Automatic generation of FPTASes for stochastic monotone dynamic programs made easier

Tzvi Alon [*] Nir Halman [†]

December 19, 2019

Abstract

In this paper we go one step further in the automatic generation of FPTASes for multi-stage stochastic dynamic programs with scalar state and action spaces, in where the cost-to-go functions have a monotone structure in the state variable. While there exist a few frameworks for automatic generation of FPTASes, so far none of them is general and simple enough to be extensively used. We believe that our framework has this two attributes, and has great potential to attract interest from both the operations research and theoretical computer science communities. Moreover, it seems very reasonable that many intractable problems, that currently do not admit an FPTAS, can be formulated as DPs that fit into our framework, and therefore will admit a first FPTAS. Our results are achieved by a combination of Bellman equation formulations, the technique of K-approximation sets and functions, and in particular – the calculus of K-approximation functions.

Keywords: Dynamic programming, FPTAS, K-approximation sets and functions.

1 Introduction

Multi-stage stochastic decision making and approximation. Stochastic dynamic programming (DP) is a common tool in multi-stage decision making. Unfortunately, stochastic dynamic programs (DPs – to be distinguished from dynamic programming by context) are generally intractable both in theory and in practice, implying that they cannot be solved exactly except for toy examples. This motivates the study of approximate solution methods. Among approximation algorithms, Fully Polynomial Time Approximation Schemes (FPTASes) are widely considered the best. An FPTAS runs in time polynomial in the (binary) input size and in the error parameter ϵ, and allows trading off the error parameter with the runtime. Soon after the first FPTASes were designed, see the seminal works of Ibarra and Kim, and Sahni on the Knapsack problem [IK75, Sah76], it was clear that conditions sufficient for the existence of an FPTAS are of interest [KS81]. There is only a handful of papers dealing with constructive frameworks for FPTASes. Such frameworks consist of: (i) a DP model, (ii) a set of conditions, and (iii) an algorithm. If a DP fits a framework, i.e., fits into the model as well as satisfies the set of conditions, then the framework assures us that the algorithm applied on the DP serves as an FPTAS for it. One such constructive framework is developed for multi-stage stochastic linear programs (LPs), where the number of stages is constant [SS12]. Moving to non-constant number of stages, the only such frameworks to the best of our knowledge, are the one of [HKL$^+$14], applied to certain families of stochastic DPs, and its extension to more complex stochastic processes [Hal19b].

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

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

Multi-stage deterministic decision making. Deterministic decision making is a very important special case, where all random variables realize to a-priori given constants with probability one, and is dealt by deterministic dynamic programming – a much better understood tool with abundant literature in the operations research and theoretical computer science communities. Still, there are very few constructive frameworks for approximating deterministic DPs, e.g., [Woe00, PW07, KK12, MS13]. However, to the best of our knowledge, none of these frameworks has yielded much more than a dozen of application FPTASes since its publication. Possible reasons for this phenomenon are that (i) the underlying model or set of conditions are somewhat too specific, or (ii) that they are generally too complicated to be verified, or (iii) that it is difficult to formulate a problem as a DP that fits the framework. Regarding the constructive frameworks for approximating stochastic DPs, i.e., [HKL+14, Hal19b], it seems that while they better address the drawback listed in points (ii)-(iii) above, their underlying models are still not general enough.

Our Results. In this paper we aim to tackle the main drawback of the frameworks [HKL+14, Hal19b], i.e., point (i) above, by designing an FPTAS framework to a *much broader* class of DPs. The modeling process of a problem inherently involves making tradeoffs between the accuracy of the models, the availability of data, and the time it takes to solve the problem. To make these tradeoffs, modelers and researchers would like to identify problems that can be approximated efficiently. This paper aims at *developing easily recognizable conditions* under which stochastic DPs can be approximated via FPTASes. In developing conditions, there is a natural tradeoff of generality vs. efficiency. The more general the conditions, the slower the general-purpose FPTAS for a problem satisfying these conditions. Moreover, in developing the conditions, there is also the tradeoff of generality vs. simplicity. The more general the conditions, the more complex the algorithm for developing the FPTASes. However, in this work we develop a framework that is general *and* simple *and* efficient at the same time. Specifically, in this paper we *design computationally-efficient FPTASes* whose running times dependency on the error parameter (on the problem (binary) input size) are typically between linear and quadratic (quadratic and cubic), up to log terms, respectively. There are no hidden constants in our FPTASes.

Our contribution. The contribution of this paper is fivefold. First, we give a *powerful FPTAS framework for stochastic DPs.* We demonstrate the generality and applicability of our framework by (re)formulating 18 problems to fit into it, and hence to admit an FPTAS, see Table 1. Second, our FPTAS framework, while general, is *efficient and has low-degree polynomial dependency on the problem size and* $1/\epsilon$: Our FPTAS, applied on the 18 problems, which were studied in about half a dozen of different papers, is generally not slower than the specific FPTASes tailored to each of the problems studied. Third, while general, *our FPTAS framework is easy to use when having at hand the calculus of* K*-approximation functions.* This is done in two stages: (i) (re)formulating the problem such that it fits into our model, and (ii) proving that the specific DP formulation satisfies the conditions of our framework. We meticulously demonstrate, mostly in the appendices, that these two stages are not hard to perform on the aforementioned 18 problems. We note in passing that our motivation to give so many examples is to provide as a wide as possible "training set" for any researcher interested to know how to use our framework. We also note that while our framework generalizes the one of [HKL+14], the current manuscript is considerably shorter, and we believe – also easier to read than [HKL+14]. Fourth, in providing a powerful FPTAS framework, we actually *exchange algorithm design with DP formulation.* I.e., instead of designing an FPTAS for each specific problem, we show how to formulate it as a DP that fits into our framework, which in turn, automatically generates FPTASes, hence the title of this work. Last, because our framework (strictly) contains the frameworks for stochastic DPs of [HKL+14, Hal19b] as special cases, it can be viewed as a *Meta-framework for FPTAS design.* We believe that many more intractable problems, that currently do not admit an FPTAS, can be formulated as a DP that fits into our framework, and therefore will admit a first FPTAS. (This is actually the case with problem 4 in Table 1, for which we report a first FPTAS.) In fact, if the current paper have already existed, the papers reporting tailor-made FPTASes for problems 6-11 in Table 1 could have used it and consequently become simpler and shorter at the same time. This means that our framework has great potential to become a standard tool in the toolbox of FPTAS design.

Table 1: Reformulated applications. n stands for number of items while T stands for number of time periods. See the specific section for problem description and additional notations.

	Problem	Running time	Previous FPTAS via K-approximation sets and functions	Section
1	0/1 Knapsack	$O\left(\frac{n^2}{\epsilon}\log\frac{n\log\Pi}{\epsilon}\log B\log\Pi\right)$	Elaborated in Sec. 3.4	3.4
2	Counting 0/1 Knapsack	$O\left(\frac{n^3}{\epsilon}\log\frac{n}{\epsilon}\log C\right)$	Elaborated in Sec. 3.5	3.5
3	Capacity expansion	$O\left(\frac{T^3}{\epsilon^2}\log\left(\frac{T}{\epsilon}\log(T\Pi)\right)\log^2(T\Pi)\log C_T+\frac{Tn^3}{\epsilon^2}\log\frac{n}{\epsilon}\log C_T\log\Pi\right)$	[HKL+14, Appx. A.3]	3.7
4	Counting n-tuples	$O\left(\frac{n^3M\log M}{\epsilon}\log\frac{n\log M}{\epsilon}\log B\right)$	First FPTAS ever	A.1
5	Counting shortest s-t paths in directed acyclic graph (V,E)	$O\left(\frac{\lvert E\rvert\lvert V\rvert^2}{\epsilon}\log\frac{\lvert V\rvert}{\epsilon}\log L\right)$	First FPTAS via K-apx. sets and functions	A.2
6	Bi-criteria path problem with maximum survival probability in directed graph (V,E)	$O\left(\frac{\lvert V\rvert^2\lvert E\rvert}{\epsilon}\log\frac{\lvert V\rvert\log\frac{1}{p_{\min}}}{\epsilon}\log\frac{1}{p_{\min}}\log(L-\ell_{\min})\right)$	[HKQ+19]	A.3
7	Boolean programming problem under nested knapsack constraints	$O\left(\frac{n^2}{\epsilon}\log\left(\frac{n}{\epsilon}\log(Z^{UB}+A_n)\right)\log(Z^{UB}+A_n)\log d_n\right)$	[HKS18]	A.4
8	Minimizing makespan of deteriorating jobs	$O\left(\frac{n^3}{\epsilon^2}\log^2 L\log(nL)\log\frac{n\log(nL)}{\epsilon}\right)$	[Hal19a]	A.5
9	Counting integer Knapsack	$O\left(\frac{n^3}{\epsilon}\log^3 U\log C\log\frac{n\log U}{\epsilon}\right)$	[Hal16]	A.6
10	Single-item economic lot-sizing	$O\left(\frac{T^3}{\epsilon^2}\log^2\bar{U}\log d^*\log\left(\frac{T}{\epsilon}\log\bar{U}\right)\right)$	[HOS12]	B.1
11	Time-cost trade-off problem in series parallel project network	$O\left(\frac{n^3}{\epsilon^2}\log\left(\frac{n}{\epsilon}\log\bar{U}\right)\log^2\bar{U}\log U\right)$	[HLS09]	B.2
12	Stochastic ordered adaptive knapsack problem	$O\left(\frac{n^*n^2}{\epsilon}\log\frac{n\log\Pi}{\epsilon}\log B\log\Pi\right)$	[HKL+14, Appx. A.1]	C.1.1(1)
13	Nonlinear knapsack problem	$O\left(\frac{n^3}{\epsilon^2}\left(\log u_{\max}+\log\left(\frac{n}{\epsilon}\log\Pi\right)\right)\log^2\Pi\log B\right)$	[HKL+14, Appx. A.2]	C.1.1(2)
14	Time-cost trade-off machine scheduling	$O\left(\frac{n^3}{\epsilon^2}\log^2 d_1\log^2 U\log\left(\frac{n}{\epsilon}\log U\right)\right)$	[HKL+14, Appx. A.4]	C.1.1(3)
15	Single-item stochastic batch dispatch	$O\left(\frac{n^*T^3\log^2 U\log(I_1+D^*)}{\epsilon^2}\left(n^*+\log(I_1+D^*)+\log\frac{T\log U}{\epsilon}\right)\right)$	[HKL+14, Appx. A.6]	C.1.1(4)
16	Single-resource revenue management	$O\left(\frac{n^*T^3\log^2(Tr_{\max}C)\log C}{\epsilon^2}\left(n^*+\log C+\log\frac{T\log(Tr_{\max}C)}{\epsilon}\right)\right)$	[HKL+14, Appx. A.7]	C.1.1(5)
17	Lifetime consumption of risky capital	$O\left(\frac{n^*T^3}{\epsilon^2}\left(n^*+\log U+\log(\frac{T}{\epsilon}\log\breve{U})\right)\log^2(\breve{U})\log U\right)$	[HKL+14, Appx. A.8]	C.1.1(6)
18	Stochastic growth model	$O\left(\frac{n^*T^3}{\epsilon^2}\left(n^*+\log\bar{I}^{\max}+\log(\frac{T}{\epsilon}\log\breve{u})\right)\log^2\breve{u}\log\bar{I}^{\max}\right)$	[HKL+14, Appx. A.9]	C.1.1(7)

Previous approaches to FPTAS design. Standard techniques to design FPTASes, such as rounding and scaling the input data (the goal is to make the resulting rounded problem easy to solve and to bring the running time of the DP down to polynomial), e.g., [Sah76], or trimming the state space (the idea is to iteratively thin out the state space of the DP, and to collapse states that are close to each other. The goal is to bring the size of the state spaces down to polynomial), e.g., [IK75], are applied on DP tables (i.e., matrices) of size that is pseudopolynomial in the (binary) input size, whose elements are *numbers*, in order to access only a polynomial number of entries. Because numerous NP-hard problems admit tailor-made FPTASes, the concept of having some framework for designing FPTASes and giving sufficient conditions on its applicability soon emerged, see works as early as the one of Korte and Schrader [KS81] and as recent as the ones of [Woe00, PW07, MS13], all of which are based on variations of the techniques stated above and deal only with deterministic DPs.

Our approach and innovation of this paper. We take on a different approach: we conceptualize the DP as a recursion that consists of a polynomial number of univariate *functions* whose domain size is pseudopolynomial in the (binary) input size. We next use a polynomial number of points from the domain to approximate these functions, and then use the calculus of K-approximation functions to show that the DP recursion which use these approximate functions indeed runs in polynomial time. This approach is used in the framework of [HKL$^+$14] in order to design an FPTAS to certain stochastic problems that can be casted as equation (3) below, and satisfy Conditions 1-3 in [HKL$^+$14]. (For the sake of completeness we overview their framework in Appendix C.1 - see therein Conditions 6-8.) The same approach is also taken in the framework of [Hal19b], which applies for similar problems but with complex stochastic processes, providing an FPTAS when the random variables are described by value oracles to their CDF. This allows it to handle discrete random variables that can take on exponentially many values, and bounded-support continuous random variables. (For the sake of completeness we overview their framework in Appendix C.2.) The innovation of our framework with respect to these two is elaborated in Section 3, where our framework is introduced and where we claim that both frameworks for monotone DPs in [HKL$^+$14, Hal19b] are special cases of the framework we introduce in that section.

In this paper we take the same approach but consider DPs with almost no requirement on their recursive structure. We state a set of relatively simple conditions, as well as simple algorithms, and use the technique of K-approximation sets and functions to show that our algorithms serve as FPTASes for any DP that satisfies these conditions. Our starting point is the framework of [HKL$^+$14] for monotone DPs with finite-time discrete-time horizons. We first relax the requirement on the structure of the DP recursion. We then relax the set of conditions about the functions that are involved in the DP formulations.

Our ability to perform fine error analysis permits us to deal with problems that can be casted as the more general equation (1) and to use auxiliary FPTASes within our FPTAS, e.g., we do not require that single-stage cost functions nor the Bellman equation itself are computable in polynomial time for any given state variable, and require instead that they admit an FPTAS. We show that our innovative technique enables us to develop a very powerful FPTAS framework, that is, we show how various algorithmic and approximation ideas are combined together so as to give a framework that can handle (i) some problems (e.g., problems 3-11 from Table 1) that were not handled by previous frameworks, as well as (ii) problems that fit into the stochastic models dealt by [HKL$^+$14, Hal19b] (e.g., problems 1-3, 12-18 from Table 1). We add in passing that we are unaware of any FPTAS framework for stochastic DPs but the ones state above that use the technique of K-approximation sets and functions.

Organization of the paper. In Section 2 we overview the technique of K-approximation sets and functions. In Section 3 we state and prove our FPTAS framework, i.e., give the DP model, the set of conditions and the FPTAS, and demonstrate its applicability on three problems. In the appendix we show how the remaining application problems fit into our framework.

2 K-approximation sets and functions

In this section, we give an overview of the technique of K-approximation sets and functions.

Related research. Halman et al. [HLS09] introduced the technique of K-approximation sets and functions, and used it to develop an FPTAS for a certain stochastic inventory control problem. Halman et al. [HKL$^+$14] applied this technique to develop an FPTAS framework for three rather general classes of stochastic DPs. Halman [Hal19b] extended the aforementioned framework to more general stochastic processes as well as to sample-based DPs. This technique has also been used to design tailor-made FPTASes to various optimization problems, see the very recent papers [DHV19, Hal16, Hal19a, HKS18, HKQ$^+$19, HN19, HNO18].

Notation. We denote by $\mathbb{R}, \mathbb{R}^+$ the set of real and nonnegative real numbers, respectively. For any pair of integers $A \leq B$, let $\{A, \dots, B\}$ denote the set of integers between and including A and B. Let $D \subset \mathbb{R}$ be a finite set of real numbers. Let $D^{\min}$ and $D^{\max}$ denote the minimal and maximal element in D, respectively. For $x < D^{\max}$, let $\text{next}(x, D) = \min\{y \in D \mid y > x\}$. For $x > D^{\min}$, let $\text{prev}(y, D) = \max\{y \in D \mid y < x\}$. In all algorithms discussed in this paper we assume that for an input function $\varphi : D \to \mathbb{R}$ an oracle is available which for any x returns the value $\varphi(x)$ in t_φ time. Note that t_φ may be a constant, or a function that depends on the size of the domain, i.e., on $|D|$. For any such function φ that is not identically zero we denote $\varphi^{\min} := \min_{x \in D}\{|\varphi(x)| : \varphi(x) \neq 0\}$, and $\varphi^{\max} := \max_{x \in D}\{|\varphi(x)|\}$. We want to preprocess a representation $\tilde{\varphi}$ for φ such that any further evaluation of $\varphi(\cdot)$ will be done by querying $\tilde{\varphi}(\cdot)$ instead of querying $\varphi(\cdot)$ directly. Of course, by querying all values $\varphi(x)$ in $x \in D$ and storing them in a sorted array of the form $\{(x, \varphi(x)) \mid x \in D\}$, we can obtain in $O(|D| t_\varphi)$ time a representation of size $O(|D|)$ which can return the (exact) value $\varphi(x)$ for any x in $O(\log |D|)$ time. However, whenever either $|D|$ or t_φ is large, we would like to have a representation that takes less space or queries to construct. We say that a representation is *succinct* if its size is polylogarithmic in $|D| + \frac{\varphi^{\max}}{\varphi^{\min}}$, and that a representation is *efficient* if it can be built in time polylogarithmic in both these terms (if $\varphi \equiv 0$, then $\varphi^{\min}$ is undefined, and we will refer to $\frac{\varphi^{\max}}{\varphi^{\min}}$ as 2). The base two logarithm of z is denoted by $\log z$. For simplicity of presentation, and because we use the following assumption only in big "$O()$" terms, we slightly abuse notation and assume in the sequel that for any base $b > 1$ and $x > 0$ the value of $\log_b x$ is the maximum between this term (without the assumption) and 1.

Definition 2.1 *Let $K \geq 1$, $\alpha, \tilde{\alpha} \geq 0$ be arbitrary real numbers and let $\varphi, \tilde{\varphi} : D \to \mathbb{R}^+$ be arbitrary functions. We say that $\tilde{\alpha}$ is a* K-approximation value *of α if $\alpha \leq \tilde{\alpha} \leq K\alpha$. We say that $\tilde{\varphi}$ is a* K-approximation function *of φ if $\varphi(x) \leq \tilde{\varphi}(x) \leq K\varphi(x)$ (i.e., if $\tilde{\varphi}(x)$ is a K-approximation value of $\varphi(x)$) for all $x \in D$. (Note that if $\tilde{\varphi}$ is a K-approximation of φ then it has relative error $K - 1$ with respect to φ. We usually denote by $\epsilon = K - 1$ the relative error, and by K the multiplicative error.)*

Remark 1 *The definition of K-approximation functions is tailored to minimization problems, where the approximated function lies "above" the exact function. A K-approximation function for maximization problems should lie "below" the exact function, and is defined by $z/K \leq \tilde{z} \leq z$. Clearly, if $\tilde{z}$ K-approximates z from above, then $\tilde{z}/K$ K-approximates z from below. Similarly, if $\tilde{z}$ K-approximates z from below, then $K\tilde{z}$ K-approximates z from above. Hence, although this paper focuses on the definition of K-approximation functions for minimization problems, it is rather straightforward to extend its definitions and results to deal with maximization problems.*

The following proposition provides a set of general computational rules of K-approximation functions, and is a key ingredient in our FPTAS design and analysis, as well as in verifying that specific problems satisfy the framework's conditions.

Proposition 2.2 (Calculus of K-approximation Functions [HKL$^+$14, Prop. 5.1]) *For $i = 1, 2$, let $K_i \geq 1$, let $\varphi_i : D \to \mathbb{R}^+$ be an arbitrary function over any finite domain D, and let $\tilde{\varphi}_i : D \to \mathbb{R}^+$ be a K_i-approximation of φ_i. Let $\psi_1 : D' \to D$ be an arbitrary function over any finite domain D'. Let $\alpha, \beta \in \mathbb{R}^+$. The following properties hold:*

1. *φ_1 is a 1-approximation of itself.*
2. *(Linearity of approximation) $\alpha + \beta\tilde{\varphi}_1$ is a K_1-approximation of $\alpha + \beta\varphi_1$.*
3. *(Summation of approximation) $\tilde{\varphi}_1 + \tilde{\varphi}_2$ is a $\max\{K_1, K_2\}$-approximation of $\varphi_1 + \varphi_2$.*
4. *(Composition of approximation) $\tilde{\varphi}_1(\psi_1)$ is a K_1-approximation of $\varphi_1(\psi_1)$.*
5. *(Maximization of approximation) $\max\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $\max\{K_1, K_2\}$-approximation of $\max\{\varphi_1, \varphi_2\}$.*
6. *(Minimization of approximation) $\min\{\tilde{\varphi}_1, \tilde{\varphi}_2\}$ is a $\max\{K_1, K_2\}$-approximation of $\min\{\varphi_1, \varphi_2\}$.*
7. *(Approximation of approximation) If $\varphi_2 = \tilde{\varphi}_1$, then $\tilde{\varphi}_2$ is a K_1K_2-approximation of φ_1.*

In this paper we consider DP formulations in which the functions involved are monotone, and therefore concentrate on definitions for K-approximation sets and functions specialized to monotone functions (cf. [HKL$^+$14]). The idea behind K-approximation sets is to choose a small (i.e., polynomially bounded size) subset of points from the domain of a function, ensuring that step interpolation between the values of the function on this subset guarantees rigorous error bounds.

