Ann Oper Res (2016) 240:39–94
DOI 10.1007/s10479-015-2018-y

Optimizing the half-product and related quadratic Boolean functions: approximation and scheduling applications

Hans Kellerer[1] · **Vitaly A. Strusevich**[2]

Published online: 22 September 2015

Abstract This paper reviews the problems of Boolean non-linear programming related to the half-product problem. All problems under consideration have a similar quadratic non-separable objective function. For these problems, we focus on the development of fully polynomial-time approximation schemes, especially of those with strongly polynomial time, and on their applications to various scheduling problems.

Keywords Quadratic knapsack · Half-product · Single machine scheduling · FPTAS

1 Introduction

This paper provides a review of results on Boolean programming problems of optimizing a particular objective function, known as the *half-product*. The function is a quadratic non-separable function of Boolean variables, and we consider the problems of its minimization and maximization either with no additional constraints or under a linear constraint of a knapsack type. The problems of this range serve as mathematical models for numerous scheduling problems. We pay special attention to developing fast approximation schemes and algorithms for the problems related to optimizing the half-product, as well as for the relevant scheduling applications.

This is an updated version of the paper that appeared in *4OR*, 12, 111–161 (2012).

✉ Vitaly A. Strusevich
v.strusevich@gre.ac.uk; V.Strusevich@greenwich.ac.uk

Hans Kellerer
hans.kellerer@uni-graz.at

1 Institut für Statistik und Operations Research, Universität Graz, Universitätsstraße 15, 8010 Graz, Austria

2 Department of Mathematical Sciences, University of Greenwich, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, UK

This review is a modified version of the survey Kellerer and Strusevich (2012). The current version exhibits a shift of focus of the presentation. The earlier survey was centered around the symmetric quadratic knapsack problem, which consists in minimizing a special form of the half-product function under a linear knapsack constraint. Our interest in that specific model was justified by the fact that it would serve as a universal model of most scheduling applications. In this paper, the stress is on the half-product problem in its pure form, and on its variants of practical importance such as the positive half-product and the symmetric quadratic knapsack.

Our main goal is to review algorithmic ideas that help to develop approximation schemes for these Boolean quadratic problems, and to survey all known scheduling applications. New topics addressed in this paper compared to Kellerer and Strusevich (2012) include fast approximation schemes for minimizing the positive half-product and its scheduling applications, development of differential approximation schemes and approximability for the maximization counterparts of the problems under consideration.

The remainder of this paper is organized as follows. Section 2 gives formal descriptions of all versions of problems related to the half-product optimization, in various forms: Boolean, matrix and in terms of set-functions. In Sect. 3 we formulate a number of scheduling problems and for each problem present its reformulation in terms of a half-product related problem. Sections 4, 5 and 6 address the problems of minimizing the half-product, the positive half-product and the symmetric quadratic knapsack problem, respectively. For each of this models, we describe the principles of design of approximation schemes and discuss their adaptations to the relevant scheduling problems. Section 7 reviews the whole range of problems from the point of view of differential approximation. The maximization versions are discussed in Sect. 8. The concluding remarks and open questions can be found in Sect. 9.

2 Formulation of half-product and related problems

In this section, we present formulations of the half-product problem and its versions. In this section, and in fact in most of this paper, we focus on problems of minimization of the relevant functions. Their maximization counterparts are discussed in Sect. 8.

2.1 Boolean programming formulations

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector with n Boolean components. Consider the function

$$H(\mathbf{x}) = \sum_{1 \le i < j \le n}^{n} \alpha_i \beta_j x_i x_j - \sum_{j=1}^{n} \gamma_j x_j, \tag{1}$$

where for each j, $1 \le j \le n$, the coefficients α_j and β_j are non-negative integers, while γ_j is an integer that can be either negative or positive. Problems of minimizing quadratic functions similar to (1) were introduced in 1990s as mathematical models for various scheduling problems by Kubiak (1995) and Jurisch et al. (1997). Function (1) and the term "half-product" were introduced by Badics and Boros (1998), who considered the problem of minimizing the function $H(\mathbf{x})$ with respect to Boolean decision variables with no additional constraints. The function $H(\mathbf{x})$ is called a *half-product* since its quadratic part consists of roughly half of the terms of the product $\left(\sum_{j=1}^{n} \alpha_j x_j\right)\left(\sum_{j=1}^{n} \beta_j x_j\right)$. Notice that we only are interested in the instances of the problem for which the optimal value of the function is strictly negative; otherwise, setting all decision variables to zero solves the problem.

In this paper, we refer to the problem of minimizing function $H(\mathbf{x})$ of the form (1), as *Problem HP*. This problem is NP-hard in the ordinary sense, even if $\alpha_j = \beta_j$ for all

$j = 1, 2, \dots, n$, as proved by Badics and Boros (1998). It has numerous applications, mainly to machine scheduling; see Erel and Ghosh (2008) and Kellerer and Strusevich (2012) for reviews. Notice that in those applications a scheduling objective function usually is written in the form

$$F(\mathbf{x}) = H(\mathbf{x}) + K, \tag{2}$$

where K is a given additive constant. We refer to the problem of minimizing function $F(\mathbf{x})$ of the form (2), as *Problem HPAdd*.

Consider the function

$$P(\mathbf{x}) = \sum_{1 \le i < j \le n}^{n} \alpha_i \beta_j x_i x_j + \sum_{j=1}^{n} \mu_j x_j + \sum_{j=1}^{n} \nu_j \left(1 - x_j\right) + K, \tag{3}$$

where all coefficients $\alpha_j,\ \beta_j,\ \mu_j,\ \nu_j$ and K are non-negative integers. Following Janiak et al. (2005), we call the problem of minimizing the function $P(\mathbf{x})$ of the form (3) the *Positive Half-Product Problem* or *Problem PosHP*.

In the two problems introduced above, the minimum is sought for over all $n-$dimensional Boolean vectors, i.e., they are quadratic Boolean programming problems with no additional constraints. In this paper, we review a restricted version of these problems, in which an additional knapsack constraint is introduced. In particular, the knapsack constrained variant of Problem PosHP can be written as

$$\begin{aligned} &\text{Minimize } \ P(\mathbf{x}) = \sum_{1 \le i < j \le n}^{n} \alpha_i \beta_j x_i x_j + \sum_{j=1}^{n} \mu_j x_j + \sum_{j=1}^{n} \nu_j \left(1 - x_j\right) + K \\ &\text{Subject to } \sum_{j=1}^{n} \alpha_j x_j \le A \\ &\qquad\qquad\quad x_j \in \{0, 1\},\ \ j = 1, 2, \dots, n, \end{aligned} \tag{4}$$

which we call the *Positive Half-Product Knapsack Problem* and denote *Problem PosHPK*.

Similarly to the classical Linear Knapsack Problem (see the comprehensive monographs Martello and Toth (1990) and Kellerer et al. (2004) on this most studied problem of Combinatorial Optimization), Problem PosHPK contains a linear *knapsack* constraint

$$\sum_{j=1}^{n} \alpha_j x_j \le A. \tag{5}$$

We can view the value α_j as the weight of item j, $1 \le j \le n$, i.e., $x_j = 1$ means that item j is placed into a knapsack with capacity A, while $x_j = 0$ means that the corresponding item is not placed into the knapsack. An important feature is that the coefficients α_j in the knapsack constraint are the same as in the quadratic terms of the objective function. The latter feature makes Problem PosHPK to be a special case of another quadratic knapsack problem, namely the problem

$$\begin{aligned} &\text{Minimize } \ Z(\mathbf{x}) = \sum_{1 \le i < j \le n} \alpha_i \beta_j x_i x_j + \sum_{1 \le i < j \le n} \alpha_i \beta_j (1 - x_i)(1 - x_j) \\ &\qquad\qquad + \sum_{j=1}^{n} \mu_j x_j + \sum_{j=1}^{n} \nu_j (1 - x_j) + K \\ &\text{Subject to } \sum_{j=1}^{n} \alpha_j x_j \le A \\ &\qquad\qquad\quad x_j \in \{0, 1\},\ \ j = 1, 2, \dots, n. \end{aligned} \tag{6}$$

Following Kellerer and Strusevich (2010a, b), we call the latter problem the *Symmetric Quadratic Knapsack Problem*, or *Problem SQK* . We use the term "*symmetric*" because both the quadratic and the linear parts of the objective function are separated into two terms, one depending on the variables x_j, and the other depending on the variables $(1-x_j)$. Problem SQK is no easier than the Linear Knapsack Problem and therefore is at least NP-hard in the ordinary sense.

Clearly, Problem SQK is a generalization of Problem PosHPK, since its objective contain an additional quadratic term. The objective function $Z(\mathbf{x})$ can be rewritten as

$$Z(\mathbf{x}) = 2\sum_{1\le i<j\le n}\alpha_i\beta_j x_i x_j + \sum_{i=1}^{n}\left(\mu_i - \alpha_i\sum_{j=i+1}^{n}\beta_j - \beta_i\left(\sum_{j=1}^{i-1}\alpha_j\right) - \nu_i\right)x_i + K + \sum_{i=1}^{n}\left(\alpha_i\sum_{j=i+1}^{n}\beta_j + \nu_i\right),$$

i.e., in a form close to that used in the formulation of Problem HPAdd.

