-LS with lead time $L = 2$. The other parameters are set as follows:

1. Without loss of generality, the ordering cost in each period is set to 0 (Janakiraman and Muckstadt 2004).

2. We consider two sets of holding and penalty costs. In the first set, the unit holding cost h is \$1 per period and the unit penalty cost p for lost sales is \$1. In the second set, these costs are \$1 and \$4, respectively.

3. The demands across the periods are independent and identically distributed. We use two distributions: a uniform distribution with support $\{0, 1, \ldots, 1{,}024\} \subset \mathbb{Z}$ and a symmetric triangular distribution with the same support.

For the four combinations of the problem parameters (namely, the two choices of holding/penalty costs and the two choices of the demand distribution), we generate 10 instances for each combination by varying the horizon length $T = 20, 40, \ldots, 200$ for a total of $4 \times 10 = 40$ instances. Note that for a horizon length of T, we let the costs accrue until period $T + L = T + 2$.

To generate informed choices of the additive-error guarantee ε that is required by Algorithm 2, we first obtain a lower bound on the optimal expected cost starting from any initial state $\mathbf{v}_1$; i.e., $f_1(\mathbf{v}_1) \geq LB$, for all $\mathbf{v}_1$. This lower bound, referred to here as LB, is calculated as follows:

Let

$$l = \min_{q \in \{0, 1, \ldots, 1{,}024\} \subset \mathbb{Z}} \left[h \left(\sum_{d=0}^{q-1} P[\text{Demand} = d](q - d) \right) + p \left(\sum_{d=q}^{1{,}024} P[\text{Demand} = d](d - q) \right) \right],$$

where $P[\text{Demand} = d] = 1/1{,}025$ for the uniform distribution, and $P[\text{Demand} = d] = \min\{d, 1{,}024 - d\}/(512 \times 513)$ for the symmetric triangular distribution. Notice that l is a lower bound on the expected cost achievable in any period t. Therefore,

$$LB = (T + L) \times l$$

is a lower bound on the optimal total expected cost over the entire horizon, over all feasible policies.

Using the lower bound LB, we consider three choices of the additive-error guarantee ε as follows: $\varepsilon = \epsilon \times LB$, where $\epsilon \in \{0.01, 0.02, 0.05\}$. Note that an additive guarantee of $\varepsilon = \epsilon \times LB$ implies a relative-error guarantee of ϵ. Recall from the proof of Theorem 1 that the time complexity of Algorithm 2 increases with the value of the ratio s/ε', where $\varepsilon' = \varepsilon/(2(T + L))$. In the computational results presented in the next section, we also report an estimated value of this ratio to understand its actual effect on the computational time of Algorithm 2.

8.2. Results

Algorithms 1 and 2 were implemented in C under Linux.[15] The computations were carried out on a computer with an Intel i7-870 processor and 4GB of memory. For each of the 40 instances in our test bed, Tables 1 through 4 report (i) the CPU time taken by Algorithm 1, (ii) the CPU times required by Algorithm 2 and the solution quality for the three choices of the additive error, and (iii) the ratio s/ε'. To speed up the computation, we use a state-space reduction technique due to Zipkin (2008a) in both Algorithms 1 and 2.

We now explain the entries of each of these tables. The first column indicates the horizon length T. The second column shows the CPU time (in seconds) for Algorithm 1. In the third, fourth, and fifth columns, we report the time, cost performance, and the ratio s/ε' of Algorithm 2 when the guaranteed upper bound ε on the additive error equals $0.01 \times LB$. The third column shows the CPU time (in seconds). The fourth column shows the maximum (over all initial states) actual percentage error, i.e.,

$$\max_{\mathbf{v}_1} \frac{[\breve{f}_1(\mathbf{v}_1) - f_1(\mathbf{v}_1)] \times 100}{f_1(\mathbf{v}_1)}\%,$$

where the maximum is taken over all possible initial states $\mathbf{v}_1$ in Zipkin's reduced state space. Note that the calculation of the maximum actual percentage error uses the result of Algorithm 1. The fifth column shows the ratio s/ε'. The remaining columns of each table are organized in a similar fashion.