Definition 2.3 ([HKL$^+$14, Def. 4.4]) *Let D be a finite set of real numbers and $\varphi : D \to \mathbb{R}^+$ be a monotone nonnegative function. For any subset $W \subseteq D$ satisfying $D^{\min}, D^{\max} \in W$, the* approximation of φ induced by W *is the piecewise constant (step) function*

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

Remark 2 *Once one stores in the computer memory an array $\{(x, \varphi(x)) \mid x \in W\}$ sorted in increasing order of x, then via binary search one can query $\hat{\varphi}(x)$ in $O(\log|W|)$ time, for any $x \in D$.*

Definition 2.4 ([HKL$^+$14, Def. 4.2 and Prop. 4.5]) *Let $K \geq 1$ and let $\varphi : D \to \mathbb{R}^+$ be a monotone nonnegative function. Let $W \subseteq D$ be such that $D^{\min}, D^{\max} \in W$ and let $\hat{\varphi}$ be the approximation of φ induced by W. We say that W is a* K-approximation set of φ *if for every $x \in D$, we have $\hat{\varphi}(x) \leq K\varphi(x)$.*

The above two definitions tell us that the approximation of φ induced by a K-approximation set of φ is a K-approximation function of φ, see Figure 1.

Halman *et al.* design an FPTAS, called in this paper FPTASApxSet(φ, ϵ), that constructs a succinct $(1 + \epsilon)$-approximation set for a nonnegative monotone function $\varphi : D \to \mathbb{R}^+$. The algorithm essentially finds a subset W of points from the domain D of the function, such that the ratio between the values of φ on successive points in W is not more than $(1 + \epsilon)$ [HKL$^+$14, Alg. 1]. The cardinality of W is therefore $O(\log_{1+\epsilon} \frac{\varphi^{\max}}{\varphi^{\min}})$, and because the algorithm uses binary search over the domain D of φ to find each point in W, its running time is $O(t_\varphi \log_{1+\epsilon} \frac{\varphi^{\max}}{\varphi^{\min}} \log D) = O(\frac{t_\varphi \log D}{\epsilon} \log \frac{\varphi^{\max}}{\varphi^{\min}})$. For the sake of completeness we state their algorithm in Appendix D. (This algorithm produced the set W in Figure 1). We summarize the discussion above in the following proposition.

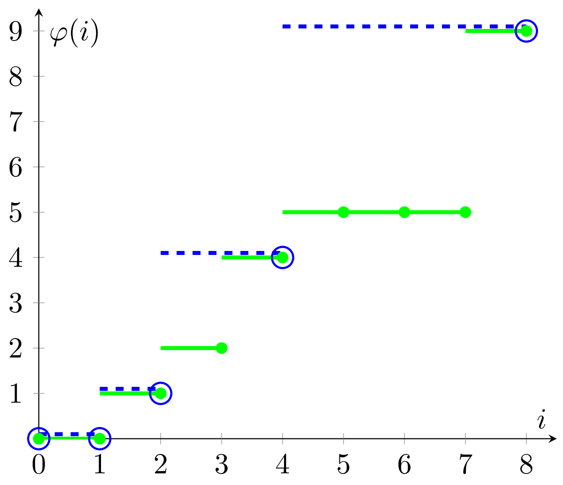

Figure 1: Function $\varphi : \{0,\ldots,8\}\to\mathbb{Z}^+$ is defined by the green dots. The set $W = \{0,1,2,4,8\}$ serves as a 2-approximation set for φ, and is depicted by the blue circles. The function $\hat{\varphi}$ induced by W is depicted by the blue dashed lines.

Proposition 2.5 (FPTAS for a monotone function via direct access, based on [HKL$^+$14, Prop. 4.6]) *Let $0 < \epsilon \le 1$ be a real number, $K := 1+\epsilon$, and $\varphi : D\to\mathbb{R}^+$ be a monotone nonnegative function over a finite domain D of real numbers. Then, function* FPTASCompress(φ,ϵ) *is an FPTAS that returns in $O\big(t_\varphi \log_K \frac{\varphi^{\max}}{\varphi^{\min}} \log|D|\big) = O\big(\frac{t_\varphi}{\epsilon}\log\frac{\varphi^{\max}}{\varphi^{\min}}\log|D|\big)$ time a representation of a monotone step function $\tilde{\varphi}$ with $O\big(\log_K \frac{\varphi^{\max}}{\varphi^{\min}}\big)$ steps that approximates φ with relative error ϵ, and of which the query time is $t_{\tilde{\varphi}} = O(\log\log_K \frac{\varphi^{\max}}{\varphi^{\min}})$.*

Algorithm 1 An FPTAS that returns a monotone $(1+\epsilon)$-approximation of a monotone function φ

```
1: Function FPTASCompress($\varphi, \epsilon$)
2: $W \leftarrow$FPTASApxSet($\varphi, \epsilon$)
3: return the appx. of $\varphi$ induced by $W$ as an array $\{(x,\varphi(x)) \mid x \in W\}$ sorted in increasing order of $x$.
```

K-approximations of a function φ can be found even if the function itself is not available, but there exists an oracle that computes values of some monotone function $\bar{\varphi}$ that approximates φ: Below we present a proposition, adapted from [HKL$^+$14, Prop. 4.5, Prop. 4.6], that applies to finding approximations of a monotone function φ by calling FPTASCompress. The estimates of computation times and approximation quality follow from the discussion above and an application of the calculus of approximation (the approximation of approximation rule).

Proposition 2.6 (FPTAS for a monotone function via a monotone L-approximation oracle) *Let $0 < \epsilon \le 1, L \ge 1$ be real numbers, $K = 1+\epsilon$, and let $\varphi : D\to\mathbb{R}^+$ be a monotone function over a finite domain D of real numbers. Let $\bar{\varphi}$ be a monotone L-approximation function of φ. Then Function* FPTASCompress$(\bar{\varphi},\epsilon)$ *returns in $O\big(t_{\bar{\varphi}} \log_K \frac{\varphi^{\max}}{\varphi^{\min}} \log|D|\big) = O\big(\frac{t_{\bar{\varphi}}}{\epsilon}\log\frac{\varphi^{\max}}{\varphi^{\min}}\log|D|\big)$ time a representation of a monotone step function $\tilde{\varphi}$ with $O\big(\log_K \frac{\varphi^{\max}}{\varphi^{\min}}\big)$ steps that approximates $\bar{\varphi}$ with relative error ϵ, and of which the query time is $t_{\tilde{\varphi}} = O\big(\log\log_K \frac{\varphi^{\max}}{\varphi^{\min}}\big)$. Moreover, $\tilde{\varphi}$ is a KL-approximation function of φ.*

As demonstrated in [HKL$^+$14, Sec. 4.2], it is also possible to compute an approximation of a monotone function φ that is accessed via an *unnecessarily-monotone* approximation oracle $\bar{\varphi}$, as is the case with Condition 4(ii) below. Here we encounter an additional difficulty: Since $\bar{\varphi}$ is unnecessarily monotone, a K-approximation set of it is not well defined. However, for this purpose [HKL+14] design a simple procedure we call FPTASIndirectApxSet (see Appendix E for the algorithm statement) that for every given parameters $\bar{\varphi}$ and ϵ, returns a set $\bar{W}$. The maximal monotone function $\tilde{\varphi}$ that is bounded from above by $\bar{\varphi}$ over $\bar{W}$ serves as a $(1+\epsilon)$-approximation function of φ, and for which $\bar{W}$ is a $(1+\epsilon)$-approximation set [HKL+14, Prop. 4.5 and 4.7].

We define a function FPTASIndirectCompress, see Algorithm 2, for the efficient construction of a succinct representation of an approximation of a monotone function $\varphi : D \to \mathbb{R}^+$ which is accessed via an unnecessarily monotone L-approximation $\bar{\varphi}$.

Algorithm 2 Returns a monotone $(1+\epsilon)L$-approximation of a monotone function φ via a non-monotone L-approximation oracle $\bar{\varphi}$ of φ

```
1: Function FPTASIndirectCompress($\bar{\varphi}, \epsilon$)
2: $\bar{W} \leftarrow$ FPTASIndirectApxSet($\bar{\varphi}, \epsilon$)
3: return the maximal monotone function $\tilde{\varphi}$ that is bounded from above by $\bar{\varphi}$ over $\bar{W}$, represented as an
   array $\{(x, \tilde{\varphi}(x)) \mid x \in \bar{W}\}$ sorted in increasing order of $x$.
```

Applying the approximation of approximation rule (Proposition 2.2(7)) with [HKL+14, Prop. 4.5 and 4.7] yields the following result.

Proposition 2.7 *(based on [HKL+14, Prop. 4.7]) Let $\varphi : D \to \mathbb{R}^+$ be a monotone nonnegative function over a finite domain D of real numbers. Let $\bar{\varphi}$ be an (unnecessarily monotone) L-approximation function of φ ($L > 1$). Then,* FPTASIndirectCompress$(\bar{\varphi}, K)$ *returns in $O\big(t_{\bar{\varphi}} \log_K \frac{\varphi^{\max}}{\varphi^{\min}} \log |D|\big) = O\big(\frac{t_{\bar{\varphi}}}{\epsilon} \log \frac{\varphi^{\max}}{\varphi^{\min}} \log |D|\big)$ time a representation of a monotone step function $\tilde{\varphi}$ with $O\big(\log_K \frac{\varphi^{\max}}{\varphi^{\min}}\big)$ steps that KL-approximates φ, and of which the query time is $t_{\tilde{\varphi}} = O\big(\log \log_K \frac{\varphi^{\max}}{\varphi^{\min}}\big)$.*

3 Model and FPTASes

This paper deals with an abstract stochastic DP model of the following form. Let $0 \le t \le T, 1 \le i \le m$, be a pair of integers where T represents, e.g., the length of a discrete time period, and m repersents, e.g., the number of stages in each time period. For every $0 \le t \le T, 1 \le i \le m$, let $z_{t,i} : \mathcal{S}_{t,i} \to \mathbb{R}^+$ be a nonnegative real-valued function, where $\mathcal{S}_{t,i}$ is the finite state space at time t and stage i. We require that the k'th largest element in $\mathcal{S}_{t,i}$ can be identified in constant time for every $1 \le k \le |\mathcal{S}_{t,i}|$, e.g., $\mathcal{S}_{t,i}$ is a contiguous interval of the form $\{A, \dots, B\}$, $0 \le t \le T, 1 \le i \le m$. For any $I_{t,i} \in \mathcal{S}_{t,i}$, let $A_{t,i}(I_{t,i})$ be additional information which is available in the beginning of time t and stage i when being in state $I_{t,i}$. Typical information in $A_{t,i}(I_{t,i})$ includes the description of the action space at time t and stage i, and of the random variable $D_{t,i}$ which is revealed (i.e., realized) at the end of time t and stage i. Our DP formulation reads as follows:

$$z_{t,i}(I_{t,i}) = f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r \le t, j \le i : r+j < t+i}\right) \qquad \forall I_{t,i} \in \mathcal{S}_{t,i},\ 0 \le t \le T,\ 1 \le i \le m. \tag{1}$$

That is, in order to calculate the value of $z_{t,i}$ on the state $I_{t,i}$ one needs to know also the additional information $A_{t,i}(I_{t,i})$, the previously-calculated functions $\{z_{r,j}\}_{r \le t, j \le i : r+j < t+i}$, as well as to be able to query (calculate) the value of function $f_{t,i}$ on $(I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r \le t, j \le i : r+j < t+i})$. Before continuing any further we give as an example an important special case of (1), to which problems 12-18 in Table 1 belong.

3.1 An Example - finite-time discrete-time horizon stochastic DPs

Consider a finite-time discrete-time horizon stochastic DP, as defined in Bertsekas [Ber95]. The model has two principal features: (i) an underlying *discrete time dynamic system*, and (ii) a *cost function that is additive over time.* The system dynamics are of the form

$$I_{j+1} = \text{tran}_j(I_j, x_j, D_j), \quad j = 1, \ldots, T,$$

where j is the discrete time index,

I_j is the state of the system, which belongs to the *state space* $\mathcal{S}_j$,
x_j is the action or decision to be selected in time period j, which belongs to the *action space* $\mathcal{A}_j(I_j)$,
D_j is a discrete random variable, which takes values in a given set $\mathcal{D}_j$, and
T is the number of time periods.

The cost function, denoted by $g_j(I_j, x_j, D_j)$, is nonnegative and additive in the sense that the cost incurred in time period j is accumulated over time. Let I_1 be the initial state of the system. Given a realization d_j of D_j, for $j = 1, \ldots, T$, the total cost is

$$g_{T+1}(I_{T+1}) + \sum_{j=1}^{T} g_j(I_j, x_j, d_j),$$

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

The problem is to determine

$$z^*(I_1) = \min_{x_j \in \mathcal{A}_j(I_j) | 1 \le j \le T} E\left\{ g_{T+1}(I_{T+1}) + \sum_{j=1}^{T} g_j(I_j, x_j, D_j) \right\}, \tag{2}$$

where the expectation is taken with respect to the joint distribution of the random variables involved. The optimization is over the actions $x_1, \ldots, x_T$. Here, x_j is selected with the knowledge of the current state I_j but before the realization of D_j takes place. The state space and the action space are one-dimensional. Note that the domain of the *single-period cost functions* g_j and *transition functions* tran_j is $(\mathcal{S}_j \otimes \mathcal{A}_j) \times \mathcal{D}_j$, where $\mathcal{S}_j \otimes \mathcal{A}_i = \bigcup_{I_j \in \mathcal{S}_j} (\{I_j\} \times \mathcal{A}_j(I_j))$. For every initial state I_1, the optimal cost $z^*(I_1)$ of the DP is equal to $z_1(I_1)$, where the function z_1 is given by the last step of the following recursion, which proceeds backward from period T to period 1 (Bellman recursion):

$$\begin{aligned} z_{T+1}(I_{T+1}) &= g_{T+1}(I_{T+1}), \\ z_j(I_j) &= \min_{x_j \in \mathcal{A}_j(I_j)} E_{D_j} \left\{ g_j(I_j, x_j, D_j) + z_{j+1}(\text{tran}_j(I_j, x_j, D_j)) \right\}, \quad j = 1, \ldots, T, \end{aligned} \tag{3}$$

where the expectation is taken with respect to the probability distribution of D_j. $z_j(I_j)$ is the *cost-to-go function* whose value is the minimal cost when starting at time period t with state I_t and continuing until the end of the time horizon.

We note that (3) is a special case of DP formulation (1) in the following sense. (i) We reverse the order of recursion so that $t = T + 1 - j$, (ii) we set $m = 1$ and for convenience drop the index i from the double index t, i, (iii) the information given in $A_t(I_t)$ is the description of the action space $\mathcal{A}_{T+1-j}(I_{T+1-j})$ and of the random variable D_{T+1-j}, (iv) when considering time period t in (1), instead of using all previously-calculated $\{z_r\}_{r<t}$, we use only z_{t-1}, i.e., when considering time period $T + 1 - t$ in (3), we only use the previously-calculated cost-to-go function z_{T+2-t}, and (v) $f_0 = g_{T+1}$ and for $1 \le t \le T$ we have (let $j = T + 1 - t$)

$$f_t = \min_{x_j \in \mathcal{A}_j(I_j)} E_{D_j} \left\{ g_j(I_j, x_j, D_j) + z_{j+1}(\text{tran}_j(I_j, x_j, D_j)) \right\}.$$

3.2 Our model

Denote by $z^{\max}$ ($z^{\min}$) an upper (lower) bound on the value (strictly-positive value) that $z_{t,i}$ can achieve, $0 \le t \le T, 1 \le i \le m$, respectively. Denote by $U_z = \max\left\{\frac{z^{\max}}{z^{\min}}, 2\right\}$, by $U_{\mathcal{S}}$ an upper bound on the cardinality of $\mathcal{S}_{t,i}$, and by t_f an upper bound on the time needed to evaluate $f_{t,i}$ when assuming that calls to the previous $\{z_{r,j}\}_{r\le t, j\le i: r+j<t+i}$ are done in constant time $0 \le t \le T, 1 \le i \le m$.

Our model applies for DPs that satisfy the four conditions below. The first two conditions relate to the structure of the functions; Condition 1 requires monotonicity, and Condition 2 requires that their domain and range are not “too big”. The objective of Condition 3 is to make sure that things are not “ruined” when we use approximate (as opposed to exact) functions in the DP recursion. The last condition comes to verify that the computation time of each of the functions is not “too big”.

Condition 1 (monotonicity) *The function $z_{t,i}(\cdot)$ is monotone, $0 \le t \le T, 1 \le i \le m$.*

Condition 2 (bounded size of elements) *The values $\log U_z$ and $\log U_{\mathcal{S}}$ are polynomially bounded in the (binary) input size.*

Condition 3 (approximation invariance under approximate input) *For any real number $L \ge 1$ and any $I_{t,i}, A_{t,i}(I_{t,i})$, if $\{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}$ are L-approximation functions of $\{z_{r,j}\}_{r\le t, j\le i: r+j<t+i}$, then $f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}\right)$ is an L-approximation value of $f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r\le t, j\le i: r+j<t+i}\right)$.*

Condition 4 (computation time) *For all $t = 0, \dots, T$ and $i = 1, \dots, m$, and for any real number $L \ge 1$, if $\{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}$ are L-approximation functions of $\{z_{r,j}\}_{r\le t, j\le i: r+j<t+i}$, then one of the following holds:*

(i) There exists an upper bound $t_{\tilde{f}}$ on the calculation time of $\bar{z}_{t,i} = f_{t,i}((I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}))$, that is polynomially bounded in the (binary) input size and in $t_{\tilde{z}_{r,j}}$.

(ii) (Existence of an auxiliary FPTAS) For every real-valued parameter $\epsilon > 0$ there exists an FPTAS, denoted by $\mathrm{FPTAS}\bar{f}_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}, \epsilon\right)$, that computes a $(1+\epsilon)L$-approximation value for $f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r\le t, j\le i: r+j<t+i}\right)$, and its running time, denoted by $t_{\bar{\bar{f}}}$, is polynomially bounded in the (binary) input size, $\frac{1}{\epsilon}$, and in $t_{\tilde{z}_{r,j}}$.

If DP (1) satisfies Conditions 1-3 and Condition 4(i), then we call it a *direct DP*. If the DP satisfies Conditions 1-3 and Condition 4(ii), we have only indirect (and approximated) access to it, and therefore call it an *indirect DP*. Among reasons for a DP not to be direct is that the calculation of $f_{t,i}$ takes exponential time because the action space is exponential, and/or it involves functions that are not computable in polynomial time. The Capacity Expansion application in Section 3.7 demonstrates these two issues.

Main differences from the DP model of [HKL$^+$14]. The crux in the differences between our DP model and the one of [HKL$^+$14] (which we overview in Appendix C.1) lies in the fact that while (1) is an implicit recursive formula (by means of the implicit function $f_{t,i}$), the recursive formula (3.4) in [HKL$^+$14], i.e., (3), is way too specific. Specifically, since the publication of [HKL$^+$14] there is not even a single problem which was solved directly through that framework. However, there are many problems which were solved using the techniques developed in that work, i.e., problems 4-11 in Table 1. This showcases that while [HKL$^+$14] developed a very useful technique, its framework is too specific in order to be applied to a large body of problems. As we will show in detail in Appendices C.1 and C.2, our stochastic DP model fully generalizes the monotone DP models of [HKL$^+$14, Hal19b] in the sense that the class of problems that can be formulated as to satisfy (1) strictly contains the one that satisfy (3).

3.3 FPTAS for direct DP

Algorithm 3 serves as an FPTAS for direct DPs. The essence of the algorithm lies in line 6, where it calls FPTASCompress that builds an approximate succinct representation of $\bar{z}_{t,i}$. Due to the approximation of approximation rule (Proposition 2.2(7)), a multiplicative error of K is introduced in every iteration of the algorithm. Due to the value of K set in line 2, and the number of iterations, we get that $\tilde{z}_{T,m}$ is a $(1+\epsilon)$-approximation function of $z_{T,m}$.