The half-product function and its variants above are special cases of the general quadratic function of Boolean variables. Let $(q_{ij})_{n\times n}$ be a symmetric quadratic matrix. For a Boolean vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ define the function

$$Q(\mathbf{x}) = \sum_{1\le i<j\le n} q_{ij}x_i x_j - \sum_{j=1}^{n}\gamma_j x_j. \tag{7}$$

We refer to a Boolean programming of optimizing function (7) subject to a linear constraint (5) as the *Quadratic Knapsack Problem* (*Problem QK*). In general, Problem QK is NP-hard in the strong sense. See Chapter 12 of the book by Kellerer et al. (2004) and a survey by Pisinger (2007) for an overview of principal results on Problem QK.

Table 1 summarizes the notation introduced above for all Boolean programming problems under consideration.

2.2 Matrix formulations and convexity

Sometimes it is convenient to reformulate the introduced functions in an alternative way, e.g., in the matrix form. Below we illustrate this for Problem SQK. Let $q(\mathbf{x})$ be the quadratic term of the objective function $Z(\mathbf{x})$ in (6). We can rewrite $q(\mathbf{x})$ in the form

$$q(\mathbf{x}) = 2\sum_{i=1}^{n}\alpha_i x_i\sum_{j=i}^{n}\beta_j x_j - \sum_{i=1}^{n}\left(\alpha_i\sum_{j=i}^{n}\beta_j + \beta_i\left(\sum_{j=1}^{i}\alpha_j\right)\right)x_i + \sum_{i=1}^{n}\left(\alpha_i\sum_{j=i+1}^{n}\beta_j\right).$$

Table 1 Notation for Boolean programming problems under consideration

Notation	Objective/formulation	Additional constraints
HP	$H(\mathbf{x})$ (1)	None
HPAdd	$F(\mathbf{x})$ (2)	None
PosHP	$P(\mathbf{x})$ (3)	None
PosHPK	$P(\mathbf{x})$ (4)	(5)
SQK	$Z(\mathbf{x})$ (6)	(5)
QK	$Q(\mathbf{x})$ (7)	(5)

See Appendix A of Kellerer and Strusevich (2010b) for a detailed proof that uses the obvious fact that

$$x_j = x_j^2 \tag{8}$$

for a Boolean variable x_j. This allows us to rewrite the objective function of (6) as

$$Z(\mathbf{x}) = 2\sum_{i=1}^{n} \alpha_i x_i \sum_{j=i}^{n} \beta_j x_j + \sum_{i=1}^{n} \left(\mu_i - \alpha_i \sum_{j=i}^{n} \beta_j - \beta_i \left(\sum_{j=1}^{i} \alpha_j \right) - \nu_i \right) x_i$$
$$+ K + \sum_{i=1}^{n} \left(\alpha_i \sum_{j=i+1}^{n} \beta_j + \nu_i \right)$$

and to write down a matrix form of that function. Define the matrix

$$G(M) = \begin{bmatrix} M\alpha_1\beta_1 & \alpha_1\beta_2 & \cdots & \alpha_1\beta_n \\ \alpha_1\beta_2 & M\alpha_2\beta_2 & \cdots & \alpha_2\beta_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1\beta_n & \alpha_2\beta_n & \cdots & M\alpha_n\beta_n \end{bmatrix}. \tag{9}$$

Lemma 1 [Kellerer and Strusevich (2010b)] *The objective function in* (6) *admits a representation*

$$Z(\mathbf{x}) = \mathbf{x}^T G \mathbf{x} + \gamma^T \mathbf{x} + K', \tag{10}$$

where G is a positive definite $n \times n$ matrix $G(2)$ of the form (9) *with $M = 2$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ are $n-$dimensional column vectors and K' is a constant such that*

$$\gamma_i = \mu_i - \alpha_i \sum_{j=i}^{n} \beta_j - \beta_i \sum_{j=1}^{i} \alpha_i - \nu_i,\ i = 1, 2, \ldots, n;$$

$$K' = K + \sum_{i=1}^{n} \left(\alpha_i \sum_{j=i+1}^{n} \beta_j \right) + \sum_{i=1}^{n} \nu_i.$$

In fact, the objective functions of all versions of the formulated problems admit a matrix representation similar to (10) with matrix $G = G(2)$.

Skutella (2001) proves that matrix $G(1)$ of the form (9) with α_i and β_j numbered according to

$$\frac{\alpha_1}{\beta_1} \le \frac{\alpha_2}{\beta_2} \le \ldots \le \frac{\alpha_n}{\beta_n} \tag{11}$$

is positive semi-definite. Thus, under condition (11) matrix $G(M)$ is positive definite for each $M > 1$ (as the sum of a positive semi-definite matrix $G(1)$ and a positive definite diagonal matrix). This implies that under condition (11) the half-product function and its variants are convex. The assumption on convexity is essential for developing fast approximation schemes for the relevant problems and their scheduling applications; see Sects. 5 and 6.

2.3 Set-function form and supermodularity

Another alternative form of the introduced problems of Boolean programming is based on reformulation of the objective functions as set-functions, rather than functions of 0–1 variables.

Following Kellerer et al. (2015), we illustrate this for the half-product function $H(\mathbf{x})$ of the form (1). For a set $N = \{1, 2, \ldots, n\}$, let 2^N denote the family of all subsets of N. For a sequence $(p_1, p_2, \ldots, p_n)$ of n numbers define $p(U) = \sum_{j \in U} p_j$ for every non-empty set $U \in 2^N$ and define $p(\emptyset) = 0$. Similarly, we use notation $\alpha(U)$, $\gamma(U)$, etc. to denote partial sums of the corresponding sequences $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ or $(\gamma_1, \gamma_2, \ldots, \gamma_n)$, etc.

Given a function $\varphi(\mathbf{x})$ with Boolean arguments $x_j \in \{0, 1\}$, we can associate it with a set-function $\varphi(U)$. More precisely, a Boolean vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ can be associated with a set $U \in 2^N$ in such a way that element $j \in N$ belongs to U if and only if $x_j = 1$. We see the Boolean and the set representation of a function as equivalent, and use both types of notation, $\varphi(\mathbf{x})$ and $\varphi(U)$, whichever is more convenient. Using this notation, the knapsack constraint (5) can be written as $\alpha(U) \le A$. See Foldes and Hammer (2005) for a detailed discussion of the link between the set-functions and Boolean functions.

We can rewrite function $H(\mathbf{x})$ in the set-function form as

$$H(U) = \sum_{i,j \in U;\ i<j;} \alpha_i \beta_j - \gamma(U), \tag{12}$$

and the general quadratic function $Q(\mathbf{x})$ as

$$Q(U) = \sum_{i,j \in U;\ i<j} q_{ij} - \gamma(U). \tag{13}$$

In a similar way, the set-function representations can be derived for functions $F(\mathbf{x})$, $P(\mathbf{x})$ and $Z(\mathbf{x})$.

Problem HP of minimizing $H(\mathbf{x})$ can be understood as the problem of finding a set-minimizer U_* such that the inequality $H(U_*) \le H(U)$ holds for all sets $U \in 2^N$. The concepts of set-minimizers for set-functions $\varphi \in \{F, P, S\}$ are defined analogously. In terms of the set-functions, the problems that we consider in this paper can be formulated as $\min\left\{\varphi(U) \mid U \in 2^N\right\}$ if no additional constraints are imposed, and as $\min\left\{\varphi(U) \mid \alpha(U) \le A,\ U \in 2^N\right\}$, if an additional knapsack constraint is introduced.

A set-function $\varphi : 2^N \to \mathbb{R}$ is called *submodular* if for all sets $X, Y \in 2^N$ the inequality

$$\varphi(X \cup Y) + \varphi(X \cap Y) \le \varphi(X) + \varphi(Y)$$

holds, and *supermodular* if for all sets $X, Y \in 2^N$ the inequality

$$\varphi(X \cup Y) + \varphi(X \cap Y) \ge \varphi(X) + \varphi(Y)$$

holds. A pseudo-Boolean function $\varphi(\mathbf{x})$ is submodular (supermodular) if and only if all its second order derivatives are non-positive (non-negative); see Nemhauser et al. (1978). Thus, a quadratic pseudo-Boolean function is submodular (supermodular) if and only if all its quadratic terms have non-positive (non-negative, respectively) coefficients while the signs of the coefficients in the linear part are irrelevant; see Boros and Hammer (2002). Since the half-product function $H(U)$ is a special case of function $Q(U)$ with non-negative coefficients $q_{ij} = \alpha_i \beta_j$, it follows that $H(U)$ is a supermodular function. This fact is useful for deciding the complexity status of the problem of maximizing the half-product function; see Sect. 8.