Tables 1 through 4 demonstrate that Algorithm 2 is indeed of practical value. For example, in Table 1, when $T = 20$ and $\varepsilon = 0.05 \times LB$, Algorithm 2 runs in three minutes, whereas Algorithm 1 takes approximately three hours. Note that as the guaranteed additive error increases from $0.01 \times LB$ to $0.05 \times LB$, the ε-coarse cubes used by Algorithm 2 become coarser, resulting in a significant reduction of computing time.

The effect of s/ε'

- (Changing ε) We refer the readers to Tables 1 through 4. For a fixed value of T, as the imposed additive error ε reduces (and therefore $\varepsilon' = \varepsilon/(2(T + L))$ reduces), the ratio s/ε' increases and therefore the time taken by Algorithm 2 increases. However, even for an additive error of $\varepsilon = 0.01 \times LB$, this time is significantly below that of Algorithm 1.
- (Changing T) We refer the readers to Tables 1 through 4. For a fixed value of the relative error ϵ (i.e., additive error of $\epsilon \times LB$), as T increases, we find that the ratio s/ε' increases slightly and then remains stable. Correspondingly, we find that the ratio of the running time of Algorithm 2 over that of Algorithm 1 also remains stable.
- (Changing the unit penalty cost p): Table 5 shows the effect of increasing the unit penalty cost p which, in turn, has the overall effect of increasing the ratio s/ε'. In this table, we see that the ratio of the time taken by Algorithm 2 relative to that taken by Algorithm 1 indeed increases. In particular, there is a significant jump in this ratio when p changes from 16 to 32. This sudden change is because Algorithm 2 shifts to a finer discretization (i.e., the lengths of the coarse cubes reduce).
- We have also investigated other situations where the ratio s/ε' changes. Overall, we observe that an increase in this ratio is usually accompanied by an increase in the ratio of the time taken by Algorithm 2 to that of Algorithm 1. However, for reasonable parameter ranges, we find that Algorithm 2 continues to exhibit significant computational time savings over Algorithm 1, even at a relative error guarantee of 1%.

Table 1. Experimental results for $p=3$, $h=3$, and uniform distribution.

		Performance of Algorithm 2								
		$\varepsilon=0.01\times LB$			$\varepsilon=0.02\times LB$			$\varepsilon=0.05\times LB$		
T	Algorithm 1 time	Time	% error	s/ε'	Time	% error	s/ε'	Time	% error	s/ε'
2	1,499.8	97.7	0.0343	3.0545	27.0	0.0900	1.5261	20.6	0.2077	0.6104
4	2,472.9	160.0	0.0257	3.7815	43.6	0.0825	1.8862	36.6	0.1487	0.7544
6	3,677.1	220.9	0.0235	3.9457	59.7	0.0784	1.9670	53.6	0.1268	0.7868
8	4,727.1	280.8	0.0221	3.9685	76.3	0.0761	1.9782	70.8	0.1143	0.7913
10	5,253.1	347.9	0.0215	3.9706	94.4	0.0745	1.9792	88.0	0.1064	0.7917
12	6,687.3	409.2	0.0210	3.9707	111.2	0.0735	1.9793	105.1	0.1009	0.7917
14	7,611.8	467.9	0.0207	3.9707	127.2	0.0728	1.9793	120.6	0.0967	0.7917
16	8,729.9	536.6	0.0204	3.9707	145.0	0.0722	1.9793	138.2	0.0935	0.7917
18	9,839.1	589.4	0.0202	3.9707	160.0	0.0717	1.9793	153.7	0.0909	0.7917
20	10,885.4	651.6	0.0200	3.9707	176.6	0.0713	1.9793	170.5	0.0888	0.7917

Table 2. Experimental results for $p=3$, $h=3$, and triangular distribution.