Algorithm 3 FPTAS for direct monotone DPs

```
1: Function FPTASDirectDP($\epsilon$)
2: $K \leftarrow \sqrt[m(T+1)]{1+\epsilon}$
3: for $t = 0, \dots, T$ do
4:     for $i = 1, \dots, m$ do
5:         $\bar{z}_{t,i}(I_{t,i}) = f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r \le t, j \le i : r+j < t+i}\right)$
6:         $\tilde{z}_{t,i} \leftarrow$ FPTASCompress($\bar{z}_{t,i}, K-1$)
7:     end for
8: end for
9: return $\tilde{z}_{T,m}$
```

Remark 3 *Usually, the monotonicity of $z_{t,i}(\cdot)$, $0 \le t \le T, 1 \le i \le m$, follows from the monotonicity of $\{z_{r,j}\}_{r \le t, j \le i : r+j < t+i}$, e.g., if $z_{t,i}$ is either a sum or minimization of functions from former stages. In these cases $\{\tilde{z}_{r,j}\}_{r \le t, j \le i : r+j < t+i}$ are monotone as an output of* FPTASCompress, *and so are $\bar{z}_{t,i}$ monotone, $0 \le t \le T, 1 \le i \le m$. Hence, for simplicity, we assume over Algorithm 3 and the proof of Theorem 3.1 that $\bar{z}_{t,i}$ are monotone, $0 \le t \le T, 1 \le i \le m$. If this is not the case, i.e., there exist t, i such that $\bar{z}_{t,i}$ is not monotone, then we will replace the call to* FPTASCompress *in line 6 of Algorithm 3 with a call to* FPTASIndirectCompress *(with the same parameters), as is done in Algorithm 4.*

Theorem 3.1 *Suppose that DP (1) is a direct DP. For every $0 < \epsilon < 1$, the output of Algorithm 3, $\tilde{z}_{T,m}$, is a $(1+\epsilon)$-approximation of $z_{T,m}$. The running time of the algorithm is $O\left(\frac{t_f (mT)^2}{\epsilon} \log \frac{mT \log U_z}{\epsilon} \log U_z \log U_{\mathcal{S}}\right) =$ $O\left(\frac{t_{\tilde{f}} (mT)^2}{\epsilon} \log U_z \log U_{\mathcal{S}}\right)$, which is polynomial in the (binary) input size and ϵ^{-1}.*

Recall that t_f is an upper bound on the evaluation time of $f_{t,i}((I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r \le t, j \le i : r+j < t+i}))$, and $t_{\tilde{f}}$ is an upper bound on the evaluation time of $f_{t,i}((I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r \le t, j \le i : r+j < t+i}))$.

Proof. We first show by induction that $\tilde{z}_{t,i}$ is a monotone $K^{i(t+1)}$-approximation of $z_{t,i}$, for $0 \le t \le T, 1 \le i \le m$. For the base case of $t = 0, i = 1$, we have $\bar{z}_{0,1} = z_{0,1}$, since $\{\tilde{z}_{r,j}\}_{r \le 0, j \le 1 : r+j < 1}$ is an empty set. Hence, by Proposition 2.5 FPTASCompress($\bar{z}_{0,1}, K-1$) returns a monotone K-approximation function of $z_{0,1}$.

We next assume that the claim holds for every $r \le t, j \le i : r+j < t+i$; so, $\tilde{z}_{r,j}$ is a monotone $K^{j(r+1)}$-approximation of $z_{r,j}$ for $r \le t, j \le i : r+j < t+i$. By the induction hypothesis $\{\tilde{z}_{r,j}\}_{r \le t, j \le i : r+j < t+i}$ are monotone $K^{i(t+1)-1}$-approximation functions of $\{z_{r,j}\}_{r \le t, j \le i : r+j < t+i}$, and by Condition 3, $\bar{z}_{t,i}$ is a monotone (see Remark 3) $K^{i(t+1)-1}$-approximation function of $z_{t,i}$. Due to Proposition 2.5, FPTASCompress($\bar{z}_{t,i}, K-1$) returns a monotone K-approximation of $\bar{z}_{t,i}$. Then, by the approximation of approximation rule (Proposition 2.2(7)), $\tilde{z}_{t,i}$ is a $K^{i(t+1)}$-approximation of $z_{t,i}$. This concludes the induction proof.

We now apply the claim for $t = T, i = m$ and get that $\tilde{z}_{T,m}$ is a $K^{m(T+1)}$-approximation function of $z_{T,m}$. By the value of K set in line 2 we get that $K^{m(T+1)} = 1 + \epsilon$ and the claimed approximation ratio follows.

We next turn to analyze the running time. Algorithm 3 performs $m(T+1)$ iterations, in each of them the running time is dominated by the call to FPTASCompress. By Proposition 2.5, the running time of FPTASCompress is $O\left(t_f \log\log_K U_z \log_K U_z \log U_{\mathcal{S}}\right) = O\left(t_{\tilde{f}} \log_K U_z \log U_{\mathcal{S}}\right)$ (the calculation time of $\bar{z}_{t,i}$, $1 \le t \le T, 1 \le i \le m$, is $t_{\tilde{f}} = O(t_f \log\log_K U_z)$, where the term $\log\log_K U_z$ is due to the query time to $\tilde{z}_{r,j}$ from previous stages). Since $\log_K U_z = O\left(\frac{mT \log U_z}{\log(1+\epsilon)}\right) = O\left(\frac{mT \log U_z}{\epsilon}\right)$ for every $0 < \epsilon < 1$, the running time of a single call to FPTASCompress is $O\left(\frac{mT}{\epsilon} t_f \log \frac{mT \log U_z}{\epsilon} \log U_z \log U_{\mathcal{S}}\right) = O\left(\frac{mT}{\epsilon} t_{\tilde{f}} \log U_z \log U_{\mathcal{S}}\right)$. Recalling that there are $m(T+1)$ iterations, the claimed running time follows. By Conditions 2 and 4(i), this term is indeed polynomial in the (binary) input size and $1/\epsilon$. □

In the next two sections we give simple examples of direct DPs. More examples of direct DPs, some of which are quite sophisticated, are given in Appendix A.

3.4 First example of a direct DP - 0/1 Knapsack

In this section we show how a classical 0/1 Knapsack DP formulation can be viewed as a direct DP. Although our FPTAS is not the fastest one known for this problem in terms of the dependency in n and $1/\epsilon$, see, e.g., [KPP04], it is rather simple once Algorithm 3 is understood. We note in passing that a more general version of this problem, i.e., the non-linear Knapsack, is solved via the method of K-approximation sets and functions, and the FPTAS running time is given in Appendix C.1.1(2).

In the classical 0/1 Knapsack problem one is given a positive integer knapsack volume B and a set of n items. For every item $i = 1, \dots, n$ two positive integers are given: its volume $v_i \le B$ and its profit π_i that is resulted if the item is placed in the knapsack. The objective is to select a subset of the items that maximizes the total profit while not exceeding the knapsack volume B. The input consists of B and the n pairs (v_i, π_i), $i = 1, \dots, n$.

Let $z_i(j)$ be the maximal profit, considering only items $1, \dots, i$, and using at most j volume. The DP recurrence reads as follows.

$$z_i(j) = \begin{cases} 0 & i = 0, \\ z_{i-1}(j) & j < v_i, i \ge 1, \\ \max\{z_{i-1}(j), \pi_i + z_{i-1}(j - v_i)\} & j \ge v_i, i \ge 1. \end{cases} \tag{4}$$

The optimal solution is $z_n(B)$.

We next show that DP (4) is a direct DP. We first set the parameters $T = n$ and $m = 1$. Next, we set the functions $f_{i,1}(\cdot) = z_i(\cdot)$, where z_i are defined in (4), $0 \le t \le T$. Clearly, the functions $z_i(\cdot)$ are non-decreasing over $\{0, \dots, B\}$, and so Condition 1 holds. Condition 2 holds with $U_{\mathcal{S}} = B$ and $U_z = \Pi := \sum_{i=1}^n \pi_i$. Condition 3 holds by the summation, composition, and maximization of approximation rules (Proposition 2.2(3,4,5)), and Condition 4(i) holds since z_i can be calculated in $t_f = O(1)$ time for every $1 \le i \le n$, as a maximization of two former z functions. Therefore, by Theorem 3.1, Algorithm 3 serves as an FPTAS for our problem, and its running time is $O\left(\frac{n^2}{\epsilon} \log \frac{n \log \Pi}{\epsilon} \log B \log \Pi\right)$.

3.5 Second example for direct DP - Counting 0/1 knapsack solutions

In the counting version of 0/1 knapsack, the objective is to count the number of feasible solutions. The input consists of the volumes $v_1, \dots, v_n \in \mathbb{N}$ of the n items, and an integer number B, the capacity of the knapsack. The problem is to determine the cardinality of $\left\{S \subseteq \{1, \dots, n\} \middle| \sum_{k \in S} v_k \le C\right\}$. This section is based on [Alo16, Appx. B]. The first FPTAS for this problem was introduced by [ŠVV12]. Their FPTAS is faster than ours by a factor of $\log C$, but we think ours is much simpler. [Hal16] considers the more general problem of counting integer knapsack solutions, introduces a somewhat involved DP formulation, and designs an FPTAS by the method of K-approximation sets and functions, whose running time (when applied to 0/1 knapsack) is equal to our FPTAS. For simplicity, we bring here the FPTAS for the 0/1 version of the problem as is reported in [Alo16, Appx. B].

We next formulate the problem as a DP. Let $z_i(j)$ be the number of feasible solutions when considering items $1, \dots, i$, and the knapsack capacity is j. This problem admits the following DP recursion:

$$z_i(j) = \begin{cases} 0 & i = 0 \text{ or } j < 0, \\ z_{i-1}(j) + z_{i-1}(j - v_i) & 1 \le i \le n,\ 0 \le j \le C. \end{cases} \tag{5}$$

Next, we show that DP (5) fits into our framework. Set $T = n$ and $m = 1$. It is easy to see that $z_i(\cdot)$ is non-decreasing, $i = 1, \dots, n$, and hence Condition 1 holds. Condition 2 holds since $U_z = 2^n$ and $U_{\mathcal{S}} = C$. Regarding Condition 3, it holds due to the summation and composition of approximation rules (Proposition 2.2(3,4)). Condition 4(i) holds, as functions z_i are calculated by adding two previously-calculated z functions, $0 \le i \le n$, therefore $t_f = O(1)$. From Theorem 3.1 we conclude that Algorithm 3 serves as an FPTAS to the problem, and it runs in time $O\left(\frac{n^3}{\epsilon} \log \frac{n}{\epsilon} \log C\right)$, which is equal to the result presented in [Alo16].

In Appendix A.6 we deal with a related problem - counting integer knapsack solutions. As is shown therein, while the DP formulation we give is way more complex than (5), we still use the same FPTAS algorithm, i.e., Algorithm 3. This exemplifies once more one of the contributions of this paper - trading-off FPTAS design with DP formulations.

3.6 FPTAS for indirect DP

In this section we unveil the full strength of our DP framework, e.g., in the cases where neither the single-stage cost functions, nor the Bellman equation itself are computable in polynomial time for any given state variable. A possible reason for this to happen is when the action space is exponential in the input size. This is the case with problems 3 and 10-18 in Table 1. Moreover, Woeginger writes that his framework does not apply to DP formulations in which the action space size is pseudopolynomial in the input size, and explicitly mentions the single-item economic lotsizing problem as such a problem [Woe00, Sec. 8]. We note in passing that we provide FPTASes for all aforementioned 10 problems, and in particular to the economic lot-sizing problem, see the specific sections numbers mentioned in Table 1 w.r.t. these problems.

Algorithm 4 serves as an FPTAS for indirect DPs. While it is similar to Algorithm 3, it is more powerful because it repeatedly calls *two* FPTASes in each iteration - one in line 5 to approximate a value oracle, and one in line 6 (similarly to line 6 in Algorithm 3), to build an approximate succinct representation of a value oracle to $\bar{z}_{t,i}$. By the approximation of approximation rule (Proposition 2.2(7)), the overall accumulated error in lines 5 and 6 is K^2 for every iteration. Since there are $m(T+1)$ iterations and due to the value of K set in line 2, we get that $\tilde{z}_{T,m}$ is indeed a $(1+\epsilon)$-approximation function of $z_{T,m}$

Theorem 3.2 *Suppose that DP (1) is an indirect DP. For every $0 < \epsilon < 1$, function $\tilde{z}_{T,m}$ returned by line 9 of Algorithm 4 is a $(1+\epsilon)$-approximation function of $z_{T,m}$. The running time of the algorithm is $O\left(\frac{(mT)^2}{\epsilon} t_{\bar{f}} \log \frac{mT \log U_z}{\epsilon} \log U_z \log U_{\mathcal{S}}\right) = O\left(\frac{(mT)^2}{\epsilon} t_{\bar{\bar{f}}} \log U_z \log U_{\mathcal{S}}\right)$, which is polynomial in the (binary) input size and $\frac{1}{\epsilon}$.*

Recall that $t_{\bar{f}}$ is an upper bound on the running time of $\text{FPTAS}\bar{f}_{t,i}((I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r \le t, j \le i: r+j < t+i}, \epsilon))$, and $t_{\bar{\bar{f}}}$ is an upper bound on the running time of $\text{FPTAS}\bar{f}_{t,i}((I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r \le t, j \le i: r+j < t+i}, \epsilon))$.

Algorithm 4 FPTAS for indirect monotone DPs

```
1: Function FPTASIndirectDP($\epsilon$)
2: $K \leftarrow \sqrt[2m(T+1)]{1+\epsilon}$
3: for $t = 0, \ldots, T$ do
4:     for $i = 1, \ldots, m$ do
5:         $\bar{z}_{t,i}(I_{t,i}) = \text{FPTAS}\bar{f}_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r \le t, j \le i : r+j < t+i}, K-1\right)$ /* as defined in Condition 4(ii) */
6:         $\tilde{z}_{t,i} \leftarrow \text{FPTASIndirectCompress}(\bar{z}_{t,i}, K-1)$
7:     end for
8: end for
9: return $\tilde{z}_{T,m}$
```

The proof of Theorem 3.2 is similar to the one of Theorem 3.1, and therefore is deferred to Appendix F; the differences stem from the call to the auxiliary FPTAS in line 5. The main differences from Algorithm 3 are (i) the multiplicative error caused in each iteration is K^2 (as opposed to K), as explained above, and (ii) we use FPTASIndirectCompress (as opposed to FPTASCompress), since the auxiliary FPTAS in line 5 computes the function $\bar{z}_{t,i}(\cdot)$ which is not necessarily monotone.

We end this section by stating a powerful proposition about an auxiliary FPTAS that we will use in the case where the action space size is exponential. For simplicity of presentation the proposition refers to DP equation (3), which is a special case of DP (1) and in where we have $m = 1$. Furthermore, in the sequel, as a convention, if $m = 1$, i.e., in every time $0 \le t \le T$ there is only one function to evaluate, we omit the second index in the functions and states.

Proposition 3.3 (FPTAS to compute a single Bellman recursion over an exponential-size action space, based on [HKL$^+$14, Prop. 7.1]) *Suppose that DP (1) is an indirect DP where:*

$$z_t(I_t) = f_t\left(I_t, A_t(I_t), \{z_r\}_{r<t}\right) := \min_{x_t \in A_t(I_t)} E_{D_t}\left\{g_t(I_t, x_t, D_t) + z_{t-1}(\text{tran}_t(I_t, x_t, D_t))\right\}, \tag{6}$$

where $\text{tran}_t(I_t, x_t, D_t)$ is the transition function from time t to time $t-1$ and $g_t(I_t, x_t, D_t)$ is the single-period cost function, when the state in the beginning of time t is I_t, the action taken is x_t and D_t is a random variable realized at time t. Moreover, assume both these functions are monotone in their first two variables. Suppose further that the distribution function of the random variable D_t is given as the n_t pairs $\left(d_{t,1}, Prob(D_t = d_{t,1})\right), \ldots, \left(d_{t,n_t}, Prob(D_t = d_{t,n_t})\right)$. Let $\epsilon > 0$, L', L'', t, and I_t be fixed values, where $K' = 1+\epsilon, L' \ge 1$, $1 \le L'' \le K'L'$, $I_t \in \mathcal{S}_t$, and $t \in \{1, \ldots, T\}$. Let $\tilde{z}_{t-1}$ be a monotone L'-approximation of z_{t-1}, and W_ϵ be a $(1+\epsilon)$-approximation set of $\tilde{z}_{t-1}$. Let $W_\epsilon^{-1}(I_t) = \bigcup_{i=1}^{n_t} \text{tran}_{t,i}^{-1}(I_t, W_\epsilon)$, where $\text{tran}_{t,i}^{-1}(I_t, W_\epsilon) = \big\{ \max\{x_t \mid \text{tran}_t(I_t, x_t, d_{t,i}) \le w\}, \min\{x_t \mid \text{tran}_t(I_t, x_t, d_{t,i}) \ge w\} \bigm| w \in W_\epsilon \big\}$ if tran_t is nondecreasing in its second variable, and $\text{tran}_{t,i}^{-1}(I_t, W_\epsilon) = \big\{ \max\{x_t \mid \text{tran}_t(I_t, x_t, d_{t,i}) \ge w\}, \min\{x_t \mid \text{tran}_t(I_t, x_t, d_{t,i}) \le w\} \bigm| w \in W_\epsilon \big\}$ if tran_t is nonincreasing in its second variable. Let $\tilde{g}_t(I_t, \cdot, D_t)$ be a monotone L''-approximation of $g_t(I_t, \cdot, D_t)$. Let

$$\bar{z}_t(I_t) = \text{FPTAS}\bar{f}_t\left(I_t, A_t(I_t), \{\tilde{z}_r\}_{r<t}, \epsilon\right) := \min_{x_t \in W_\epsilon^{-1}(I_t)} E_{D_t}\left\{\tilde{g}_t(I_t, x_t, D_t) + \tilde{z}_{t-1}(\text{tran}_t(I_t, x_t, D_t))\right\}. \tag{7}$$

Then, $\bar{z}_t(I_t)$ is a $(1+\epsilon)L'$-approximation value of $z_t(I_t)$, and it can be determined in $O\big(n_t^2(t_{\tilde{g}_t} + t_{\text{tran}_t} + t_{\tilde{z}_{t-1}})|W_\epsilon|\big)$ time.

We demonstrate the power and applicability of Proposition 3.3 in the next section, as well as in Appendices B.1 and C.1. In Appendix B.2 we use a similar proposition, namely, Proposition B.1, which serves as an FPTAS to compute a different single Bellman recursion over an exponential-size action space.

3.7 An example of an indirect DP - capacity expansion

In this section we show how to formulate the capacity expansion problem as an indirect DP. We bring it here as a somewhat sophisticated example for an application which is an indirect DP due to (i) the action space being exponential in the (binary) problem input size, and (ii) the need to compute functions that are NP-hard to compute – the functions Π_t in (9) below are such functions, $1 \leq t \leq T$.

Before formulating our problem we give a proposition which serves as an easy and efficient tool to build from the set W that is constructed by FPTASCompress or FPTASIndirectCompress a 1-approximation set for the function that they return. We use this proposition later on in this section.

Proposition 3.4 *(based on [HKL+14, Prop. 4.5(2)]) Let $\epsilon > 0$, $L \geq 1$ be a real numbers, $K = 1+\epsilon$, and let $\varphi : D \to \mathbb{R}^+$ be a monotone nonnegative function over a finite domain D of real numbers. Let $\breve{\varphi}$ =FPTASCompress(φ, ϵ) be the function returned by a call to* FPTASCompress *and let W be the set constructed by* FPTASCompress. *Let $W^- = \{prev(x, D) \mid x \in W \setminus \{D^{\min}\}\}$ and $W^+ = \{next(x, D) \mid x \in W \setminus \{D^{\max}\}\}$. Then, W is a K-approximation set of $\breve{\varphi}$, and $W \cup W^- \cup W^+$ is a 1-approximation set of $\breve{\varphi}$. If φ is nondecreasing, then $W \cup W^+$ is a 1-approximation set of $\breve{\varphi}$. If φ is nonincreasing, then $W \cup W^-$ is a 1-approximation set of $\breve{\varphi}$. Furthermore, the same claims hold for $W, W \cup W^- \cup W^+, W \cup W^+$ and $W \cup W^-$ if $\bar{\varphi}$ is an unnecessarily monotone L-approximation of φ, $\breve{\varphi}$ =FPTASIndirectCompress$(\bar{\varphi}, \epsilon)$ is the function returned by a call to* FPTASIndirectCompress *and W is the set constructed by* FPTASIndirectCompress.

We turn now to formulate our problem. Consider the following multi-period capacity expansion problem in telecommunication network planning [San95]: Given a set of transmission technologies $\{1, \ldots, n\}$ such as copper cables of various sizes, optical fiber cables with different bit rates, etc., we would like to determine a combination of sizes of these technologies to be installed in each time period. Our objective is to satisfy a given demand of circuits in each time period of the planning horizon at minimum cost. The problem is formulated as follows:

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^{T} \sum_{i=1}^{n} \pi_{t,i}(x_{t,i}) & \\ \text{subject to} & \sum_{\ell=1}^{t} \sum_{i=1}^{n} v_i x_{\ell,i} \geq C_t, & t = 1, \ldots, T \\ & x_{t,i} \in \mathbb{Z}^+, & t = 1, \ldots, T;\ i = 1, \ldots, n. \end{array} \tag{8}$$

In this formulation, the planning horizon is divided into T time periods. Variable $x_{t,i}$ is the amount of technology i installed in period t. Parameter v_i is the unit capacity of technology i, where $v_i > 0$. Parameter C_t is the accumulated demand over time periods $1, \ldots, t$; that is, $C_t = \sum_{\ell=1}^{t} c_\ell$, where c_ℓ is the added demand requirement (expansion) in period ℓ. We assume that v_i and c_ℓ are integers for $i = 1, \ldots, n$ and $\ell = 1, \ldots, T$. The quantity $\pi_{t,i}(x_{t,i})$ is the present value of the monetary resources spent on technology i in period t, where $\pi_{t,i} : \mathbb{Z}^+ \to \mathbb{Q}^+$ is a nondecreasing function. The input data for the problem consists of (i) the number of time periods T, (ii) the accumulated demand C_t (for each $t = 1, \ldots, T$), (iii) the number of transmission technologies n, (iv) the unit capacity v_i of technology i (for each $i = 1, \ldots, n$), and (v) an oracle that computes function $\pi_{t,i}$ (for each time period t and technology i). We assume that the values of its domain are polynomially bounded in the (binary) input size.