2.4 Approximation algorithms and schemes

Since Problem HP is NP-hard, the main focus of research of the outlined range of problems is on design and evaluation of approximation algorithms and schemes.

For a collection of decision variables $\mathbf{x}$, consider a problem of minimizing a function $\varphi(\mathbf{x})$ that takes positive values. Recall that a polynomial-time algorithm that finds a feasible solution $\mathbf{x}^H$ such that $\varphi(\mathbf{x}^H)$ is at most $\rho \geq 1$ times the optimal value $\varphi(\mathbf{x}^*)$ is called a $\rho-$ *approximation* algorithm; the value of ρ is called a *worst-case ratio bound*. A family of $\rho-$approximation algorithms is called a *fully polynomial-time approximation scheme (FPTAS)* if $\rho = 1 + \varepsilon$ for any $\varepsilon > 0$ and the running time is polynomial with respect to both the length of the problem input and $1/\varepsilon$. If a function $\varphi(\mathbf{x})$ takes both positive and negative values, then an FPTAS delivers a feasible solution $\mathbf{x}^H$ such that $\varphi(\mathbf{x}^H) - \varphi(\mathbf{x}^*) \leq \varepsilon\, |\varphi(\mathbf{x}^*)|$. The latter definition is applicable to Problem HP, while the former definition is suitable for problems PosHP, PosHPK, SQK and their scheduling applications. A special attention is paid to the design of FPTASs that require *strongly polynomial* running time, i.e., time bounded by a polynomial that depends on n and $1/\varepsilon$ only.

Among the results on the general Problem QK a lack of approximation algorithms is especially noticeable, while for the linear knapsack problems the design of approximation algorithms and schemes is one of the major directions of research, see Kellerer et al. (2004) and Pisinger (2007). Unlike the general Problem QK, the problems related to minimization of the half-product are NP-hard only in the ordinary sense and are solvable in pseudopolynomial time, see Sects. 4.1, 5.1 and 6.1. This gives a hope for developing fully polynomial-time approximation schemes for these problems, at least under some additional conditions that may appear relevant for scheduling applications.

The problems of the outlined range, with and without a linear knapsack constraint, serve as mathematical models of many scheduling problems, and approximation algorithms and schemes for the Boolean programming problems can be adapted for the relevant scheduling problems.

An interesting feature of the problems under consideration initially brought by Janiak et al. (2005) and Erel and Ghosh (2008) is that from the point of view of approximability there is a difference between minimizing a pure half-product function $H(\mathbf{x})$ of the form (1) and minimizing the function $F(\mathbf{x}) = H(\mathbf{x}) + K$. The presence of an additive constant K may influence the behavior of approximation algorithms, e.g., an FPTAS for the problem of minimizing function $H(\mathbf{x})$ does not need to yield an FPTAS for the problem of minimizing F. We discuss these issues in Sect. 4.

Additional aspects, also reviewed in this paper, include approximability issues of the problems of maximizing Boolean functions related to the half-product; see Sect. 8. Besides, we also discuss differential approximation algorithms and schemes, which rely on an alternative approach to evaluating the quality of an approximate solution; see Sect. 7.

3 Scheduling problems: formulations and reductions to Boolean programming

In this section, we present a number of scheduling problems that have initiated the study on the half-product minimization and on the symmetric quadratic knapsack problem. For each scheduling problem discussed below we only mention its complexity status and provide its reduction either to one of the Boolean programming problems introduced in Sect. 2 , such as Problems HP, HPAdd, PosHP, PosHPK or SQK. The issues of developing approximation schemes for these problems are discussed later in the paper.

In most scheduling problems reviewed in this paper, we are given a set $N = \{1, 2, \ldots, n\}$ of jobs to be processed without preemption on a single machine. The processing of job $j \in N$ takes p_j time units. There is a positive weight w_j associated with job j, which indicates its relative importance. All values p_j and w_j are positive integers. The machine processes at most one job at a time. The completion time of job $j \in N$ in a feasible schedule S is denoted by $C_j(S)$, or shortly C_j if it is clear which schedule is referred to. In a specific problem, it is required to minimize a function $Z(S)$ that depends on the completion times $C_j(S)$. For all problems under consideration S^* denotes an optimal schedule, i.e., $Z(S^*) \le Z(S)$ for any feasible schedule S.

For all scheduling problems we use a classification scheme widely accepted in scheduling theory that associates each problem with a three-field descriptor $\alpha|\beta|\gamma$ where α represents the machine environment, β defines the job characteristics, and γ is the optimality criterion.

Unless stated otherwise, the jobs are numbered in such a way that

$$\frac{p_1}{w_1} \le \frac{p_2}{w_2} \le \cdots \le \frac{p_n}{w_n}. \tag{14}$$

We call the sequence of jobs numbered in accordance with (14) a *Smith* sequence or a *WSPT* sequence (Weighted Shortest Processing Time). Recall that in an optimal schedule for the classical single machine problem of minimizing the sum of the weighted completion times, the jobs are processed according to the WSPT sequence, see Smith (1956).

Throughout this paper W denotes the sum of all weights, i.e.,

$$W = \sum_{j=1}^{n} w_j, \tag{15}$$

while the total processing time of all jobs is denoted by

$$p(N) = \sum_{j=1}^{n} p_j. \tag{16}$$

Similarly, for a non-empty subset N' define $p(N') := \sum_{j\in N'} p_j$ and additionally define $p(\emptyset) := 0$.

3.1 Scheduling with machine non-availability

Consider a scheduling model that belongs to the family of scheduling models with *machine availability constraints*. We refer to the surveys by Lee (2004) and by Ma et al. (2010) for the most recent reviews of deterministic scheduling under availability constraints. Assume that for the processing machine there is a known *non-availability* interval $I = [s, t]$, during which the machine cannot perform the processing of any job. This non-availability interval can be due to some scheduled activity other than processing (a maintenance period, a rest period, etc.). The job that is affected by the non-availability interval is called the *crossover* job. There are several possible scenarios to handle the crossover job. Under the *non-resumable* scenario, the crossover job that cannot be completed by time s is restarted from scratch at time t. Under the *resumable* scenario the crossover job is interrupted at time s and resumed from the point of interruption at time t. It is required to minimize the total weighted completion time, i.e., the function

$$Z(S) = \sum_{j=1}^{n} w_j C_j(S).$$

Extending standard scheduling notation, we denote the resulting problem under the non-resumable scenario by $1|h(1), N-res|\sum w_j C_j$, and that under the resumable scenario by $1|h(1), Res|\sum w_j C_j$. Both problems are NP-hard in the ordinary sense and solvable in pseudopolynomial time by dynamic programming (DP), see Adiri et al. (1989) and Lee (1996). On the other hand, if the weights are equal, the resulting problem $1|h(1), Res|\sum C_j$ is solvable in $O(n \log n)$ time, while $1|h(1), N-res|\sum w_j C_j$ remains NP-hard in the ordinary sense; see Adiri et al. (1989) and Lee and Liman (1992).

For each problem $1|h(1), N-res|\sum w_j C_j$ and $1|h(1), Res|\sum w_j C_j$, there exists an optimal schedule in which the jobs sequenced before and after the crossover job follow the WSPT rule. Both problems can be formulated in terms of Problem SQK, as shown by Kellerer and Strusevich (2010a).

Given problem $1|h(1), N-res|\sum w_j C_j$, introduce a Boolean variable x_j in such a way that

$$x_j = \begin{cases} 1, & \text{if job } j \text{ completes before interval } I \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

for each job j, $1 \le j \le n$. If job j completes before the interval I , then

$$C_j = \sum_{i=1}^{j} p_i x_i, \tag{18}$$

while if it completes after the interval I then

$$C_j = t + \sum_{i=1}^{j} p_j (1 - x_j),$$

Thus, the sum of the weighted completion times can be written as

$$\begin{aligned} Z(\mathbf{x}) &= \sum_{j=1}^{n} w_j x_j \sum_{i=1}^{j} p_i x_i + \sum_{j=1}^{n} w_j (1 - x_j) \left(t + \sum_{i=1}^{j} p_i (1 - x_i) \right) \\ &= \sum_{1 \le i \le j \le n} p_i w_j x_i x_j + \sum_{1 \le i \le j \le n} p_i w_j (1 - x_i)(1 - x_j) + t \sum_{j=1}^{n} w_j (1 - x_j). \end{aligned}$$

Taking into account (8), we deduce that problem $1\,|h(1), N-res|\sum w_j C_j$ can be formulated as the following Boolean quadratic programming problem

$$\begin{aligned} \text{Minimize } Z(\mathbf{x}) = & \sum_{1 \le i < j \le n} p_i w_j x_i x_j + \sum_{1 \le i < j \le n} p_i w_j (1 - x_i)(1 - x_j) \\ & + t \sum_{j=1}^{n} w_j (1 - x_j) + \sum_{j=1}^{n} p_j w_j \\ \text{subject to } & \sum_{j=1}^{n} p_j x_j \le s \\ & x_j \in \{0, 1\}, \ \ j = 1, 2, \ldots, n. \end{aligned} \tag{19}$$

If in (6) we define

$$\alpha_j = p_j, \ \beta_j = w_j, \ \mu_j = 0, \ \nu_j = w_j t, \ j = 1, 2, \ldots, n; \ A = s, \ K = \sum_{j=1}^{n} p_j w_j,$$

then (19) becomes an instance of (6).