		Performance of Algorithm 2								
		$\varepsilon=0.01\times LB$			$\varepsilon=0.02\times LB$			$\varepsilon=0.05\times LB$		
T	Algorithm 1 time	Time	% error	s/ε'	Time	% error	s/ε'	Time	% error	s/ε'
2	1,727.4	213.1	0.0409	4.6590	104.3	0.0534	2.3292	28.6	0.2140	0.9315
4	2,442.3	493.7	0.0268	5.8735	170.3	0.0478	2.9331	46.6	0.1787	1.1702
6	3,461.5	765.6	0.0202	6.0364	234.5	0.0447	3.0137	63.9	0.1764	1.2019
8	4,449.0	1,031.5	0.0180	6.0472	299.8	0.0425	3.0190	81.6	0.1741	1.2039
10	5,857.5	1,305.5	0.0166	6.0475	369.4	0.0409	3.0192	100.7	0.1732	1.2040
12	6,888.0	1,594.4	0.0156	6.0476	430.5	0.0397	3.0192	117.3	0.1725	1.2040
14	7,631.0	1,851.5	0.0148	6.0476	495.9	0.0388	3.0192	135.0	0.1720	1.2040
16	8,540.7	2,120.5	0.0143	6.0475	569.5	0.0381	3.0192	154.9	0.1716	1.2040
18	9,330.6	2,486.5	0.0138	6.0475	639.9	0.0376	3.0192	172.6	0.1713	1.2040
20	10,098.4	2,703.1	0.0134	6.0475	691.5	0.0371	3.0192	190.8	0.1710	1.2040

Table 3. Experimental results for $h=3$, $p=4$, and uniform distribution.

		Performance of Algorithm 2								
		$\varepsilon=0.01\times LB$			$\varepsilon=0.02\times LB$			$\varepsilon=0.05\times LB$		
T	Algorithm 1 time	Time	% error	s/ε'	Time	% error	s/ε'	Time	% error	s/ε'
2	1,538.2	95.0	0.0225	1.9470	26.0	0.0891	0.9714	25.7	0.1240	0.3885
4	1,846.2	157.5	0.0206	2.1333	43.0	0.0815	1.0632	42.7	0.1033	0.4252
6	2,779.5	218.5	0.0193	2.1816	59.5	0.0773	1.0868	59.2	0.0937	0.4347
8	3,241.0	276.5	0.0188	2.1860	75.1	0.0751	1.0890	75.0	0.0879	0.4356
10	3,375.8	337.2	0.0184	2.1863	91.6	0.0734	1.0891	91.3	0.0841	0.4356
12	3,547.7	397.6	0.0181	2.1863	107.9	0.0723	1.0891	107.6	0.0814	0.4357
14	4,746.9	458.1	0.0179	2.1863	124.3	0.0715	1.0892	124.1	0.0794	0.4356
16	7,869.5	525.1	0.0177	2.1863	140.6	0.0708	1.0891	140.5	0.0779	0.4357
18	8,126.6	579.2	0.0176	2.1863	159.0	0.0703	1.0892	158.8	0.0766	0.4356
20	8,816.7	647.2	0.0174	2.1863	175.5	0.0698	1.0891	175.3	0.0756	0.4356

9. Future Research Direction

The following question is open: Does an FPTAS exist for our problem under Assumptions 1 through 4?

Related to this question, in Halman et al. (2011), the authors prove that if f_t is two-dimensional and convex in the sense of Miller (1971), then a generalization of the K-approximation technique is unlikely to succeed, in a way such that the amount of information stored is still polynomial in the size of the binary input and $1/\varepsilon$. The following two comments are relevant. First, since Miller's notion of discrete convexity is more general than $L^\natural$-convexity, we do not know whether the above result holds for two-dimensional $L^\natural$-convex functions. Our efforts to answer this question using the counting argument in Halman et al. (2011) were unsuccessful. Second, even if we can show that the specific technique of a K-approximation is unlikely to succeed, we cannot rule out the possibility of deriv-

Table 4. Experimental results for $h=1$, $p=4$, and triangular distribution.