Consider a single time period t, and let $\Pi_t(X_t)$ be the optimal cost to meet X_t units of demand in that period (assuming that there is no capacity carried over from the previous period). Problem (8) can be formulated as a pseudo-polynomial time DP as follows: Define $z_t(I_j)$ as the minimum total cost to meet the demands of periods $t \ldots, T$, given that there are already I_t units of accumulated capacity available from period $t-1$ (i.e., $I_t = \sum_{\ell=1}^{t-1} \sum_{i=1}^{n} v_i x_{\ell,i}$), for $t = 1, \ldots, T$ and $I_t = 0, \ldots, C_T$.

The recurrence relation is

$$z_t(I_t) = \begin{cases} 0 & t = T+1, I_t \geq 0, \\ \min_{X_t=(C_t-I_t)^+,\dots,C_T} \Big\{ \Pi_t(X_t) + z_{t+1}\big(\min\{I_t + X_t, C_T\}\big) \Big\} & I_t = 0,\dots,C_T \text{ and } t = 1,\dots,T, \end{cases} \tag{9}$$

where X_t is the increase in capacity in period t. The optimal solution value is $z_1(0)$.

The value of $\Pi_t(X_t)$ is the optimal objective value of the following nonlinear minimization Knapsack problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} \pi_{t,i}(x_{t,i}) \\ \text{subject to} & \sum_{i=1}^{n} v_i x_{t,i} \geq X_t \\ & x_{t,i} = 0,\dots,\bar{x}_{t,i}, \quad i = 1,\dots,n. \end{array} \tag{10}$$

Here, $\bar{x}_{t,i}$ represents an upper bound on $x_{t,i}$. For example, we may set $\bar{x}_{t,i} = \lceil X_t/v_i \rceil$. Problem (10) is NP-hard and admits an FPTAS that constructs the oracle $\tilde{\Pi}_t(\cdot)$ with running time $O\big(\frac{n^3}{\epsilon^2} \log \frac{n}{\epsilon} \log C_T \log \Pi\big)$, where $\Pi \leq \max_{t=1,\dots,T} \min_{i=1,\dots,n} \pi_{t,i}(\lceil C_T/v_i \rceil)$ is an upper bound on the value of (10) [HKL+14, Appx. A.2]. Once we run this FPTAS, any query to $\tilde{\Pi}_t(\cdot)$ is realized in time $O\big(\log(\frac{T}{\epsilon}\log\Pi)\big)$. As a preprocessing phase for the FPTAS for problem (8), we will run such an FPTAS to construct each one of $\tilde{\Pi}_t(\cdot)$, $t = 1,\dots,T$.

We next show that DP (9) is an indirect DP. We first note that while DP (1) is a forward recursion formulation, DP (9) is a backward recursion. Therefore, we reverse the order of time periods to $t \leftarrow T+1-t$. We set $m = 1$. Note that the functions $z_t(\cdot)$ are non-increasing, as the more accumulated capacity available from previous stages, the less the payment in the current period, and hence Condition 1 holds. Next, we set $U_z = T\Pi$ and $U_{\mathcal{S}} = C_T$, and therefore get that Condition 2 holds as well. As for Condition 3, it follows by the summation, composition, and minimization of approximation rules (Proposition 2.2 (3,4,6)). Regarding Condition 4(ii), note that (i) since the size of the action space is pseudopolynomial in the (binary) input size (it can be up to C_T), and (ii) computing the value of $g_t = \Pi_t$ is NP-hard, we cannot compute in polynomial time the minimization part of DP (9) directly. We will use instead Proposition 3.3 with the following parameters: $\tilde{g}_t$ is $\tilde{\Pi}_{t+1}$ as described above, $n_t = 1$ as we are dealing with a deterministic equation, $L' = L$, where L is as defined in Condition 4, we set L'' to be as small as needed to satisfy the condition $L'' \leq (1+\epsilon)L'$, we let $W_\epsilon = W \cup W^-$, where W is the approximation set constructed by FPTASIndirectCompress$(\bar{z}_{t-1}, \epsilon)$, so by Proposition 3.4 W_ϵ is a 1-approximation set of $\tilde{z}_{t+1}$ (and consequently – also a $(1+\epsilon)$-approximation set of it). Then, by Proposition 3.3 we get that $\bar{z}_t(I_t)$ is a $(1+\epsilon)L$-approximation value of $z_t(I_t)$, so Condition 4(ii) also holds.

Since DP (9) is an indirect DP, by Theorem 3.2, it admits an FPTAS which runs in time

$$O\left(\frac{T^2}{\epsilon} t_{\bar{f}} \log U_z \log U_{\mathcal{S}}\right) \tag{11}$$

where $t_{\bar{f}}$ is an upper bound on the computational time of $\bar{z}_t(I_t) = \text{FPTAS}\bar{f}_t\,(I_t, A_t(I_t), \{\tilde{z}_r\}_{r<t}, \epsilon)$, i.e., the value of (9) when using $\tilde{z}_{t+1}$ (i.e., an FPTAS for z_{t+1}) instead of the exact z_{t+1} and using $\tilde{\Pi}_t(X_t)$ (i.e., an FPTAS to $\Pi_t(X_t)$) instead of the exact $\Pi_t(X_t)$. As written above, we realize the calculation of $\bar{z}_t(I_t)$ by Proposition 3.3 where $t_{\tilde{g}_t} = O\big(\log(\frac{T}{\epsilon}\log\Pi)\big)$ is the query time of $\tilde{g}_t$ and $t_{\tilde{z}_{t+1}} = O\big(\log(\frac{T}{\epsilon}\log(T\Pi))\big)$ is the query time of $\tilde{z}_{t+1}(\cdot)$. Because the size of an approximation set of $\tilde{z}_{t+1}$ is $O\big(\frac{T}{\epsilon}\log(T\Pi)\big)$, we get by Proposition 3.3 that

$$t_{\bar{z}_t} = O\left(\frac{T}{\epsilon}\big(\log(\frac{T}{\epsilon}\log\Pi) + \log(\frac{T}{\epsilon}\log(T\Pi))\big)\log(T\Pi)\right)$$

Substituting the values of $t_{\bar{z}_t}, U_z, U_{\mathcal{S}}$ with their values as explained above, and taking into account the computational effort for performing the preprocessing phase, we conclude that the total running time of the FPTAS for problem (8) is

$$O\left(\frac{T^3}{\epsilon^2}\log\left(\frac{T}{\epsilon}\log(T\Pi)\right)\log^2(T\Pi)\log C_T + \frac{Tn^3}{\epsilon^2}\log\frac{n}{\epsilon}\log C_T\log\Pi\right).$$

We note in passing that the following FPTASes are involved in the design of the FPTAS for the capacity expansion problem: for executing FPTAS$\bar{f}_{t,i}\,(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r\le t,j\le i:r+j<t+i}, K-1)$ in line 5 of Algorithm 4 we apply the FPTAS (7) in Proposition 3.3, that in turn, for evaluating $\tilde{g}_t$, calls the oracle $\tilde{\Pi}_t(\cdot)$ constructed by the FPTAS for (10). Moreover, when executing FPTASIndirectCompress in line 6 of Algorithm 4, this FPTAS calls FPTASIndirectApxSet.

4 Concluding remarks and future research

In this paper we introduce a new framework for the automatic generation of FPTASes that unifies and simplifies previous results, and consequently enables and eases future design of FPTASes.

We introduce our framework for a pretty general class of monotone functions, i.e., (1) coupled with Condition 1. However, we can tradeoff the monotonicity requirement in Condition 1 with a different requirement on the structure of the functions. That is, instead of calling FPTASCompress in line 6 of Algorithm 3, which is used to approximate the monotone functions $\bar{z}_{t,i}$, we will call another FPTAS to approximate these functions, using their structural property. Consequently, developing FPTASes for functions which are not necessarily monotone, but have other structure, is of interest. Due to the modularity of the method of K-approximation sets and functions, e.g., the Calculus of K-approximation functions, developing such FPTASes may be not too difficult. A natural candidate for future research following this approach is to consider convex functions, using the elements introduced in [HKL$^+$14] in what they call convex DPs.

References

[Alo16] T. Alon. Fully polynomial time approximation schemes (FPTAS) for some counting problems. *arXiv preprint arXiv:1611.00992*, 2016. Master thesis of the author, held in the department of statistics, the Hebrew University of Jerusalem, Israel.

[Ber95] D. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[DHV19] F. Díaz-Núñez, N. Halman, and Ó. Vásquez. The TV advertisements scheduling problem. *Optimization Letters*, 13:81–94, 2019.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, 1979.

[GMW18] P. Gawrychowski, L. Markin, and O. Weimann. A faster fptas for# knapsack. *arXiv preprint arXiv:1802.05791*, 2018.

[Hal16] N. Halman. A deterministic fully polynomial time approximation scheme for counting integer knapsack solutions made easy. *Theoretical Computer Science*, 645:41–47, 2016.

[Hal19a] N. Halman. Fully polynomial time approximation schemes for minimizing makespan of deteriorating jobs with non-linear processing times. *Journal of Scheduling*, 2019. To appear.

[Hal19b] N. Halman. Provably near-optimal approximation schemes for implicit stochastic and for sample-based dynamic programs. *INFORMS Journal on Computing*, 2019. To appear.

[HKL$^+$14] N. Halman, D. Klabjan, C.L. Li, J. Orlin, and D. Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *SIAM Journal on Discrete Mathematics*, 28(4):1725–1796, 2014.

[HKQ$^+$19] N. Halman, M. Kovalyov, A. Quilliot, D. Shabtay, and M. Zofi. Bi-criteria path problem with minimum length and maximum survival probability. *OR Spectrum*, 41(2):469–489, 2019.

[HKS18] N. Halman, H. Kellerer, and V. Strusevich. Approximation schemes for non-separable non-linear Boolean programming problems under nested knapsack constraints. *European Journal of Operational Research*, 270(2):435–447, 2018.

[HLS09] N. Halman, C.L. Li, and D. Simchi-Levi. Fully polynomial-time approximation schemes for time–cost tradeoff problems in series–parallel project networks. *Operations research letters*, 37(4):239–244, 2009.

[HN19] N. Halman and G. Nannicini. Toward breaking the curse of dimensionality: an FPTAS for stochastic dynamic programs with multidimensional actions and scalar states. *SIAM Journal on Optimization*, 29(2):1131–1163, 2019.

[HNO18] N. Halman, G. Nannicini, and J. Orlin. On the complexity of energy storage problems. *Discrete Optimization*, 28:31–53, 2018.

[HOS12] N. Halman, J. Orlin, and D. Simchi-Levi. Approximating the nonlinear newsvendor and single-item stochastic lot-sizing problems when data is given by an oracle. *Operations Research*, 60(2):429–446, 2012.

[IK75] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[KK98] M. Kovalyov and W. Kubiak. A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs. *Journal of Heuristics*, 3(4):287–297, 1998.

[KK12] M.Y. Kovalyov and W. Kubiak. A generic FPTAS for partition type optimisation problems. *International Journal of Planning and Scheduling*, 1:209–233, 2012.

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springler, Berlin, 2004.

[KS81] B. Korte and R. Schrader. On the existence of fast approximation schemes. In *Nonlinear Programming 4*, pages 415–437. Elsevier, 1981.

[KvdV98] W. Kubiak and S. van de Velde. Scheduling deteriorating jobs to minimize makespan. *Naval Research Logistics (NRL)*, 45(5):511–523, 1998.

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

[MŠW16] M. Mihalák, R. Šrámek, and P. Widmayer. Approximately counting approximately-shortest paths in directed acyclic graphs. *Theory of Computing Systems*, 58(1):45–59, 2016.

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

[RT13] R. Rizzi and A. Tomescu. Combinatorial decomposition approaches for efficient counting and random generation fptases. *arXiv preprint arXiv:1307.2347*, 2013.

[Sah76] S.K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

[San95] I. Saniee. An efficient algorithm for the multiperiod capacity expansion of one location in telecommunications. *Operations Research*, 43(1):187–190, 1995.

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

[Smi56] W. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

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

[ŠVV12] D. Štefankovič, S. Vempala, and E. Vigoda. A deterministic polynomial-time approximation scheme for counting knapsack solutions. *SIAM Journal on Computing*, 41(2):356–366, 2012.

[VTL82] J. Valdes, R. Tarjan, and E. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, 1982.

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

APPENDIX

A More examples of direct DPs

A.1 Counting n-tuples

The following problem is taken from Garey and Johnson's [GJ79, p. 225] list of NP-hard problems. This section is based upon [Alo16, Sec. 2]. To the best of our knowledge we give here the first FPTAS for this problem.

Given are finite ordered sets $X_1, \ldots, X_n \subseteq \mathbb{Z}^+$ and an integer number $B \in \mathbb{N}$. For all $1 \le i \le n$ and $1 \le j \le |X_i|$ denote by x_{ij} the j-th element of the set X_i. The problem is to find how many distinct n-tuples $(x_{1\ell_1}, \ldots, x_{n\ell_n})$, $1 \le \ell_i \le |X_i|$ are there such that $\sum_{k=1}^{n} x_{k\ell_k} \ge B$. The input of the problem consists of the sets $X_1, \ldots, X_n$ and the integer B.

We next formulate the problem as a DP. Let $z_i(j) = \#\left\{(x_{1\ell_1}, \ldots, x_{i\ell_i}) \,|\, \sum_{k=1}^{i} x_{k\ell_k} \ge j\right\}$ be a restriction of the problem to the sets $X_1, \ldots, X_i$ with weight-threshold j. The DP formulation reads as follows.

$$\begin{array}{rcll} z_1(j) & = & \#\{x_{1k} | 1 \le k \le |X_1|,\, x_{1k} \ge j\} & j = 0, \ldots, B, \\ z_i(j) & = & \sum_{k=1}^{|X_i|} z_{i-1}(j - x_{ik}) & i = 2, \ldots, n;\ j = 0, \ldots, B, \\ z_i(j) & = & \prod_{k=1}^{i} |X_k| & i = 1, \ldots, n;\ j < 0. \end{array} \tag{12}$$

As a preprocessing phase we calculate $\prod_{k=1}^{i} |X_k|$ for $i = 1, \ldots, n$ in total $O(n)$ time. Next, we show that the DP (12) fits into our framework. Set $T = n$ and $m = 1$. It is easy to see that $z_i(\cdot)$ is non-increasing, $i = 1, \ldots, n$, and hence Condition 1 holds. Condition 2 holds since $U_z = \prod_{k=1}^{n} |X_k|$ and $U_{\mathcal{S}} = B$. Regarding Condition 3, it holds due to the summation and composition of approximation rules (Proposition 2.2(3,4)). Condition 4(i) holds, as functions z_i are calculated by either adding previously-calculated z functions, or using values of $\prod_{k=1}^{i} |X_k|$ that were already calculated in the preprocessing phase, $1 \le i \le n$, and hence its calculation time is $t_f = O(M)$, where $M = \max_{i=1}^{n} |X_i|$. From Theorem 3.1 we conclude that Algorithm 3 serves as an FPTAS to the problem, and it runs in time $O(\frac{n^3 M \log M}{\epsilon} \log \frac{n \log M}{\epsilon} \log B)$.

A.2 Counting shortest s-t paths

Let $G = (V, E)$ be a directed acyclic graph (DAG) with vertices $v_1, \ldots, v_{|V|}$ that defines a topological ordering, i.e., there is no path from v_i to v_j unless $i < j$. We assume that $v_1 = s$ and $v_{|V|} = t$ (the vertices before s and after t are never visited in a path from s to t). Every edge ij has a positive edge-weight w_{ij}. Let L be an integer weight-threshold. The problem is to find the number of s-t paths with weight no larger than L.

The fastest FPTAS published for counting shortest s-t paths is due to Mihalák *et al.* and runs in time $O(|E||V|^3\epsilon^{-1} \log |V|)$ [MŠW16, Thm. 2]. There is an unpublished manuscript of Rizzi and Tomescu that contains an FPTAS with a better running time of $O\left(|E||V|^2\epsilon^{-1} \log \frac{|E|}{|V|} \left\lceil \frac{\log \frac{1}{\epsilon}}{\log |V|} \right\rceil\right)$ [RT13, Thm. 5]. Our FPTAS, although not strongly polynomial, is faster than the one of [RT13] for some parameters, and slower for others.

We next formulate the problem as a DP. Let $z_i(j)$ be the number of paths from 1 to i with length at most j. The recurrence reads as follows.

$$z_i(j) = \begin{cases} \sum_{ki \in E} z_k(j - w_{ki}) & 3 \leq i \leq |V| \, , \, 1 \leq j \leq L \\ 1 & i = 2 \, , \, w_{12} \leq j \leq L \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

We next show that DP (13) is a direct DP; to see this, set $T = |V| - 2, m = 1$. Note that here, to calculate z_i, we do not need all previously calculated z_i, but only those about incoming edges to v_i. By its definition, function $z_i(\cdot)$ is nondecreasing for every i, and so Condition 1 holds. In addition, $U_{\mathcal{S}} = L$ and $U_z = 2^{|V|}$, and so Condition 2 holds. Condition 3 holds by the summation and composition of approximation rules (Proposition 2.2(3,4)), and Condition 4(i) holds since the computation time at every iteration is $t_f = O(\text{InDeg}^{\max})$, where $\text{InDeg}^{\max}$ is the maximal indegree of the vertices $v_2, \ldots, v_{|V|}$. Then, according to Theorem 3.1 there exists an FPTAS for this problem. The running time stated in this theorem for this problem is $O\left(\frac{\text{InDeg}^{\max}|V|^3}{\epsilon} \log \frac{|V|}{\epsilon} \log L\right)$. However, a finer analysis for the running time of the same algorithm shows that its running is actually faster: in every iteration the computation time of $\bar{z}_i$ is $O(\text{InDeg}(v_i))$, and not $O(\text{InDeg}^{\max})$, where $\text{InDeg}(v_i)$ is the indegree of v_i. Hence, the running time of our algorithm is $O\left(\frac{|E||V|^2}{\epsilon} \log \frac{|V|}{\epsilon} \log L\right)$.

We note in passing that one can reduce counting n-tuples to counting shortest s-t paths as follows. Let the set of vertices contain the vertices $0, \ldots, n$. Between each pair of consecutive vertices $i-1, i$ there are $|X_i|$ parallel edges with weights from the set X_i, so $|E| = O(\sum_{i=1}^{n} |X_i|)$. We label vertex 0 as s and vertex n as t. There is a one-to-one correspondence from paths on this graph with weight less than B to n-tuples with weight less than B. Note that the existence of parallel edges is not necessary for the reduction; we can bisect each parallel edge of weight greater than one, creating an auxiliary vertex, to form a graph of the same functionality but without parallel edges, in where $|V| = O(\sum_{i=1}^{m} |X_i|)$. However, the running time of the specific algorithm for the n-tuples problem, which is described in Appendix A.1, is faster.

A.3 Bi-criteria path problem with maximum survival probability

The following problem is an extension of the shortest path problem. It is taken from [HKQ+19], see literature review therein. There is a directed graph $G = (V, E)$, where V is the set of vertices, and E is the set of edges. There are two fixed vertices $s, t \in V$. For each edge $(i, j) \in E$, let p_{ij} be a rational number, satisfying $0 < p_{ij} \leq 1$, which represents the survival probability while moving across the edge, and let $\ell_{ij} \in \mathbb{Z}^+$ be the length of the edge. Let $\pi = (s, i_1, \ldots, i_k, t)$ be a path from s to t. For every path π, two independent criteria are considered: (i) $F_1(\pi) = \prod_{(i,j)\in\pi} p_{ij}$ is the survival probability over the path, and (ii) $F_2(\pi) = \sum_{(i,j)\in\pi} \ell_{ij}$ is the path length. The problem of finding a simple path which maximizes F_1 and minimizes F_2, denoted by Product-Sum Path Problem, or PSPP for short, is considered in [HKQ+19]. That paper considers four versions of the problem, all of them are proved to be NP-hard. We focus here on two of these versions: (i) in PSPP Max-Probability the objective is to maximize $F_1(\pi)$ s.t. $F_2(\pi) \leq L$ where L is a given upper bound on the path length, and (ii) in PSPP-Pareto the objective is to find a set P of Pareto-optimal solutions (paths) such that there is at least one solution in P for each Pareto-optimal point (F_1, F_2). A point (F_1, F_2) is called Pareto-optimal if there exists a path π with values $F_1(\pi) = F_1$ and $F_2(\pi) = F_2$ and there exists no other path π' such that $F_1(\pi') \geq F_1(\pi)$ and $F_2(\pi') \leq F_2(\pi)$, with at least one of the inequalities being strict. In fact, we focus here in developing an FPTAS for the PSPP Max-Probability, and [HKQ+19] explain how this FPTAS serves also as an FPTAS for the PSPP-Pareto problem. The input of the problem consists of: (i) the sets V, E, (ii) the values p_{ij} and ℓ_{ij} for every $(i, j) \in E$, and (iii) L.