For problem $1|h(1), Res|\sum w_j C_j$, suppose that a certain job is chosen as the crossover job. Denote the processing time and the weight of the chosen crossover job by p and w, respectively, and renumber the remaining jobs taken according to the WSPT rule by the integers $1, 2, \dots, m$, where $m = n - 1$.

A feasible schedule for problem $1|h(1), Res|\sum w_j C_j$ with a fixed crossover job can be found by inserting the crossover job into a schedule for processing the jobs $1, 2, \dots, m$ under the non-resumable scenario.

Let S^* denote the optimal schedule that delivers the smallest value $Z(S^*)$ of the objective function, while $S(p)$ denote a feasible schedule with a fixed crossover job with the processing time p and weight w. Denote the smallest value of the function among all schedules with the chosen crossover job by $Z^*(p)$.

Define the Boolean decision variables x_j such that (17) holds for each j, $1 \le j \le m$. It follows from (19) that for an arbitrary assignment of variables x_j the sum of the weighted completion times of the corresponding schedule S_m under the non-resumable scenario is given by

$$Z_m = \sum_{1 \le i < j \le m} p_i w_j x_i x_j + \sum_{1 \le i < j \le m} p_i w_j (1 - x_i)(1 - x_j) + t \sum_{j=1}^{m} w_j (1 - x_j) + \sum_{j=1}^{m} p_j w_j,$$

where

$$\sum_{j=1}^{m} p_j x_j \le s, \ x_j \in \{0, 1\}, \ j = 1, 2, \dots, m.$$

To convert a schedule S_m into a schedule $S(p)$ that is feasible for problem $1|h(1), Res|\sum w_j C_j$ with the chosen crossover job, we process the crossover job for px time units before the non-availability interval starting at time

$$y_m = \sum_{j=1}^{m} p_j x_j,$$

where either $x = 1$, if

$$p < s - y_m,$$

or

$$x = \frac{s - y_m}{p},$$

otherwise. The former case should be ignored, since the chosen job completes earlier than time s and is not a crossover job. In the latter case, the chosen job is the crossover job that is additionally processed for $p(1 - x)$ time units starting at time t, and this increases the starting (and the completion) time of each job with $x_j = 0$ by $p(1 - x)$.

The value of the objective function of the resulting schedule $S(p)$ can be written as

$$Z(p) = Z_m + F(y_m, W_m, x), \tag{20}$$

where $W_m = \sum_{j=1}^{m} w_j (1 - x_j)$ and

$$F(y_m, W_m, x) = w\,(t + p(1 - x)) + W_m p(1 - x).$$

3.2 Scheduling with a floating maintenance period

In the problems in Sect. 3.1, the start time of the machine non-availability interval of duration $\Delta = t - s$ is fixed. By contrast, a *floating* machine non-availability interval can be viewed as a non-availability period of length Δ that may start at any time, provided that it is completed no later than a given deadline d. A possible meaningful interpretation of this situation is that the machine is subject to a compulsory maintenance during the planning period, the length of the maintenance is Δ time units, it must be completed by the deadline d, and the decision-maker has to decide when to start the maintenance period (MP). We denote the problem of minimizing the total weighted completion time in these settings by $1|C_{MP} \le d|\sum w_j C_j$, where C_{MP} means the completion time of the MP. This problem is closely related to one of the single machine scheduling problems with two competing agents, introduced and studied by Agnetis et al. (2004) together with other two-agent scheduling problems. Suppose that two agents intend to use a single machine, Agent A owns the $A-$jobs, while Agent B owns the $B-$jobs. Agent A wants to minimize the sum of the weighted completion times of the $A-$jobs, while Agent B wants to have all the $B-$jobs completed by a given deadline d. It is easily verified that there exists an optimal schedule the $B-$jobs can be processed as a block, without intermediate idle time, and this will not increase the objective function of Agent A. Thus, provided that the processing times and weights of the $A-$jobs are equal to p_j and w_j, respectively, and the total processing time of the $B-$jobs is equal to Δ, the two-agent problem is equivalent to problem $1|C_{MP} \le d|\sum w_j C_j$. Notice that the two-agent problem is proved NP-hard in the ordinary sense; see Agnetis et al. (2004).

Given problem $1|C_{MP} \le d|\sum w_j C_j$, introduce the *associated* problem $1|h(1), Res|\sum w_j C_j$ discussed in Sect. 3.1, in which the fixed non-availability interval of length Δ is defined by $[s, t] = [d - \Delta, d]$, while the processing times of the jobs and their weights remain equal to p_j and w_j, respectively. As proved by Kellerer and Strusevich (2010a), problem $1|C_{MP} \le d|\sum w_j C_j$ and the associated problem $1|h(1), Res|\sum w_j C_j$ are equivalent, i.e., any schedule feasible for the associated problem $1|h(1), Res|\sum w_j C_j$ can be transformed into a schedule for the original problem $1|C_{MP} \le d|\sum w_j C_j$ and vice versa, without any change in the objective function value.

In practice, the main reason to run maintenance of a piece of equipment is to restore its conditions, that may get worse during the previous processing. Although scheduling models that address various deterioration effects have been extensively studied since the early 1990s, the integrated models that combine machine deterioration and its maintenance are fairly new, see, e.g., Yang and Yang (2010), Zhao and Tang (2010) and Rustogi and Strusevich (2014, 2015).

The model that we discuss below is based on the paper by Kellerer et al. (2013). A single machine is subject to a so-called *cumulative* deterioration. Each job $j \in N$ is associated with an integer p_j that is called its "normal" processing time. A maintenance period has to be run exactly once during the planning period and it will restore the machine conditions completely, i.e., after the MP the machine is as good as new. Under cumulative deterioration, the actual processing time of a job depends on the sum of the normal times of the earlier sequenced jobs. Wu et al. (2011) list about a dozen of various cumulative effects. In this paper, we focus on the models with a specific cumulative deterioration effect, assuming that the actual processing time $p_j^{[r]}$ of a job j that is sequenced in position r, $1 \le r \le n$, of a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ is given by

$$p_j^{[r]} = p_j \left(1 + \sum_{k=1}^{r-1} p_{\pi(k)}\right). \tag{21}$$

This formula is a variant of the most common model for cumulative deterioration, see Kuo and Yang (2006) and Gordon et al. (2008). We distinguish between two versions of the maintenance periods.

(i) Constant Maintenance: the duration of the MP is Δ time units, where $\Delta > 0$.
(ii) Start Time Dependent Maintenance: the duration of the MP is $\Phi\tau + \Delta$ time units, provided that the MP starts at time τ; here $\Phi > 0$ and $\Delta \geq 0$.

For the latter type of maintenance, the later a machine is sent for maintenance, the longer it takes to restore it to an acceptable condition. This type of maintenance has been introduced by Kubzin and Strusevich (2005, 2006). We denote the problem of minimizing the *makespan*, i.e., the maximum completion time, by $1\,|Cumu, MP(\Phi)|\,C_{\max}$, provided that the machine is subject to cumulative deterioration and the duration of the MP is equal to $\Phi\tau + \Delta$. The version with constant maintenance, i.e., with $\Phi = 0$, is denoted by $1\,|Cumu, MP(0)|\,C_{\max}$.

In a schedule with a single MP the jobs are split into two groups: group 1 consists of the jobs scheduled before the maintenance and group 2 contains all other jobs. For problem $1\,|Cumu, MP(\Phi)|\,C_{\max}$, consider a schedule S with two groups. Let N_i be the set of jobs in group i and $|N_i| = n_i$ for $i \in \{1, 2\}$. Let $\pi = (\pi(1), \ldots, \pi(n_1))$ and $\sigma = (\sigma(1), \ldots, \sigma(n_2))$ denote a sequence of jobs of set N_1 and N_2, respectively. In accordance with (21), the makespan of schedule S is given by

$$\begin{aligned}C_{\max}(S) = {} & p_{\pi(1)} + \sum_{r=2}^{n_1} p_{\pi(r)}\left(1 + \sum_{k=1}^{r-1} p_{\pi(k)}\right)\\ & + \Phi\left(p_{\pi(1)} + \sum_{r=2}^{n_1} p_{\pi(r)}\left(1 + \sum_{k=1}^{r-1} p_{\pi(k)}\right)\right) + \Delta\\ & + p_{\sigma(1)} + \sum_{r=2}^{n_2} p_{\sigma(r)}\left(1 + \sum_{k=1}^{r-1} p_{\sigma(k)}\right)\end{aligned}$$

which implies

$$\begin{aligned}C_{\max}(S) = {} & p(N) + \frac{1}{2}\left(p(N_1)^2 + p(N_2)^2 - \sum_{j\in N} p_j^2\right)\\ & + \Phi\left(p(N_1) + \frac{1}{2}\left(p(N_1)^2 - \sum_{j\in N_1} p_j^2\right)\right) + \Delta\end{aligned} \tag{22}$$

for problem $1\,|Cumu, MP(\Phi)|\,C_{\max}$ and

$$C_{\max}(S) = p(N) + \frac{1}{2}\left(p(N_1)^2 + p(N_2)^2\right) + \Delta - \frac{1}{2}\sum_{j\in N} p_j^2 \tag{23}$$

for problem $1\,|Cumu, MP(0)|\,C_{\max}$.