		Performance of Algorithm 2								
		$\varepsilon=0.01\times LB$			$\varepsilon=0.02\times LB$			$\varepsilon=0.05\times LB$		
T	Algorithm 1 time	Time	% error	s/ε'	Time	% error	s/ε'	Time	% error	s/ε'
2	1,705.1	105.2	0.0532	2.3881	103.1	0.0532	1.1940	28.5	0.2100	0.4776
4	2,726.1	169.6	0.0476	2.7774	166.8	0.0476	1.3887	46.2	0.1905	0.5538
6	3,658.1	233.7	0.0443	2.8247	233.9	0.0443	1.4124	64.0	0.1810	0.5631
8	4,437.0	299.2	0.0424	2.8274	299.3	0.0424	1.4137	81.7	0.1744	0.5636
10	5,297.1	364.9	0.0413	2.8275	360.2	0.0413	1.4138	99.8	0.1698	0.5636
12	6,742.7	430.7	0.0404	2.8275	429.4	0.0404	1.4137	117.7	0.1665	0.5636
14	7,840.2	495.9	0.0398	2.8275	495.1	0.0398	1.4138	137.3	0.1641	0.5636
16	9,049.4	561.2	0.0394	2.8275	553.5	0.0394	1.4138	154.0	0.1622	0.5636
18	9,994.8	628.4	0.0390	2.8275	618.1	0.0390	1.4138	172.3	0.1607	0.5636
20	11,186.7	694.7	0.0387	2.8275	690.8	0.0387	1.4138	189.2	0.1594	0.5636

Table 5. Experimental results for $h=1$, $T=2$, $\varepsilon=0.01\times LB$, and uniform distribution.

		Performance of Algorithm 2				
p	Algorithm 1 time	Time	% error	s/ε'	s	Lengths of the coarse cubes for periods 3, 2, 1, 0
2	1,384.73	97.26	0.032402	2.2656	3.87	32 16 16 16
4	1,338.25	98.03	0.022459	1.9470	3.99	16 16 16 16
8	1,249.25	97.78	0.025333	3.5083	7.99	16 16 16 16
16	1,293.11	97.73	0.032359	6.6302	15.99	16 16 16 16
32	1,259.59	364.81	0.011637	12.8760	31.99	8 8 8 8
64	1,250.81	364.83	0.018686	25.3639	63.99	8 8 8 8

ing an FPTAS for our problem via a completely different technique.

For our inventory application INV-LS, if the lead time is either 0 or 1, the corresponding DP is single-dimensional. Thus, one can obtain an FPTAS in each of these two cases; see §§7.2 and 7.3 of Halman et al. (2009, 2011). When the lead time is greater than 1, the existence of an FPTAS for INV-LS is an open question.

We briefly discuss other possible research directions. Given $\varepsilon,\delta>0$ and oracle access to a submodular function $f:\{0,1\}^n\to\mathbb{Z}$, Raskhodnikova and Yaroslavtsev (2013) have shown that there exists a randomized algorithm that can output the value of f correctly (with probability $1-\delta$) on at least $(1-\varepsilon)2^n$ points of the domain. The running time of the algorithm is polynomial in n, $(2k)^{O(2k\log(k/\varepsilon))}$ and $\log(1/\delta)$. It would be interesting to investigate the applicability of their results to approximate a dynamic program. With the objective of robust optimization, Iancu et al. (2013) have characterized convexity and supermodularity based conditions on the value function of a dynamic program to identify simple optimal policies. It would also be interesting to investigate the applicability of discrete convexity in this setting.

Endnotes

1. The details about how the input is specified are in §5.

2. As we will see later in §3, the notion of discrete convexity in $\mathbb{Z}^n$ we use ($L^\natural$-convexity) coincides with the notion of discrete convexity used in Halman et al. (2008, 2009) when $n=1$.

