When does a dynamic programming formulation guarantee the existence of an FPTAS?

Gerhard J. Woeginger / Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria, Email: gwoegi@opt.math.tu-graz.ac.at

Abstract

We derive results of the following flavor: If a combinatorial optimization problem can be formulated via a dynamic program of a certain structure and if the involved cost and transition functions satisfy certain arithmetical and structural conditions, then the optimization problem automatically possesses a fully polynomial time approximation scheme (FPTAS).

Our characterizations provide a natural and uniform approach to fully polynomial time approximation schemes. We illustrate their strength and generality by deducing from them the existence of FPTASs for a multitude of scheduling problems. Many known approximability results follow as corollaries from our main result.

Keywords: Approximation algorithm; approximation scheme; FPTAS; worst case analysis; dynamic programming; combinatorial optimization; scheduling theory.

1 Introduction

Assume that we are working on a combinatorial optimization problem in which each feasible solution has a positive integer cost. We are looking for an *optimal* solution. Our problem may either be a *minimization* problem where the optimal solution is a feasible solution with *minimum* possible cost, or a *maximization* problem where the optimal solution has the *maximum* possible cost. For many important optimization problems it is $\mathcal{NP}$-hard to determine the exact optimal solution. One approach to getting around $\mathcal{NP}$-hardness is not to look for exact optimal solutions, but to look for *near-optimal* solutions instead, which often is good enough in practice. An algorithm that returns near-optimal solutions is called an *approximation algorithm*.

Assume that we have an approximation algorithm that always returns a near-optimal solution whose cost is at most a factor of ρ away from the optimal cost, where $\rho > 1$ is some real number: In minimization problems the near-optimal cost is at most a multiplicative factor of ρ above the optimum, and in maximization problems it is at most a factor of ρ below the optimum. Such an approximation algorithm is called a ρ-approximation algorithm. A family of $(1+\varepsilon)$-approximation algorithms over all $\varepsilon > 0$ with polynomial running times is called a polynomial time approximation scheme or PTAS, for short. If the time complexity of a PTAS is also polynomially bounded in $1/\varepsilon$, then it is called a *fully* polynomial time approximation scheme or FPTAS, for short. An FPTAS is the strongest possible polynomial time approximation result that we can derive for an $\mathcal{NP}$-hard problem (unless, of course, $\mathcal{P} = \mathcal{NP}$ holds).

The first fully polynomial time approximation schemes were developed in the mid-1970s in the work of Horowitz, Ibarra, Kim, and Sahni [20, 21, 39]. The term 'approximation scheme' itself seems to be due to a paper by Garey & Johnson [12] from 1978 which mainly discusses negative results: It identifies certain conditions on optimization problems that guarantee that the problem does *not* possess an FPTAS. The results in [12] lead to the natural problem of finding a 'meaningful' and 'useful' characterization of the fully polynomial time approximable optimization problems. What we ideally would like to have is a mathematical theorem from which we could tell at first glance whether some given optimization problem does have an FPTAS or whether it does not have an FPTAS unless $\mathcal{P} = \mathcal{NP}$ or whether it does not have an FPTAS at all. Theoretical computer scientists studied this problem (cf. e.g. Ausiello, D'Atri & Protasi [2], Ausiello, Marchetti-Spaccamela & Protasi [3], Paz & Moran [36], Ausiello, Crescenzi & Protasi [1]), and they derived deep and beautiful characterizations of fully polynomial time approximable problems. Unfortunately, all these characterizations seem to be of the form "An optimization problem has an FPTAS if and only if there exists a *polynomial time computable* function for which ....." The whole difficulty of the problem seems to be concentrated in this "polynomial time computable function", and the characterization does not help us at all in detecting this function and in designing the FPTAS (we advise the reader to study e.g. Theorem 4.20 in [36]). This is of course a consequence of the fact that all these characterizations are *exact* characterizations of the class of fully polynomial time approximable problems and thus implicitly suffer from the difficulty of the $\mathcal{P} = \mathcal{NP}$ question.

In this paper, we are going for a more modest goal: We will be satisfied with a set of conditions on optimization problems that (i) guarantee the existence of an FPTAS, that (ii) cover a relatively large subclass of the fully polynomial time approximable problems, and that — probably the most important — (iii) are easy to verify. In order to reach this goal, we will carefully analyze one of the standard constructive approaches for designing an FPTAS.

The first FPTASs in [20, 21, 39] and apparently all the FPTASs that were developed since then are based on dynamic programming formulations which do *always* find the exact optimal solution, but not necessarily in polynomial time. There are two standard approaches for transforming a dynamic programming formulation into an FPTAS. The first approach is probably due to Sahni [39]. The main idea of this approach is to round the input data of the instance. The goal is to make the resulting rounded problem easy to solve and to bring the running time of the dynamic program down to polynomial. We call this technique the *rounding-the-input-data* technique. The second approach is due to Ibarra & Kim [21]. This approach does not round any numbers. The main idea is to iteratively thin out the state space of the dynamic program, and to collapse states that are 'close' to each other. The goal is to bring the size of the state spaces down to polynomial. In a pictorial setting, Ibarra & Kim are cleaning up the state space of the dynamic program, and therefore their technique is sometimes called the *trimming-the-state-space* technique.

Of course in both techniques the crucial point is how to control the error that is introduced by the changes in the input data and in the state space, respectively. Just from searching through the literature and from counting research articles, we conclude that the rounding-the-input-data technique by far is the more popular approach. This is strange, since the trimming-the-state-space technique seems to be more powerful and flexible. If we just round the input data and then let the dynamic program run as a black box until it terminates, we can neither control the propagation of error inside of the dynamic program, nor can we control the size or the

structure of the state space. As a consequence, the initial rounding in the rounding-the-input-data technique must yield an input of a very primitive and severely restricted form. In contrast to this, in the trimming-the-state-space technique we thin out the state space at the end of *every phase* in the dynamic program. Hence, we always have full control over the dynamic program and we can smooth out the propagation of error.

In this paper, we will identify a class of optimization problems that we call *DP-benevolent* problems. A DP-benevolent problem possesses a simple dynamic programming formulation that fulfills four structural conditions. The first three of these conditions are purely combinatorial and just enforce certain inequalities on the underlying functions in the dynamic program. The fourth condition imposes the existence of polynomial time algorithms for the initialization phase of the dynamic program and for the computation of the state spaces; clearly, this condition is rather mild. As our main result, we prove that every DP-benevolent optimization problem is fully polynomial time approximable. The proof is based on a very general implementation of the trimming-the-state-space technique of Ibarra & Kim [21]. In the rest of the paper, we then discuss a sequence of basic scheduling problems from the literature. All discussed scheduling problems have an FPTAS, and most of these FPTASs were derived via the rounding-the-input-data technique. We will demonstrate that all these problems in fact are DP-benevolent; hence, their fully polynomial time approximability also follows as a corollary from our main result.

Organization of the paper. The paper is organized as follows. Section 2 collects and clarifies some notations that are used throughout the paper. Section 3 defines the concept of DP-benevolence and formulates the main result; the proof of the main result is given in Section 4. Section 5 introduces the ex-benevolent problems which form an extremely simple subclass of the DP-benevolent problems. Section 6 discusses five scheduling problems that are all shown to be ex-benevolent and hence possess an FPTAS. Section 7 introduces the cc-benevolent problems, another simple subclass of the DP-benevolent problems. Section 8 analyzes six optimization problems that are all cc-benevolent and hence are fully polynomial time approximable. Section 9 discusses the possibilities for extending the notion of DP-benevolence to a wider class of optimization problems. Section 10 completes the paper with a short conclusion.

2 Notations and elementary definitions

The set of real numbers is denoted by $\mathbb{R}$, and the set of non-negative integers is denoted by $\mathbb{N}$; note that $0 \in \mathbb{N}$. The base two logarithm of z is denoted by $\log z$, and the natural logarithm by $\ln z$.

We recall the following well-known properties of binary relations $\preceq$ on a set Z. The relation $\preceq$ is called

- *reflexive*, if for any $z \in Z$: $z \preceq z$,
- *symmetric*, if for any $z, z' \in Z$: $z \preceq z'$ implies $z' \preceq z$,
- *anti-symmetric*, if for any $z, z' \in Z$: $z \preceq z'$ and $z' \preceq z$ implies $z = z'$,
- *transitive*, if for any $z, z', z'' \in Z$: $z \preceq z'$ and $z' \preceq z''$ implies $z \preceq z''$.

A relation on Z is called a *partial order*, if it is reflexive, anti-symmetric, and transitive. A relation on Z is called a *quasi-order*, if it is reflexive and transitive. A quasi-order on Z is called a *quasi-linear order*, if any two elements of Z are comparable.

The *trivial relation* on Z is the relation $\{z \preceq z : z \in Z\}$. The *universal relation* on Z is the relation $\{z \preceq z' : z, z' \in Z\}$. Note that the trivial relation is a partial order, and the universal relation is a quasi-linear order.

For $Z' \subseteq Z$, an element $z \in Z'$ is called a *maximum* in Z' with respect to $\preceq$, if $z' \preceq z$ holds for all $z' \in Z'$. The element $z \in Z'$ is called *maximal* in Z' with respect to $\preceq$, if the only $z' \in Z'$ with $z \preceq z'$ is z itself.

Proposition 2.1 *For any binary relation $\preceq$ on a set Z, and for any finite subset Z' of Z the following holds.*

(i) If $\preceq$ is a partial order, then there exists a maximal element in Z.

(ii) If $\preceq$ is a quasi-linear order, then there exists at least one maximum element in Z. ∎

A *polynomial* function P in v variables $x_1, \ldots, x_v$ is a function that can be written in the following form:

$$P(x_1, \ldots, x_v) \;=\; \sum_{K=(a_1,\ldots,a_v)} c_K \cdot \prod_{i=1}^{v} x_i^{a_i},$$

where the summation is done over all tuples K in $\mathbb{N}^v$, where the coefficients c_K are integers, and where only finitely many of the values c_K are non-zero. Every term $c_K \prod_{i=1}^{v} x_i^{a_i}$ in the summation with $c_K \neq 0$ is called a *monomial* of P. The degree of a univariate polynomial function $f : \mathbb{N} \to \mathbb{N}$ is denoted by $\deg(f)$.

3 Formulation of the main result

Throughout this section, we consider a GENEric optimization problem GENE. We start with four definitions that specify the representation of problem GENE and the relationship of GENE to its dynamic programming formulation DP.

Definition 3.1 *(Structure of the input in* GENE*)*
In any instance I of GENE*, the input is structured into n vectors $X_1, \ldots, X_n \in \mathbb{N}^\alpha$. Every vector X_k $(k = 1, \ldots, n)$ consists of α non-negative integer components $[x_{1,k}, x_{2,k}, \ldots, x_{\alpha,k}]$. All components of all vectors X_k are encoded in binary.*

The number α is a positive integer whose value may depend on the input I. Let $\overline{x}$ denote the sum $\sum_{k=1}^{n} \sum_{i=1}^{\alpha} x_{i,k}$. The size of the input of instance I is $\Theta(n + \log \overline{x})$. Note that in case instance I was encoded in unary, then its size would be $\Theta(\overline{x})$.

Definition 3.2 *(Structure of the dynamic program DP)*
The dynamic program DP for problem GENE *goes through n phases. The k-th phase $(k = 1, \ldots, n)$ processes the input piece X_k and produces a set $\mathcal{S}_k$ of states. Any state in the state space $\mathcal{S}_k$ is a vector $S = [s_1, \ldots, s_\beta] \in \mathbb{N}^\beta$. The number β is a positive integer whose value depends on* GENE*, but does not depend on any specific instance of* GENE*.*

Intuitively speaking, every state in the state space $\mathcal{S}_k$ encodes a solution to the subproblem specified by the partial input $X_1, \ldots, X_k$.

Definition 3.3 *(Iterative computation of the state space in DP)*
The set $\mathcal{F}$ is a finite set of mappings $\mathbb{N}^\alpha \times \mathbb{N}^\beta \to \mathbb{N}^\beta$. The set $\mathcal{H}$ is a finite set of mappings $\mathbb{N}^\alpha \times \mathbb{N}^\beta \to \mathbb{R}$. For every mapping $F \in \mathcal{F}$ there is a corresponding mapping $H_F \in \mathcal{H}$.

In the initialization phase of DP, the state space $\mathcal{S}_0$ is initialized by a finite subset of $\mathbb{N}^\beta$. In the k-th phase $(k = 1, \ldots, n)$ of DP, the state space $\mathcal{S}_k$ is obtained from the state space $\mathcal{S}_{k-1}$ via $\mathcal{S}_k = \{F(X_k, S) : \ F \in \mathcal{F},\ S \in \mathcal{S}_{k-1},\ H_F(X_k, S) \leq 0\}$.

The mappings in $\mathcal{F}$ translate the states of the 'old' state space $\mathcal{S}_{k-1}$ into states of the 'new' state space $\mathcal{S}_k$. Some of these new states may be infeasible and may not correspond to solutions for the subproblem specified by the partial input $X_1, \ldots, X_k$. The mappings in $\mathcal{H}$ serve as a tool for keeping such infeasible states out of the state space $\mathcal{S}_k$. For several illustrations how this tool can be applied, the reader is referred to the examples in Section 8.

Definition 3.4 *(Objective value in DP)*
The function $G : \mathbb{N}^\beta \to \mathbb{N}$ is a non-negative function. The optimal objective value of an instance I of GENE *is denoted by* $\textsc{Opt}(I)$.

- *If* GENE *is a minimization problem, then* $\textsc{Opt}(I) = \min\ \{G(S) : S \in \mathcal{S}_n\}$.
- *If* GENE *is a maximization problem, then* $\textsc{Opt}(I) = \max\ \{G(S) : S \in \mathcal{S}_n\}$.

An optimization problem GENE is called *DP-simple*, if it can be expressed via a simple dynamic programming formulation DP as described in Definitions 3.1–3.4. The dynamic programming formulation for a DP-simple problem GENE is summarized in Figure 1. Our next goal is to state a set of conditions that automatically guarantee the existence of an FPTAS for a DP-simple optimization problem GENE.

```
1  Initialize S_0
2  For k = 1 to n do
3      Let S_k := ∅
4      For every S ∈ S_{k-1} and every F ∈ F do
5          If H_F(X_k, S) ≤ 0 then add F(X_k, S) to S_k
6      EndFor
7  EndFor
8  Output  min / max  {G(S) : S ∈ S_n}
```

Figure 1: The dynamic programming formulation for a DP-simple optimization problem GENE.

In dynamic programs we often have a dominance relation among the states: Consider two states S and S' in the state space that represent solutions to the same subproblem. Assume that for any extension T of S to a solution for the complete problem, the corresponding extension T' of S' yields a solution whose objective value is at least as good as the objective value of the extension T. In such a situation we usually say that the state S is dominated by the state S'.