Notice that (22) and (23) demonstrate that for problem $1\,|Cumu, MP(\Phi)|\,C_{\max}$ the order of jobs in each group does not affect the makespan. This complies with Gordon et al. (2008), where for the single machine problem with the deterioration effect (21) and no maintenance the makespan has been shown to be sequence independent. Thus, the main issue in solving problem $1\,|Cumu, MP(\Phi)|\,C_{\max}$, including its simpler version $1\,|Cumu, MP(0)|\,C_{\max}$, is to find an appropriate partition of the jobs into two groups. It is proved by Kellerer et al. (2013) that both problems are NP-hard.

Given problem $1\,|Cumu, MP(\Phi)|\,C_{\max}$, introduce a Boolean variable x_j in such a way that

$$x_j = \begin{cases} 1, & \text{if job } j \text{ is scheduled in the first group}\\ 0, & \text{otherwise.}\end{cases}$$

Then problem $1\,|Cumu, MP(\Phi)|\, C_{\max}$ reduces to minimizing the function

$$F_{\Phi}(\mathbf{x}) = (\Phi + 1)\left(\sum_{1\le i<j\le n} p_i p_j x_i x_j + \sum_{j=1}^{n} p_j x_j\right) + \sum_{1\le i<j\le n} p_i p_j (1-x_i)(1-x_j) + \sum_{j=1}^{n} p_j (1-x_j) + \Delta, \tag{24}$$

which is a half-product plus a constant.

It is worth noticing that for $\Phi = 0$ the function (24) is the simplest form of function (6) with $\alpha_j = \beta_j = \mu_j = \nu_j = p_j,\ j = 1, 2, \ldots, n.$

3.3 Minimizing total weighted earliness and tardiness

In this model, the jobs have a common due date d. In a schedule S, a job is said to be *early* if $C_j(S) - d \le 0$, and its *earliness* is defined as $E_j(S) = d - C_j(S)$. On the other hand, a job is said to be *late* if $C_j(S) - d > 0$, and its *tardiness* is defined as $T_j(S) = C_j(S) - d$. The aim is to find a schedule that minimizes the function $\sum_{j\in N} w_j \left(E_j(S) + T_j(S)\right)$.

Problems with an earliness-tardiness criterion are important in just-in-time manufacturing, where the earliness generates holding costs and the tardiness incurs a penalty for a late delivery. Notice that the weights are symmetric, i.e., for job j the same weight w_j is applied, no matter the job is late or early. Let $p(N)$ be the total processing time of all jobs defined by (16). If $p(N) \le d$, the due date is called *large* or *nonrestrictive*; otherwise, for $p(N) > d$, the due date is called *small* or *restrictive*. This classification is traditional and is justified by the fact that the size of the due date may influence a possible structure of a feasible schedule, as well as the complexity and the approximability status of the problem. We denote the problems to minimize the total weighted earliness-tardiness by $1|d_j = d, p(N) \le d|\sum w_j(E_j + T_j)$ if the due date is large and by $1|d_j = d, p(N) > d|\sum w_j(E_j + T_j)$ if the due date is small. As far as problem $1|d_j = d, p(N) \le d|\sum w_j(E_j + T_j)$ is concerned, it is solvable in $O(n \log n)$ time, provided that the weights are equal; otherwise, it is NP-hard in the ordinary sense as proved by Hall and Posner (1991). If the due date is small then problem $1|d_j = d, p(N) > d|\sum(E_j + T_j)$ is NP-hard in the ordinary sense even if the weights are equal; see Hall et al. (1991) and Hoogeveen and van de Velde (1991).

As proved by Hall and Posner (1991), for problem $1|d_j = d, p(N) \le d|\sum w_j(E_j + T_j)$ there exists an optimal schedule in which some job completes exactly at time d, i.e., it will have neither earliness nor tardiness. There is no intermediate idle time in job processing, but some idle time may occur before the first early job; we call this class of schedules *Class 1*.

As demonstrated by Hall et al. (1991), for problem $1|d_j = d, p(N) > d||\sum w_j(E_j + T_j)$, an optimal schedule can be sought for either in Class 1 described above or in *Class 2* of schedules, in which the early jobs are processed starting at time zero and are followed by the *straddling* job that starts before time d and is completed after time d; in turn, the straddling job is followed by the block of late jobs.

In a schedule of either class the early jobs are processed in the order opposite to their numbering by the WSPT rule, while the jobs that start either at or after the due date are processed in the order of their numbering.

Following Kellerer and Strusevich (2010b), to establish the relevance of these two scheduling problems to Problem SQK and Problem HPAdd, introduce Boolean decision variables

$$x_j = \begin{cases} 1, & \text{if job } j \text{ completes by the due date } d \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

In order to find a Class 1 schedule that is optimal for problem $1|d_j = d, p(N) \le d|\sum w_j(E_j + T_j)$, consider the jobs in the order of their numbering defined by (14), and compute the completion time and the earliness of a job j that completes by time d as

$$C_j = d - \sum_{i=1}^{j-1} p_i x_i, \ E_j = \sum_{i=1}^{j-1} p_i x_i.$$

If job j is starts after the due date, then its completion time and tardiness are given by

$$C_j = d + \sum_{i=1}^{j} p_i(1 - x_i), \ T_j = \sum_{i=1}^{j} p_i(1 - x_i).$$

Thus, we obtain that the objective function can be written as

$$\sum_{j=1}^{n} w_j(E_j + T_j) = \sum_{1 \le i < j \le n} p_i w_j x_i x_j + \sum_{1 \le i < j \le n} p_i w_j (1 - x_i)(1 - x_j) + \sum_{j=1}^{n} p_j w_j (1 - x_j), \tag{26}$$

which implies that the problem reduces to Problem HPAdd. This fact has also been pointed out in Erel and Ghosh (2008).

On the other hand, for problem $1|d_j = d, p(N) > d|\sum w_j \left(E_j + T_j\right)$ finding the best schedule in Class 1 reduces to minimizing (26) subject to the knapsack constraint, i.e., reduces to the problem

$$\begin{aligned} \text{Minimize } Z(\mathbf{x}) = &\sum_{1 \le i < j \le n} p_i w_j x_i x_j + \sum_{1 \le i < j \le n} p_i w_j (1 - x_i)(1 - x_j) \\ &+ \textstyle\sum_{j=1}^{n} p_j w_j (1 - x_j) \\ \text{subject to } &\sum_{j=1}^{n} p_j x_j \le d \\ &x_j \in \{0, 1\}, \ j = 1, 2, \ldots, n. \end{aligned} \tag{27}$$

If we set

$$\alpha_j = p_j, \ \beta_j = w_j, \ \mu_j = 0, \ \nu_j = w_j p_j, \ j = 1, 2, \ldots, n, \ A = d, \ K = 0 \tag{28}$$

it follows that (27) and (6) coincide, i.e., the problem defined by (27) is Problem SQK.

For problem $1|d_j = d, p(N) > d|\sum w_j \left(E_j + T_j\right)$, in order to find the best schedule in Class 2 suppose that a certain job is chosen as the straddling job. Renumber the remaining jobs taken according to the WSPT rule by the integers $1, 2, \ldots, m$, where $m = n - 1$.

A feasible schedule of Class 2 with a fixed straddling job can be found by inserting the chosen job into a schedule S_m, the best Class 1 schedule for processing the jobs $1, 2, \ldots, m$. If such an insertion is successful, it will increase the earliness of each early job and the tardiness of each tardy job in schedule S_m.

Let p and w denote the processing time and the weight of the job that is chosen as a possible straddling job. Renumber the remaining jobs in the WSPT order by the numbers $1, 2, \ldots, m$. Take a feasible schedule S_m for these jobs that belongs to Class 1. Such a schedule is defined by a partition of the jobs into early and late, i.e., by an assignment of the Boolean variables (25). The value of the objective function for schedule S_m is given by

$$Z_m = \sum_{1 \le i < j \le m} p_i w_j x_i x_j + \sum_{1 \le i < j \le m} p_i w_j (1 - x_i)(1 - x_j) + \sum_{j=1}^{m} p_j w_j \left(1 - x_j\right),$$

where

$$\sum_{j=1}^{m} p_j x_j \leq d.$$

The cases that either $\sum_{j=1}^{m} p_j x_j = d$ or $\sum_{j=1}^{m} p_j x_j + p \leq d$ must be ignored, since the chosen job cannot be inserted as straddling. It is obvious that the problem of minimizing Z_m is of the same structure as problem (27).