3. K is an upper bound on the ratio of the values of the approximate function to those of the original function; for an approximation scheme, $K=1+\varepsilon$.

4. Note that we are using the word "extension" in a loose manner, in the sense that a continuous extension of a discrete function f does not necessarily coincide with f on all of its domain points.

5. The fact that unconstrained minimization preserves $L^\natural$-convexity is established in Theorem 6.19, item (6), of Murota (2007). However, the result extends to the case of constrained minimization as long as the constraint set is $L^\natural$-convex. This lemma and the proof idea were suggested to the authors by Professor Kazuo Murota.

6. We are thankful to Professor Kazuo Murota for suggesting this proof.

7. Since Murota (2007) is in Japanese, we also provide the English translations of Theorems 7.1(1) and 7.3(1):

"*Theorem* 7.1(1): *The discretization* (7.1) *of an $L^\natural$-convex function in continuous variables is an $L^\natural$-convex function in discrete variables.*

Theorem 7.3(1): *The elongation* (7.2) *of an $L^\natural$-convex function in discrete variables is an $L^\natural$-convex convex function in continuous variables.* (*Elongation means the convex extension.*)"

These English translations are due to Professor Kazuo Murota.

8. To be precise, we should attach the sub/superscript ε to $\check{f}$ and H in Definition 16. We avoid using this sub/supercript to keep the notation simple.

9. Assumption 3 is only for notational simplicity. This assumption can be replaced with the weaker requirement that $\mathbf{x}_t\in\Pi_{i=1}^{k_1} L_{1,i}U_{1,i}$, $\mathbf{a}_t\in\Pi_{i=1}^{k_2} L_{2,i}U_{2,i}$, and $\mathbf{w}_t\in\Pi_{i=1}^{k_3} L_{3,i}U_{3,i}$.

10. Note that for period T, $\tilde{f}_T$ is constructed using f_T directly; for period $T-1$ (and other periods), we use $\tilde{f}_{T-1}$ to construct $\check{f}_{T-1}$.

11. In our case, by Assumption 1 the constraint set is assumed to be $L^\natural$-convex.
12. To clarify, $\check{F}_t$ is not necessarily integer-valued. This is due to the linear interpolation step used in taking the Lovász Extension of the function obtained by restricting F_t to the ε-coarse cube. This fact will be considered when calculating the time complexity of our algorithm.
13. From Halman et al. (2009), given a function $f: [L, U]_{\mathbb{Z}} \to \mathbb{R}^+$ and a subset S of $[L, U]_{\mathbb{Z}}$, we refer to the continuous function obtained by making f linear between successive points in S as the piecewise-linear extension of f induced by S.
14. The actual procedure in Halman et al. (2009) involves additional steps; we only capture the basic idea here.
15. The source code, as well as instructions for carrying out the experiments, can be downloaded from http://www.utdallas.edu/~wei.chen/dp.tar.bz2.

Acknowledgments

The authors thank Nir Halman, Retsef Levi, and Chung-Lun Li for answering their questions regarding Halman et al. (2008, 2009, 2011) and Levi et al. (2008b). They are also indebted to Kazuo Murota for answering their numerous questions, suggesting the proofs of some lemmas, and sending them a copy of his book (Murota 2007). The authors thank the associate editor and three anonymous referees for their valuable suggestions, which have significantly improved the paper.

References

Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M (1999) *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties* (Springer-Verlag, Berlin).

Falk JE, Hoffman KR (1976) A successive underestimation method for concave minimization problems. *Math. Oper. Res.* 1(3):251–259.

Favati P, Tardella F (1990) Convexity in nonlinear integer programming. *Ricerca Operativa* 53:3–44.

Fujishige S, Murota K (2000) Notes on L-/M-convex functions and the separation theorems. *Math. Programming* 88(1):129–146.