To capture the concept of dominance with respect to the DP formulation, we introduce two binary relations $\preceq_{dom}$ and $\preceq_{qua}$ on $\mathbb{N}^\beta$. The *dominance relation* $\preceq_{dom}$ is a partial order on $\mathbb{N}^\beta$, and $\preceq_{qua}$ is a quasi-linear order on $\mathbb{N}^\beta$. Moreover, $\preceq_{qua}$ is any extension of $\preceq_{dom}$, *i.e.*, $S \preceq_{dom} S'$ always implies $S \preceq_{qua} S'$. For two vectors $S, S' \in \mathbb{N}^\beta$ with $S \preceq_{dom} S'$, we say that S' *dominates* S. The dominance relation $\preceq_{dom}$ will typically be the product of linear or trivial orders on the components of the states. For several illustrating examples for relations $\preceq_{dom}$ and $\preceq_{qua}$, the reader is referred to Section 8.

In the fully polynomial time approximation scheme, we are going to trim the state space by merging certain states that are 'close' to each other into single states. While performing these mergings, we must ensure that the resulting errors cannot propagate in an uncontrolled way. To this end, the concept of $[D, \Delta]$-closeness is introduced in the following paragraph.

We fix a vector $D = [d_1, \ldots, d_\beta] \in \mathbb{N}^\beta$ which throughout the paper will be called the *degree-vector*. The degree-vector depends on GENE and on the DP formulation, but it does not depend on any specific instance of GENE. For a real number $\Delta > 1$ and two vectors $S, S' \in \mathbb{N}^\beta$ with $S = [s_1, \ldots, s_\beta]$ and $S' = [s'_1, \ldots, s'_\beta]$, we say that S is $[D, \Delta]$-*close* to S' if

$$\Delta^{-d_\ell} \cdot s_\ell \;\leq\; s'_\ell \;\leq\; \Delta^{d_\ell} \cdot s_\ell \qquad\qquad \text{for } \ell = 1, \ldots, \beta. \tag{1}$$

Note that for any $\Delta > 1$, the relation of being $[D, \Delta]$-close is a symmetric and reflexive relation on $\mathbb{N}^\beta$. Moreover, note that if two vectors are $[D, \Delta]$-close to each other, then they must agree in all coordinates ℓ with $d_\ell = 0$.

Now let us state some conditions that relate the functions in DP to the concepts of dominance and $[D, \Delta]$-closeness.

Condition C.1 *(Conditions on the functions in $\mathcal{F}$)*
For any $\Delta > 1$, for any $F \in \mathcal{F}$, for any $X \in \mathbb{N}^\alpha$, and for any $S, S' \in \mathbb{N}^\beta$, the following holds:

(i) If S is $[D, \Delta]$-close to S' and if $S \preceq_{qua} S'$, then (a) $F(X, S) \preceq_{qua} F(X, S')$ holds and $F(X, S)$ is $[D, \Delta]$-close to $F(X, S')$, or (b) $F(X, S) \preceq_{dom} F(X, S')$ holds.

(ii) If $S \preceq_{dom} S'$, then $F(X, S) \preceq_{dom} F(X, S')$.

Condition C.2 *(Conditions on the functions in $\mathcal{H}$)*
For any $\Delta > 1$, for any $H \in \mathcal{H}$, for any $X \in \mathbb{N}^\alpha$, and for any $S, S' \in \mathbb{N}^\beta$, the following holds:

(i) If S is $[D, \Delta]$-close to S' and if $S \preceq_{qua} S'$, then $H(X, S') \leq H(X, S)$.

(ii) If $S \preceq_{dom} S'$, then $H(X, S') \leq H(X, S)$.

Suppose for the moment that the degree-vector does not contain any zero entry. Then for any $S, S' \in \mathbb{N}^\beta$, there exists some sufficiently large Δ such that S is $[D, \Delta]$-close to S'. Since moreover $\preceq_{qua}$ is an extension of $\preceq_{dom}$, in this case the statements (i) and (ii) in Condition C.2 boil down to: If $S \preceq_{qua} S'$, then $H(X, S') \leq H(X, S)$.

Condition C.3 *(Conditions on the function G)*

(i) There exists an integer $g \geq 0$ (whose value only depends on the function G and on the degree-vector D) such that for any $\Delta > 1$ and for any $S, S' \in \mathbb{N}^\beta$ the following property holds: If S is $[D, \Delta]$-close to S' and if $S \preceq_{qua} S'$, then $G(S') \leq \Delta^g \cdot G(S)$ holds (in minimization problems), respectively, $\Delta^{-g} \cdot G(S) \leq G(S')$ holds (in maximization problems).

(ii) For any $S, S' \in \mathbb{N}^\beta$ with $S \preceq_{dom} S'$, $G(S') \leq G(S)$ holds (in minimization problems), respectively, $G(S') \geq G(S)$ holds (in maximization problems).

Finally we want that in DP every atomic step can be performed in polynomial time and that the lengths of the involved numbers do not explode. This is ensured by the following set of technical conditions.

Condition C.4 *(Technical conditions)*

(i) Every $F \in \mathcal{F}$ can be evaluated in polynomial time. Every $H \in \mathcal{H}$ can be evaluated in polynomial time. The function G can be evaluated in polynomial time. The relation $\preceq_{qua}$ can be decided in polynomial time.

(ii) The cardinality of $\mathcal{F}$ is polynomially bounded in n and $\log \overline{x}$.

(iii) For every instance I of GENE*, the state space S_0 can be computed in time that is polynomially bounded in n and $\log \overline{x}$. As a consequence, also the cardinality of the state space $\mathcal{S}_0$ is polynomially bounded in n and $\log \overline{x}$.*

(iv) For an instance I of GENE*, and for a coordinate ℓ $(1 \leq \ell \leq \beta)$, let $\mathcal{V}_\ell(I)$ denote the set of the values of the ℓ-th components of all vectors in all state spaces $\mathcal{S}_k$ $(1 \leq k \leq n)$. Then the following holds for every instance I.*

For all coordinates ℓ, $(\ell = 1, \ldots, \beta)$, the natural logarithm of every value in $\mathcal{V}_\ell(I)$ is bounded by a polynomial $\pi_1(n, \log \overline{x})$ in n and $\log \overline{x}$; equivalently, one may say that the length of the binary encoding of every value is polynomially bounded in the input size.

Moreover, for coordinates ℓ with $d_\ell = 0$, the cardinality of $\mathcal{V}_\ell(I)$ is bounded by a polynomial $\pi_2(n, \log \overline{x})$ in n and $\overline{x}$.

A DP-simple optimization problem GENE is called *DP-benevolent* iff there exist a partial order $\preceq_{dom}$, a quasi-linear order $\preceq_{qua}$, and a degree-vector D such that its dynamic programming formulation DP fulfills the Conditions C.1–C.4. DP-benevolent problems are easy to approximate:

Theorem 3.5 *(Main result on DP-benevolent problems)*
If an optimization problem GENE *is DP-benevolent, then it has an FPTAS.*

4 Proof of the main result

This section is devoted to the proof of Theorem 3.5. Before describing the FPTAS, let us state two simple but useful propositions.

Proposition 4.1

(i) For any $x \geq 1$, $\ln x \geq (x-1)/x$ holds.

(ii) For any $0 \leq x \leq 1$ and for any real $m \geq 1$, $(1+x/m)^m \leq 1+2x$ holds.

Proof. The inequality in (i) can be derived from the Taylor expansion of $\ln x$. In (ii) the left-hand side is a convex function in x, and the right-hand side is a linear function in x. Moreover, the claimed inequality holds at $x = 0$ and $x = 1$. ∎

Proposition 4.2 *Let S, S', and S'' be vectors in $\mathbb{N}^\beta$. Let $\Delta_1, \Delta_2 \geq 1$. If S is $[D, \Delta_1]$-close to S', and if S' is $[D, \Delta_2]$-close to S'', then S is $[D, \Delta_1 \cdot \Delta_2]$-close to S''.* ∎

Throughout the rest of this section we deal with the dynamic programming formulation DP of the DP-benevolent problem GENE in the statement of Theorem 3.5. By applying the trimming-the-state-space technique of Ibarra & Kim [21] to DP, we will derive the claimed FPTAS for GENE. The main idea in trimming-the-state-space is to clean up and to thin out the state space of the dynamic program DP. If this is done in the right way, the cardinality of the state space shrinks down to polynomial, whereas the loss in precision and the loss in information can be kept under control. The dynamic program that results from trimming DP will be abbreviated by TDP.

We work with three types of state spaces: with $\mathcal{S}_k$, $\mathcal{U}_k$, and $\mathcal{T}_k$. The state space that is produced at the end of the k-th phase in the original dynamic program DP is denoted by $\mathcal{S}_k$. In the trimmed dynamic program TDP, we have two types of state spaces: The iterative computation in the k-th phase of TDP expands an old state space from the $(k-1)$-th phase to a new state space; this new (untrimmed) state space is denoted by $\mathcal{U}_k$. In the second half of the k-th phase in TDP, the new state space $\mathcal{U}_k$ is thinned out and trimmed; this yields the (trimmed) state space $\mathcal{T}_k$. The trimming is based on the so-called *trimming parameter* $\Delta > 1$, where

$$\Delta \doteq 1 + \frac{\varepsilon}{2gn}. \tag{2}$$

Here $\varepsilon > 0$ is the desired precision of approximation, n is the length of the input sequence as in Definition 3.1, and g is the integer constant that exists by Condition C.3(i). Moreover, define an integer L by

$$L \doteq \left\lceil \frac{\pi_1(n, \log \overline{x})}{\ln \Delta} \right\rceil \leq \left\lceil (1 + \frac{2gn}{\varepsilon})\, \pi_1(n, \log \overline{x}) \right\rceil \tag{3}$$

Here $\pi_1(\cdot,\cdot)$ is the polynomial function introduced in Condition C.4(iv); the upper bound on L follows by setting $x := \Delta$ in Proposition 4.1(i). Next, define $L+1$ intervals as follows: $\mathcal{I}_0 = [0]$, $\mathcal{I}_i = [\Delta^{i-1}, \Delta^i)$ for $i = 1, \ldots, L-1$, and $\mathcal{I}_L = [\Delta^{L-1}, \Delta^L]$. Note that every integer in the range $[0, \Delta^L]$ is contained in precisely one of these intervals.

Define a partition of the integer vectors in $[0, \Delta^L]^\beta$ into orthogonal, axes-parallel boxes: For every coordinate ℓ $(1 \leq \ell \leq \beta)$ with $d_\ell \geq 1$, the integer range $[0, \Delta^L]$ is partitioned into the intervals $\mathcal{I}_i$, $i = 0, \ldots, L$. For every coordinate ℓ $(1 \leq \ell \leq \beta)$ with $d_\ell = 0$, the integer range $[0, \Delta^L]$ is partitioned into $\Delta^L + 1$ intervals that each contain a single integer. The boxes in the thereby induced orthogonal partition of $[0, \Delta^L]^\beta$ will be called Δ-*boxes*.

Observation 4.3 *Every state in every state space $\mathcal{S}_k$ in DP is contained in one of the Δ-boxes. If two states $S, S' \in \mathbb{N}^\beta$ are contained in the same Δ-box, then they are $[D, \Delta]$-close to each other.*

Proof. By Condition C.4(iv), the values of all components of all vectors in all state spaces $\mathcal{S}_k$ are integers less or equal to $e^{\pi_1(n, \log \overline{x})}$. By the definition of L in (3), the latter bound in turn is less or equal to Δ^L. This proves the first statement. If two states $S = [s_1, \ldots, s_\beta]$ and $S' = [s'_1, \ldots, s'_\beta]$ are contained in the same Δ-box, then $\Delta^{-1} \cdot s_\ell \leq s'_\ell \leq \Delta \cdot s_\ell$ holds for all coordinates ℓ with $d_\ell \geq 1$, and $s_\ell = s'_\ell$ holds for all coordinates ℓ with $d_\ell = 0$. This implies (1), and thus proves the second statement. ∎

Definition 4.4 *(Trimming the state space in TDP).*
Let $\mathcal{U}, \mathcal{T} \subseteq \mathbb{N}^\beta$ be sets of non-negative integer vectors whose components are bounded by Δ^L. We say that $\mathcal{T}$ is a trimmed copy *of $\mathcal{U}$, if the following holds:*

(i) $\mathcal{T}$ is a subset of $\mathcal{U}$.

(ii) For every Δ-box $\mathcal{B}$ with $\mathcal{U} \cap \mathcal{B} \neq \emptyset$, the set $\mathcal{T}$ contains exactly one state from $\mathcal{U} \cap \mathcal{B}$. This state is a maximum of $\mathcal{U} \cap \mathcal{B}$ with respect to $\preceq_{qua}$; such a maximum state exists by Proposition 2.1(ii).

Figure 2 presents the full description of the trimmed dynamic program TDP; the trimming is performed in Line 7.

```
1  Initialize T_0 := S_0
2  For k = 1 to n do
3      Let U_k := ∅
4      For every T ∈ T_{k-1} and every F ∈ F do
5          If H_F(X_k, T) ≤ 0 then add F(X_k, T) to U_k
6      EndFor
7      Compute a trimmed copy T_k of U_k
8  EndFor
9  Output  min / max  {G(T) : T ∈ T_n}
```

Figure 2: The trimmed dynamic program TDP for the DP-benevolent problem GENE.

Observation 4.5 *In the trimmed dynamic program TDP, the state spaces fulfill the following properties for $k = 1, \ldots, n$.*

(i) $\mathcal{T}_k \subseteq \mathcal{U}_k \subseteq \mathcal{S}_k$.

(ii) For every state U in the untrimmed state space $\mathcal{U}_k$, the trimmed state space $\mathcal{T}_k$ contains a state T that is $[D, \Delta]$-close to U and that fulfills $U \preceq_{qua} T$. ∎

This observation is a straightforward consequence of Definition 4.4 and Observation 4.3. The statement in Observation 4.5(ii) tells us that the trimmed copy $\mathcal{T}_k$ forms a good system of representatives for the state space $\mathcal{U}_k$ with respect to $[D, \Delta]$-closeness.