The following DP formulation of the problem is introduced in [HKQ+19]. Let $P_{v,j}(\ell)$ be the maximum survival probability of a path from s to j using exactly v edges, and length at most ℓ. Let $\ell_{\min} = \min_{(i,j)\in E} \ell_{ij}$.

$$P_{v,j}(\ell) = \begin{cases} 1 & v = 0, j = s, \ell_{\min} \le \ell \le L, \\ 0 & v = 0, j \neq s, \ell_{\min} \le \ell \le L, \\ 0 & \ell < \ell_{\min}, \\ \max_{(i,j)\in \mathrm{In}_j, \ell_{ij} \le \ell} p_{ij} \cdot P_{v-1,i}(\ell - \ell_{ij})\} & 1 \le v \le |V| - 1, j \neq s, \ell_{\min} \le \ell \le L, \end{cases} \tag{14}$$

where In_j is the set of incoming edges of j. The optimal value is $\max_{v=1,\dots,|V|-1} P_{v,t}(L)$. Note that if the graph is acyclic, then the vertices can be numbered in a topological order such that $i < j$ for any two vertices i and j connected by an edge, and the state variable v counting the number of arcs can be omitted.

We turn now to show that (14) is a direct DP. We first set the parameters $T = |V| - 1$ and $m = |V|$. The functions $f_{t,i}$ are set to the functions $P_{t,i}$ as defined in recursion (14). Note that here, to calculate $P_{v,\ell}$, we do not need all previously calculated $P_{v,\ell}$, but only those about incoming edges to vertex j at stage $v - 1$. The functions $P_{v,j}(\cdot)$ are non-decreasing, as a maximum of non-decreasing functions, $0 \le v \le T, 1 \le i \le m$, and so Condition 1 holds. Condition 2 holds since $U_z = \frac{1}{p_{\min}}$, where $p_{min} = \min_{(i,j)\in E} p_{ij}$, and $U_{\mathcal{S}} = L - \ell_{\min}$. Regarding Condition 3, it holds due to the linearity, composition, and maximization of approximation rules (Proposition 2.2(2,4,5)). Condition 4(i) holds, as functions $P_{v,j}(\cdot)$ are calculated by maximizing over $t_f = O(\mathrm{InDeg}^{\max})$ previously-calculated P functions, $0 \le v \le T, 1 \le i \le m$. From Theorem 3.1 we conclude that Algorithm 3 serves as an FPTAS to the problem. The running time as stated in this theorem is $O\left(\frac{|V|^4 \mathrm{InDeg}^{\max}}{\epsilon} \log \frac{|V| \log \frac{1}{p_{\min}}}{\epsilon} \log \frac{1}{p_{\min}} \log(L - \ell_{\min})\right)$. However, a finer analysis for the running time of the algorithm shows that its actual running is faster. It relies on the observation, that for a fixed v and j, the calculation in (14) requires only $O(\mathrm{In}_j)$ time, and since $E = \bigcup_{j\in V\setminus s} \mathrm{In}_j$, then for a fixed v and *all* values of $j \in V \setminus s$ the maximization in (14) requires $|E|$ steps. Therefore, the actual running time of our algorithm is $O\left(\frac{|V|^2|E|}{\epsilon} \log \frac{|V| \log \frac{1}{p_{\min}}}{\epsilon} \log \frac{1}{p_{\min}} \log(L - \ell_{\min})\right)$, which is equal to the running time presented in [HKQ+19].

A.4 Boolean programming problems under nested knapsack constraints

The following problem is taken from [HKS18], see literature review therein. There are n sequential "leave-or-take" decisions. It is required to either minimize or maximize a non-linear function that facilitates the leave-or-take decision making. Each decision j is associate with a positive integer weight w_j. At every stage k there is a knapsack constraint, which keeps that the total weight of the "take" decisions up to stage k to be no more than d_k. Furthermore, it is assumed that $d_1 \le \ldots \le d_n$. The problem can be formulated in the following way:

$$\begin{aligned} \max & \sum_{j=1}^{n} h_j \left(\sum_{i=1}^{j} w_i x_i\right) x_j + \sum_{j=1}^{n} g_j \left(\sum_{i=1}^{j} w_i (1 - x_i)\right) (1 - x_j) \\ \text{s.t.} & \sum_{j=1}^{k} w_j x_j \le d_k & 1 \le k \le n, \\ & x_j \in \{0, 1\} & j = 1, \ldots, n. \end{aligned}$$

where h_j, g_j are non-decreasing non-negative functions that depend on the total accumulated weight of the taken (left) items i, respectively, $1 \le i \le j$. The boolean variables x_j equals 1 if item j is taken, and 0 if it is left. The function h_j represents the cost for taking items, and the function g_j represents the cost for leaving them. Another assumption on these function is that there exists a number q such that for every positive x

$$\varphi(Kx) \leq K^q \varphi(x) \tag{15}$$

for every function $\varphi = h_1, \ldots, h_n, g_1, \ldots, g_n$. I.e., if $\tilde{x}$ is a K-approximation value of x, than $\varphi(\tilde{x})$ is a K^q-approximation value of $\varphi(x)$.

The input of the problem consists of: (i) n–the number of items, (ii) w_j–the items weights, (iii) the weight constraints d_k for every period k, and (iv) the number q from (15). We assume that the binary size of any of function values of h_j, g_j, $j = 1, \ldots, n$, is polynomially bounded by the (binary) input size.

The following DP formulation is introduced in [HKS18]. Denote by $A_k = \sum_{j=1}^{k} w_j$, $1 \leq k \leq n$. Let y , $0 \leq y \leq d_n$, denote a possible value of the total weight of all taken items; we refer to this value as the available space. At every stage $k = 1, \ldots, n$ three functions are considered. The function $s_k(y)$ is the optimal value of the objective function, provided that only items $1, \ldots, k$ have been considered and the available space is $\min\{y, d_k\}$. The function $a_k(y) \leq y$ is the actual used space, i.e., the total weight of the taken items in the optimal solution associated with $s_k(y)$, and $b_k(y) \leq A_k$ is the total weight of the left items in the optimal solution associated with $s_k(y)$.

The DP recurrence reads as follows. Let $s_0(y) = a_0(y) = b_0(y) = 0$ for every y. For every stage $k = 1, \ldots, n$, compute:

$$\begin{aligned}
s'(y) =& h_k(a_{k-1}(y) + w_k) + s_{k-1}(y - w_k)\\
s''(y) =& g_k(b_{k-1}(y) + w_k) + s_{k-1}(y)\\
s_k(y) =& \begin{cases} s''(y) & 0 \leq y < w_k,\\ \min\{s'(y), s''(y)\} & w_k \leq y \leq d_k,\\ s_k(d_k) & d_k < y \leq d_n. \end{cases}\\
a_k(y) =& \begin{cases} a_{k-1}(y) & 0 \leq y < w_k,\\ a_{k-1}(y) & w_k \leq y \leq d_k \text{ and } s_k(y) = s''(y),\\ a_{k-1}(y) + w_k & w_k \leq y \leq d_k \text{ and } s_k(y) = s'(y),\\ a_k(d_k) & d_k < y \leq d_n. \end{cases}\\
b_k(y) =& \begin{cases} b_{k-1}(y) + w_k & 0 \leq y < w_k,\\ b_{k-1}(y) + w_k & w_k \leq y \leq d_k \text{ and } s_k(y) = s''(y),\\ b_{k-1}(y) & w_k \leq y \leq d_k \text{ and } s_k(y) = s'(y),\\ b_k(d_k) & d_k < y \leq d_n. \end{cases}
\end{aligned} \tag{16}$$

for every $0 \leq y \leq d_k$.

The boolean programming problem does not fall directly into our meta-framework; in order to design an FPTAS for this problem, we need to introduce an extension first. We replace Condition 3 with the following weaker version:

Condition 5 *there exists a positive integer q such that for any real number $L \geq 1$, and for any $t = 0, \ldots, T$, $1 \leq i \leq m$, if $\{\tilde{z}_{r,j}\}_{r \leq t, j \leq i : r+j < t+i}$ are L^k-approximation functions of $\{z_{r,j}\}_{r \leq t, j \leq i : r+j < t+i}$ for some integer k, then $f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r \leq t, j \leq i : r+j < t+i}\right)$ is an L^{k+q}-approximation function of $f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{z_{r,j}\}_{r \leq t, j \leq i : r+j < t+i}\right)$.*

We replace Algorithm 3 with the following Algorithm 5, in which the only difference from Algorithm 3 is in line 2, where we change the value of K from $\sqrt[m(T+1)]{1+\epsilon}$ to $\sqrt[(q+1)m(T+1)]{1+\epsilon}$:

Algorithm 5 FPTAS for DPs that satisfy Conditions 1,2,4(i), and 5

```
1: Function FPTASDirectDP*
2: $K \leftarrow \sqrt[(q+1)m(T+1)]{1+\epsilon}$
3: for t=0,...,T do
4:     for i=1,...,m do
5:         $\bar{z}_{t,i}(I_{t,i}) = f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}\right)$
6:         $\tilde{z}_{t,i} \leftarrow$ FPTASCompress($\bar{z}_{t,i}, K-1$)
7:     end for
8: end for
9: return $\tilde{z}_{T,m}$
```

Proposition A.1 *If Conditions 1,2,4, and 5 are met for DP* (1), *then it admits an FPTAS, which runs in time* $O\left(\frac{(mT)^2}{\epsilon} t_f \log \frac{mT \log U_z}{\epsilon} \log U_z \log U_{\mathcal{S}}\right) = O\left(\frac{(mT)^2}{\epsilon} t_{\tilde{f}} \log U_z \log U_{\mathcal{S}}\right)$.

Proof. We prove here the case where Condition 4(i) is met, the proof for the case where Condition 4(ii) is met uses identical arguments. Algorithm 5 serves as an FPTAS for the case where Condition 4(i) is met. This algorithm and its analysis are similar to the case where Condition 3 is met. Therefore, we focus here on the differences between Algorithms 3 and 5. We start with the induction part in the proof of Theorem 3.1.

We show by induction that $\tilde{z}_{t,i}$ is a monotone $K^{(q+1)i(t+1)}$-approximation of $z_{t,i}$, for $0 \le t \le T, 1 \le i \le m$, i.e., the approximation ratio is $q+1$ multiplied by the iteration number. For the base case of $t = 0, i = 1$, $\bar{z}_{0,1} = z_{0,1}$, since $\{\tilde{z}_{r,j}\}_{r\le 0, j\le 1: r+j<1}$ is an empty set, function FPTASCompress($\bar{z}_{0,1}, K-1$) returns a monotone K-approximation function of $z_{0,1}$, see Proposition 2.5, and since $q \ge 1$, $\tilde{z}_{0,1}$ is K^{q+1}-approximation of $z_{0,1}$.

Assume that the claim holds for every $r \le t, j \le i : r+j < t+i$; so, $\tilde{z}_{r,j}$ is a monotone $K^{(q+1)j(r+1)}$-approximation of $z_{r,j}$ for $r \le t, j \le i : r+j < t+i$. By the induction hypothesis $\{\tilde{z}_{r,j}\}_{r\le t, j\le i: r+j<t+i}$ are monotone, at most $K^{(q+1)(i(t+1)-1)}$-approximation functions of $\{z_{r,j}\}_{r\le t, j\le i: r+j<t+i}$, and by Condition 5, $\bar{z}_{t,i}$ is a monotone (see Remark 3) $K^{(q+1)(i(t+1)-1)+q} = K^{(q+1)(i(t+1))-1}$-approximation function of $z_{t,i}$. The function FPTASCompress($\bar{z}_{t,i}, K-1$) returns a monotone K-approximation of $\bar{z}_{t,i}$, see Proposition 2.5. Then, by the approximation of approximation rule (Proposition 2.2(7)), $\tilde{z}_{t,i}$ is a $K^{(q+1)i(t+1)}$-approximation of $z_{t,i}$. This concludes the induction proof.

The rest of the arguments in the proof of Theorem 3.1 hold in our case, including the running time analysis, and therefore omitted. □

We are ready now to show that (16) is a direct DP. Let $T = n$ and $m = 3$. For every $k = 0, \dots, T$, we denote $z_{k,1}(\cdot) = s_k(\cdot), z_{k,2}(\cdot) = a_k(\cdot), z_{k,3}(\cdot) = b_k(\cdot)$. We start with Condition 1. Note that each function $s_k(\cdot)$ is monotone non-increasing, since as y grows the problem becomes less constrained, i.e., the more space is available for items to be taken, the less we are forced to make the "leave" decisions. Using similar arguments, note that each function $a_k(\cdot)$ is monotone non-decreasing, since the more space y is available to accommodate the taken items, the more items we may take. Furthermore, note that each function $b_k(\cdot)$ is monotone non-increasing: by a symmetric argument, the more space y is available to accommodate the taken items, the less items we may actually leave. As for Condition 2, let $U_{\mathcal{S}} = d_n$, $Z^{UB} := \sum_{j=1}^n h_j(d_j) + \sum_{j=1}^n g_j(d_j)$, and $U_z = Z^{UB} + A_n$. The logarithms of $U_{\mathcal{S}}, U_z$ are polynomially bounded in the (binary) input size. We now turn to prove that Condition 5 holds. For every $k = 1, \dots, n$, if $\tilde{a}_{k-1}, \tilde{b}_{k-1}$ are K-approximation functions of a_{k-1}, b_{k-1}, respectively, then by the summation of approximation rule (Proposition 2.2(3)), $\bar{a}_k, \bar{b}_k$, as defined in line 5 of Algorithm 5, are K-approximation of a_k, b_k, respectively. For the functions s_k we prove by induction that $\bar{s}_k$, as defined in line 5 of Algorithm 5, is $K^{q(k-1)}$-approximation of s_k for $k = 1, \dots, n$

(note that $\bar{s}_0$ is trivially a 1-approximation of s_0). The base case of $k = 1$ follows as a_0, b_0 are exact functions. Now, note that by the approximation of approximation rule (Proposition 2.2(7)) at stage $k-1$, $\tilde{a}_{k-1}, \tilde{b}_{k-1}$ are K^{k-1}-approximation of a_{k-1}, b_{k-1}. In addition, we assume that $\tilde{s}_{k-1}$ is $K^{q(k-2)}$-approximation of s_{k-1}. By assumption (15), and the summation and decomposition of approximation rules (Proposition 2.2(3,4)), we have that $h_k(\tilde{a}_{k-1}(\cdot)+w_k), g_k(\tilde{b}_{k-1}(\cdot)+w_k)$ are $K^{q(k-1)}$-approximations of $h_k(a_{k-1}(\cdot)+w_k), g_k(b_{k-1}(\cdot)+w_k)$, and so, by the summation of approximation rule (Proposition 2.2(3)) and the induction hypothesis, $\bar{s}'(\cdot) = h_k(\tilde{a}_{k-1}(\cdot)+w_k) + s_{k-1}(\cdot - w_k), \bar{s}''(\cdot) = g_k(\tilde{b}_{k-1}(\cdot)+w_k) + s_{k-1}(\cdot)$ are $K^{\max\{q(k-1),q(k-2)\}} = K^{q(k-1)}$-approximation of s', s''. By the minimization of approximation rule (Proposition 2.2(6)), the same approximation ratio holds for $s_k(\cdot)$, and so Condition 5 holds. Condition 4(i) holds, as the computation time is $t_f = O(1)$ for all the recursive functions. Hence, Algorithm 3 serves as an FPTAS for this problem, and its running time is $O\left(\frac{n^2}{\epsilon}\log\left(\frac{n}{\epsilon}\log(Z^{UB}+A_n)\right)\log(Z^{UB}+A_n)\log d_n\right)$, which is identical to the running time of [HKS18].

A.5 Minimizing the makespan of deteriorating jobs

This problem is taken from [Hal19a], see therein for a literature review. There are n independent jobs that need to be processed by a single machine without preemption. Each job J_j has a basic processing time $p_j, j = 1, \ldots, n$. The jobs have a given common critical date d after which they start to deteriorate, and a common maximum deterioration date D ($D > d$) after which they deteriorate no further. The actual processing time $a_j(t)$ of J_j depends on its start time t in the following way:

$$a_j(t) = \begin{cases} p_j & t \le d, \\ p_j + w_j(\min\{t, D\} - d) & \text{otherwise.} \end{cases}$$

where $w_j \ge 0$ is the deterioration rate of J_j. If $D < \infty$, the deterioration is called *bounded*, and if $D = \infty$ the deterioration is called *unbounded*. We assume that d, D, and all p_j and w_j are integers for $j = 1, \ldots, n$. Let $p_{\max} = \max_j\{p_j\}$ and $w_{\max} = \max_j\{w_j\}$. If we set $L = \max\{p_{\max}, w_{\max}, D\}$ (or $L = \max\{p_{\max}, w_{\max}, d\}$ in the unbounded case) to be the largest numerical parameter in the input, then the problem (binary) input size is $O(n \log L)$. The problem is to schedule the jobs in order to minimize the makespan, i.e., the completion time of the last job in the schedule. Following the notation in [Hal19a] we shall denote this problem by DET.

Before we approach the DP formulation, some notations and structural properties of optimal schedules for DET are presented. Note that it is easy to realize that there is no machine idle time in any optimal schedule. Therefore, there is a unique job for any given schedule that starts not after d and completes after d; we call such a job *pivotal*. Any job that ends before d is called early. We call the early jobs and the pivotal one *non-deteriorating*. Any job that starts after d and ends not after (after) D is called *tardy* (respectively, *suspended*). We call the tardy and suspended jobs *deteriorating*.

Let DET(k, s) denote a DET problem in which the pivotal job is J_k and the start time of the earliest deteriorating job is s, $d \le s \le d + p_k$. Kovalyov and Kubiak observe that there exist $1 \le k^* \le n$ and s^*, where $\max\{d+1, p_{k^*}\} \le s^* \le d + p_{k^*}$, such that any optimal solution to DET(k^*, s^*) is an optimal solution to DET [KK98]. Thus, DET reduces to solving $\sim nd$ problems DET(k, s). Since this amount of problems is pseudo-polynomial in the input size, solving (or even approximating) all of them and taking the best solution results in a pseudo-polynomial algorithm. Therefore, [KK98] turn to approximating only a logarithmic number of DET(k, s) problems.

Proposition A.2 (Lemma 2 in [KK98]) *The best solution in the family of* $(1+\frac{\epsilon}{3})$*-approximate solutions for* DET$(k, d + (\max\{d+1, p_k\} - d)(1+\frac{\epsilon}{3})^{i_k})$, $k = 1, ..., n$, $i_k = 1, ..., 1 + \frac{3\log p_k}{\epsilon}$ *, is a* $(1+\epsilon)$*-approximate solution for* DET.

Proposition A.2 reduces the problem of designing an FPTAS for DET, to designing an FPTAS for DET(k,s). For this end, denote the pivotal job by J_0, and index all the other jobs according to Smith's rule so that $\frac{p_1}{w_1} \leq \ldots \leq \frac{p_{n-1}}{w_{n-1}}$ [Smi56]. Kubiak and van de Velde observe that there exists an optimal schedule, where early jobs are sequenced arbitrarily, followed by the pivotal job, which in turn is followed by tardy and suspended jobs, if any [KvdV98]. The former are sequenced in increasing order of their indices; the order of the latter is arbitrary. Let $f_j(y)$ be the makespan of an optimal schedule of jobs $0,\ldots,j$, where at most y time is allocated to early jobs. Note that the makespan of DET(k,s) equals $z_{n-1}(s-p_0)$. The DP recursion reads as follows:

$$z_j(y) = \begin{cases} s & j=0 \\ z_{j-1}(y) + p_j + w_j(\min\{z_{j-1}(y,D)\} - d) & 1 \leq j \leq n-1 \text{ and } y = 0,\ldots,p_j - 1, \\ \min\{z_{j-1}(y-p_j), z_{j-1}(y) + p_j + w_j(\min\{z_{j-1}(y,D)\} - d)\} & 1 \leq j \leq n-1 \text{ and } y = p_j,\ldots,s-p_0. \end{cases} \tag{17}$$

We turn now to show that (17) is a direct DP. We first set the parameters $T = n-1$ and $m = 1$. Note that here, in order to calculate z_j, we need to know only z_{j-1}. The functions $z_j(\cdot)$ are non-decreasing by their definition, and so Condition 1 holds. Condition 2 holds since $U_z = n(p_{\max} + Dw_{\max}) = O(nL^2)$ in the bounded case and $U_z = O(p_{\max} w_{\max}^{n-1}) = O(L^n)$ in the unbounded case. In both cases $U_{\mathcal{S}} = d = O(L)$. Regarding Condition 3, it holds due to the linearity, composition, and minimization of approximation rules (Proposition 2.2(2,4,6)). Condition 4(i) holds, as functions $z_j(\cdot)$ are calculated in $t_f = O(1)$ time. From Theorem 3.1 we conclude that Algorithm 3 serves as an FPTAS to problem DET(k,s) with running time $O\left(\frac{n^2}{\epsilon}\log L \log(nL)\log\left(\frac{n\log(nL)}{\epsilon}\right)\right)$ in the bounded case, and $O\left(\frac{n^3}{\epsilon}\log^2 L \log\left(\frac{n\log L}{\epsilon}\right)\right)$ in the unbounded case. This results in FPTASes for the bounded and unbounded cases with running times $O\left(\frac{n^3}{\epsilon^2}\log^2 L \log(nL)\log\left(\frac{n\log(nL)}{\epsilon}\right)\right)$ and $O\left(\frac{n^4}{\epsilon^2}\log^3 L \log\left(\frac{n\log L}{\epsilon}\right)\right)$, respectively, which is equal to the running times presented in [Hal19a].