Compute

$$x = \frac{d - \sum_{j=1}^{m} p_j x_j}{p}.$$

We only need to consider the case that $0 < x < 1$. To convert a schedule S_m into a schedule S that is feasible for the original problem with the chosen straddling job, we reduce the starting time of each early job by px and start the straddling job at time $\sum_{j=1}^{m} p_j x_j$. The straddling job is processed for $p(1-x)$ time units after time d, thereby creating tardiness and forcing all other tardy jobs to start $p(1-x)$ time units later.

3.4 Minimizing total weighted tardiness

Here it is required to minimize the total weighted tardiness with respect to a common due date. We denote this problem by $1|d_j = d|\sum w_j T_j$. Obviously, it only makes sense to consider the instances for which $p(N) > d$. This problem is NP-hard in the ordinary sense, as proved by Yuan (1992). For this problem, Lawler and Moore (1969) provide a dynamic programming (DP) algorithm that requires $O(n^2 d)$ time and demonstrate that the problem with equal weights is solvable in $O(n^2)$ time.

This problem can be handled similarly to finding a schedule of Class 2 for problem $1|d_j = d, p(N) > d|\sum w_j(E_j + T_j)$, see Kellerer and Strusevich (2006). Introduce the Boolean decision variables

$$x_j = \begin{cases} 1, & \text{if job } j \text{ completes after the due date } d \\ 0, & \text{otherwise.} \end{cases}$$

Let p and w denote the processing time and the weight of the job that is chosen as a possible straddling job. Renumber the remaining jobs in the WSPT order by the numbers $1, 2, \ldots, m$. The problem of scheduling the jobs $1, 2, \ldots, m$ reduces to maximizing the function

$$Z_m = \sum_{j=1}^{m} w_j \left(\sum_{i=1}^{j} p_i x_i \right) x_j = \sum_{1 \leq i \leq j \leq m} p_i w_j x_i x_j,$$

subject to

$$\sum_{j=1}^{m} p_j \left(1 - x_j\right) \leq d$$
$$x_j \in \{0, 1\}, \ j = 1, 2, \ldots, m.$$

Given the values of $x_j \in \{0, 1\}$, $j = 1, 2, \ldots, m$, we can create the corresponding schedule S_m scheduling the jobs with $x_j = 0$ as the block of early jobs and the jobs with $x_j = 1$ as the block of late jobs; the jobs of each block are sequenced in the order of their numbering. The chosen straddling job is inserted to start at time $\sum_{j=1}^{m} p_j \left(1 - x_j\right)$.

Although the function Z_m above is not symmetric in the sense of the objective in (6), it possesses structural properties that allow us to use for its optimization and approximation the algorithmic ideas developed for the symmetric functions.

3.5 Minimizing completion time variance

Given a schedule S for a single machine scheduling problem, the average completion time is defined as

$$\overline{C}(S) = \frac{1}{n}\sum_{j=1}^{n} C_j(S),$$

and the completion time variance (CTV) is defined as

$$V(S) = \frac{1}{n}\left(\sum_{j=1}^{n} C_j(S) - \overline{C}(S)\right)^2.$$

We denote the problem of minimizing the CTV by $1\,||\,V$. Since the 1970s, the problem has been known to be applicable in various contexts, see, e.g., Merten and Muller (1972) where this objective function is first introduced. Cheng and Kubiak (2005) refer to Kanet (1981) to stress that the CTV as a measure of the schedule quality "is applicable to any service and manufacturing setting where it is desirable to provide jobs or customers with approximately the same level of service" .

An extended version of the problem in which the jobs have weights w_j and the purpose is to minimize the weighted CTV defined by

$$WV(S) = \sum_{j=1}^{n} w_j\left(C_j(S) - \frac{1}{W}\sum_{j=1}^{n} w_j C_j(S)\right)^2,$$

where W is the sum of all weights as defined by (15); see Merten and Muller (1972) and Cai (1995). We call this problem $1\,||\,WV$. As agreed earlier, the jobs are numbered in accordance with (14); in the non-weighted case this numbering reduces to $p_1 \le p_2 \le \ldots \le p_n$.

The objective function $V(S)$ possesses several interesting properties. One of them, established by Merten and Muller (1972), holds for any sequence $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ of jobs. The value of the CTV for the jobs taken in this sequence is equal to that for the jobs taken in the "almost reversed" sequence $\pi' = (\pi(1), \pi(n), \pi(n-1), \ldots, \pi(2))$.

A permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ in which the shortest job 1 is placed in a position k, $1 \le k \le n$, is called $V-$*shaped* if

$$p_{\pi(1)} \ge \cdots \ge p_{\pi(k)} \le p_{\pi(k+1)} \le \cdots p_{\pi(n)}.$$

Eilon and Chowdhury (1972) show that for problem $1\,||\,V$ there exists an optimal sequence π that is $V-$shaped and the longest job is in the first position, i.e., $\pi(1) = n$. Cai (1995) extends this result to problem $1\,||\,WV$, provided that the weights are agreeable, i.e., the jobs can be numbered so that

$$p_1 \le p_2 \le \cdots \le p_n \text{ and } w_1 \ge w_2 \ge \cdots \ge w_n. \tag{29}$$

It is pointed out by Bagchi et al. (1987) that problem $1\,||\,V$ is equivalent to problem $1\left|d_j = d\right|\frac{1}{n}\sum\left(C_j - d\right)^2$ of minimizing the mean squared deviation of the completion times with respect to a common due date d. De et al. (1989) discuss the differences in the properties of problem $1\left|d_j = d\right|\frac{1}{n}\sum\left(C_j - d\right)^2$ that depend on a relative value of d; these issues are similar to restrictive and non-restrictive due dates for the problem from Sect. 3.3.

Kubiak (1993) settles the complexity status of problem $1\,||\,V$ by proving its NP-hardness in the ordinary sense. Several DP algorithms are known to solve the problem in pseudopolynomial time, but it remains unknown whether problem $1\,||\,WV$ with general weights is NP-hard in the strong sense. Kubiak (1995) reduces problem $1\,||\,V$ to (an NP-hard) problem of maximizing a quadratic submodular function.

Starting from Kubiak (1995), there have been several attempts to write out problem $1\,||\,V$ as a Boolean programming problem with a quadratic function, which we now call the half-product; see, e.g., Jurisch et al. (1997) and Badics and Boros (1998). In particular, Badics and Boros (1998) give a formulation of problem $1\,||\,V$ in terms of Problem HPAdd. The jobs are scanned in the order of their numbering and following decision variables are used:

$$x_j = \begin{cases} 1, & \text{if job } j \text{ is sequenced before job 1} \\ 0, & \text{otherwise,} \end{cases}$$

where $x_n = 1$, since an optimal V−shaped sequence starts with that job. In this case, the completion time of job $j \in N$ is given by

$$C_j = \left(1 - x_j\right)\sum_{k=1}^{j} p_k + p_j + \sum_{k=j+1}^{n-1} p_k x_k + p_n,$$

so that the objective function V as a function of n Boolean variables with $x_n = 1$ can be written as

$$n^2 V\left(x_1, x_2, \ldots, x_{n-1}, 1\right) = H\left(x_1, x_2, \ldots, x_{n-1}\right) + K,$$

where $H\left(x_1, x_2, \ldots, x_{n-1}\right)$ is a half-product with $n-1$ variables and

$$K = \sum_{j=1}^{n} j\left(n-j\right)p_j^2 + 2\sum_{1\le i<j\le n} i\left(n-j\right)p_i p_j.$$

Cheng and Kubiak (2005) reduce problem $1\,||\,WV$ with agreeable weights to Problem HPAdd with the decision variables

$$x_j = \begin{cases} 1, & \text{if job } j \text{ is sequenced after job 1} \\ -1, & \text{otherwise,} \end{cases}$$

where $x_1 = -1$.

3.6 Scheduling with controllable processing times

In scheduling with controllable processing times, the actual durations of the jobs are not fixed in advance, but have to be chosen from a given interval. This area of scheduling has been active since the 1980s, see surveys by Nowicki and Zdrzałka (1990) and by Shabtay and Steiner (2007).

Normally, for a scheduling model with controllable processing times two types of decisions are required: (i) each job has to be assigned its actual processing time, and (ii) a schedule has to be found that provides a required level of quality. There is a penalty for assigning shorter actual processing times, since the reduction in processing time is usually associated with an additional effort, e.g., allocation of additional resources or improving processing conditions. A quality of the resulting schedule is measured with respect to the cost of assigning the actual processing times that guarantee a certain scheduling performance.