Lemma 4.6 *The running time of TDP is polynomial in n, in $\log \overline{x}$, and in $1/\varepsilon$.*

Proof. First, let us estimate the cardinality of $\mathcal{T}_k$. By Definition 4.4, $|\mathcal{T}_k|$ equals the number of Δ-boxes that have non-empty intersection with $\mathcal{U}_k$. In coordinates ℓ with $d_\ell \geq 1$, these non-empty Δ-boxes arise from at most $L+1$ distinct intervals in the partition. In coordinates ℓ with $d_\ell = 0$, these non-empty Δ-boxes arise from at most $\pi_2(n, \log \overline{x})$ distinct intervals in the partition; cf. Condition C.4(iv). Hence,

$$|\mathcal{T}_k| \;\leq\; (L + 1 + \pi_2(n, \log \overline{x}))^\beta \;\leq\; \left\lceil (1 + \frac{2gn}{\varepsilon})\, \pi_1(n, \log \overline{x}) + 1 + \pi_2(n, \log \overline{x}) \right\rceil^\beta . \tag{4}$$

Here we used the upper bound on L in (3). Since the values β and g only depend on DP and hence are constants, and since $\pi_1(\cdot,\cdot)$ and $\pi_2(\cdot,\cdot)$ are polynomials, we conclude that for every $k = 1, \ldots, n$, the cardinality of $\mathcal{T}_k$ is polynomially bounded in n, $\log \overline{x}$, and $1/\varepsilon$.

Next, let us discuss how to perform the trimming. We compute for every state in $\mathcal{U}_k$ its Δ-box and thereby get a list of Δ-boxes that have non-empty intersection with $\mathcal{U}_k$. For every Δ-box $\mathcal{B}$ in this list, we compute the relation $\preceq_{qua}$ on $\mathcal{U}_k \cap \mathcal{B}$ and find a maximum element with respect to $\preceq_{qua}$. By Condition C.4(i) all this can be done in polynomial time per Δ-box. Since the list contains at most $|\mathcal{U}_k|$ Δ-boxes, the overall time needed for the trimming is polynomial in n, $\log \overline{x}$, and $1/\varepsilon$.

Finally we observe that the trimmed dynamic program TDP goes through several nested for-loops. In every for-loop, the index either runs through a range of n, a range of $|\mathcal{T}_k|$, or a range of $|\mathcal{F}|$ values. By the above discussion and by Conditions C.4, the index range of every for-loop and the lengths of all encountered numbers are polynomially bounded in n, $\log \overline{x}$, and $1/\varepsilon$. This completes the proof of the lemma. ∎

For a set $\mathcal{S} \subseteq \mathbb{N}^\beta$ and a state $S \in \mathcal{S}$, we say that S is *undominated* in $\mathcal{S}$ if S is maximal in $\mathcal{S}$ with respect to $\preceq_{dom}$; otherwise, S is *dominated* in $\mathcal{S}$.

Lemma 4.7 *Let $S \in \mathcal{S}_k$ be a state that is undominated in $\mathcal{S}_k$, for some $1 \leq k \leq n$. Then there exists a state $S^\# \in \mathcal{S}_{k-1}$ which is undominated in $\mathcal{S}_{k-1}$, and a mapping $F^\# \in \mathcal{F}$ with corresponding mapping $H^\# \in \mathcal{H}$, such that*

$$H^\#(X_k, S^\#) \;\leq\; 0 \qquad \textit{and} \qquad F^\#(X_k, S^\#) \;=\; S. \tag{5}$$

Proof. Let us assume that S enters the state space $\mathcal{S}_k$ when the function $F^\# \in \mathcal{F}$ is applied to X_k and to some state in $\mathcal{S}_{k-1}$ in Line 5 in DP. Let $H^\# \in \mathcal{H}$ be the mapping that corresponds to $F^\#$. Consider the non-empty subset $\mathcal{S}^\diamondsuit$ of $\mathcal{S}_{k-1}$ that contains all states $S^\diamondsuit$ in $\mathcal{S}_{k-1}$ for which

$$H^\#(X_k, S^\diamondsuit) \;\leq\; 0 \qquad \text{and} \qquad F^\#(X_k, S^\diamondsuit) \;=\; S. \tag{6}$$

Let $S^\#$ be a state in $\mathcal{S}^\diamondsuit$ that is undominated in $\mathcal{S}^\diamondsuit$; such a state exists by Proposition 2.1(i). Suppose that $S^\#$ is dominated by some state S' in $\mathcal{S}_{k-1} \setminus \mathcal{S}^\diamondsuit$. Since $S^\# \preceq_{dom} S'$, Condition C.2(ii) together with (6) yields

$$H^\#(X_k, S') \;\leq\; H^\#(X_k, S^\#) \;\leq\; 0. \tag{7}$$

Consequently, $F^\#(X_k, S')$ is in $\mathcal{S}_k$. Since S' dominates $S^\#$, the state $F^\#(X_k, S')$ dominates S by Condition C.1(ii). Since S' is not contained in the set $\mathcal{S}^\diamondsuit$, $F^\#(X_k, S') \neq S$ must hold. Hence, S would be dominated in $\mathcal{S}_k$. This contradiction shows that $S^\#$ is undominated in $\mathcal{S}_{k-1}$, and it also completes the proof. ∎

Lemma 4.8 *For every $k = 0, \ldots, n$ and for every state S that is undominated in $\mathcal{S}_k$, there exists a state $T \in \mathcal{T}_k$ that is $[D, \Delta^k]$-close to S and that fulfills $S \preceq_{qua} T$.*

Proof. The proof is by induction on k. Since $\mathcal{S}_0 = \mathcal{T}_0$ holds, there is nothing to show for $k = 0$. As induction hypothesis we assume that for every state $S^\#$ that is undominated in $\mathcal{S}_{k-1}$, there exists a state $T^\#$ in $\mathcal{T}_{k-1}$ that is $[D, \Delta^{k-1}]$-close to $S^\#$ and fulfills $S^\# \preceq_{qua} T^\#$.

Consider an arbitrary undominated state S in $\mathcal{S}_k$. By Lemma 4.7, there exists a state $S^\#$ that is undominated in $\mathcal{S}_{k-1}$, and there exists a mapping $F^\# \in \mathcal{F}$ with corresponding mapping $H^\# \in \mathcal{H}$, such that (5) is fulfilled. Now by the induction hypothesis, there exists a state $T^\#$ in $\mathcal{T}_{k-1}$ that is $[D, \Delta^{k-1}]$-close to $S^\#$ and that fulfills $S^\# \preceq_{qua} T^\#$. Then Condition C.2(i) and (5) yield that

$$H^\#(X_k, T^\#) \;\leq\; H^\#(X_k, S^\#) \;\leq\; 0. \tag{8}$$

By (8) and by Line 5 of TDP, the state space $\mathcal{U}_k$ contains the state $U = F^\#(X_k, T^\#)$. Since $T^\#$ is $[D, \Delta^{k-1}]$-close to $S^\#$ and since $S^\# \preceq_{qua} T^\#$, Condition C.1(i) implies that either (Case a) U is $[D, \Delta^{k-1}]$-close to S and $S \preceq_{qua} U$, or that (Case b) U dominates S.

In (Case a), Observation 4.5(ii) yields that the state space $\mathcal{T}_k$ contains a state T that is $[D, \Delta]$-close to U and that fulfills $U \preceq_{qua} T$. Since T is $[D, \Delta]$-close to U and since U is $[D, \Delta^{k-1}]$-close to S, Proposition 4.2 yields that T is $[D, \Delta^k]$-close to S. And since $S \preceq_{qua} U$ and $U \preceq_{qua} T$ hold, we have $S \preceq_{qua} T$. This completes the analysis of (Case a).

In (Case b), Observation 4.5(i) implies that U is in $\mathcal{S}_k$. Since S is undominated in $\mathcal{S}_k$, we infer that $S = U$. Then Observation 4.5(ii) yields that the state space $\mathcal{T}_k$ contains a state T that is $[D, \Delta]$-close to $U = S$, and that fulfills $S \preceq_{qua} T$. Then trivially, T is also $[D, \Delta^k]$-close to S. This completes the analysis of (Case b), and it also completes the inductive argument. ∎

Completing the proof of Theorem 3.5 now is routine. Recall from Definition 3.4 that there exists a state $S^* \in \mathcal{S}_n$ such that $G(S^*) = \textsc{Opt}$. By Condition C.3(ii), we may assume without loss of generality that S^* is undominated in $\mathcal{S}_n$. By Lemma 4.8, there exists a state $T^* \in \mathcal{T}_n$ that is $[D, \Delta^n]$-close to S^* and that fulfills $S^* \preceq_{qua} T^*$. Then for minimization problems, we conclude from Condition C.3(i) that

$$G(T^*) \;\leq\; \Delta^{gn} \cdot G(S^*) \;=\; \left(1 + \frac{\varepsilon}{2gn}\right)^{gn} \textsc{Opt} \;\leq\; (1+\varepsilon)\,\textsc{Opt}. \tag{9}$$

Here we have applied Proposition 4.1(ii) with $x := \varepsilon/2$ and $m := gn$ to get the final inequality. Similarly, for maximization problems we conclude from Condition C.3(i) that

$$G(T^*) \;\geq\; \Delta^{-gn} \cdot G(S^*) \;=\; \left(1 + \frac{\varepsilon}{2gn}\right)^{-gn} \textsc{Opt} \;\geq\; \frac{1}{1+\varepsilon}\,\textsc{Opt} \;\geq\; (1-\varepsilon)\,\textsc{Opt}. \tag{10}$$

Summarizing, if GENE is a minimization problem then the objective value of the approximate solution TDP is at most a factor of $1 + \varepsilon$ above the optimal objective value, and if GENE is a maximization problem then the objective value of the approximate solution TDP is at most a factor of $1 - \varepsilon$ below the optimal objective value. This yields the desired approximation guarantee. Moreover, by Lemma 4.6 the running time of TDP is polynomial in the input size and in $1/\varepsilon$. This completes the proof of Theorem 3.5.

We remark that the FPTAS described above only produces an approximation of the optimal objective value, whereas it does not produce the corresponding feasible solution. However, by storing appropriate auxiliary informations in the dynamic program and by performing some backtracking, we can also explicitly compute the corresponding feasible solution. This will increase the running time only by a constant factor. Since these are standard techniques, we do not elaborate on them.

5 Definition of ex-benevolent problems

In this section we consider a subclass of the DP-simple optimization problems that possesses extremely simple DP formulations: A DP-benevolent optimization problem GENE is called *extremely benevolent* (or *ex-benevolent*, for short), if

$H \equiv 0$ for all $H \in \mathcal{H}$,

$\preceq_{dom}$ is the trivial relation on $\mathbb{N}^\beta$,

$\preceq_{qua}$ is the universal relation on $\mathbb{N}^\beta$.

In an ex-benevolent problem, the condition in the 'If-then' statement in Line 5 of the DP in Figure 1 is always fulfilled. Hence, the iterative computation of the state space becomes unconditional: For *every* state S in the old state space $\mathcal{S}_{k-1}$ and for *every* mapping F in $\mathcal{F}$, the dynamic program DP must put the state $F(X_k, S)$ into the new state space $\mathcal{S}_k$. This unconditional computation will prevent us from modeling any data dependent constraints (like hard due dates on the jobs in a scheduling problem) and it will not allow us to get rid of any infeasible states.

The statements in Conditions C.2 and C.3(ii) are by default fulfilled in any ex-benevolent problem GENE. Moreover, Condition C.1 boils down to the following statement.

Condition C.5 *(Condition C.1 in ex-benevolent problems)*
For any $\Delta > 1$, for any $F \in \mathcal{F}$, for any $X \in \mathbb{N}^\alpha$, and for any $S, S' \in \mathbb{N}^\beta$, the following holds: If S is $[D, \Delta]$-close to S', then $F(X, S)$ is $[D, \Delta]$-close to $F(X, S')$.

By the above discussion, a DP-benevolent problem is ex-benevolent iff there exists a degree-vector D such that the dynamic programming formulation DP fulfills the Conditions C.5, C.3(i), and C.4. By Theorem 3.5 every ex-benevolent optimization problem GENE has an FPTAS.

Now suppose that we have found a DP formulation of a DP-simple optimization problem GENE and that we want to know whether GENE is also ex-benevolent. How do we check whether the Conditions C.5, C.3(i), and C.4 are satisfied? Whereas Condition C.4 is rather transparent and usually easy to verify, Conditions C.5 and C.3(i) depend on the degree-vector in a non-trivial way. In the remaining part of this section we will identify a simple special case of these conditions that is based on polynomial mappings and that is straightforward to recognize.

With a polynomial function $f : \mathbb{N}^{\alpha+\beta} \to \mathbb{N}$ in $\alpha + \beta$ variables, we associate the univariate polynomial $f^{(D)} : \mathbb{N} \to \mathbb{N}$ that depends on the degree-vector in the following way:

$$f^{(D)}(\xi) \doteq f(\underbrace{1, 1, \ldots, 1,}_{\alpha-times} \xi^{d_1}, \xi^{d_2}, \ldots, \xi^{d_\beta}). \tag{11}$$

Lemma 5.1

(i) Let $\mathcal{F}$ be a set of mappings of the following form: Every $F \in \mathcal{F}$ is a vector $[f_1, \ldots, f_\beta]$ of polynomial functions $f_\ell : \mathbb{N}^{\alpha+\beta} \to \mathbb{N}$ in $\alpha + \beta$ variables with non-negative coefficients. Moreover for $\ell = 1, \ldots, \beta$, the inequality $\deg(f_\ell^{(D)}) \leq d_\ell$ holds.

Then the set $\mathcal{F}$ fulfills Condition C.5.

(ii) Let $G : \mathbb{N}^\beta \to \mathbb{N}$ be a polynomial function with non-negative coefficients.

Then the function G fulfills Condition C.3(i) for any degree-vector D.

Lemma 5.1 provides us with a simple tool for proving the ex-benevolence of optimization problems. This tool will be applied many times in the illustrating examples in Section 6. Lemma 5.1 will be proved in Appendix A.

6 Examples of ex-benevolent problems

This section discusses five scheduling problems. With the help of Lemma 5.1 we will show that these scheduling problems are all ex-benevolent, and consequently do possess an FPTAS by Theorem 3.5. We will roughly follow the standard scheduling notation (see e.g. Graham, Lawler, Lenstra & Rinnooy Kan [15] and Lawler, Lenstra, Rinnooy Kan & Shmoys [32]). The discussed problems are: minimizing the makespan on two identical machines (Section 6.1), minimizing the sum of the cubed job completion times on two identical machines (Section 6.2), minimizing the total weighted job completion time on two identical machines (Section 6.3), minimizing the total completion time on two identical machines with time dependent processing times (Section 6.4), and minimizing weighted earliness-tardiness about a common non-restrictive due date on a single machine (Section 6.5).

6.1 Makespan on two identical machines

The problem. In the scheduling problem $P2|\,|C_{\max}$, the input consists of n jobs $J_1, \ldots, J_n$ with positive integer processing times $p_1, \ldots, p_n$. All jobs are available for processing at time 0. The goal is to schedule the jobs without preemption on two identical machines such that the largest job completion time is minimized. Karp [23] shows that $P2|\,|C_{\max}$ is $\mathcal{NP}$-hard in the ordinary sense, and Sahni [39] gives an FPTAS for the problem.