A.6 Counting integer knapsack solutions

The following problem is taken from [Hal16]. It demonstrates well the trade-off between problem formulation and algorithm design: while the DP formulation of this problem is somewhat involved, see (19) below, ensuring that the conditions of the framework hold for this DP formulation is rather easy, and therefore it is straightforward to apply Theorem 3.1 to get an FPTAS for the problem. The problem formulation goes as follows. Given n elements with nonnegative integer weights $w = (w_1,\ldots,w_n)$, an integer capacity C, and positive integer ranges $u = (u_1,\ldots,u_n)$, we consider the counting version of the classic integer knapsack problem: find the size of the set of feasible solutions $\{x \mid \sum_{i=1}^n w_i x_i \leq C, 0 \leq x_i \leq u_i\}$. Let $U = \max_{1\leq i\leq n} u_i$. We will give below an FPTAS which runs in $O\left(\frac{n^3}{\epsilon}\log\left(\frac{n}{\epsilon}\log U\right)\log^2 U\right)$ time. In addition to the literature review given in [Hal16], we mention here the recent work [GMW18], in which an FPTAS for the problem with running time $O\left(\frac{n^{2.5}}{\epsilon^{1.5}}\log\left(\frac{n}{\epsilon}\log U\right)\log(n\epsilon)\log^2 U\right)$ is designed, using an extension to the method of K-approximation sets and functions. Their algorithm runs faster than ours for values of ϵ greater than $1/n$.

A pseudo-polynomial time algorithm is achieved using the following recurrence:

$$s_i(j) = \begin{cases} \sum_{k=0}^{m_i(j)} s_{i-1}(j - kw_i) & 2 \leq i \leq n, j = 1,\ldots,C, \\ m_1(j) + 1 & i = 1, j = 1,\ldots,C, \end{cases} \tag{18}$$

where function $m_i : [0, \ldots, C] \to \mathbb{Z}^+$ is defined as $m_i(j) := \max\{x \in \mathbb{Z}^+ \mid x \le u_i, xw_i \le j\}$ and returns the maximum number of copies of item i that can be placed in a knapsack with capacity j. Here $s_i(j)$ is the number of integer knapsack solutions that use a subset of the items $1, \ldots, i$ whose weights sum up to at most j. The solution of the counting problem is therefore $s_n(C)$. The complexity of this pseudo-polynomial time algorithm is $O(nUC)$, i.e., exponential in both the (binary) sizes of U and C.

In order to get our FPTAS we give next a more careful DP formulation that is exponential only in the (binary) size of C. Instead of deciding at once how many copies of item i to put in the knapsack, we break the decision into $\lfloor \log^+ m_i(j) \rfloor + 1$ binary sub-decisions, where $\log^+ x := \max\{0, \log x\}$. Sub-decision $\ell = 1, \ldots, \lfloor \log^+ m_i(j) \rfloor + 1$ checks the possibility of putting $2^{\ell-1}$ copies of item i in the knapsack. It is done using the idea of *binding constraints*: For $\ell \ge 1$ let $z_{i,\ell,1}(j)$ be the number of solutions for a knapsack of capacity j that use a subset of the items $\{1, \ldots, i\}$, put no more than $u_i \mod 2^\ell$ copies of item i, and no more than u_k copies of item k, for $k = 1, \ldots, i-1$. In this way, considering the third index of $z_{i,\ell,r}(j)$, if $r = 0$ then the constraint $x \le u_i$ is assumed to be non binding. If, on the other hand, $r = 1$ then the constraint $x \le u_i$ may be binding. Before giving the formal recurrences we define $\text{msb}(x, i) := \lfloor \log(x \mod 2^i) \rfloor + 1$. $\text{msb}(x, i)$ is therefore the most significant 1-digit of $(x \mod 2^i)$ if $(x \mod 2^i) > 0$, and is $-\infty$ otherwise. E.g., $\text{msb}(5, 2) = 1$ and $\text{msb}(4, 1) = -\infty$. Our recurrences are as follows:

$$z_{i,\ell,0}(j) = z_{i,\ell-1,0}(j) + z_{i,\ell-1,0}(j - 2^{\ell-1}w_i) \qquad \ell = 2, \ldots, \lfloor \log^+ m_i(j) \rfloor + 1 \tag{19a}$$

$$z_{i,\ell,1}(j) = z_{i,\ell-1,0}(j) + z_{i,\text{msb}(u_i,\ell-1),1}(j - 2^{\ell-1}w_i) \qquad \ell = 2, \ldots, \lfloor \log^+ m_i(j) \rfloor + 1 \tag{19b}$$

$$z_{i,1,r}(j) = z_{i-1,\lfloor \log^+ m_{i-1}(j) \rfloor+1,1}(j) + z_{i-1,\lfloor \log^+ m_{i-1}(j-w_i) \rfloor+1,1}(j - w_i) \tag{19c}$$

$$z_{i,-\infty,1}(j) = z_{i-1,\lfloor \log^+ m_{i-1}(j) \rfloor+1,1}(j) \tag{19d}$$

$$z_{1,\ell,r}(j) = m_1(j) + 1 \qquad \ell = 1, \ldots, \lfloor \log^+ m_i(j) \rfloor + 1, \tag{19e}$$

$$z_{i,\ell,r}(j) = 0 \qquad j < 0 \tag{19f}$$

where $r = 0, 1, i = 2, \ldots, n$, and $j = 0, \ldots, C$, unless otherwise specified. The solution of the counting problem is therefore $z_{n,\lfloor \log u_n \rfloor+1,1}(C)$. The time needed to solve this program is only $O(nC \log U)$. We extend DP formulation (19) to any integer positive index ℓ by letting $z_{i,\ell,r}(j) = 0$ for $i = 1, \ldots, n, r = 0, 1$ and $\ell > \lfloor \log^+ m_i(j) \rfloor + 1$. We now explain the six equations in formulation (19) in more detail. Equation (19a) deals with the case where the constraint $x \le u_i$ is non binding, so putting $2^\ell - 1$ more copies of item i in a knapsack of remaining capacity j is a feasible possibility. Clearly, in the following steps the constraint $x \le u_i$ remains non binding. As for equation (19b), it deals with the case where the constraint $x \le u_i$ may be binding when putting $2^\ell - 1$ copies of item i in the knapsack. If we do put this number of copies, the constraint may be binding, otherwise it is assured to be non binding. Equation (19c) deals with the possibility of putting an odd number of copies of item i in the knapsack. Equation (19d) is called by either equation (19b) or equation (19c). The former case occurs when exactly u_i copies of item i are put in the knapsack, and the latter case occurs when $m_i(j) = 0$, i.e., there is not enough capacity to put even a single copy of the item. Equation (19e) deals with the initial condition of one item only, and the last equation deals with the boundary condition that there is not enough capacity in the knapsack.

We next show that DP (19) is a direct DP. Set $T = n - 1, m = 2(\lfloor \log U \rfloor + 1)$. Now, the boundary condition is $f_{0,k} = z_{1,\lceil \frac{k}{2} \rceil,r}$ for every $k = 1, \ldots, m$. (Note that $z_{1,\ell,r}$ is independent of the value of r.) For $i \ge 1$, set $f_{i,1} = z_{i,-\infty,1}$, $f_{i,2} = z_{i,1,r}$ ($z_{i,1,r}$ is independent of the value of r), and

$f_{i,k} = z_{i,\lceil \frac{k}{2} \rceil, 1-(k \mod 2)}$. By its definition, function $z_{i,\ell,r}(\cdot)$ is nondecreasing for every i, ℓ, r, and so Condition 1 holds. In addition, $U_{\mathcal{S}} = C$ and $U_z = U^n$, and so Condition 2 holds. Condition 3 holds by the summation and composition of approximation rules (Proposition 2.2(3,4)), and Condition 4(i) holds since the computation time at every iteration is $t_f = O(1)$, as a sum of at most two former z functions. Then, according to Theorem 3.1, there exists an FPTAS for this problem. The running time of the algorithm is $O\left(\frac{n^3 \log^3 U \log C}{\epsilon} \log \frac{n \log U}{\epsilon}\right)$, which equals the running time stated in [Hal16].

B More examples of indirect DPs

In this section we give more examples of indirect DPs. The first two examples, given in the next two subsections, are of tailor-made FPTASes to specific application problems. In the subsequent two subsections we present two frameworks for the automatic generation of FPTASes for stochastic monotone DPs that have a specific DP recursion, as detailed in Section 3.1. We show that these two frameworks are special cases of our meta-framework.

B.1 Single-item economic lot-sizing

We consider here the classical single-item economic lotsizing problem with varying demand, see [SCB14, Chap. 8], or any textbook on inventory theory. The DP formulation for this problem, see (21) below, is taken from [HOS12], who proved that it fits the framework of [HKL+14] as a special deterministic monotone DP, see literature review therein. We show later in Appendix C.1 that the monotone case of the framework of [HKL+14] fits our meta-framework as a special case, so our meta-framework applies to this problem as well. However, for the sake of completeness, and since the single-item economic lotsizing problem is a fundamental problem in inventory theory, we explicitly show in this section that (21) is an indirect DP. We note in passing that although numerous tailor-made FPTASes were designed for this problem, see discussion in [HOS12, Sec. 6.1], somewhat surprisingly, none of them beats our FPTAS, which is automatically generated by our general framework, i.e., our FPTAS is the fastest one known to date.

Consider the following single-item inventory control problem. The planning horizon is divided into T' time periods. At the beginning of time period t, the inventory level I_t is observed, and then a replenishment decision is made. Let $x_t \geq 0$ denote the replenishment quantity in period t. We assume that the replenishment lead time is zero; that is, the x_t units arrive in period t. After that, a deterministic demand of $d_t \geq 0$ units of the item occurs, and a disposal decision of y_t units is made. The y_t units are disposed of immediately and at no cost. The ending inventory level of period t equals $I_{t+1} = I_t + x_t - d_t - y_t$. Backlogging is allowed, which implies that the inventory level I_{t+1} can be negative. If $I_{t+1} > 0$, then a holding cost of $h_t(I_{t+1})$ is charged; the holding cost is monotone non-decreasing in the amount of inventory held at the end of the time period. If $I_{t+1} < 0$, then a backlogging cost of $h_t(I_{t+1}) := b_t(|I_{t+1}|)$ is incurred; the backlogging cost $b(\cdot)$ is monotone non-decreasing of the amount of inventory backlogged to the next time period. (In this way $h_t(\cdot)$ is a unimodal function which is minimized at 0 with $h_t(0) = 0$.) Thus, our objective is

$$\min_{x_1,\ldots,x_{T'};y_1,\ldots,y_{T'}} \sum_{t=1}^{T'} c_t(x_t) + h_t(I_t + x_t - d_t - y_t), \tag{20}$$

subject to the system dynamics $I_{t+1} = I_t + x_t - d_t - y_t$ and the procurement cost $c_t(\cdot)$ for $t = 1, \ldots, T'$. We assume that $I_1 = 0$ (i.e., we start with zero inventory). We also assume that $c_t(\cdot)$ is a non-decreasing function over $\mathbb{Z}^+$, $t = 1, \ldots, T'$. We assume that $h_{T'}(x)$ is very high for any $x \neq 0$, which calls to zero inventory at the end of period T'. The input data for the problem consists of (i) the number of time periods T', (ii) the values $d_1, \ldots, d_{T'}$, and (iii) an oracle that computes functions c_t

and h_t (for each period t). All demand, procurement, disposal and inventory levels are integral. For every $t = 1, \ldots, T'$, functions c_t and h_t are nonnegative integer-valued, and the binary size of any of their values is polynomially bounded by the (binary) input size. It is easy to see that because we are dealing with a deterministic setting, in an optimal policy there will be no disposal. We allow disposal in order to show that the DP formulation for the problem is monotone as follows.

Let $d^* = \sum_{t=1}^{T'} d_t$ be an upper bound on the total demand. The following DP formulation of the minimization problem (20) is introduced in [HOS12] and consists of a transformed problem into $2T'$ time periods, in which demands appear and procurement decisions are made only in odd time periods while holding/backloggong costs occur and disposal decisions are made only in even time periods. Let $z_t(I_t)$ be the minimal cost incurred in time periods $t, \ldots, 2T'$, with initial inventory I_t. For every $-d^* \le I_t \le d^*$ the recurrence reads as follows.

$$z_t(I_t) = \begin{cases} 0 & t = 2T'+1, \\ \min_{x \in \{0,\ldots,d^*\}} c_{\lceil \frac{t}{2} \rceil}(x) + z_{t+1}(I_t + x - d_{\lceil \frac{t}{2} \rceil}) & t = 1, 3, \ldots, 2T'-1, \\ \min_{x \in \{-I_t^*,\ldots,0\}} h_{\frac{t}{2}}(I_t + x) + z_{t+1}(I_t + x) & t = 2, 4, \ldots, 2T'. \end{cases} \tag{21}$$

We next show that DP (21) is an indirect DP. We set $T = 2T'$ and reverse the order of the recursion by setting $t \leftarrow 2T' + 1 - t$. Halman *et al.* [HOS12, Sec. 6.1] prove that the functions $z_t(\cdot)$ are non-increasing for every $1 \le t \le T+1$, and hence Condition 1 holds. We also have $U_{\mathcal{S}} = 2d^* + 1$, and $U_z := \bar{U} = \sum_{t=1}^{T} \left(\{c_t(d^*) + \max\{h_t(d^*), h_t(-d^*)\}\} \right)$, and so Condition 2 holds. From the summation, composition and minimization of approximation rules (Proposition 2.2 (3,4,6)), Condition 3 holds as well. As for Condition 4(ii), we get an FPTAS for (21), that is, for the case of minimization over an action space of pseudo-polynomial range, via Proposition 3.3 with parameters set as follows. The random variables D_t realize to 0 with probability one whenever t is even and to $d_{\lceil \frac{t}{2} \rceil}$ with probability one whenever t is odd. We set $\text{tran}_t(I, x, D) = I - x + D$. We also set $g_t(I, x, D) = c_{\lceil \frac{t}{2} \rceil}(x)$ for odd values of t and $g_t(I, x, D) = h_{\frac{t}{2}}(I + x)$ for even values of t. Note that both $\text{tran}_t(I, x, D)$ and $g_t(I, x, D)$ are monotone functions in their first two variables. Therefore, by Proposition 3.3 an FPTAS for $\bar{f}_t = \bar{z}_t$ exists with running time $t_{\bar{f}} = O(\frac{T}{\epsilon} \log \bar{U})$. Consequently, an FPTAS for the single-item economic lotsizing problem exists, and its running time is $O\left(\frac{T^3}{\epsilon^2} \log^2 \bar{U} \log d^* \log\left(\frac{T}{\epsilon} \log \bar{U}\right)\right)$, which is equal to the running time of [HOS12, Thm. 7].

B.2 Time-cost trade-off problems in series parallel project networks

This problem is taken from [HLS09], see literature review therein. Given is a directed acyclic series-parallel project network of n activities in activity-in-arc representation. Associated with each activity i are two non-increasing functions $g_i : C_i \to \mathbb{Z}^+$ and $h_i : D_i \to \mathbb{Z}^+$, where $g_i(c_i)$ is the activity time when an amount c_i of monetary resource is spent on the activity, $h_i(d_i)$ is the cost incurred when the activity time is d_i, $C_i = \{\underline{c}_i, \underline{c}_i + 1, \ldots, \overline{c}_i\} \subseteq \mathbb{Z}^+$ is the set of all possible cost consumptions of activity i, and $D_i = \{\underline{d}_i, \underline{d}_i + 1, \ldots, \overline{d}_i\} \subseteq \mathbb{Z}^+$ is the set of all possible time durations of activity i.

Let $\phi(d_1, \ldots, d_n)$ denote the total duration of the project when the time duration of activity i is d_i for $i = 1, \ldots, n$. Halman *et al.* [HLS09] consider two variants of the problem: (i) *deadline problem*: given a deadline a, determine $d_1, \ldots, d_n$ so that $\phi(d_1, \ldots, d_n) \le a$, and that $h_1(d_1) + \ldots + h_n(d_n)$ is minimized, and (ii) *budget problem*: given a budget b, determine $c_1, \ldots, c_n$ so that $c_1 + \ldots + c_n \le b$, and that $\phi(g_1(c_1), \ldots, g_n(c_n))$ is minimized. The input of the deadline problem consists of n, the number of activities, and $\underline{d}_i, \overline{d}_i$, $i = 1, \ldots, n$. We assume that the binary size of each of the values $h_i(\cdot)$, $i = 1, \ldots, n$, is polynomially bounded by the (binary) input size. The input of the budget problem consists of n, the number of activities, and $\underline{c}_i, \overline{c}_i$, $i = 1, \ldots, n$. We assume that the binary size of each of the values of $g_i(\cdot)$, $i = 1, \ldots, n$, is also polynomially bounded by the (binary) input size.

Before we approach the DP formulation, we would like to set an order on the series and parallel operations over the graph. As demonstrated in Valdes *et al.* [VTL82], for every series-parallel graph consisting of the n edges $e_1, \ldots, e_n$ it is possible to build in polynomial time a *binary decomposition tree*. The binary decomposition tree is a rooted tree, in which every leaf v_i is associated with the edge e_i of the series-parallel graph (activity in our context), and every non-leaf vertex is denoted by P for parallel, or by S for series. The tree is associated with the series-parallel graph, such that given the binary decomposition tree, the series-parallel graph can be reduced into a single edge in the following iterative way: in every iteration, if the vertex v' combining leaves v_i, v_j is denoted by P, unify the two activities in e_i, e_j by a parallel reduction, i.e., replace in the series-parallel graph the two parallel arcs e_i, e_j by a single arc e_{ij} and in the decomposition tree delete the two leaves v_i, v_j so that their parent v' becomes a leaf denoted as v_{ij}. If the vertex v' combining these leaves is denoted by S, unify the two activities by a parallel reduction, i.e., replace the in the series-parallel graph the two series arcs e_i, e_j by a single arc e_{ij} and in the decomposition tree delete the two leaves v_i, v_j so that their parent v' becomes a leaf denoted as v_{ij}. Note that the process of reducing a series-parallel graph to a single arc from the associated binary decomposition tree induces an order on the non-leaf vertices of the tree. According to such an order, we denote the non-leaf vertices of the tree by $v_{n+1}, \ldots, v_{2n-1}$, i.e., given an order from the tree for reducing the series-parallel graph to an arc, for every $n+1 \leq i < j \leq 2n-1$, the operation associated with v_i comes before the operation associated with v_j.

As [HLS09] show, the budget and deadline problems admit monotone DP formulations. Let $U = \max_i\{\max\{\bar{c}_i, \bar{d}_i\}\}$ be the maximal size of an action space, $U_h := \sum_{i=1}^n h_i(\underline{d}_i)$ be an upper bound on the value of the deadline problem and $U_g := \sum_{i=1}^n g_i(\underline{c}_i)$ be an upper bound on the value of the budget problem. Expand the domain of functions g_i, h_i to $\{0, \ldots, U\}$ such that $g_i(c_i) = U_g$ and $h_i(d_i) = U_h$ for $c_i < \underline{c}_i, d_i < \underline{d}_i$, and $g_i(c_i) = g_i(\underline{c}_i), h_i(d_i) = h_i(\underline{d}_i)$ for $c_i > \bar{c}_i, d_i > \bar{d}_i$, $i = 1, \ldots, n$. We start with the deadline problem. For $i = n+1, \ldots, 2n-1$, let $z_i(d)$ be the minimal cost of finishing all the tasks in the series-parallel sub-graph associated with the sub-tree rooted by v_i in time not more than d. Then for all $0 \leq d \leq a$ the following recursion holds, where v_{i_1}, v_{i_2} are the two siblings of v_i:

$$z_i(d) = \begin{cases} h_i(d) & 1 \leq i \leq n, \\ \min_{0 \leq d' \leq d} z_{i_1}(d') + z_{i_2}(d - d') & n+1 \leq i \leq 2n-1, v_i \text{ is marked with S}, \\ z_{i_1}(d) + z_{i_2}(d) & n+1 \leq i \leq 2n-1, v_i \text{ is marked with P}. \end{cases} \tag{22}$$

As for the budget problem, for $i = n+1, \ldots, 2n-1$, let $s_i(c)$ be the minimal time in the series-parallel sub-graph associated with the sub-tree rooted by v_i, with cost no more than c. Then for all $0 \leq c \leq b$ the following recursion holds, where v_{i_1}, v_{i_2} are the two siblings of v_i:

$$s_i(c) = \begin{cases} g_i(c) & 1 \leq i \leq n, \\ \min_{0 \leq c' \leq c} s_{i_1}(c') + s_{i_2}(c - c') & n+1 \leq i \leq 2n-1, v_i \text{ is marked with S}, \\ \min_{0 \leq c' \leq c}\{\max\{s_{i_1}(c'), s_{i_2}(c - c')\}\} & n+1 \leq i \leq 2n-1, v_i \text{ is marked with P}. \end{cases} \tag{23}$$

We turn now to show that each one of the DPs (22) and (23) is an indirect DP. We set $T = 2n-2$, $m = 1$ and shift the indices of the z_i and s_i functions down by one to be between 0 to $2n-2$. For $0 \leq i \leq n-1$ the functions z_i, s_i are non-increasing by definition. A simple direct proof shows that the non-increasing property for these functions holds also for $n \leq i \leq 2n-2$. Hence, Condition 1 holds. The logarithms of $U_z = U_h, U_g$, and $U_{\mathcal{S}} := U$ are polynomially bounded in the (binary) input size, and therefore Condition 2 holds. Condition 3 for both problems follows from the summation, composition, maximization, and minimization of approximation rules (Proposition 2.2(3,4,5,6)).