The model that is of interest for the purpose of this survey is the following problem of scheduling jobs on a single machine. For each job $j \in N$, its processing time p_j is not given in advance but has to be chosen by the decision-maker from a given interval $\left[\underline{p}_j, \overline{p}_j\right]$. That selection process can be seen as either compressing (also known as crashing) the longest processing time $\overline{p}_j$ down to p_j, and the value $y_j = \overline{p}_j - p_j$ is called the compression amount of job j. Compression may decrease the completion time of each job j but incurs additional cost $v_j y_j$, where v_j is a given non-negative unit compression cost. The goal is to find the actual processing times and the sequence of jobs such that the sum of the total weighted completion time $\sum_{j \in N} w_j C_j$ and the total compression cost $\sum_{j \in N} v_j y_j$ is minimized. We denote this problem by $1 \left| p_j = \overline{p}_j - y_j \right| \sum_{j \in N} w_j C_j + \sum_{j \in N} v_j y_j$.

Vickson (1980) proves that in an optimal schedule each job is either fully compressed, i.e., $p_j = \underline{p}_j$ or fully decompressed, i.e., $p_j = \overline{p}_j$.

In this review, we focus of a special case of the problem in which $\underline{p}_j = 0$. The resulting problem in NP-hard in the ordinary sense, as independently proved by Hoogeveen and Woeginger (2002) and by Wan et al. (2001). Combining the results by Vickson (1980) and the optimality of the WSPT rule (14) for minimizing $\sum_{j \in N} w_j C_j$ on a single machine established by Smith (1956), it follows that in an optimal sequence some jobs will have zero processing times (and therefore zero completion times), while the other jobs will be sequenced in non-decreasing order of $\overline{p}_j / w_j$ and for each of these jobs the compression cost is zero.

Janiak et al. (2005) and Kellerer and Strusevich (2013) show that problem $1 \left| p_j = \overline{p}_j - y_j \right| \sum_{j \in N} w_j C_j + \sum_{j \in N} v_j y_j$ reduces to Problem PosHP. Introduce the Boolean decision variables

$$x_j = \begin{cases} 1, & \text{if } p_j = \overline{p}_j \\ 0, & \text{otherwise} \end{cases}.$$

The completion time of job j satisfies (18), and the objective function can be written as as

$$\begin{aligned} \sum_{j=1}^{n} w_j C_j + \sum_{j=1}^{n} v_j y_j &= \sum_{j=1}^{n} w_j x_j \sum_{i=1}^{j} \overline{p}_i x_i + \sum_{j=1}^{n} \overline{p}_j v_j \left(1 - x_j\right) \\ &= \sum_{1 \le i < j \le n} \overline{p}_i w_j x_i x_j + \sum_{j=1}^{n} \overline{p}_j w_j x_j + \sum_{j=1}^{n} \overline{p}_j v_j \left(1 - x_j\right), \end{aligned}$$

The last expression is a positive half-product function of the form (3) with

$$\alpha_j = \overline{p}_j, \ \beta_j = w_j, \ \mu_j = \overline{p}_j w_j, \ \nu_j = \overline{p}_j v_j, \ K = 0.$$

3.7 Scheduling with rejection

Consider the following model of scheduling with rejection introduced by Engles et al. (2003). The decision-maker has to decide which of the jobs of set N to accept for processing and which to reject. This decision splits the set of jobs into two subsets, N_A and $N_R = N \backslash N_A$ of accepted and rejected jobs, correspondingly. Each rejected job j incurs a penalty of v_j. The purpose is to minimize the sum of the total weighted completion time $\sum_{j \in N_A} w_j C_j$ of the accepted jobs and the total rejection penalty $\sum_{j \in N_R} v_j$. We denote this problem by $1 \left| rej \right| \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j$. Engles et al. (2003) show that this problem is NP-hard in the ordinary sense.

In practice rejection decisions are often taken when the processing capabilities will not allow the completion of all jobs by a given deadline. Kellerer and Strusevich (2013) introduce a restricted version of the problem with rejection, in which all accepted jobs must be completed by a given time d. We denote this problem by $1\left|rej, C_j \le d\right|\sum_{j\in N_A} w_jC_j + \sum_{j\in N_R} v_j$.

The objective function in each of the problems $1\left|rej\right|\sum_{j\in N_A} w_jC_j + \sum_{j\in N_R} v_j$ and $1\left|rej, C_j \le d\right|\sum_{j\in N_A} w_jC_j + \sum_{j\in N_R} v_j$ is a positive half-product function of the form (3).

As before, it follows from the optimality of the WSPT rule for minimizing the total weighted completion time $\sum_{j\in N} w_jC_j$ on a single machine that in an optimal sequence the accepted jobs will be sequenced in accordance with (14). Renumber the jobs in this order, and introduce the Boolean decision variables

$$x_j = \begin{cases} 1, & \text{if } j \text{ is accepted} \\ 0, & \text{otherwise} \end{cases}.$$

Then an accepted job j completes at time C_j given by (18), and the objective function can be written as

$$\begin{aligned}\sum_{j\in N_A} w_jC_j + \sum_{j\in N_R} v_j &= \sum_{j=1}^{n} w_jx_j \sum_{i=1}^{j} p_ix_i + \sum_{j=1}^{n} v_j\left(1-x_j\right) \\ &= \sum_{1\le i<j\le n} p_iw_jx_ix_j + \sum_{j=1}^{n} p_jw_jx_j + \sum_{j=1}^{n} v_j\left(1-x_j\right),\end{aligned}$$

i.e., as a positive half-product function of the form (3) with

$$\alpha_j = p_j,\ \beta_j = w_j,\ \mu_j = p_jw_j,\ \nu_j = v_j,\ K = 0.$$

Thus, problem $1\left|rej\right|\sum_{j\in N_A} w_jC_j + \sum_{j\in N_R} v_j$ is Problem PosHP. For problem $1\left|rej, C_j \le d\right|\sum_{j\in N_A} w_jC_j + \sum_{j\in N_R} v_j$ the condition that all accepted jobs complete no later than time d can be written in the form of an additional knapsack constraint

$$\sum_{j=1}^{n} p_jx_j \le d,$$

so that the problem can be seen as Problem PosHPK of the form (4).

3.8 Scheduling with controllable release dates

In scheduling with controllable release dates, the actual times at which the jobs enter the system are not fixed in advance, but have to be chosen from a given interval. These problems can serve as mathematical models of situations that arise in supply chain scheduling, i.e., when the times by which the supplier delivers the required materials to the manufacturer can be negotiated.

If the due dates r_j are fixed and the jobs are numbered in non-decreasing order of their values, then the optimal makespan, i.e., the maximum completion time, $C_{\max}$ is given

$$C_{\max}(S^*) = \max_{1\le u\le n}\left\{r_u + \sum_{j=u}^{n} p_j\right\}.$$

In this review, we focus on the model studied by Shakhlevich and Strusevich (2006), in which the processing times are fixed and equal to p_j, and the decision-maker chooses the actual values of the release dates r_j from a given interval $[\underline{r}, \bar{r}]$, the same for all jobs $j \in N$. We further assume that the length of the interval exceeds the sum of all processing times. Reducing $\bar{r}$ to some actual value r_j, $\underline{r} \le r_j \le \bar{r}$, incurs additional cost $\beta_j y_j$, where $y_j = \bar{r} - r_j$ is the compression amount of the corresponding release date. The goal is to find the actual release dates and the sequence of jobs such that the sum of the makespan $C_{\max}$ and the total compression cost of the release dates $\sum_{j \in N} v_j y_j$ is minimized. We denote this problem by $1 \left| r_j \in [\underline{r}, \bar{r}] \right| C_{\max} + \sum_{j \in N} v_j y_j$.

Let the jobs that become available earlier than time $\bar{r}$ be called *early* jobs, while the other jobs are called *late*. Notice that the late jobs have a common release date $\bar{r}$, while for the early jobs the release dates have been reduced individually. As proved by Shakhlevich and Strusevich (2006), in an optimal schedule either all jobs are late or there exists a sequence of early jobs with the last early job completed at time $\bar{r}$. It follows from the optimality of the WSPT rule (14) for minimizing $\sum_{j \in N} w_j C_j$ on a single machine established by Smith (1956), that in an optimal sequence the early jobs will be sequenced in non-increasing order of p_j / v_j.

As in Kellerer and Strusevich (2013), considering the jobs in this order, introduce the Boolean decision variables

$$x_j = \begin{cases} 1, & \text{if } j \text{ is sequenced early} \\ 0, & \text{otherwise} \end{cases}.$$

Then $1 \left| r_j \in [\underline{r}, \bar{r}] \right| C_{\max} + \sum_{j \in N} v_j y_j$ reduces to minimizing the function

$$C_{\max} + \sum_{j \in N} v_j y_j = \left(\bar{r} + \sum_{j=1}^{n} p_j \left(1 - x_j\right) \right) + \sum_{1 \le i < j \le n}^{n} v_i p_j x_i x_j + \sum_{j=1}^{n} p_j v_j x_j,$$

i.e., to Problem PosHP.