The dynamic program. Let $\alpha = 1$ and $\beta = 2$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k]$. A state $S = [s_1, s_2]$ in $\mathcal{S}_k$ encodes a partial schedule for the first k jobs $J_1, \ldots, J_k$: s_1 is the total processing time on the first machine in the partial schedule, and s_2 is the total processing time on the second machine. The set $\mathcal{F}$ consists of two functions F_1 and F_2:

$$\begin{aligned} F_1(p_k, s_1, s_2) &= [s_1 + p_k,\ s_2] \\ F_2(p_k, s_1, s_2) &= [s_1,\ s_2 + p_k] \end{aligned}$$

Intuitively speaking, the function F_1 puts the job J_k on the first machine in every partial schedule $[s_1, s_2] \in \mathcal{S}_{k-1}$ for the jobs $J_1, \ldots, J_{k-1}$, and function F_2 puts J_k on the second machine.

Finally, set

$$G(s_1, s_2) \quad = \quad \max\{s_1, s_2\}.$$

The initial state space $\mathcal{S}_0$ is set to $\{[0,0]\}$. This dynamic programming formulation is due to Horowitz & Sahni [19].

Benevolence. Consider the degree-vector $D = [1,1]$. The functions F_1 and F_2 are vectors of polynomial functions with non-negative coefficients; Lemma 5.1(i) yields that Condition C.5 is satisfied. Condition C.3(i) is satisfied with $g = 1$. The statements (i), (ii) and (iii) in Condition C.4 are straightforward; the statement (iv) is fulfilled, since all components in all states are upper bounded by $\sum_{j=1}^{n} p_j$. Hence, problem $P2|\,|C_{\max}$ is ex-benevolent and Theorem 3.5 yields the following result.

Corollary 6.1 (Sahni [39], 1976)
The scheduling problem $P2|\,|C_{\max}$ has an FPTAS. ∎

The result can easily be carried over to the problems $Pm|\,|C_{\max}$, $Qm|\,|C_{\max}$, and $Rm|\,|C_{\max}$ with any fixed number m of identical, respectively, uniform or unrelated machines.

6.2 Sum of cubed job completion times on two machines

The problem. In the scheduling problem $P2|\,|\sum C_j^3$, the input consists of n jobs $J_1, \ldots, J_n$ with positive integer processing times $p_1, \ldots, p_n$. All jobs are available for processing at time 0. The goal is to schedule the jobs without preemption on two identical machines such that the sum of the cubes of the job completion times is minimized. Lenstra [34] proves that this problem is $\mathcal{NP}$-hard in the ordinary sense.

The dynamic program. We renumber the jobs such that $p_1 \leq p_2 \leq \cdots \leq p_n$ holds. A straightforward job interchange argument shows that there is an optimal schedule in which both machines process the jobs in increasing order of index. Moreover, an optimal schedule will not contain any idle time.

Now let $\alpha = 1$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k]$. A state $S = [s_1, s_2, s_3]$ in $\mathcal{S}_k$ encodes a partial schedule for the first k jobs $J_1, \ldots, J_k$: s_1 (respectively s_2) is the total processing time on the first (respectively second) machine in the partial schedule, and s_3 is the sum of the cubes of the job completion times in the partial schedule. The set $\mathcal{F}$ consists of two functions F_1 and F_2:

$$\begin{aligned} F_1(p_k, s_1, s_2, s_3) &= [s_1 + p_k,\ s_2,\ s_3 + (s_1 + p_k)^3] \\ F_2(p_k, s_1, s_2, s_3) &= [s_1,\ s_2 + p_k,\ s_3 + (s_2 + p_k)^3] \end{aligned}$$

Intuitively speaking, the function F_1 puts the job J_k at the end of the first machine in every partial schedule $[s_1, s_2, s_3] \in \mathcal{S}_{k-1}$ for the jobs $J_1, \ldots, J_{k-1}$. The function F_2 puts J_k at the end of the second machine in a symmetric way. Finally, set $G(s_1, s_2, s_3) = s_3$ to extract the objective value from a partial schedule, and set the initial state space $\mathcal{S}_0$ to $\{[0,0,0]\}$.

Benevolence. Consider the degree-vector $D = [1,1,3]$. The functions F_1 and F_2 are vectors of polynomial functions with non-negative coefficients. Let $F_1 = [f_1, f_2, f_3]$ where f_1, f_2, and f_3 are the induced polynomial functions $\mathbb{N}^4 \to \mathbb{N}$. Then

$$f_1^{(D)}(\xi) = \xi + 1, \qquad f_2^{(D)}(\xi) = \xi, \qquad f_3^{(D)}(\xi) = 2\xi^3 + 3\xi^2 + 3\xi + 1.$$

Hence $\deg(f_1^{(D)}) = 1$, $\deg(f_2^{(D)}) = 1$, and $\deg(f_3^{(D)}) = 3$. In an analogous way it can be shown that function F_2 also fulfills the conditions in Lemma 5.1(i). The function G is a polynomial with non-negative coefficients and thus fulfills the conditions in Lemma 5.1(ii). Condition C.4(iv) is fulfilled, since all components in all states are upper bounded by $n(\sum_{j=1}^{n} p_j)^3$. The statements (i), (ii) and (iii) in Condition C.4 in this case are trivial. Summarizing, Lemma 5.1 and Theorem 3.5 now yield the following result.

Corollary 6.2 *The scheduling problem $P2||\sum C_j^3$ is ex-benevolent and has an FPTAS.* ∎

An analogous result holds true for any problem $Pm||\sum C_j^\lambda$, and $Qm||\sum C_j^\lambda$ with any fixed number m of machines and with any fixed exponent $\lambda \geq 1$.

6.3 Total weighted job completion time on two identical machines

The problem. In the scheduling problem $P2||\sum w_jC_j$, the input consists of n jobs J_j with positive integer processing times p_j and positive integer weights w_j $(j = 1, \ldots, n)$. All jobs are available for processing at time 0. The goal is to schedule the jobs without preemption on two identical machines such that the weighted sum of job completion times is minimized. The problem $P2||\sum w_jC_j$ is $\mathcal{NP}$-hard in the ordinary sense (Bruno, Coffman & Sethi [7] and Lenstra, Rinnooy Kan & Brucker [35]). Sahni [39] constructs an FPTAS for the problem.

The dynamic program. We renumber the jobs such that $p_1/w_1 \leq p_2/w_2 \leq \cdots \leq p_n/w_n$. A job interchange argument shows that there always exists an optimal schedule which does not contain any idle time and in which both machines process the jobs in increasing order of index.

Let $\alpha = 2$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, w_k]$. A state $S = [s_1, s_2, s_3]$ in $\mathcal{S}_k$ encodes a partial schedule without idle time for the first k jobs: s_1 (respectively s_2) is the total processing time on the first (respectively second) machine, and s_3 is the objective value for the partial schedule.

$$\begin{aligned} F_1(p_k, w_k, s_1, s_2, s_3) &= [s_1 + p_k,\ s_2,\ s_3 + w_k(s_1 + p_k)] \\ F_2(p_k, w_k, s_1, s_2, s_3) &= [s_1,\ s_2 + p_k,\ s_3 + w_k(s_2 + p_k)] \end{aligned}$$

Set $\mathcal{F} = \{F_1, F_2\}$, set $G(s_1, s_2, s_3) = s_3$, initialize the state space $\mathcal{S}_0 = \{[0, 0, 0]\}$, and define the degree-vector $D = [1, 1, 1]$. Lemma 5.1 implies that $P2||\sum w_jC_j$ is ex-benevolent, and Theorem 3.5 yields the existence of an FPTAS.

Corollary 6.3 (Sahni [39], 1976)
The scheduling problem $P2||\sum w_jC_j$ has an FPTAS. ∎

For any fixed number m of machines, also the scheduling problems $Pm||\sum w_jC_j$ and $Qm||\sum w_jC_j$ are ex-benevolent and thus fully polynomial time approximable.

6.4 Total completion time on two identical machines with time dependent processing times

The problem. In the scheduling problem $P2|\text{time-dep}|\sum C_j$, the input consists of n jobs $J_1, \ldots, J_n$ and of $m = 2$ identical machines. For every $j = 1, \ldots, n$, a positive integer b_j is

associated with job J_j. All jobs are available for processing at time 1. The actual processing time of a job depends on its starting time: If job J_j is started at time t_j, then its processing time equals $b_j t_j$. The goal is to schedule the jobs without preemption on two identical machines such that the total completion time is minimized. Chen [9] proves that problem $P2|\text{time-dep}|\sum C_j$ is $\mathcal{NP}$-hard in the ordinary sense, and he also poses as an open problem to determine the precise approximability status of $P2|\text{time-dep}|\sum C_j$.

The dynamic program. Assume that in some schedule the total processing time on one of the machines is P, and and that job J_j is scheduled at the end of this machine. Then the total processing time becomes $P(1+b_j)$. We renumber the jobs such that $b_1 \leq b_2 \leq \cdots \leq b_n$. A job interchange argument shows that under this numbering, there always exists an optimal schedule in which on every machine the jobs are processed in increasing order of index.

Now let $\alpha = 1$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [b_k]$. A state $[s_1, s_2, s_3]$ encodes a partial schedule for the first k jobs $J_1, \ldots, J_k$: s_1 is the total processing time on the first machine in the partial schedule, and s_2 is the total processing time on the second machine. s_3 is the objective value of the current schedule. The set $\mathcal{F}$ consists of two functions F_1 and F_2:

$$\begin{aligned} F_1(b_k, s_1, s_2, s_3) &= [s_1(1+b_k),\ s_2,\ s_3 + s_1(1+b_k)] \\ F_2(b_k, s_1, s_2, s_3) &= [s_1,\ s_2(1+b_k),\ s_3 + s_2(1+b_k)] \end{aligned}$$

Hence, function F_1 puts the job J_k at the end of the first machine in every partial schedule for the jobs $J_1, \ldots, J_{k-1}$, and function F_2 puts J_k at the end of the second machine. Finally, set $G(s_1, s_2, s_3) = s_3$. The initial state space $\mathcal{S}_0$ is set to $\{[1, 1, 1]\}$.

Benevolence. The values of all components in all states are upper bounded by $n\prod_{j=1}^{n}(1+b_j) \leq n\overline{x}^n$. Hence, their natural logarithms are bounded by $n\ln\overline{x} + \ln n$, and statement (iv) in Condition C.4 is fulfilled. Lemma 5.1 implies that $P2|\text{time-dep}|\sum C_j$ is ex-benevolent for the degree-vector $D = [1, 1, 1]$, and Theorem 3.5 yields the existence of an FPTAS.

Corollary 6.4 *The scheduling problem $P2|\text{time-dep}|\sum C_j$ has an FPTAS.* ∎

The result can easily be carried over to the corresponding problems with any fixed number m of identical or uniform machines. Moreover, the corresponding problems with the objective of minimizing the total *weighted* job completion time are also ex-benevolent and fully polynomial time approximable.

6.5 Weighted earliness-tardiness about a common non-restrictive due date on a single machine

The problem. In the scheduling problem $1||\sum w_j|C_j|$, the input consists of n jobs J_j with positive integer processing times p_j and positive integer weights w_j $(j = 1, \ldots, n)$. All jobs are available for processing at time $-\infty$. The goal is to schedule the jobs without preemption on a single machine such that the sum of the weighted absolute values of the job completion times C_j is minimized. Intuitively speaking, every job should have its completion time as close as possible to a common due date at time 0. Hall & Posner [16] show that this problem is $\mathcal{NP}$-hard in the ordinary sense and provide an FPTAS for the special case where the total

weight $\sum_{j=1}^{n} w_j$ of the jobs is polynomially bounded in n. Jurisch, Kubiak & Józefowska [22] establish that this special case in fact is solvable in polynomial time. Kovalyov & Kubiak [24] present an FPTAS for the general case without putting any restriction on the total weight of the jobs.

The dynamic program. We renumber the jobs such that $p_1/w_1 \leq p_2/w_2 \leq \cdots \leq p_n/w_n$ holds. Hall & Posner [16] show by a job interchange argument that under this numbering, there always exists an optimal schedule which does not contain any idle time, in which some job is completed at time 0, in which the jobs with non-positive completion times are processed in decreasing order of index, and in which the jobs with positive completion times are processed in increasing order of index.

Now let $\alpha = 2$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, w_k]$. A state $S = [s_1, s_2, s_3]$ in $\mathcal{S}_k$ encodes a partial schedule without idle time for the first k jobs: s_1 is the total processing time scheduled before time 0, s_2 is the total processing time scheduled after time 0, and s_3 is the objective value for the partial schedule.

$$\begin{aligned} F_1(p_k, w_k, s_1, s_2, s_3) &= [s_1 + p_k,\ s_2,\ s_3 + w_k s_1] \\ F_2(p_k, w_k, s_1, s_2, s_3) &= [s_1,\ s_2 + p_k,\ s_3 + w_k(s_2 + p_k)] \end{aligned}$$

Intuitively speaking, the function F_1 puts J_k at the left end of a partial schedule for the jobs $J_1, \ldots, J_{k-1}$, and function F_2 puts J_k at the right end. Set $\mathcal{F} = \{F_1, F_2\}$, set $G(s_1, s_2, s_3) = s_3$, and initialize the state space $\mathcal{S}_0 = \{[0, 0, 0]\}$.

Benevolence. For the degree-vector $D = [1, 1, 1]$, Lemma 5.1 implies Conditions C.5 and C.3(i). Condition C.4(iv) holds since all components in all states are upper bounded by $(\sum_{j=1}^{n} p_j)(\sum_{j=1}^{n} w_j)$. Summarizing, the problem $1||\sum w_j|C_j|$ is ex-benevolent. Now Theorem 3.5 yields another proof for the following result.

Corollary 6.5 (Kovalyov & Kubiak [24], 1997)
The scheduling problem $1||\sum w_j|C_j|$ *has an FPTAS.* ∎

Of course, the result carries over to any fixed number m of machines.

7 Definition of cc-benevolent problems

In this section we discuss another subclass of the DP-simple optimization problems. The definition of this subclass is based on a special quasi-linear order on $\mathbb{N}^\beta$.

From now on the first coordinate s_1 of any vector $S = [s_1, \ldots, s_\beta]$ in $\mathbb{N}^\beta$ will be called its *critical coordinate*. For two vectors $S, S' \in \mathbb{N}^\beta$ we write $S \preceq_{cc} S'$ iff the critical coordinate of S' is less or equal to the critical coordinate of S. Then the relation $\preceq_{cc}$ is a quasi-linear order on $\mathbb{N}^\beta$ which is called the *critical coordinate quasi-linear order*. A DP-benevolent optimization problem GENE is called *DP-benevolent under critical coordinates* (or *cc-benevolent*, for short), if

the relation $\preceq_{qua}$ is the critical coordinate quasi-linear order $\preceq_{cc}$.