Halman N, Nannicini G, Orlin JB (2013) A computationally efficient FPTAS for convex stochastic dynamic programs. *Eur. Sympos. Algorithms* (Springer-Verlag, Berlin), 577–588.

Halman N, Klabjan D, Li C-L, Orlin JB, Simchi-Levi D (2008) Fully polynomial time approximation schemes for stochastic dynamic programs. *Proc. ACM-SIAM Sympos. Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia), 700–709.

Halman N, Klabjan D, Li C-L, Orlin JB, Simchi-Levi D (2011) Fully polynomial time approximation schemes for stochastic dynamic programs. Technical Report 3918, http://optimization-online.org.

Halman N, Klabjan D, Mostagir M, Orlin JB, Simchi-Levi D (2009) A fully polynomial-time approximation scheme for single-item stochastic inventory control with discrete demand. *Math. Oper. Res.* 34(3):674–685.

Iancu DA, Sharma M, Sviridenko M (2013) Supermodularity and affine policies in dynamic robust optimization. *Oper. Res.* 61(4):941–956.

Janakiraman G, Muckstadt JA (2004) Inventory control in directed networks: A note on linear costs. *Oper. Res.* 52(3):491–495.

Levi R, Janakiraman G, Nagarajan M (2008a) A 2-approximation algorithm for stochastic inventory control models with lost sales. *Math. Oper. Res.* 33(2):351–374.

Levi R, Lodi A, Sviridenko M (2008b) Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Math. Oper. Res.* 33(2):461–474.

Levi R, Pál M, Roundy R, Shmoys DB (2007) Approximation algorithms for stochastic inventory control models. *Math. Oper. Res.* 32(2): 284–302.

Levi R, Roundy R, Shmoys DB, Truong VA (2008c) Approximation algorithms for capacitated stochastic inventory control models. *Oper. Res.* 56(5):1184–1199.

Lovász L (1983) Submodular functions and convexity. Bachem A, Grötschel M, Korte B, eds. *Mathematical Programming—The State of the Art* (Springer-Verlag, Berlin), 235–257.

Miller BL (1971) On minimizing nonseparable functions defined on the integers with an inventory application. *SIAM J. Appl. Math.* 21(1): 166–185.

Morton TE (1969) Bounds on the solution of the lagged optimal inventory equation with no demand backlogging and proportional costs. *SIAM Rev.* 11(4):572–596.

Murota K (2003) *Discrete Convex Analysis*, SIAM Monographs on Discrete Mathematics and Applications (Society for Industrial and Applied Mathematics, Philadelphia).

Murota K (2007) *Primer of Discrete Convex Analysis—Discrete versus Continuous Optimization* (Kyoritsu Publishing Company, Tokyo).

Murota K, Shioura A (2000) Extension of M-convexity and L-convexity to polyhedral convex functions. *Adv. Appl. Math.* 25(4):352–427.

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Wiley Series in Probability and Statistics (Wiley-Interscience, Hoboken, NJ).

Raskhodnikova S, Yaroslavtsev G (2013) Learning pseudo-Boolean k-DNF and submodular functions. *Proc. ACM-SIAM Sympos. Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia), 1356–1368.

Woeginger GJ (2000) When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS J. Comput.* 12(1):57–74.

Zipkin PH (2008a) Old and new methods for lost-sales inventory systems. *Oper. Res.* 56(5):1256–1263.

Zipkin PH (2008b) On the structure of lost-sales inventory models. *Oper. Res.* 56(4):937–944.

Wei Chen is a Ph.D. candidate of operations management in the Naveen Jindal School of Management at the University of Texas at Dallas. His research interests include inventory management and procurement auctions.

Milind Dawande is Ashbel Smith Professor of Operations Management in the Naveen Jindal School of Management at the University of Texas at Dallas. His research interests are in optimization theory and its applications to problems in manufacturing and service operations management.

Ganesh Janakiraman is professor of operations management in the Naveen Jindal School of Management at the University of Texas at Dallas. His research interests include stochastic inventory theory and auctions for procurement.