We next consider Condition 4(ii). Halman *et al.* design an FPTAS for the second case of DP (22), i.e., the minimization formula in (22). The arguments of this FPTAS are similar to the ones in Proposition 3.3.

Proposition B.1 (Based on Property 3 in [HLS09]) *For any fixed integers i, d, and any fixed reals $L > 1$ and $\epsilon > 0$, let $\tilde{z}_{i_1}$ and $\tilde{z}_{i_2}$ be monotone L-approximation functions of z_{i_1} and z_{i_2}, respectively, and let S_{i_1} and S_{i_2} be $(1+\epsilon)$-approximation sets of $\tilde{z}_{i_1}$ and $\tilde{z}_{i_2}$, respectively. Let*

$$\bar{z}_i(d) = \min_{\{0 \le d' \le d \mid d' \in S_{i_1} \text{ or } d-d' \in S_{i_2}\}} \tilde{z}_{i_1}(d') + \tilde{z}_{i_2}(d-d').$$

Then, $\bar{z}_i(d)$ is a $(1+\epsilon)L$-approximation of $z_i(d)$ that can be computed in $O\big((|S_{i_1}| + |S_{i_2}|)(t_{\tilde{z}_{i_1}} + t_{\tilde{z}_{i_2}})\big)$ time.

This proposition tells us that that an FPTAS for $\bar{f}_t = \bar{z}_i$ exists with running time $t_{\bar{f}_t} = O(\log_K U_h) = O\left(\frac{n \log U_h}{\epsilon}\right)$ and therefore Condition 4(ii) holds for the deadline problem. Using similar arguments, see [HLS09, Sec. 3.2], there exists an FPTAS for the second and third cases of (23), and its computation time is $t_{\bar{f}_t} = O(\log_K U_g) = O\left(\frac{n \log U_g}{\epsilon}\right)$. Hence, by Theorem 3.2, Algorithm 4 serves as an FPTAS for these problems, with running time (denote by $\bar{U} = U_h$ for the deadline problem, and $\bar{U} = U_g$ for the budget problem) $O\left(\frac{n^3}{\epsilon^2} \log\left(\frac{n}{\epsilon} \log \bar{U}\right) \log^2 \bar{U} \log U\right)$, which is equal to the running time stated in the end of [HLS09, Sec. 3].

C Previous frameworks

The principles of the K-approximation sets and functions method were used to design FPTASes for some specific problems, see some of them in Appendices A,B, as well as some general frameworks, see [HKL+14] for the monotone and convex explicit stochastic, [Hal19b] for the monotone and convex implicit stochastic, and stochastic DPs with multidimensional actions and scalar states [HN19]. In this section we show that the frameworks for monotone explicit and implicit stochastic are special cases of our meta-framework.

C.1 Explicit stochastic

The framework introduced in Halman *et al.* [HKL+14] for explicit stochastic monotone and convex DPs applies for a specific DP model and gives an FPTAS via the method of K-approximation sets and functions. While, as we show here, their framework for monotone DPs is a special case of our meta-framework, the conditions of [HKL+14] are more cumbersome than ours.

Halman *et al.* consider the formulation for a finite horizon stochastic DP as described in Section 3.1. We note that DP recursion (3) yields an exact solution for $z_1(I_1)$ but may require a pseudo-polynomial running time. For example, if $\mathcal{A}_j(I_j) \equiv \mathcal{A}$ and $\mathcal{S}_j \equiv \mathcal{S}$ for every j and I_j, then this DP has a running time of $O(T|\mathcal{A}||\mathcal{S}|)$, but $|\mathcal{A}|$ and $|\mathcal{S}|$ may be exponential in the (binary) input size.

Their framework assumes that the random variables are given explicitly in the following way: For each D_j, we are given n_j, the number of different values it admits with positive probability, and its support $\mathcal{D}_j := \{d_{j,1}, \ldots, d_{j,n_j}\}$, where $d_{j,\ell} < d_{j,k}$ for $\ell < k$. We are also given positive integers $q_{j,1}, \ldots, q_{j,n_j}$ such that

$$\text{Prob}(D_j = d_{j,i}) = \frac{q_{j,i}}{\sum_{k=1}^{n_j} q_{j,k}}.$$

For every $j = 1, \dots, T$ and $i = 1, \dots, n_j$, we denote $p_{j,i} = \text{Prob}(D_j = d_{j,i})$. Then, we have

$$E_{D_j}\left\{g_j(I_j, x_j, D_j) + z_{t+1}(h_j(I_j, x_j, D_j))\right\} = \sum_{k=1}^{n_j} p_{j,k}\Big[g_j(I_j, x_j, d_{j,k}) + z_{j+1}(\text{tran}_j(I_j, x_j, d_{j,k}))\Big].$$

In the proceeding, the following notations will be used:

$n^* = \max_j n_j$	$=$	maximum number of different values that D_j can take over the entire time horizon;
$D^* = \sum_{j=1}^{T} \lvert d_{j,n_j}\rvert$	$=$	maximum possible total value that the random variables can take over the entire time horizon;
$U_{\mathcal{S}} = \max_{j=1,\dots,T+1} \lvert\mathcal{S}_j\rvert$	$=$	maximal size of the state space;
$U_{\mathcal{A}} = \max_{j=1,\dots,T} \max_{I_j \in \mathcal{S}_j} \lvert\mathcal{A}_j(I_j)\rvert$	$=$	maximal size of the action space;
$U_{\mathcal{S}}^{\max} = \max_{j=1,\dots,T+1} \max_{x \in \mathcal{S}_j} \lvert x\rvert$	$=$	maximal size of an element in the state space;
$U_{\mathcal{A}}^{\max} = \max_{j=1,\dots,T} \max_{I_j \in \mathcal{S}_j} \max_{x \in \mathcal{A}(I_j)} \lvert x\rvert$	$=$	maximal size of an element in the action space.

Let $g_j^{\max} = \max_{I \in \mathcal{S}_j,\, x \in \mathcal{A}_j(I),\, d \in \mathcal{D}_j} g_j(I, x, d)$ be the maximal cost value in time period j, for $j = 1, \dots, T$. Let $g_{T+1}^{\max} = \max_{I \in \mathcal{S}_{T+1}} g_{T+1}(I)$. Let $g_j^{\min} = \min_{I \in \mathcal{S}_j,\, x \in \mathcal{A}_j(I),\, d \in \mathcal{D}_j}\{g_j(I, x, d) \mid g_j(I, x, d) > 0\}$ be the minimal positive cost value in time period j, for $j = 1, \dots, T$. Let $g_{T+1}^{\min} = \min_{I \in \mathcal{S}_{T+1}}\{g_{T+1}(I) \mid g_{T+1}(I) > 0\}$ (note: For $j = 1, \dots, T+1$, if $g_j \equiv 0$, then $g_j^{\min} = +\infty$.) Let

$$U_g = \frac{\max_{j=1,\dots,T+1} g_j^{\max}}{\min_{j=1,\dots,T+1} g_j^{\min}}.$$

A DP is called *monotone* whenever it satisfies the following 3 conditions (i.e., Conditions 1-3 in [HKL+14]):

Condition 6 *$\mathcal{S}_{T+1}, \mathcal{S}_j, \mathcal{A}_j(I_j) \subset \mathbb{Z}$ for $I_j \in \mathcal{S}_j$ and $j = 1, \dots, T$. For any set X among these sets, $\log \max_{x \in X}(|x| + 1)$ is bounded polynomially by the (binary) input size, and the kth largest element in X can be identified in constant time for any $1 \le k \le |X|$. For every $j = 1, \dots, T$, the number of different values the random variable D_j admits with positive probability is a given integer n_j, and its probability distribution function is given as n_j ordered pairs $(d_{j,i}, p_{j_i})$, where $p_{j,i} = Prob(D_j = d_{j,i}) \in \mathbb{Q}$ for $i = 1, \dots, n_j$. Moreover, $\mathcal{D}_j \subset \mathbb{Q}$ for $j = 1, \dots, T$.*

Condition 7 *For every $j = 1, \dots, T+1$, functions* tran_j, g_j *are either given explicitly (i.e., as explicit formula) or are accessed via oracle calls. Moreover, the values of g_j are nonnegative rational numbers that are polynomially bounded by the (binary) input size.*

Condition 8 *At least one of the following properties holds:*

(i) **(Non-decreasing DP)** *Function g_{T+1} is non-decreasing. For $j = 1, \dots, T$, function h_j is non-decreasing in its first variable and monotone in its second variable, and g_j is monotone in its second variable. Moreover, for each $j = 1, \dots, T$, either z_j is non-decreasing, or g_j is non-decreasing in its first variable and $\mathcal{A}_j(I) \subseteq \mathcal{A}_j(I')$ for all $I, I' \in \mathcal{S}_j$ with $I \ge I'$.*

(ii) **(Non-increasing DP)** *Function g_{T+1} is non-increasing. For $j = 1, \dots, T$, function h_j is non-decreasing in its first variable and monotone in its second variable, and g_j is monotone in its second variable. Moreover, for each $j = 1, \dots, T$, either z_j is non-increasing, or g_j is non-increasing in its first variable and $\mathcal{A}_j(I) \subseteq \mathcal{A}_j(I')$ for all $I, I' \in \mathcal{S}_j$ with $I \le I'$.*

The input data of the problem consists of the number of time periods T, the initial state I_1, the maximal size of an element in the state space $U_{\mathcal{S}}^{\max}$, the maximal size of an element in the action space $U_{\mathcal{A}}^{\max}$, the ratio U_g, and the explicit description of the random variables as described in Condition 6. Halman *et al.* show that any minimization optimization problem that can be cast as a monotone DP admits an FPTAS [HKL$^+$14, Thm. 3.2]. They also give a similar set of conditions such that if a maximization DP (i.e, the case where max replaces the min in (2)) satisfies them, then it admits an FPTAS [HKL$^+$14, Thm. 3.3].

We next show that DP (3) is an indirect DP. Condition 1 holds due to the following proposition.

Proposition C.1 (Proposition 8.1 in [HKL$^+$14]) *If Condition 8(i) (8(ii)) is satisfied, then for every $t = 1, \ldots, T+1$, function z_j in DP formulation* (3) *is non-decreasing (non-increasing) over $\mathcal{S}_j$.*

Note that Condition 6 implies that $\log U_{\mathcal{S}}$ is polynomially bounded by the (binary) input size, and $\log U_g$ is polynomially bounded by the (binary) input size as a part of the problem input, and hence Condition 2 holds. Condition 3 follows from summation, composition, and minimization of approximation (Proposition 2.2(3,4,6)), since in (3) we minimize over the expectation, which is a finite weighted sum with a number of expressions that is bounded linearly by n^*.

We next show that Condition 4 follows from Proposition 3.3. We note first that (7) serves as the FPTAS$\bar{f}_t\left(I_t, A_t(I_t), \{\tilde{z}_r\}_{r<t}, \epsilon\right)$ in Condition 4(ii), i.e., (7) is an FPTAS for (3). Its running time $O(n_t(t_{\tilde{g}_t} + t_{\text{tran}_t} + t_{\tilde{z}_{t-1}})|W^{-1}(I_t)|)$, as stated in Proposition 3.3 equals

$$O\left(\frac{n^*T}{\epsilon}\log(TU_g)\left(n^*t_g + (n^* + \log U_{\mathcal{A}})t_{\text{tran}} + n^*\log\left(\frac{T}{\epsilon}\log(TU_g)\right)\right)\right),$$

see also the detailed calculations in the proof of [HKL$^+$14, Thm 8.2]. By Condition 6, $\log U_{\mathcal{A}}$ is polynomially bounded in the (binary) input size, and because U_g is part of the problem input, we get that the running time stated in Proposition 3.3 is polynomial in the (binary) input size and $\frac{1}{\epsilon}$.

Since (3) is an indirect DP, by Theorem 3.2, Algorithm 4 serves as an FPTAS for it, and its running time is

$$O\left(\frac{n^*T^3}{\epsilon^2}\left(n^*t_g + (n^* + \log U_{\mathcal{A}})t_{\text{tran}} + n^*\log\left(\frac{T}{\epsilon}\log(TU_g)\right)\right)\log^2(TU_g)\log U_{\mathcal{S}}\right), \tag{24}$$

which is equal, to the running time stated in [HKL$^+$14].

Remark 4 *In Condition 7, a direct access to the function g_j is required for $1 \le j \le T+1$. However, we note that by Proposition 3.3,* (7) *serves as the* FPTAS$\bar{f}_t\left(I_t, A_t(I_t), \{\tilde{z}_r\}_{r<t}, \epsilon\right)$ *in Condition 4(ii) even if there is no direct access to g, and instead there exists a monotone FPTAS for calculating the value of g_t, $1 \le t \le T+1$. Hence, Condition 7 can be replaced with the following one:*

Condition 9 *For every $j = 1, \ldots, T+1$, function* tran_j *is either given explicitly (i.e., as explicit formula) or accessed via oracle calls, and a monotone FPTAS for evaluating g_j is given. Moreover, the values returned by the FPTAS are nonnegative rational numbers that are polynomially bounded by the (binary) input size.*

The only change in the running time analysis for this case is that now t_g serves as an upper bound on the calculation time of the FPTAS for evaluating g, and not on the calculation time of g itself. Hence, Extension 10.5 in [HKL$^+$14] is also included in our meta-framework.

C.1.1 Applications

The authors of [HKL$^+$14] provide eight application problems for the monotone explicit stochastic model. We already described one of them, i.e, the Dynamic Capacity Expansion problem in Section 3.7. For the remaining seven applications, we provide here the short descriptions appearing in [HKL$^+$14, Sec. 1]. We refer the reader to [HKL$^+$14, Appx. A] for full descriptions of the applications, as well as proofs that they are indeed monotone explicit stochastic DPs. We express here the running time of the problems. Note that except for the first application, all the other six applications are formulated as indirect DPs, and hence the running time of their FPTASes are based on (24).

1. Stochastic ordered adaptive knapsack problem [HKL$^+$14, Appx. A.1]: A number of items are to be considered sequentially for placing into a knapsack. Each item i has a deterministic profit π_i and a stochastic volume v_i in which the distribution is known in advance. The actual volume of an item is unknown until we instantiate the item by attempting to place it into the knapsack, and we have to decide whether or not to select the item for packing. The packing process will be terminated once the knapsack capacity is exceeded. The objective is to maximize the expected total profit of the packed items.

In a similar manner to what we show in Section 3.4 for the classic 0/1 knapsack problem, this problem also can be proven to be a direct DP, and its FPTAS runs in time $O(\frac{n^* n^2}{\epsilon} \log \frac{n \log \Pi}{\epsilon} \log B \log \Pi)$.

2. Nonlinear knapsack problem [HKL$^+$14, Appx. A.2]: This problem is similar to the classical integer knapsack problem, in which a quantity of each given item is selected and packed into the knapsack. However, instead of having fixed volumes and profits per unit, a general nondecreasing volume function and a general nondecreasing profit function are given; that is, placing x units of item i into the knapsack will result in a profit of $\pi_i(x)$ and take up a volume of $v_i(x)$. The objective is to maximize the total profit without exceeding the knapsack capacity of the knapsack.

Let u_i be the maximum units from the i'th item which can be placed in the knapsack, $1 \leq i \leq n$, let $u_{\max} = \max_{1\leq i \leq n} u_i$, and let $\Pi = \sum_{i=1}^{n} \pi(u_i)$. The running time is $O\left(\frac{n^3}{\epsilon^2}\left(\log u_{\max} + \log\left(\frac{n}{\epsilon}\log \Pi\right)\right)\log^2 \Pi \log B\right)$.

3. Time-cost tradeoff machine scheduling [HKL$^+$14, Appx. A.4]: There is a single machine and a given set of jobs. The processing time of a job is a nonincreasing function of the amount of monetary resources allocated to it. Each job is given a due date, and a late penalty will be incurred if the job completes after its due date. The objective is to determine the job processing times and to schedule the jobs on the machine in such a way that the sum of the total late penalty and the total resource consumption is minimized.

Let $U = \sum_{j=1}^{n} \min\{w_j, \rho(0)\}$, and let d_1 be the maximal due date. The running time is $O\left(\frac{n3}{\epsilon^2}\log^2 d_1 \log^2 U \log\left(\frac{n}{\epsilon}\log U\right)\right)$.

4. Single-item stochastic batch dispatch [HKL$^+$14, Appx. A.6]: Consider running a dispatch station over a finite time horizon, where a vehicle with a finite capacity is available to dispatch goods in batches. In each time period, goods arrive randomly based on a time-dependent distribution known in advance. The decision in each time period is whether we should send off the vehicle and if yes, how many units of the goods should be carried by the vehicle. Dispatching the vehicle will incur a fixed cost as well as a per-unit cost of the dispatched goods, while the goods left at the dispatch station will incur a per-unit holding cost.

Let $U = TK_{\max} + c_{\max}(I_1 + D^*) + h_{\max}(I_1 + D^*)$, where $K_{\max}$ is the maximal fixed cost over the time horizon, $c_{\max}$ the maximal per-unit cost, and $h_{\max}$ the maximal holding cost. The running time is $O\left(\frac{n^* T^3}{\epsilon^2}\left(n^* + \log(I_1 + D^*) + \log\left(\frac{T}{\epsilon}\log(U)\right)\right)\log^2(U)\log(I_1 + D^*)\right)$.

5. Single-resource revenue management [HKL$^+$14, Appx. A.7]: There is a single resource with a given limited capacity C (e.g., an airplane with seat capacity C for a specific flight). There are T customer classes, in which class t has a revenue contribution of r_t per arrival. All customers in class t arrive in time period t, and the number of such customers is distributed randomly based on a random variable D_t with a known distribution. We assume no cancellations or no-shows, no overbookings, and independent customer arrivals. The problem is to find acceptance policies to maximize the expected total revenue.

Let $r_{\max}$ be the maximal revenue contribution per arrival. The running time is

$$O\left(\frac{n^* T^3}{\epsilon^2}\left(n^* + \log C + \log\left(\frac{T}{\epsilon}\log(Tr_{\max}C)\right)\right)\log^2(Tr_{\max}C)\log C\right).$$

6. Lifetime consumption of risky capital [HKL$^+$14, Appx. A.8]: Consider an individual managing her capital over a finite time horizon. In each time period, she can consume some of her capital, and the subsequent utility is derived from her consumption based on an underlying utility function. The remaining capital yields a stochastic return. In addition, she receives an income at the end of the period. The problem is to determine an optimal consumption strategy which maximizes her expected total utility.

Denote by $y_{\max}$ the maximal income she receives. Since every $d_{t,i} \in \mathcal{D}_t$ is rational, we write it as $d_{t,i} = \frac{r_{t,i}}{q_{t,i}}$. Let $U = T\log(1+D^*) + \log(I_1 + Ty_{\max}) + \sum_{j=1}^{T}\sum_{i=1}^{n_j}\log q_{j,i}$, and $\breve{U} = \sum_{i=1}^{T} u_t(U)$. The running time is $O\left(\frac{n^*T^3}{\epsilon^2}\left(n^* + \log U + \log\left(\frac{T}{\epsilon}\log(\breve{U})\right)\right)\log^2(\breve{U})\log U\right)$.

7. Stochastic growth model [HKL$^+$14, Appx. A.9]: This is a variant of "lifetime consumption of risky capital." In each time period, a household decides how much of its capital it should consume, and utility is derived from its consumption. The rest of the capital can be used to produce output via a production process. There is a deterministic depreciation of the remaining capital, but fluctuations in capital are created by random shocks to the production process. The objective is to maximize the expected total utility throughout the time horizon. As [HKL$^+$14] shows, the problem can be converted to a problem with integer state space. W.l.o.g, we relate here to the converted problem. Denote by $I_j^{\max}$ an upper bound on $\mathcal{S}_j$ for $1 \le j \le T$. The value of $I_j^{\max}$ can be calculated by the recursion $I_{j+1}^{\max} = I_j^{\max} + d_{j,n_j} p_j(I_j^{\max})$. Then, $I_T^{\max}$ is an upper bound on the state space, and $\breve{u} = \sum_{i=1}^{T} u_j(I_j^{\max})$ is an upper bound on the value of the problem. Hence, the running time is

$$O\left(\frac{n^*T^3}{\epsilon^2}\left(n^* + \log I_T^{\max} + \log\left(\frac{T}{\epsilon}\log\breve{u}\right)\right)\log^2(\breve{u})\log I_T^{\max}\right).$$

C.2 Implicit stochastic

The implicit stochastic framework introduced in [Hal19b] comes to deal with the same DP formula (3), but in the case where the random variables are not given explicitly, i.e., they are not part of the input, but are given as an oracle or a formula. This is done in order to deal with more complex (and realistic) stochastic processes, when the random variables are described by value oracles to their CDF. This allows to handle discrete random variables that can take on exponentially many values, and bounded-support continuous random variables. The conditions for the existence of their FPTAS framework are similar to Conditions 6-8.

Before we turn to state their conditions, we bring here a sufficient condition for approximating the expected value of a single-period cost function, i.e., for approximating $E_{D_j} g_j(I_j, x_j, D_j)$, which assumes that g_j has some special structure. (For every $j = 1, ...T$ denote by F_j an oracle to the CDF of D_j).