By Theorem 3.5 every cc-benevolent optimization problem GENE has an FPTAS. The following Lemma 7.1 will be proved in Appendix A. Lemma 7.1 yields a convenient way for proving the cc-benevolence of DP-simple optimization problems.

Lemma 7.1 *The following statements hold for a DP-simple optimization problem in which the relation $\preceq_{qua}$ is the critical coordinate quasi-linear order $\preceq_{cc}$.*

(i) Assume that the set $\mathcal{F}$ is a set of mappings of the following form: Every $F \in \mathcal{F}$ is a vector $[f_1, \ldots, f_\beta]$ of polynomial functions $f_\ell : \mathbb{N}^{\alpha+\beta} \to \mathbb{N}$ with non-negative coefficients $(\ell = 1, \ldots, \beta)$. For $\ell = 1, \ldots, \beta$, the inequality $\deg(f_\ell^{(D)}) \leq d_\ell$ holds; cf. Lemma 5.1(i). Furthermore, the value of function $f_1(X, S)$ only depends on X, on the critical coordinate of S, and on those coordinates ℓ of S for which the entry d_ℓ in the degree-vector equals 0.

Then Condition C.1(i) on the functions in $\mathcal{F}$ is fulfilled.

(ii) Assume that every function $H \in \mathcal{H}$ is a polynomial $H(X, S)$ that only depends on X, on the critical coordinate of S, and on those coordinates ℓ of S for which the entry d_ℓ in the degree-vector equals 0. Furthermore, every monomial in $H(X, S)$ that does depend on the critical coordinate has a non-negative coefficient.

Then Condition C.2(i) on the functions in $\mathcal{H}$ is fulfilled.

(iii) Let $G : \mathbb{N}^\beta \to \mathbb{N}$ be a polynomial function with non-negative coefficients.

Then the function G fulfills Condition C.3(i) for any degree-vector D.

(iv) Assume that $\preceq_{dom}$ is the trivial relation.

Then the statements in Conditions C.1(ii), C.2(ii), and C.3(ii) are fulfilled.

8 Examples of cc-benevolent problems

In this section, we present six illustrating examples for cc-benevolent optimization problems. The discussed problems are the 0/1-knapsack problem (Section 8.1), the scheduling problem of minimizing the weighted number of tardy jobs on a single machine (Section 8.2), batch scheduling to minimize the weighted number of tardy jobs (Section 8.3), minimizing the makespan of deteriorating jobs on a single machine (Section 8.4), minimizing total late work on a single machine, (Section 8.5), and finally minimizing total *weighted* late work (Section 8.6).

By Theorem 3.5 all these problems do possess an FPTAS. The example in Section 8.3 uses a degree-vector one of whose coordinates is 0. In the examples in Sections 8.5 and 8.6, one of the functions in $\mathcal{F}$ is *not* a polynomial and the dominance relation is *not* the trivial relation.

8.1 The 0/1-knapsack problem

The problem. In the 0/1-knapsack problem the input consists of n pairs of positive integers (p_k, w_k) and a positive integer W. The p_k are called profits, the w_k are called weights, and W is called the weight bound. The goal is to select an index set $K \subseteq \{1, \ldots, n\}$ such that the selected

weight obeys the weight bound $\sum_{k\in K} w_k \leq W$ and such that the selected profit $\sum_{k\in K} p_k$ is maximized. The 0/1-knapsack problem is $\mathcal{NP}$-hard in the ordinary sense (Karp [23]).

The dynamic program. Let $\alpha = 2$ and $\beta = 2$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, w_k]$. A state $S = [s_1, s_2]$ in $\mathcal{S}_k$ encodes a partial solution for the first k indices: The critical coordinate s_1 stands for the total selected weight, and s_2 stands for the total selected profit in the partial solution. The set $\mathcal{F}$ consists of two functions F_1 and F_2.

$$\begin{aligned} F_1(w_k, p_k, s_1, s_2) &= [s_1 + w_k,\ s_2 + p_k] \\ F_2(w_k, p_k, s_1, s_2) &= [s_1,\ s_2] \end{aligned}$$

Intuitively, the function F_1 adds the index k to the partial solution, and function F_2 does not add it. In the set $\mathcal{H}$ there is a function $H_1(w_k, p_k, s_1, s_2) = s_1 + w_k - W$ which corresponds to F_1 and a function $H_2(w_k, p_k, s_1, s_2) \equiv 0$ which corresponds to F_2. With this, the iterative computation in Line 5 of DP for both functions in $\mathcal{F}$ reads

$$\begin{array}{lrl} \text{If} & s_1 + w_k - W \leq 0 & \text{then add } [s_1 + w_k,\ s_2 + p_k] \\ \text{If} & 0 \leq 0 & \text{then add } [s_1,\ s_2]. \end{array}$$

The reader will observe that this is the standard dynamic programming formulation of the 0/1-knapsack (Bellman & Dreyfus [4]). Finally, set $G(s_1, s_2) = s_2$ to extract the total profit from a solution. Define the initial state space $\mathcal{S}_0 = \{[0, 0]\}$.

Benevolence. Use the trivial dominance relation $\preceq_{dom}$ and the degree-vector $D = [1, 1]$. By Lemma 7.1(iii) and (iv), the above dynamic programming formulation fulfills Condition C.3. The functions F_1 and F_2 are vectors of polynomials with non-negative coefficients, and the polynomial functions in F_1 and F_2 that yield the critical coordinates are polynomials which do not depend on the second coordinate s_2. The functions H_1 and H_2 are polynomials; the monomials do not depend on s_2, and the monomial that depends on s_1 has a positive coordinate. With this, Lemma 7.1 yields that the Conditions C.1 and C.2 are satisfied. Verifying the correctness of Conditions C.4 is straightforward. Summarizing, the 0/1-knapsack problem indeed is cc-benevolent. After all, the well-known result of Ibarra & Kim can be given as a corollary to Theorem 3.5.

Corollary 8.1 (Ibarra & Kim [21], 1975)
The 0/1-knapsack problem possesses an FPTAS. ∎

Lawler [30] elaborates on the running time of FPTASs for knapsack problems. He constructs a set of extremely fast algorithms that only perform $O(n \log(1/\varepsilon) + 1/\varepsilon^4)$ operations on integers with $O(\log \overline{x})$ bits.

8.2 Weighted number of tardy jobs on a single machine

The problem. In the scheduling problem $1||\sum w_j U_j$, the input consists of n jobs J_j with positive integer processing times p_j, weights w_j, and due dates d_j $(j = 1, \ldots, n)$. All jobs are available for processing at time 0. In some schedule a job is *early* if its processing is completed by its deadline, and otherwise it is *tardy*. The goal is to schedule the jobs without preemption on a

single machine such that the total weight of the tardy jobs is minimized. Problem $1||\sum w_jU_j$ is $\mathcal{NP}$-hard in the ordinary sense (Karp [23]).

The dynamic program. We renumber the jobs such that $d_1 \leq d_2 \leq \cdots \leq d_n$. Under this numbering, there always exists an optimal schedule in which all early jobs are processed before all tardy jobs and in which all early jobs are processed in increasing order of index.

Now let $\alpha = 3$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, w_k, d_k]$. A state $S = [s_1, s_2, s_3]$ in $\mathcal{S}_k$ encodes a partial schedule for the first k jobs: The critical coordinate s_1 measures the total processing time of the scheduled early jobs, s_2 measures the total weight of the tardy jobs, and s_3 measures the total weight of the early jobs. Define functions F_1 and F_2:

$$\begin{aligned} F_1(p_k, w_k, d_k, s_1, s_2, s_3) &= [s_1 + p_k,\ s_2,\ s_3 + w_k] \\ F_2(p_k, w_k, d_k, s_1, s_2, s_3) &= [s_1,\ s_2 + w_k,\ s_3] \end{aligned}$$

Intuitively, the function F_1 causes job J_k to be early and function F_2 causes job J_k to be tardy. The function $H_1(p_k, w_k, d_k, s_1, s_2, s_3) = s_1 + p_k - d_k$ corresponds to F_1 and the function $H_2(p_k, w_k, d_k, s_1, s_2, s_3) \equiv 0$ corresponds to F_2. Now the iterative computation in Line 5 of DP for both functions in $\mathcal{F}$ reads

$$\begin{array}{lrl} \text{If} & s_1 + p_k - d_k \leq 0 & \text{then add } [s_1 + p_k,\ s_2,\ s_3 + w_k] \\ \text{If} & 0 \leq 0 & \text{then add } [s_1,\ s_2 + w_k,\ s_3]. \end{array}$$

The reader will recognize the Lawler & Moore [33] dynamic programming formulation for $1||\sum w_jU_j$. Finally, set $G(s_1, s_2, s_3) = s_2$ and initialize the state space $\mathcal{S}_0 = \{[0, 0, 0]\}$.

Benevolence. Use the trivial dominance relation $\preceq_{dom}$ and the degree-vector $D = [1, 1, 1]$. By applying Lemma 7.1 in very much the same way as in Section 8.1, one argues that the scheduling problem $1||\sum w_jU_j$ is cc-benevolent. Theorem 3.5 yields the existence of an FPTAS.

Corollary 8.2 (Gens & Levner [14], 1981)
There exists an FPTAS for minimizing the weighted number of tardy jobs in the scheduling problem $1||\sum w_jU_j$. ∎

If we set $G(s_1, s_2, s_3) = s_3$ while leaving everything else unchanged, we get a DP formulation for *maximizing* the total weight of the jobs that are processed *early*.

Corollary 8.3 (Sahni [39], 1976)
There exists an FPTAS for maximizing the weighted number of early jobs in the scheduling problem $1||\sum w_jU_j$. ∎

8.3 Batch scheduling to minimize the weighted number of tardy jobs

The problem. This section discusses a batch scheduling problem $1|\text{batch}|\sum w_jU_j$ that is a simple generalization of the problem $1||\sum w_jU_j$ discussed in Section 8.2: The input consists of n jobs J_j with integer processing times p_j, weights w_j, and due dates d_j $(j = 1, \ldots, n)$, together with a batch setup time b. All jobs are available for processing at time 0. The processing of the jobs is

grouped into *batches*. Each batch starts with a setup time b, which is followed by the sequential processing of all jobs in the batch. The completion time of every job in the batch coincides with the completion time of the batch. The goal is to schedule the jobs on a single machine such that the total weight of the tardy jobs is minimized. The problem $1|\text{batch}|\sum w_jU_j$ is $\mathcal{NP}$-hard in the ordinary sense, since it contains problem $1||\sum w_jU_j$ as a special case. Brucker & Kovalyov [6] design an FPTAS for it.

The dynamic program. We renumber the jobs such that $d_1 \leq d_2 \leq \cdots \leq d_n$. Hochbaum & Landy [18] observe that under this numbering, there always exists an optimal schedule in which all early jobs are processed before all tardy jobs and in which all early jobs are processed in increasing order of index. Hence, there are three possibilities for adding a new job J_k to a partial schedule for the jobs $J_1, \ldots, J_{k-1}$: The new job may be scheduled tardy (F_1); the new job may be added to the currently last batch (F_2); the new job may start a new batch (F_3).

Let $\alpha = 3$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, w_k, d_k]$. A state $S = [s_1, s_2, s_3]$ in $\mathcal{S}_k$ encodes a partial schedule for the first k jobs: The critical coordinate s_1 measures the total processing time and setup time of the scheduled early jobs, s_2 measures the total weight of the tardy jobs, and s_3 stores the due date of the job with lowest index in the latest batch. Let $\mathcal{F}$ contain the functions F_1, F_2, and F_3:

$$\begin{aligned}
F_1(p_k, w_k, d_k, s_1, s_2, s_3) &= [s_1,\ s_2 + w_k,\ s_3] \\
F_2(p_k, w_k, d_k, s_1, s_2, s_3) &= [s_1 + p_k,\ s_2,\ s_3] \\
F_3(p_k, w_k, d_k, s_1, s_2, s_3) &= [s_1 + b + p_k,\ s_2,\ d_k]
\end{aligned}$$

The functions H_1, H_2, and H_3 in $\mathcal{H}$ correspond to F_1, F_2, and F_3, respectively.

$$\begin{aligned}
H_1(p_k, w_k, d_k, s_1, s_2, s_3) &= 0 \\
H_2(p_k, w_k, d_k, s_1, s_2, s_3) &= s_1 + p_k - s_3 \\
H_3(p_k, w_k, d_k, s_1, s_2, s_3) &= s_1 + b + p_k - d_k
\end{aligned}$$

Finally, set $G(s_1, s_2, s_3) = s_2$ and initialize the state space $\mathcal{S}_0 = \{[0, 0, 0]\}$.

Benevolence. Use the trivial dominance relation $\preceq_{dom}$ and the degree-vector $D = [1, 1, 0]$. Note that the third coordinate in the states can only take the n due dates as values; hence, Condition C.4(iv) is fulfilled. Lemma 7.1 then implies that problem $1|\text{batch}|\sum w_jU_j$ indeed is cc-benevolent, and Theorem 3.5 implies the existence of an FPTAS.

Corollary 8.4 (Brucker & Kovalyov [6], 1996)
There exists an FPTAS for maximizing the weighted number of tardy jobs in the scheduling problem $1|\text{batch}|\sum w_jU_j$. ∎

8.4 Makespan of deteriorating jobs on a single machine

The problem. In the scheduling problem $1|\text{deteriorate}|C_{\max}$, the input consists of two non-negative integers d and D, and of n jobs $J_1, \ldots, J_n$ that are specified by positive integers p_j and

w_j, $j = 1, \ldots, n$. All jobs are available for processing at time 0. The actual processing time a_j of job J_j depends on its starting time t_j:

$$a_j = \begin{cases} p_j, & \text{if } t_j \leq d, \\ p_j + w_j(t_j - d), & \text{if } d < t_j < D, \\ p_j + w_j(D - d), & \text{if } D \leq t_j. \end{cases}$$

The goal is to schedule the jobs on a single machine such that the makespan is minimized. Kubiak & van de Velde [28] show that the problem 1|deteriorate|$C_{\max}$ is $\mathcal{NP}$-hard in the ordinary sense, and they provide a pseudo-polynomial algorithm for it. Kovalyov & Kubiak [25] construct an FPTAS for 1|deteriorate|$C_{\max}$.

The dynamic program. A job that is started before or at time d is called *early*, A job that is started after time d but before or at time D is called *tardy*, and a job that is started after time D is called *suspended*. We renumber the jobs such that $p_1/w_1 \leq p_2/w_2 \leq \cdots \leq p_n/w_n$. A job interchange argument (cf. Browne & Yechiali [5]) shows that under this numbering, there always exists an optimal schedule in which all early jobs are sequenced in increasing order of index, all tardy jobs are sequenced in increasing order of index, and all suspended jobs are sequenced in increasing order of index.

Lemma 8.5 (Kubiak & van de Velde [28])
Let $J_1, \ldots, J_\ell$ be the sequence of tardy jobs in some schedule; let C_j denote the completion time of J_j; let $d + \lambda$ with $\lambda \geq 0$ be the total processing time of the early jobs and hence, the starting time of job J_1. Then for $i = 1, \ldots, \ell$,

$$\begin{aligned} C_i = {} & (1 + w_i)(1 + w_{i-1}) \cdots (1 + w_2)(1 + w_1) \cdot \lambda + \\ & (1 + w_i)(1 + w_{i-1}) \cdots (1 + w_2) \cdot p_1 + \\ & \cdots\cdots \\ & (1 + w_i)(1 + w_{i-1}) \cdot p_{i-2} + \\ & (1 + w_i) \cdot p_{i-1} + \\ & p_i + d \end{aligned}$$

Denote by x_i the coefficient of λ in the above equation, and let $y_i = C_i - x_i\lambda - d$. Define $x_0 = 1$ and $y_0 = 0$. Then for $i = 1, \ldots, \ell$,

$$x_i = (1 + w_i)x_{i-1} \qquad \text{and} \qquad y_i = (1 + w_i)y_{i-1} + p_i. \tag{12}$$

Note that with this notation, the last tardy job J_ℓ is completed at time $C_\ell = x_\ell\lambda + y_\ell + d$. ∎

Now let $\alpha = 2$ and $\beta = 4$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, w_k]$. A state $[s_1, s_2, s_3, s_4]$ encodes a partial schedule for the first k jobs $J_1, \ldots, J_k$:

- s_1 is the total processing time of the early jobs.
- s_2 is the product of the values $(1 + w_j)$ where j runs over all tardy jobs; in other words, s_2 is the current x-coefficient in the completion time of the tardy jobs in Lemma 8.5.
- s_3 is the current y-coefficient in the completion time of the tardy jobs in Lemma 8.5.

- s_4 is the total processing time of the suspended jobs.

The set $\mathcal{F}$ consists of three functions: F_1 schedules job J_k early, F_2 schedules job J_k tardy, and F_3 makes J_k a suspended job.

$$\begin{aligned} F_1(p_k, w_k, s_1, s_2, s_3, s_4) &= [s_1 + p_k,\ s_2,\ s_3,\ s_4] \\ F_2(p_k, w_k, s_1, s_2, s_3, s_4) &= [s_1,\ s_2(1+w_k),\ s_3(1+w_k) + p_k,\ s_4] \\ F_3(p_k, w_k, s_1, s_2, s_3, s_4) &= [s_1,\ s_2,\ s_3,\ s_4 + p_k + w_k(D-d)] \end{aligned}$$

These functions are justified by the recursion in (12). Moreover, set $H_1(p_k, w_k, s_1, s_2, s_3, s_4) = s_1 - d$, $H_2 \equiv 0$, and $H_3 \equiv 0$. Finally, initialize $\mathcal{S}_0$ by $\{[0,0,0,0]\}$ and set

$$G(s_1, s_2, s_3, s_4) = s_2 \cdot \max\{0, s_1 - d\} + s_3 + d + s_4.$$

Intuitively, G adds the total processing time s_4 of the suspended jobs to the last completion time of a tardy job.

The reader should observe the following: Function H_1 controls the total processing time of the early jobs; as soon as s_1 exceeds d, no more job can be scheduled early. Function H_2 does not prevent *any* job from being scheduled tardy. Hence, in some states the total processing time of the tardy jobs may be much larger than $D-d$, and the last tardy job may actually be a suspended job. In this case, however, the whole state will be dominated by another state that instead assigns the last tardy job as suspended job.

Benevolence. Consider the degree-vector $D = [1,1,1,1]$, the quasi-linear order $\preceq_{cc}$, and the trivial dominance relation. Then by Lemma 7.1, Conditions C.1, C.3(ii), and C.2 are fulfilled. Condition C.3(i) holds for $g = 1$: If $S = [s_1, s_2, s_3, s_4]$ is $[D, \Delta]$-close to $S' = [s'_1, s'_2, s'_3, s'_4]$ and if $S \preceq_{qua} S'$, then $s'_1 \leq s_1$ and $s'_\ell \leq \Delta s_\ell$ for $\ell = 2, 3, 4$. Hence, $G(S') \leq \Delta G(S)$. Since the technical Condition C.4 is easy to verify, we get that problem $1|\text{deteriorate}|C_{\max}$ is cc-benevolent.

Corollary 8.6 (Kovalyov & Kubiak [25], 1998)
The scheduling problem $1|\text{deteriorate}|C_{\max}$ *has an FPTAS.* ∎

8.5 Total late work on a single machine

The problem. In the scheduling problem $1||\sum V_j$, the input consists of n jobs J_j with positive integer processing times p_j and due dates d_j $(j = 1, \ldots, n)$. All jobs are available for processing at time 0; preemption is not allowed. Denote by C_j the completion time of job J_j in some schedule.

- If $C_j \leq d_j$, then job J_j is *early* and $V_j = 0$.
- If $d_j < C_j < d_j + p_j$, then job J_j is *partially early* and $V_j = C_j - d_j$.
- If $d_j + p_j \leq C_j$, then job J_j is *late* and $V_j = p_j$.

In other words, the *late work* V_j measures the processing of job J_j that is done after the due date d_j. The goal is to schedule the jobs on a single machine such that the total late work is

minimized. Problem $1||\sum V_j$ is $\mathcal{NP}$-hard in the ordinary sense (Potts & Van Wassenhove [37]) and it has an FPTAS (Potts & Van Wassenhove [38]).

The dynamic program. We renumber the jobs such that $d_1 \leq d_2 \leq \cdots \leq d_n$. Potts & Van Wassenhove [37] show that under this numbering, there always exists an optimal schedule in which all early and partially early jobs are processed before all late jobs, and in which the early and partially early jobs are processed in increasing order of index.

Now let $\alpha = 2$ and $\beta = 2$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, d_k]$. As in the preceding sections, a state $S = [s_1, s_2]$ in $\mathcal{S}_k$ encodes a partial schedule for the first k jobs: The critical coordinate s_1 measures the total processing time of the scheduled early and partially early jobs, and s_2 measures the total late work in the schedule. Define functions F_1 and F_2:

$$\begin{aligned} F_1(p_k, d_k, s_1, s_2) &= [s_1 + p_k,\ s_2 + \max\{0, s_1 + p_k - d_k\}] \\ F_2(p_k, d_k, s_1, s_2) &= [s_1,\ s_2 + p_k] \end{aligned}$$

Intuitively speaking, the function F_1 schedules job J_k as early or partially early job. In this case the completion time of J_k will be $C_k = s_1 + p_k$, and the late work V_k will be equal to $\max\{0, C_k - d_k\} = \max\{0, s_1 + p_k - d_k\}$. Function F_2 schedules job J_k as late job with late work $V_k = p_k$. For all functions $H \in \mathcal{H}$, $H \equiv 0$ holds. Finally, set $G(s_1, s_2) = s_2$ and initialize the state space $\mathcal{S}_0 = \{[0, 0]\}$.

Benevolence. Consider the degree-vector $D = [1, 1]$ and the dominance relation

$$[s_1, s_2] \preceq_{dom} [s_1', s_2'] \qquad \Longleftrightarrow \qquad s_1' \leq s_1 \ \text{ and } \ s_2' \leq s_2. \tag{13}$$

The critical coordinate quasi-linear order clearly is an extension of this dominance relation. Condition C.2 is trivially fulfilled. The function G is a polynomial and satisfies Condition C.3(i) with $g = 1$; Condition C.3(ii) is fulfilled since G is non-decreasing in s_1 and in s_2. Condition C.1(ii) is satisfied since every coordinate of the functions F_1 and F_2 is non-decreasing in s_1 and in s_2, and also the technical Conditions C.4 are easy to verify.

It remains to verify the Condition C.1(i). Since function F_1 is *not* a vector of polynomial functions, we *cannot* invoke Lemma 7.1(i). We will give a direct proof instead.

Lemma 8.7 *The described dynamic programming formulation with the degree-vector $D = [1, 1]$ and the dominance relation in (13) satisfies Condition C.1(i) on the functions in $\mathcal{F}$.*

Proof. Consider a real number $\Delta > 1$, a positive integer vector $X = [p, d]$, and two vectors $S = [s_1, s_2]$ and $S' = [s_1', s_2']$ that fulfill $S \preceq_{cc} S'$ and that are $[D, \Delta]$-close to each other. Hence,

$$s_1' \leq s_1 \qquad \text{and} \qquad \frac{1}{\Delta} s_i \ \leq \ s_i' \ \leq \ \Delta s_i \quad \text{for } i = 1, 2. \tag{14}$$

First, let us show that function F_1 satisfies Condition C.1(i). From (14), from $p \geq 0$, and from $\Delta > 1$, we infer that

$$\frac{1}{\Delta}(s_1' + p) \ \leq \ s_1 + p \ \leq \ \Delta(s_1' + p). \tag{15}$$

From $s_1' \leq s_1$ we conclude $\max\{0, s_1' + p - d\} \leq \max\{0, s_1 + p - d\}$, which together with $s_2'/\Delta \leq s_2$ yields that

$$\frac{1}{\Delta}(s_2' + \max\{0, s_1' + p - d\}) \ \leq \ s_2 + \max\{0, s_1 + p - d\}. \tag{16}$$

Now we distinguish two cases. In the first case, we assume that

$$s_2 + \max\{0, s_1 + p - d\} \;\leq\; \Delta(s'_2 + \max\{0, s'_1 + p - d\}). \tag{17}$$

Then (15), (16), and (17) together yield that $F_1(X,S)$ is $[D,\Delta]$-close to $F_1(X,S')$. Moreover, from $s'_1 + p \leq s_1 + p$ we get that $F_1(X,S) \preceq_{cc} F_1(X,S')$. Hence, F_1 satisfies Condition C.1(i) in the first case. In the second case, we assume that (17) does not hold. As an immediate consequence, $s_2 + \max\{0, s_1 + p - d\} > s'_2 + \max\{0, s'_1 + p - d\}$ holds. Together with $s'_1 + p \leq s_1 + p$, this implies that $F_1(X,S)$ dominates $F_1(X,S')$, and that F_1 satisfies Condition C.1(i).

Summarizing, in each of the cases function F_1 satisfies Condition C.1(i). Since F_2 is a vector of polynomials, we get from the proof of Lemma 7.1(i) that F_2 also satisfies Condition C.1(i). ∎

To summarize, problem $1||\sum V_j$ is a cc-benevolent optimization problem, and Theorem 3.5 can be applied to it.

Corollary 8.8 (Potts & Van Wassenhove [38], 1992)
The scheduling problem $1||\sum V_j$ possesses an FPTAS. ∎

8.6 Total weighted late work on a single machine

The problem. This section is closely related to the preceding section; we will mainly discuss the updates and the differences to the approach for the unweighted problem. In the weighted problem $1||\sum w_j V_j$, a job J_j $(j = 1, \ldots, n)$ is specified by its processing time p_j, its due date d_j, and its weight w_j. The goal is to schedule the jobs on a single machine such that the total *weighted* late work is minimized. Problem $1||\sum w_j V_j$ is $\mathcal{NP}$-hard in the ordinary sense, since it contains $1||\sum V_j$ as a special case. Kovalyov, Potts & Van Wassenhove [26] describe an FPTAS for $1||\sum w_j V_j$.

The dynamic program. Renumber the jobs such that $d_1 \leq d_2 \leq \cdots \leq d_n$ holds. The main difference to the unweighted problem is that in an optimal schedule for the weighted problem, the early and partially early jobs need not be scheduled in strict increasing order of index.

Proposition 8.9 (Hariri, Potts & Van Wassenhove [17])
There always exists an optimal schedule of the following form: Let σ denote the sequence of early and partially early jobs. Then for each job J_j in σ there is at most one other job J_ℓ in σ that has a smaller due date than j but is scheduled after J_j in σ (such a job J_ℓ is called a deferred *job).*

Let $\alpha = 3$ and $\beta = 5$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, d_k, w_k]$; moreover, there is a dummy vector $X_{n+1} = [0, 0, 0]$. As above, the critical coordinate s_1 in a state $[s_1, s_2, s_3, s_4, s_5]$ measures the total processing time of the scheduled early and partially early jobs, and s_2 measures the total weighted late work in the partial schedule. The three additional coordinates s_3, s_4 and s_5, respectively, store the processing time, the due date, and the weight of a deferred job.

$$\begin{aligned}
F_1(p_k, d_k, w_k, s_1, s_2, s_3, s_4, s_5) &= [s_1 + p_k,\ s_2 + w_k \max\{0, s_1 + p_k - d_k\},\ s_3,\ s_4,\ s_5] \\
F_2(p_k, d_k, w_k, s_1, s_2, s_3, s_4, s_5) &= [s_1,\ s_2 + p_k,\ s_3,\ s_4,\ s_5] \\
F_3(p_k, d_k, w_k, s_1, s_2, s_3, s_4, s_5) &= [s_1 + s_3,\ s_2 + s_5 \max\{0, s_1 + s_3 - s_4\},\ p_k,\ d_k,\ w_k]
\end{aligned}$$

F_1 schedules J_k immediately as early or partially early job. F_2 schedules J_k immediately as late job. F_3 schedules the deferred job stored in s_3, s_4 and s_5 as early or partially early job, and it stores J_k as the new deferred job. The dummy vector X_{n+1} ensures that the last (deferred or non-deferred) non-late job can be scheduled. For all functions $H \in \mathcal{H}$, $H \equiv 0$ holds. Set

$$G(s_1, s_2, s_3, s_4, s_5) = s_2 + s_3 \cdot \sum_{j=1}^{n} w_j p_j.$$

Observe that the coefficient of s_3 in G equals the objective value of a schedule in which *all* jobs are processed late. Hence, in the cases where s_3 is non-zero, the value of function G becomes prohibitively large. In the remaining cases, s_3 equals zero and the dummy vector X_{n+1} constitutes the final deferred job. Finally, initialize the state space $\mathcal{S}_0 = \{[0, 0, 0, 0, 0]\}$.

Benevolence. Consider the degree-vector $D = [1, 1, 0, 0, 0]$ and the dominance relation

$$[s_1, s_2, s_3, s_4, s_5] \preceq_{dom} [s'_1, s'_2, s'_3, s'_4, s'_5] \quad \Longleftrightarrow$$
$$s'_\ell \leq s_\ell \text{ for } \ell = 1, 2, \text{ and } s'_\ell = s_\ell \text{ for } \ell = 3, 4, 5$$

The critical coordinate quasi-linear order is an extension of this dominance relation. Conditions C.2, C.3, and C.1(ii) are satisfied. Condition C.1(i) can be verified by arguments that are analogous to those in Lemma 8.7. Summarizing, problem $1||\sum w_j V_j$ is cc-benevolent.