Condition 10 (Condition 4 in [Hal19b]) *g_j can be expressed as*

$$g_j(I,x,d) = g_j^{Ix}(I,x) + g_j^I(\text{tran}_j^I(I,d)) + g_j^x(\text{tran}^x(x,d)) + g_j^u(\text{tran}_j^u(I,x,d)),$$

where for every $\Diamond \in \{Ix, I, x, u\}$, $g_j^{\Diamond}$ is monotone. For $\Diamond \in \{I, x\}$, $\text{tran}_j^{\Diamond}$ is either linear and separable in its variables or Prob$(\text{tran}_j^{\Diamond}(\cdot, D) \geq a)$ is monotone for every fixed a and can be expressed as a linear and separable function of F_j multiplied by an indicator on the relation between its variable and a. Furthermore, tran_j^u is either linear and separable in its variables or Prob$(\text{tran}_j^u(I,x,D) \geq a)$ is monotone in x for every fixed I, a and can be expressed as a linear and separable function of F_j multiplied by an indicator on the relation between I, x, a.

For any $\gamma > 0$ we say that γ is a *bound on the CDF of the demand D* if $\text{Prob}(D \leq x) > 0$ implies $\text{Prob}(D \leq x) \geq \gamma$ and $\text{Prob}(D \leq y) < 1$ implies $\text{Prob}(D \leq y) \leq 1 - \gamma, \forall x, y$. The DP formula (3) is an implicit stochastic monotone DP if the following conditions hold (Conditions 1-3 in [Hal19b]):

Condition 11 *$\mathcal{S}_{T+1}, \mathcal{S}_j, \mathcal{A}_j(I_j) \subset \mathbb{Z}$ for $I_j \in \mathcal{S}_j$ and $j = 1, \ldots, T$. For any set X among these sets, $\log \max_{x \in X}(|x| + 1)$ is bounded polynomially by the (binary) input size, and the kth largest element in X can be identified in constant time for any $1 \leq k \leq |X|$. For every $j = 1, \ldots, T$, the support of D_j is contained in $\{A_j, \ldots, B_j\}$.*

Condition 12 *For every $j = 1, \ldots, T+1$, functions tran_j, g_j and the CDF F_j of D_j are either given explicitly (i.e., as explicit formula) or are accessed via oracle calls. Moreover, the values of g_j, F_j are nonnegative rational numbers that are polynomially bounded by the (binary) input size.*

Condition 13 *At least one of the following properties holds:*

(i) **(Non-decreasing DP)** *Function g_{T+1} is non-decreasing. For $j = 1, \ldots, T$, function tran_j is non-decreasing in its first variable and monotone in its second variable, and g_j is monotone in its second variable. Either monotone FPTASes to calculate $E_D g_j(I, \cdot, D)$ are given for any fixed I, or Condition 10 is met. $\text{tran}_j(I,x,D)$ is either linear and separable in its variables, or Prob$(\text{tran}_j(I,x,D)a)$ is monotone in x for every fixed a, I and can be expressed as a linear and separable function of F_j multiplied by an indicator on the relation between I, x, a. Moreover, for each $j = 1, \ldots, T$, either z_j is non-decreasing, or g_j is non-decreasing in its first variable and $\mathcal{A}_j(I) \subseteq \mathcal{A}_j(I')$ for all $I, I' \in \mathcal{S}_j$ with $I \geq I'$.*

(ii) **(Non-increasing DP)** *Function g_{T+1} is non-increasing. For $j = 1, \ldots, T$, function tran_j is non-decreasing in its first variable and monotone in its second variable, and g_j is monotone in its second variable. Either monotone FPTASes to calculate $E_D g_j(I, \cdot, D)$ are given for any fixed I, or Condition 10 is met. $\text{tran}_j(I,x,D)$ is either linear and separable in its variables, or Prob$(\text{tran}_j(I,x,D)a)$ is monotone in x for every fixed a, I and can be expressed as a linear and separable function of F_j multiplied by an indicator on the relation between I, x, a. Moreover, for each $j = 1, \ldots, T$, either z_j is non-increasing, or g_j is non-increasing in its first variable and $\mathcal{A}_j(I) \subseteq \mathcal{A}_j(I')$ for all $I, I' \in \mathcal{S}_j$ with $I \leq I'$.*

Next, we show that if Conditions 11-13 hold then DP (3) is an indirect DP. The verification that the first three Conditions 1-3 hold is similar to the explicit stochastic case: Condition 1 holds by Proposition C.1 with Condition 8(i) replaced by Condition 13(i). Condition 2 holds by Condition 11 and the fact that $\log U_g$ is polynomially bounded by the (binary) input size. Condition 3 follows from summation, composition, and minimization of approximation (Proposition 2.2(3,4,6)).

Before we approach Condition 4(ii), we give three propositions. The first one provides a way to approximate the expectation.

Proposition C.2 (Proposition 2 in [HOS12]) *Let D be an integer-valued random variable and suppose Prob$(f(x,D) \geq a_i)$ is monotone in x, $i = 1, \ldots, n$. Let $\xi : \{A, \ldots, B\} \rightarrow \mathbb{R}^+$ be a nonnegative increasing function. Let $K_1, K_2 \geq 1$, $\xi(a_0) = 0$, and let $S = \{a_1 < \ldots < a_n\}$ be a K_1-approximation set of ξ. Finally, let $\eta_i(x)$ denote Prob$(f(x,D) \geq a_i)$ and let $\tilde{\eta}_i(\cdot)$ be a K_2-approximation of $\eta_i(\cdot)$, $i = 1, \ldots, n$. Then*

$$\tilde{\xi}_0(x) = \sum_{i=1}^{n} (\xi(a_i) - \xi(a_{i-1}))\tilde{\eta}_i(x) \;\; \textit{is a } K_1K_2\textit{-approximation of } \; E_D(\xi(f(x,D))).$$

Moreover, if $\tilde{\eta}_i(\cdot)$ are monotone, then so is $\tilde{\xi}_0(\cdot)$.

Condition 13 holds if either monotone FPTASes to calculate $E_D g_j(I, \cdot, D)$ are given for any fixed I, or Condition 10 is met. The following proposition regards to the first case.

Proposition C.3 (Proposition E.1 in [Hal19b]) *Suppose the DP formulation* (3) *satisfies Condition 13. Let K', L', L'', j, and I_j be fixed values, where $K', L' \geq 1$, $L'' \leq K'L', I_j \in \mathcal{S}_j$, and $j \in \{0, \ldots, T\}$. Let g_j be as stated in Condition 13. Let $\tilde{G}_j(I_j, \cdot)$ be a monotone L''-approximation of $E_{D_j}\big(g_j(\mathrm{tran}_j(I_j, \cdot, D_j))\big)$. Let $\tilde{Z}\mathrm{tran}_{j+1}$ be a monotone L'-approximation of $E_{D_j}\big(z_{j+1}(\mathrm{tran}_j(I_j, \cdot, D_j))\big)$, and W be a K'-approximation set of $\tilde{Z}\mathrm{tran}_{j+1}$. Let*

$$\bar{z}_j(I_j) = \min_{x_j \in W} \{\tilde{G}_j(I_j, x_j) + \tilde{Z}\mathrm{tran}_{j+1}(I_j, x_j)\}. \tag{25}$$

Then, $\bar{z}_j(I_j)$ is an unnecessarily monotone $K'L'$-approximation value of $z_j(I_j)$, and it can be determined in $O((t_{\tilde{G}_j} + t_{\tilde{Z}\mathrm{tran}_{j+1}})|W|)$ time if the elements of W are given.

The next proposition regards to the second case, where Condition 10 is met.

Proposition C.4 (Proposition 11 in [Hal19b]) *Suppose the DP formulation* (3) *satisfies Condition 10. Let K^{Ix}, K^x, K^u, K^z, L^{Ix}, L^I, L^x, L^u, L^z, t, and I_j be fixed values, where $K^z, L^z \geq 1$, $L^{Ix}, L^I, K^xL^x, K^uL^u \leq K^zL^z$. Let g_j be as stated in Condition 10. Let $\tilde{G}^I_j(I_j)$ be an L^I-approximation value of $G^I_j(I_j) := E_D\big(g^I_j(\mathrm{tran}^I_j(I_j, D_j))\big)$. Let $\tilde{g}^{Ix}_j(I_j, \cdot)$ be a monotone L^{Ix}-approximation of $g^{Ix}_j(I_j, \cdot)$, and let W^{Ix} be a K^{Ix}-approximation set of $\tilde{g}^{Ix}_j(I_j, \cdot)$. Let $\tilde{G}^x_j(\cdot)$ be a monotone L^x-approximation of $G^x_j(\cdot) := E_{D_j}\big(g^x_j(\mathrm{tran}^x_j(\cdot, D_j))\big)$, and let W^x be a K^x-approximation set of $\tilde{G}^x_j(\cdot)$. Let $\tilde{G}^u_j(I_j, \cdot)$ be a monotone L^u-approximation of $G^u_j(I_j, \cdot) := E_{D_j}\big(g^u_j(\mathrm{tran}^u_j(I_j, \cdot, D_j))\big)$, and let W^u be a K^u-approximation set of $\tilde{G}^u_j(I_j, \cdot)$. Last, let $\tilde{Z}\mathrm{tran}_{j+1}(I_j, \cdot)$ be a monotone L^z-approximation of $Z\mathrm{tran}_{j+1}(I_j, \cdot) := E_{D_j}\big(z_{j+1}(\mathrm{tran}_j(I_j, \cdot, D_j))\big)$, and W^z be a K^z-approximation set of $\tilde{Z}\mathrm{tran}_{j+1}(I_j, \cdot)$. Let*

$$\bar{z}_j(I_j) = \tilde{G}^I_j(I_j) + \min_{x_j \in W^{Ix} \cup W^x \cup W^u \cup W^z} \{\tilde{g}^{Ix}_j(I_j, x_j) + \tilde{G}^x_j(x_j) + \tilde{G}^u_j(I_j, x_j) + \tilde{Z}\mathrm{tran}_{j+1}(I_j, x_j)\}. \tag{26}$$

Then, $\bar{z}_j(I_j)$ is an unnecessarily monotone K^zL^z-approximation value of $z_j(I_j)$, and it can be determined in $O((t_{\tilde{g}^{Ix}_j} + t_{\tilde{G}^x_j} + t_{\tilde{G}^u_j} + t_{\tilde{Z}\mathrm{tran}_{j+1}})|W^{Ix} \cup W^x \cup W^u \cup W^z| + t_{G^I_j})$ time if the elements of W^{Ix}, W^x, W^u, W^z are given.

We are now ready to show that Condition 4(ii) holds. We start with the first case, where monotone FPTASes to calculate $E_D g_j(I, \cdot, D)$ are given for any fixed I. For every $1 \leq j \leq T$, if a K'-approximation function $\tilde{z}_{j+1}$ is given for z_{j+1}, then $\tilde{Z}\mathrm{tran}_{j+1}$ and W from Proposition C.3 can be calculated in three steps: (i) Construct a K-approximation of F_j by calling FPTASCompress$(F_j, K-1)$, (ii) compute a $K'K$-approximation oracle $\bar{Z}\mathrm{tran}_{j+1}(I, \cdot)$ for $E_{D_j}\big(z_{j+1}(\mathrm{tran}_j(I_j, \cdot, D_j))\big)$, using Proposition C.2, (iii) use the output of FPTASCompress$(\bar{Z}\mathrm{tran}_{j+1}(I, \cdot), \mathcal{S}_j, K-1)$ as $\tilde{Z}\mathrm{tran}_{j+1}$ and W.

The running time of these three steps and $|W|$ are all polynomially bounded by the (binary) input size, see Proposition 2.5. Then, Condition 4(ii) holds from Proposition C.3 (Recall that $\tilde{G}_j(I_j,\cdot)$ can be calculated in polynomial times by Condition 13).

The arguments dealing with the case where Condition 10 is met are similar: First, we obtain K'-approximation functions for each of the four components of g_j. Then, using the same three steps as before, we obtain in polynomial time the $K'K^2$-approximation sets and functions W^{Ix}, W^x, W^u, W^z, $\tilde{G}^I_j(I_j), \tilde{G}^x_j(x_j), \tilde{G}^u_j(I_j,x_j), \tilde{Z}\text{tran}_{j+1}(I_j,x_j)$. Now, Condition 4(ii) holds from Proposition C.4. For more detailed explanations, see Section 6.2.1 and Appendix E in [Hal19b].

By Theorem 3.2, Algorithm 4 serves as an FPTAS for monotone implicit stochastic DPs, with running time $O\left(\frac{T^2}{\epsilon}t_{\bar{f}}\log(JU_g)\log U_{\mathcal{S}}\right)$. Similar to [Hal19b], we analyze the running time of the case where Condition 10 is met. From Proposition C.4, $t_{\bar{f}} = O((t_{\tilde{g}^{Ix}_j} + t_{\tilde{G}^x_j} + t_{\tilde{G}^u_j} + t_{\tilde{Z}\text{tran}j+1})|W^{Ix}\cup W^x \cup W^u \cup W^z| + t_{G^I_j})$ plus the time to construct $\tilde{G}^I_j(I_j), \tilde{G}^x_j(x_j), \tilde{G}^u_j(I_j,x_j), \tilde{Z}\text{tran}_{j+1}(I_j,x_j), W^{Ix}, W^x, W^u, W^z$. The detailed analysis in Section 6.2.1 shows that $t_{\bar{f}} = O\left(\left(\frac{T}{\epsilon}\right)^2 \log\left(\frac{T}{\epsilon}\log\frac{1}{\gamma}\right)\log^2 U_z \log U_{\mathcal{S}}\right)$, and the time needed to construct $\tilde{G}^I_j(I_j), \tilde{G}^x_j(x_j), \tilde{G}^u_j(I_j,x_j), \tilde{Z}\text{tran}_{j+1}(I_j,x_j), W^{Ix}, W^x, W^u, W^z$ is

$$O\left(\frac{T}{\epsilon}t_F \log\frac{1}{\gamma}\log U_{\mathcal{D}} + \left(\frac{T}{\epsilon}\right)^2\left(t_g + \log\left(\frac{T}{\epsilon}\log\frac{1}{\gamma}\right)\right)\log\left(U_{\mathcal{A}}U_{\mathcal{S}}\right)\log\frac{U_z}{\gamma}\log(JU_g)\right),$$

and so, the overall running time of the algorithm is

$$O\left(\frac{T^4}{\epsilon^3}\log(JU_g)\log U_{\mathcal{S}}\left(t_F\log\frac{1}{\gamma}\log U_{\mathcal{D}} + \left(t_g + \log\left(\frac{T}{\epsilon}\log\frac{1}{\gamma}\right)\right)\log\left(U_{\mathcal{A}}U_{\mathcal{S}}\right)\log\frac{U_z}{\gamma}\log(JU_g)\right)\right).$$

The running time here is a bit slower than the one stated in [Hal19b]: there, the factor of $\frac{T^2}{\epsilon}\log(JU_g)\log U_{\mathcal{S}}$ is multiplied only by $t_{\bar{f}}$, and the construction time is multiplied only by T. This is since the construction is done only once at every iteration. Applying this change in Algorithm 4 provides the running time of [Hal19b].

All the stochastic applications in Appendix C.1.1, i.e., applications 1,4,5,6,7, can be formulated as implicit stochastic monotone DPs, when replacing the explicit random variables with implicit ones; for a detailed explanations see [Hal19b, Appx. A].

D Statement of function FPTASApxSet

Algorithm 6 Constructing a $(1+\epsilon)$-approximation set for a monotone function $\varphi : D \to \mathbb{R}^+$.

```
1: Function FPTASApxSet($\varphi, \epsilon$)
2: $x \leftarrow D^{\max}$,   $W \leftarrow \{D^{\min}, D^{\max}\}$
3: while $x > D^{\min}$ do
4:     if $\varphi$ is monotone increasing then $x \leftarrow \min\{\text{prev}(x,D), \min\{y \in D \mid (1+\epsilon)\varphi(y) \geq \varphi(x)\}\}$
5:     else $x \leftarrow \min\{\text{prev}(x,D), \min\{y \in D \mid (1+\epsilon)\varphi(x) \geq \varphi(y)\}\}$ /* $\varphi$ is monotone decreasing */
6:     $W \leftarrow W \cup \{x\}$
7: end while
8: Return $W$
```

E Statement of function FPTASIndirectApxSet

For simplicity, we state the function for the case φ is monotone increasing. Moreover, we omit the parameter D in IndirectApxSet whenever the domain of the function is clear from the context.

Algorithm 7 Constructing a subset of D for a function $\bar{\varphi} : D \to \mathbb{R}^+$ that approximates a monotone increasing function $\varphi : D \to \mathbb{R}^+$.

```
1: Function FPTASIndirectApxSet($\bar{\varphi}, \epsilon$) [HKL+14, Alg. 2]
2: $x \leftarrow D^{\max}$ and $\bar{W} \leftarrow \{D^{\min}, D^{\max}\}$
3: while $x > D^{\min}$ and $(1+\epsilon)\bar{\varphi}(D^{\min}) < \bar{\varphi}(x)$ do
4:     $x \leftarrow x' \mid x' < x$ and $(1+\epsilon)\bar{\varphi}(x') < \bar{\varphi}(x)$ and $(1+\epsilon)\bar{\varphi}(\text{next}(x', D)) \geq \bar{\varphi}(x)$
5:     $\bar{W} \leftarrow \bar{W} \cup \{x, \text{next}(x, D)\}$
6: end while
7: Return $\bar{W}$
```

F Proof of Theorem 3.2

We first show by induction that $\tilde{z}_{t,i}$ is a monotone (in the same direction of $z_{t,i}$) $K^{2i(t+1)}$-approximation of $z_{t,i}$, for $0 \leq t \leq T, 1 \leq i \leq m$. For the base case of $t = 0, i = 1$, by calling the FPTAS for $z_{0,1}$ with ϵ, we get that $\bar{z}_{0,1} =$ is a K approximation function of $\bar{z}_{0,1}$ by Condition 4(ii), since $\{\tilde{z}_{r,j}\}_{r \leq 0, j \leq 1 : r+j<1}$ is an empty set. Function FPTASIndirectCompress($\bar{z}_{0,1}, K$) returns a monotone K-approximation function of $\bar{z}_{0,1}$, see Proposition 2.7, which is a K^2-approximation of $z_{0,1}$ by approximation of approximation (Proposition 2.2(7)).

Assume that the claim holds for every $r \leq t, j \leq i : r + j < t + i$; so, $\tilde{z}_{r,j}$ is a monotone $K^{2j(r+1)}$-approximation of $z_{r,j}$ for $r \leq t, j \leq i : r + j < t + i$. By the induction hypothesis $\{\tilde{z}_{r,j}\}_{r \leq t, j \leq i : r+j<t+i}$ are at most $K^{2i(t+1)-2}$-approximation functions of $\{z_{r,j}\}_{r \leq t, j \leq i : r+j<t+i}$, and by condition 3, $f_{t,i}\left(I_{t,i}, A_{t,i}(I_{t,i}), \{\tilde{z}_{r,j}\}_{r \leq t, j \leq i : r+j<t+i}\right)$ is a $K^{2i(t+1)-2}$ approximation function of $z_{t,i}$. By condition 4(ii), $\bar{z}_{t,i}$ is a $K^{2i(t+1)-1}$ approximation function of $z_{t,i}$. The function FPTASIndirectCompress($\bar{z}_{t,i}, K$) returns a monotone (in the same direction of $z_{t,i}$) K-approximation of $\bar{z}_{t,i}$, see Proposition 2.7. Then, by approximation of approximation (Proposition 2.2(7)), $\tilde{z}_{t,i}$ is a $K^{2i(t+1)}$-approximation of $z_{t,i}$. This concludes the induction proof.

Now, applying the claim for $t = T, i = m$, it turns out that $\tilde{z}_{T,m}$ is a $K^{2m(T+1)}$-approximation of $z_{T,m}$. By the definition of K at stage 2, $K^{2m(T+1)} = 1 + \epsilon$, which proves the approximation ratio.

Regarding the running time, Algorithm 3 performs $m(T+1)$ iterations, in each of them the running time is dominated by the call to FPTASIndirectCompress. By Proposition 2.7, the running time of FPTASIndirectCompress is $O\left(t_{\bar{z}} \log\log_K U_z \log_K U_z \log U_{\mathcal{S}}\right)$ (the calculation time of $\bar{z}_{t,i}$, $1 \leq t \leq T, 1 \leq i \leq m$, is $t_{\bar{\bar{f}}} = t_{\bar{f}} \log\log_K U_z$, where the $\log\log_K U_z$ is the query time to $\tilde{z}_{r,j}$ from previous stages). Since $O\left(\log_K U_z\right) = O\left(\frac{mT \log U_z}{\log(1+\epsilon)}\right) = O\left(\frac{mT \log U_z}{\epsilon}\right)$ for every $0 < \epsilon < 1$, then the running time of FPTASIndirectCompress is $O\left(\frac{mT}{\epsilon} t_{\bar{\bar{f}}} \log \frac{mT \log U_z}{\epsilon} \log U_z \log U_{\mathcal{S}}\right) = O\left(\frac{mT}{\epsilon} t_{\bar{\bar{f}}} \log U_z \log U_{\mathcal{S}}\right)$, which, using Conditions 2 and 4(ii), concludes the running time proof. □