Corollary 8.10 (Kovalyov, Potts & Van Wassenhove [26], 1994)
The scheduling problem $1||\sum w_j V_j$ possesses an FPTAS. ∎

9 Discussion of DP-benevolence

In this section we briefly discuss the Conditions C.1–C.4 on DP-benevolence.

Let us start our discussion with the technical Condition C.4. Although the statements in this condition look harmless at first sight (all elementary steps in the dynamic program should be performable in polynomial time), they in fact are very restrictive. There are dynamic programming formulations that use steps like the following:

For $i = 1$ **to** B **do**
    Update every state in the old state space

In case B is encoded in binary, the running time of the dynamic program will be pseudo-polynomial. One example for such a dynamic program is Lawler's dynamic programming formulation [29] for the total tardiness problem $1||\sum T_j$. Another example is the dynamic programming formulation of Florian, Lenstra & Rinnooy Kan [11] for economic lot-sizing.

Dynamic programming formulations of this type are not covered by Theorem 3.5. The number of update steps in every for-loop (= cardinality of $\mathcal{F}$) is exponential in $\log B$ (= $\log \overline{x}$). Hence, Condition C.4(ii) is violated and our main result in Theorem 3.5 cannot be applied. One way to work around this situation would be to modify TDP as follows: We do not *first* expand the old state space $\mathcal{T}_{k-1}$ to $\mathcal{U}_k$ and *then* trim $\mathcal{U}_k$ down to $\mathcal{T}_k$. Instead, we try to compute

$\mathcal{T}_k$ directly from $\mathcal{T}_{k-1}$. We look at the partition into Δ-boxes, and we check for every Δ-box whether some state in $\mathcal{T}_{k-1}$ and some function in $\mathcal{F}$ will lead to a state in this specific Δ-box. If $\mathcal{F}$ is sufficiently nicely structured, this checking can be done in polynomial time for every Δ-box. Since the total number of Δ-boxes is polynomial, this then would yield a polynomial time computation of $\mathcal{T}_k$. However, it seems that the set $\mathcal{F}$ must be highly structured and severely restricted in order to make this modification go through in polynomial time. In all applications that we can think of, the rounding-the-input-data technique yields a much simpler and more direct way to an FPTAS. This is especially true for the two problems mentioned above, the total tardiness problem and the economic lot-sizing problem. Lawler [31] uses his dynamic programming formulation as a basis for an FPTAS via the rounding-the-input-data technique, and Van Hoesel and Wagelmans [40] develop an FPTAS for economic lot-sizing that is also based on the rounding-the-input-data technique.

Next, let us discuss Condition C.3 on the function G. In Sections 9.1 and 9.2, we will exhibit two optimization problems. Both of them are very 'close' to being DP-benevolent: Both of them are DP-simple, and for both of them the dynamic programming formulation fulfills Conditions C.1, C.2, and C.4. For both of them, Condition C.3 is *not* fulfilled (except in the uninteresting case where all entries of the degree-vector are zero); cf. equations (18) and (20) below. The first problem provably does not possess an FPTAS unless $\mathcal{P} = \mathcal{NP}$. The second problem possesses an FPTAS that is based on a more elaborate implementation of the trimming-the-state-space technique. We do not know how to formally capture the difference between these two functions.

Finally, in Section 9.3 we will describe an optimization problem that has a DP-formulation that fulfills the Conditions C.3, C.2, and C.4, but does not fulfill Condition C.1. Again, the problem is very 'close' to being DP-benevolent, but it provably does not possess an FPTAS unless $\mathcal{P} = \mathcal{NP}$.

9.1 A variant of the partition problem that does not have an FPTAS

The problem. In the partition problem, the input consists of n positive integers $p_1, \ldots, p_n$. The standard version of the partition problem asks whether there exists an index set K such that $\sum_{k \in K} p_k = \sum_{k \notin K} p_k$ holds. This problem is $\mathcal{NP}$-hard in the ordinary sense (Karp [23]). We will consider a variant of the partition problem where the goal is to find an index set K that minimizes the objective function $(\sum_{k \in K} p_k - \sum_{k \notin K} p_k)^2$. Since the objective value in this variant equals zero if and only if the standard version has answer YES, the variant is also $\mathcal{NP}$-hard in the ordinary sense. Moreover, the variant cannot possess an FPTAS unless $\mathcal{P} = \mathcal{NP}$, since the FPTAS could be used to decide in polynomial time whether the optimal objective value equals zero.

The dynamic program. We recycle the dynamic programming formulation for problem $P2||C_{\max}$ in Section 6.1: Let $\alpha = 1$ and $\beta = 2$, and define input vectors $X_k = [p_k]$ for $k = 1, \ldots, n$. The set $\mathcal{F}$ consists of two functions $F_1(p_k, s_1, s_2) = [s_1 + p_k, s_2]$ and $F_2(p_k, s_1, s_2) = [s_1, s_2 + p_k]$. The function G is changed to

$$G(s_1, s_2) \;=\; (s_1 - s_2)^2. \tag{18}$$

We set $H_1(p_k, s_1, s_2) \equiv 0$ and $H_2(p_k, s_1, s_2) \equiv 0$. Moreover, the initial state space $\mathcal{S}_0$ is set to $\{[0, 0]\}$.

Discussion. If we choose $\preceq_{dom}$ to be the trivial relation and $\preceq_{qua}$ to be the universal relation, then it is easy to find degree-vectors $D \in \mathbb{N}^2$ for which the described DP formulation fulfills the three Conditions C.1, C.2, and C.4. However, the following lemma (which is proved in Appendix A) shows that not all Conditions C.1–C.4 can be fulfilled simultaneously.

Lemma 9.1 *For any degree-vector D, one of the Conditions C.3(i) and C.4(iv) is not fulfilled.*

Hence, Theorem 3.5 cannot be applied to this DP formulation.

9.2 Completion time variance on a single machine

The problem. In the scheduling problem $1||\text{CTV}$, the input consists of n jobs J_j with positive integer processing times p_j $(j = 1, \ldots, n)$. All jobs are available for processing at time $-\infty$. The goal is to schedule the jobs without preemption on a single machine such that the objective function

$$\sum_{j=1}^{n} C_j^2 - \frac{1}{n}(\sum_{j=1}^{n} C_j)^2 \tag{19}$$

is minimized. The quantity in (19) measures the variance of the job completion times. The problem $1||\text{CTV}$ is $\mathcal{NP}$-hard in the ordinary sense (Kubiak [27]) and possesses an FPTAS (Cai [8] and Woeginger [41]).

The dynamic program. We renumber the jobs such that $p_1 \leq p_2 \leq \cdots \leq p_n$ holds. Eilon & Chowdhury [10] observe that under this numbering, there always exists an optimal schedule which does not contain any idle time, in which some job is completed at time 0, in which the jobs with non-positive completion times are processed in decreasing order of index, and in which the jobs with positive completion times are processed in increasing order of index.

Let $\alpha = 1$ and $\beta = 5$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k]$. A state $S = [s_1, s_2, s_3, s_4, s_5]$ in $\mathcal{S}_k$ encodes a partial schedule where s_1 (respectively s_2) is the total processing time scheduled before time 0 (respectively, after time 0), s_3 (respectively s_4) is the sum of the absolute values of the job completion times before time 0 (respectively, after time 0), and s_5 is the sum of the squared job completion times.

$$\begin{aligned} F_1(p_k, s_1, s_2, s_3, s_4, s_5) &= [s_1 + p_k,\ s_2,\ s_3 + s_1,\ s_4,\ s_5 + (s_1)^2] \\ F_2(p_k, s_1, s_2, s_3, s_4, s_5) &= [s_1,\ s_2 + p_k,\ s_3,\ s_4 + s_2 + p_k,\ s_5 + (s_2 + p_k)^2] \end{aligned}$$

Similarly as in the DP in Section 6.5, the function F_1 puts J_k at the left end of a partial schedule and function F_2 puts J_k at the right end.

$$G(s_1, s_2, s_3, s_4, s_5) = s_5 - \frac{1}{n}(s_4 - s_3)^2. \tag{20}$$

All functions $H \in \mathcal{H}$ are set $H \equiv 0$. The initial state space $\mathcal{S}_0$ is set to $\{[0, 0, 0, 0, 0]\}$.

Discussion. For the degree-vector $D = [1, 1, 1, 1, 2]$, for the trivial relation $\preceq_{dom}$, and for the universal relation $\preceq_{qua}$, the described DP formulation fulfills the Conditions C.1, C.2, and C.4. By similar arguments as in the proof of Lemma 9.1, we see that function G does not satisfy

the Condition C.3(i), and again Theorem 3.5 cannot be applied to the DP formulation. Nevertheless, the problem $1||$CTV does possess an FPTAS. Woeginger [41] constructs an FPTAS by applying the trimming-the-state-space technique *twice*: Once the technique is applied to the dynamic program described above, and once it is applied to the dynamic program whose zero-point is shifted by p_1 time units to the right. It can be shown that one of the two resulting approximations will be within $1+\varepsilon$ of the optimal solution.

9.3 A variant of the knapsack problem that does not have an FPTAS

The problem. In the *2-weighted knapsack* problem, the input consists of n triples of positive integers (p_k, v_k, w_k) and a positive integer W. The p_k are called profits, the v_k and w_k are called weights, and W is called the weight bound. The goal is to select an index set $K \subseteq \{1, \ldots, n\}$ such that the selected weight obeys the weight bound

$$(\sum_{k\in K} v_k)^2 + (\sum_{k\in K} w_k)^2 \;\leq\; W, \tag{21}$$

and such that the selected profit $\sum_{k\in K} p_k$ is maximized. The 2-weighted knapsack problem is $\mathcal{NP}$-hard in the ordinary sense, since it contains the 0/1-knapsack problem in Section 8.1 as a special case. The following lemma is proved in Appendix A.

Lemma 9.2 *Unless $\mathcal{P} = \mathcal{NP}$, the 2-weighted knapsack problem does not have an FPTAS.*

The dynamic program. Let $\alpha = 3$ and $\beta = 3$. For $k = 1, \ldots, n$ define the input vector $X_k = [p_k, v_k, w_k]$. A state $S = [s_1, s_2, s_3]$ in $\mathcal{S}_k$ encodes a partial solution for the first k indices: The coordinate s_1 stands for the total selected v-weight, coordinate s_2 stands for the total selected w-weight, and s_3 stands for the total selected profit in the partial solution. Let the set $\mathcal{F}$ consist of two functions F_1 and F_2.

$$\begin{aligned} F_1(v_k, w_k, p_k, s_1, s_2, s_3) &= [s_1 + v_k,\ s_2 + w_k,\ s_3 + p_k] \\ F_2(v_k, w_k, p_k, s_1, s_2, s_3) &= [s_1,\ s_2,\ s_3] \end{aligned}$$

Similarly as in the DP formulation in Section 8.1, function F_1 adds the index k to the partial solution, whereas function F_2 does not add it. The functions H_1 and H_2 correspond to F_1 and F_2, respectively.

$$\begin{aligned} H_1(v_k, w_k, p_k, s_1, s_2, s_3) &= (s_1 + v_k)^2 + (s_2 + w_k)^2 - W \\ H_2(v_k, w_k, p_k, s_1, s_2, s_3) &= 0. \end{aligned}$$

Finally, define $G(s_1, s_2, s_3) = s_3$ and $\mathcal{S}_0 = \{[0, 0, 0]\}$.

Discussion. Let $D = [1, 1, 1]$ and let $\preceq_{dom}$ be the trivial relation. Consider the following quasi-linear order $\preceq_{qua}$ on $\mathbb{N}^3$: For $S = [s_1, s_2, s_3]$ and $S' = [s'_1, s'_2, s'_3]$, $S \preceq_{qua} S'$ holds iff $(s'_1)^2 + (s'_2)^2 \leq (s_1)^2 + (s_2)^2$. Then the described DP formulation fulfills Conditions C.2, C.3, and C.4. However, it can be shown that it does *not* fulfill Condition C.1(i). Consequently, Theorem 3.5 cannot be applied.

The *2-dimensional knapsack* problem is very similar to the 2-weighted knapsack problem: the input consists of n triples of positive integers (p_k, v_k, w_k) and two positive integers V and

W. The goal is to select an index set $K \subseteq \{1, \ldots, n\}$ such that the selected weight obeys the weight bounds

$$\sum_{k \in K} v_k \ \leq \ V \qquad \text{and} \qquad \sum_{k \in K} w_k \ \leq \ W,$$

and such that the selected profit $\sum_{k \in K} p_k$ is maximized. Similarly as in Lemma 9.2, one can show that the existence of an FPTAS for the 2-dimensional knapsack problem would imply $\mathcal{P} = \mathcal{NP}$. Again, Theorem 3.5 cannot be applied.

10 Conclusion

As main contribution of this paper, we have introduced the concept of DP-benevolence of a combinatorial optimization problem. We have shown that every DP-benevolent optimization problem is fully polynomial time approximable. We have demonstrated the strength and the generality of the concept of DP-benevolence by a sequence of illustrating examples: Many of the fully polynomial time approximation schemes that were developed during the last 25 years just are simple special cases of our main result in Theorem 3.5.

11 Acknowledgements

I acknowledge valuable discussions with Amos Fiat, Han Hoogeveen, Bettina Klinz, Günter Rote, Petra Schuurman, and Leen Stougie. Moreover, I thank Günter Rote for correcting several mistakes in one of the preliminary versions of this paper.

This work has been supported by the START program Y43-MAT of the Austrian Ministry of Science. An extended abstract of this paper has appeared in the *Proceedings of the tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1999.

A Appendix: Proofs of the technical lemmas

The proofs of Lemmas 5.1 and 7.1 are elementary. They are mainly based on simple algebraic manipulations and on the rewriting of expressions.

Proof of Lemma 5.1(i). Let $F : \mathbb{N}^\alpha \times \mathbb{N}^\beta \to \mathbb{N}^\beta$ be a function that fulfills the properties in the statement of the lemma. Hence, F can be represented as a vector $[f_1, \ldots, f_\beta]$ of functions $f_\ell : \mathbb{N}^{\alpha+\beta} \to \mathbb{N}$ with $1 \leq \ell \leq \beta$. Every function f_ℓ in this vector is a polynomial in $\alpha + \beta$ variables that may be written as

$$f_\ell(x_1, \ldots, x_\alpha, y_1, \ldots, y_\beta) \ = \sum_{K=(a_1,\ldots,a_\alpha,b_1,\ldots,b_\beta)} c_{\ell,K} \cdot \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} y_j^{b_j}. \tag{22}$$

The summation is over all tuples K in $\mathbb{N}^{\alpha+\beta}$. All coefficients $c_{\ell,K}$ are non-negative and only a finite number of them is positive. Since $\deg(f_\ell^{(D)}) \leq d_\ell$, every tuple $K = (a_1, \ldots, a_\alpha, b_1, \ldots, b_\beta)$ in the summation with positive coefficient $c_{\ell,K}$ fulfills

$$\sum_{j=1}^{\beta} d_j b_j \;\leq\; d_\ell. \tag{23}$$

Let $\Delta > 1$ and let $X \in \mathbb{N}^\alpha$ with $X = (x_1, \ldots, x_\alpha)$. Let $S, S' \in \mathbb{N}^\beta$ be two vectors with $S = [s_1, \ldots, s_\beta]$ and $S' = [s'_1, \ldots, s'_\beta]$ that are $[D, \Delta]$-close to each other. Consider the ℓ-th coordinate, $1 \leq \ell \leq \beta$, of $F(X, S)$ and of $F(X, S')$. Consider some fixed monomial specified by a tuple $K = (a_1, \ldots, a_\alpha, b_1, \ldots, b_\beta)$ with positive coefficient $c_{\ell,K}$ in this coordinate. Then

$$c_{\ell,K} \cdot \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} (\Delta^{-d_j} \cdot s_j)^{b_j} \;\leq\; c_{\ell,K} \cdot \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} (s'_j)^{b_j} \;\leq\; c_{\ell,K} \cdot \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} (\Delta^{d_j} \cdot s_j)^{b_j}$$

where we applied the definition of $[D, \Delta]$-closeness in (1). Rewriting yields

$$c_{\ell,K} \cdot (\Delta^{-\sum_j d_j b_j}) \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} s_j^{b_j} \leq c_{\ell,K} \cdot \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} (s'_j)^{b_j} \leq c_{\ell,K} \cdot (\Delta^{\sum_j d_j b_j}) \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} s_j^{b_j},$$

and applying (23) yields

$$c_{\ell,K} \cdot (\Delta^{-d_\ell}) \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} s_j^{b_j} \;\leq\; c_{\ell,K} \cdot \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} (s'_j)^{b_j} \;\leq\; c_{\ell,K} \cdot (\Delta^{d_\ell}) \prod_{i=1}^{\alpha} x_i^{a_i} \prod_{j=1}^{\beta} s_j^{b_j}. \tag{24}$$

Adding up (24) over all tuples K gives

$$\Delta^{-d_\ell} \cdot f_\ell(X, S) \;\leq\; f_\ell(X, S') \;\leq\; \Delta^{d_\ell} \cdot f_\ell(X, S) \tag{25}$$

for every ℓ with $1 \leq \ell \leq \beta$. In other words, $F(X, S)$ is indeed $[D, \Delta]$-close to $F(X, S')$ and Condition C.5 is fulfilled. This completes the proof of Lemma 5.1(i). ∎

Proof of Lemma 5.1(ii). Let $G : \mathbb{N}^\beta \to \mathbb{N}$ be a polynomial function with non-negative coefficients. Hence, G may be written as

$$G(y_1, \ldots, y_\beta) \;=\; \sum_{K=(b_1,\ldots,b_\beta)} c_{G,K} \cdot \prod_{j=1}^{\beta} y_j^{b_j}. \tag{26}$$

All coefficients $c_{G,K}$ are non-negative and only a finite number of them is positive. Define

$$g \;\doteq\; \max \left\{ \sum_{j=1}^{\beta} d_j b_j : \; K = (b_1, \ldots, b_\beta) \text{ with } c_{G,K} > 0 \right\}. \tag{27}$$

Note that the value of g only depends on function G and on the degree-vector D.

Let $\Delta > 1$ and let $S = [s_1, \ldots, s_\beta]$ and $S' = [s'_1, \ldots, s'_\beta]$ be two vectors in $\mathbb{N}^\beta$ that are $[D, \Delta]$-close to each other. Consider some fixed monomial specified by a tuple $K = (b_1, \ldots, b_\beta)$ in (26). Then

$$c_{G,K} \cdot \prod_{j=1}^{\beta} (\Delta^{-d_j} \cdot s_j)^{b_j} \;\leq\; c_{G,K} \cdot \prod_{j=1}^{\beta} (s'_j)^{b_j} \;\leq\; c_{G,K} \cdot \prod_{j=1}^{\beta} (\Delta^{d_j} \cdot s_j)^{b_j}$$

Rewriting and using the definition of g yields

$$c_{G,K} \cdot (\Delta^{-g}) \prod_{j=1}^{\beta} s_j^{b_j} \;\leq\; c_{G,K} \cdot \prod_{j=1}^{\beta} (s'_j)^{b_j} \;\leq\; c_{G,K} \cdot (\Delta^{g}) \prod_{j=1}^{\beta} s_j^{b_j} \tag{28}$$

By adding up (28) over all tuples K, we get that Condition C.3(i) holds for G. This completes the proof of Lemma 5.1(ii). ∎

Proof of Lemma 7.1. Consider statement (i). Let $F \in \mathcal{F}$ be a vector $[f_1, \ldots, f_\beta]$ of polynomial functions $\mathbb{N}^{\alpha+\beta} \to \mathbb{N}$ with non-negative coefficients. Moreover, let the value of function $f_1(X, S)$ only depend on X, on the critical coordinate of S, and on those coordinates ℓ of S for which the entry d_ℓ in the degree-vector equals 0. Let $X \in \mathbb{N}^\alpha$. Let $S, S' \in \mathbb{N}^\beta$ be two $[D, \Delta]$-close vectors with $S \preceq_{cc} S'$. Then by Lemma 5.1(i), $F(X, S)$ is $[D, \Delta]$-close to $F(X, S')$. Since with the exception of the critical coordinate, the $[D, \Delta]$-close vectors $[X, S]$ and $[X, S']$ agree in all coordinates that are relevant for the evaluation of f_1, $S \preceq_{cc} S'$ implies $f_1(X, S') \leq f_1(X, S)$. Consequently, $F(X, S) \preceq_{cc} F(X, S')$ holds and the function F fulfills Condition C.1(i). This proves statement (i).

The proof of statement (ii) is similar to the proof of statement (i): Observe that the values of $H(X, S)$ and $H(X, S')$ only depend on coordinates in which $[X, S]$ and $[X, S']$ do agree and on the critical coordinate. Furthermore, $H(X, S)$ is non-decreasing in the critical coordinate. Statement (iii) is identical to statement (ii) in Lemma 5.1, and the proof of statement (iv) is straightforward. This completes the proof of Lemma 7.1. ∎

Proof of Lemma 9.1. Consider a degree-vector $D = [d_1, d_2] \in \mathbb{N}^2$ that fulfills Condition C.4(iv). If the cardinalities of $\mathcal{V}_1(I)$ and of $\mathcal{V}_2(I)$ are *not* polynomially bounded in n and $\log \overline{x}$, the statement in Condition C.4(iv) implies $d_1 \geq 1$ and $d_2 \geq 1$ for the entries in D.

Now consider an arbitrary $\Delta > 1$. Let z_1 and z_2 be positive integers with $z_1 \neq z_2$ and $\frac{1}{\Delta} z_1 < z_2 < \Delta z_1$. Then the vectors $S = [z_1, z_1]$ and $S' = [z_1, z_2]$ are $[D, \Delta]$-close to each other. Since $G(S) = 0$ and $G(S') > 0$, Condition C.3(i) is violated. ∎

Proof of Lemma 9.2. We use the following $\mathcal{NP}$-hard variant of the partition problem (cf. Garey & Johnson [13]): The input consists of $2m$ positive integers $a_1, \ldots, a_{2m}$ that sum up to $2A$ and that fulfill $A/(m+1) < a_k < A/(m-1)$ for $k = 1, \ldots, 2m$. The problem is to decide whether there exists an index set K such that $\sum_{k \in K} a_k = A$ holds.

Consider the following instance of 2-weighted knapsack with $n = 2m$ items: For $k = 1, \ldots, 2m$, item k has profit $p_k = 1$, weight $v_k = (m+1)A - ma_k$ and weight $w_k = (m-1)A + ma_k$. The weight bound is $W = 2m^4 A^2$. Now suppose that the two-dimensional knapsack problem does possess an FPTAS. Set $\varepsilon = \frac{1}{2m}$ and call the FPTAS for the constructed instance.

First assume that the FPTAS returns a solution K with profit at least m; hence, $|K| \geq m$ holds. Denote $Z = \sum_{k \in K} a_k$. Then the restriction imposed by the weight bound becomes

$$[(m+1)A|K| - mZ]^2 + [(m-1)A|K| + mZ]^2 \;\leq\; 2m^4A^2. \tag{29}$$

The left-hand side of (29) is a convex function in Z that takes its unique minimum at $Z = \frac{1}{m}A|K|$. The corresponding minimum value is $2m^2|K|^2A^2$. Since $|K| \geq m$, the inequality in (29) yields $|K| = m$ and $Z = A$. Consequently, the index set K constitutes a solution to the partition problem.

Next assume that the partition problem possesses a solution K with $\sum_{k \in K} a_k = A$. Since $A/(m+1) < a_k < A/(m-1)$ holds for all k, this yields $|K| = m$. Then the index set K constitutes a feasible solution to the 2-weighted knapsack problem with total profit $|K| \geq m$. By the choice of ε, the approximate objective value that is computed by the FPTAS must be at least $(1-\varepsilon)m > m-1$. Since the objective function only takes integer values, the approximate objective value must be at least m.

Summarizing, the FPTAS would find in polynomial time a solution with profit at least m for the 2-weighted knapsack problem if and only if the partition problem has a solution. ∎

References

[1] G. Ausiello, P. Crescenzi, and M. Protasi, 1995. Approximate solution of NP optimization problems, *Theoretical Computer Science 150*, 1–55.

[2] G. Ausiello, A. D'Atri, and M. Protasi, 1980. Structure preserving reductions among convex optimization problems, *Journal of Computer and Systems Sciences 21*, 136–153.

[3] G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi, 1980. Toward a unified approach for the classification of NP-complete optimization problem, *Theoretical Computer Science 12*, 83–96.

[4] R.E. Bellman and S.E. Dreyfus, 1962. *Applied Dynamic Programming*, Princeton University Press.

[5] S. Browne and U. Yechiali, 1990. Scheduling deteriorating jobs on a single processor, *Operations Research 38*, 432–450.

[6] P. Brucker and M.Y. Kovalyov, 1996. Single machine batch scheduling to minimize the weighted number of late jobs, *ZOR – Mathematical Methods of Operations Research 43*, 1–8.

[7] J.L. Bruno, E.G. Coffman, Jr., and R. Sethi, 1974. Scheduling independent tasks to reduce mean finishing time, *Communications of the ACM 17*, 382–387.

[8] X. Cai, 1995. Minimization of agreeably weighted variance in single machine systems, *European Journal of Operational Research 85*, 576–592.

[9] Z.-L. Chen, 1996. Parallel machine scheduling with time dependent processing times, *Discrete Applied Mathematics 70*, 81–9.

[10] S. Eilon and I.G. Chowdhury, 1977. Minimizing waiting variance in the single machine problem, *Management Science 23*, 567–575.

[11] M. Florian, J.K. Lenstra, and A.H.G. Rinnooy Kan, 1980. Deterministic production planning: algorithms and complexity, *Management Science 26*, 669–679.

[12] M.R. Garey and D.S. Johnson, 1978. 'Strong' NP-completeness results: motivation, examples, and implications, *Journal of the ACM 25*, 499–508.

[13] M.R. Garey and D.S. Johnson, 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

[14] G.V. Gens and E.V. Levner, 1981. Fast approximation algorithms for job sequencing with deadlines, *Discrete Applied Mathematics 3*, 313–318.

[15] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics 5*, 287–326.

[16] N.G. Hall and M.E. Posner, 1991. Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date, *Operations Research 39*, 836–846.

[17] A.M.A. Hariri, C.N. Potts, and L.N. Van Wassenhove, 1995. Single machine scheduling to minimize total weighted late work, *ORSA Journal on Computing 7*, 232–242.

[18] D.S. Hochbaum and D. Landy, 1994. Scheduling with batching: minimizing the weighted number of tardy jobs, *Operations Research Letters 16*, 79–86.

[19] E. Horowitz and S. Sahni, 1974. Computing partitions with applications to the knapsack problem, *Journal of the ACM 21*, 277–292.

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

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

[22] B. Jurisch, W. Kubiak, and J. Józefowska, 1997. Algorithms for minclique scheduling problems, *Discrete Applied Mathematics 72*, 115–139.

[23] R.M. Karp, Reducibility among combinatorial problems, 1972. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, New York, 85–104.

[24] M.Y. Kovalyov and W. Kubiak, 1998. A fully polynomial time approximation scheme for the weighted earliness-tardiness problem, to appear in *Operations Research*.

[25] M.Y. Kovalyov and W. Kubiak, 1998. A fully polynomial time approximation scheme for minimizing makespan of deteriorating jobs, *Journal of Heuristics 3*, 287–297.

[26] M.Y. Kovalyov, C.N. Potts, and L.N. Van Wassenhove, 1994. A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work, *Mathematics of Operations Research 19*, 86–93.

[27] W. Kubiak, 1993. Completion time variance on a single machine is difficult, *Operations Research Letters 14*, 49–59.

[28] W. Kubiak and S.L. van de Velde, 1998. Scheduling deteriorating jobs to minimize makespan, to appear in *Naval Research Logistics*.

[29] E.L. Lawler, 1997. A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness, *Annals of Discrete Mathematics 1*, 331–342.

[30] E.L. Lawler, 1979. Fast approximation schemes for knapsack problems, *Mathematics of Operations Research 4*, 339–356.

[31] E.L. Lawler, 1982. A fully polynomial approximation scheme for the total tardiness problem, *Operations Research Letters 1*, 207–208.

[32] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, 1993. Sequencing and scheduling: Algorithms and complexity, in: S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.) *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 445–522.

[33] E.L. Lawler and J.M. Moore, 1969. A functional equation and its application to resource allocation and sequencing problems, *Management Science 16*, 77–84.

[34] J.K. Lenstra, 1977. Unpublished manuscript.

[35] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker, 1977. Complexity of machine scheduling problems, *Annals of Discrete Mathematics 1*, 343–362.

[36] A. Paz and S. Moran, 1981. Non deterministic polynomial optimization problems and their approximations, *Theoretical Computer Science 15*, 251–277.

[37] C.N. Potts and L.N. Van Wassenhove, 1992. Single machine scheduling to minimize total late work, *Operations Research 40*, 586–595.

[38] C.N. Potts and L.N. Van Wassenhove, 1992. Approximation algorithms for scheduling a single machine to minimize total late work, *Operations Research Letters 11*, 261–266.

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

[40] C.P.M. van Hoesel and A.P.M. Wagelmans, 1997. Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems, Economic Institute Report 9735/A, Erasmus University Rotterdam.

[41] G.J. Woeginger, 1998. An approximation scheme for minimizing agreeably weighted variance on a single machine, Technical Report Woe-21, TU Graz, Austria.