3.9 Scheduling on two identical parallel machines

Unlike in all other scheduling problems previously discussed in this section, here the jobs have to be processed without preemption on two parallel identical machines M_1 and M_2. The processing time of job j on any of these two machines is p_j. In one problem that we consider it is required to minimize the makespan $C_{\max}$, while in the other problem the objective function is the weighted sum of the completion times. We denote these two problems by $P2 || C_{\max}$ and $P2 || \sum w_j C_j$, respectively. Assume that the jobs are numbered arbitrary in the case of problem $P2 || C_{\max}$ and in accordance with the WSPT rule (14) in the case of problem $P2 || \sum w_j C_j$.

Both problems can be formulated in terms of quadratic Boolean programming (although problem $P2 || C_{\max}$ is essentially the well-known subset-sum problem, a special case of the linear knapsack problem). The first formulations are given by Jurisch et al. (1997). Below we present the formulations of these problems that are due to Kubiak (2005) and both use a half-product function, written with respect to the "exclusive OR" operation $\oplus$. Recall that for two Boolean variables x_i and x_j we have that $x_i \oplus x_j = 1$ if and only if exactly one of these variables is equal to 1. More formally, for a Boolean variable $x \in \{0, 1\}$, define $\overline{x} = 1 - x$. Then $x_i \oplus x_j = x_i \overline{x}_j + \overline{x}_i x_j$. For a scheduling problem with parallel machines define

$$x_j = \begin{cases} 1, & \text{if job } j \text{is scheduled on machine } M_1 \\ 0, & \text{otherwise} \end{cases}.$$

Then for a schedule in which the jobs are considered in the order of a chosen numbering a job j assigned to machine M_1 completes at time C_j that satisfies (18); otherwise, its completion time is

$$C_j = \sum_{k=1}^{j} p_k (1 - x_k).$$

For problem $P2 || C_{\max}$, in order to minimize the makespan it suffices to minimize the product of the completion times of the last jobs assigned to the machines, i.e., to minimize $\sum_{j=1}^{n} p_j x_j \sum_{j=1}^{n} p_j \left(1 - x_j\right)$. It can be verified that

$$\begin{aligned} \sum_{j=1}^{n} p_j x_j \sum_{j=1}^{n} p_j \left(1 - x_j\right) &= p(N) \sum_{j=1}^{n} p_j x_j - \sum_{j=1}^{n} p_j^2 x_j - 2 \sum_{1 \le i < j \le n} p_i p_j x_i x_j \\ &= \sum_{j=1}^{n} \left(p(N) - p_j\right) p_j x_j - 2 \sum_{1 \le i < j \le n} p_i p_j x_i x_j \\ &= \sum_{1 \le j < i \le n} p_i p_j x_i \oplus x_j. \end{aligned}$$

For problem $P2 || \sum w_j C_j$, we derive

$$\sum_{j=1}^{n} w_j C_j = \sum_{j=1}^{n} w_j x_j \sum_{i=1}^{j} p_i x_i + \sum_{j=1}^{n} w_j \left(1 - x_j\right) \sum_{i=1}^{j} p_i (1 - x_i),$$

i.e., the problem reduces to Problem HPAdd. It can be also seen that

$$\sum_{j=1}^{n} w_j C_j = \sum_{1 \le j < i \le n} w_i p_j x_i \oplus x_j.$$

4 Half-product: approximation and scheduling applications

In this section, we review the known results regarding the existing FPTASs for Problem HP, the problem of minimizing the half-product function $H(\mathbf{x})$ of the form (1). We also discuss the implications to the relevant scheduling problems.

4.1 Approximation schemes

Badics and Boros (1998) present the first systematic study on Problem HP, although similar problems of quadratic Boolean programming appeared in the literature earlier, normally in connection with scheduling problems, see, e.g., Kubiak (1995) and Jurisch et al. (1997). Badics and Boros (1998) give an $O(n^4)-$time algorithm that recognizes whether a quadratic function of n Boolean variables is a half-product. They also give the first FPTAS for the problem of minimizing the half-product function $H(\mathbf{x})$ of the form (1) that requires $O(n^2 \log \widehat{A}/\varepsilon)$ time, where $\widehat{A} = \sum_{j=1}^{n} \alpha_j$. Notice that this running time is not strongly polynomial with respect to the length of the input.

Erel and Ghosh (2008) give the first FPTAS that requires strongly polynomial time $O(n^2/\varepsilon)$. Below we present an extended version of their FPTAS adapted by Sarto Basso and Strusevich (2014) for solving Problem HP with an additional knapsack constraint (5). Thus, in terms of set-functions the problem under consideration is $\min \{H(U) \,|\alpha(U) \le A, U \subseteq N\}$.

We start by presenting a dynamic programming algorithm (DP) from Erel and Ghosh (2008) and Sarto Basso and Strusevich (2014) and then explain how this algorithm can be converted into an FPTAS. Our description is done in terms of set-functions. The DP algorithm scans the items in the sequence $(1, 2, \dots, n)$ and manipulates the states of the form $(U_k, \alpha(U_k), H(U_k))$, where $U_k \subseteq \{1, 2, \dots, k\}$ is the set of the selected elements that represents a partial solution, $\alpha(U_k)$ is the weight of the knapsack and $H(U_k)$ is the value of the objective function for the partial solution. Given a state $(U_k, \alpha(U_k), H(U_k))$, it is always feasible not to include the next element $k+1$ into the knapsack, while a new element is included only if it fits the knapsack and decreases the current value of the objective. Notice that for all generated partial solutions the values of the objective function are negative. Formally, the algorithm can be stated as follows.

Algorithm DPHP

```
Step 1  Initialize $(U_0, \alpha(U_0), H(U_0)) = (\varnothing, 0, 0)$.
Step 2  For all $k$ from 0 to $n-1$ do:
    (a) Make transitions from each stored state of the form $(U_k, \alpha(U_k), H(U_k))$ into the state
        $(U_{k+1}, \alpha(U_{k+1}), H(U_{k+1}))$ by setting $U_{k+1} = U_k$, $\alpha(U_{k+1}) = \alpha(U_k)$, $H(U_{k+1}) = H(U_k)$.
        Additionally, if $\alpha(U_k) + \alpha_{k+1} \le A$ (i.e., item $k+1$ fits into the knapsack) and
        $\alpha(U_k)\beta_{k+1} - \gamma_{k+1} < 0$ (item $k+1$ makes a negative contribution into the objective
        function), create another state $(U_{k+1}, \alpha(U_{k+1}), H(U_{k+1}))$ by setting $U_{k+1} = U_k \cup \{k+1\}$,
        $\alpha(U_{k+1}) = \alpha(U_k) + \alpha_{k+1}$, $H(U_{k+1}) = H(U_k) + \alpha(U_k)\beta_{k+1} - \gamma_{k+1}$.
    (b) For all generated states $(U_{k+1}, \alpha(U_{k+1}), H(U_{k+1}))$ with the same $\alpha(U_{k+1})$ value,
        retain the one with smallest value of $H(U_{k+1})$.
Step 3  Output the optimal value of the function that corresponds to the smallest value of
        $H(U_n)$ among all found states of the form $(U_n, \alpha(U_n), H(U_n))$.
```

The running time of the algorithm is $O(nA)$. Its correctness follows from the fact that for two states $(U_{k+1}, \alpha(U_{k+1}), H(U_{k+1}))$ and $(U'_{k+1}, \alpha(U'_{k+1}), H\left(U'_{k+1}\right))$ generated in iteration k we can keep only the former state, provided $\alpha(U_{k+1}) \le \alpha(U'_{k+1})$ and $H(U_{k+1}) \le H\left(U'_{k+1}\right)$. This is proved in Badics and Boros (1998) for their DP for problem $\min \{H(U) \,|U \subseteq N\}$, and the proof carries over if the knapsack constraint is added.

To convert Algorithm DP into an FPTAS, Erel and Ghosh (2008) use a popular technique of thinning the solution space, making sure that the number of states kept after each iteration is $O(n/\varepsilon)$. For an iteration k, $0 \le k \le n-1$, compute LB_{k+1}, the smallest objective function value among all states $(U_{k+1}, \alpha(U_{k+1}), H(U_{k+1}))$ generated after Step 2(a) of Algorithm DP. Recall that LB_{k+1}, as well as all other function values computed by the algorithm, is negative. Thus, since $LB_{k+1} \ge H(U_*)$, we deduce that $|LB_{k+1}| \le |H(U_*)|$. For a given $\varepsilon > 0$, define $\Delta_{k+1} = (\varepsilon\,|LB_{k+1}|)/n = -\varepsilon LB_{k+1}/n$. It follows that for each k, $0 \le k \le n-1$, the inequality $\Delta_{k+1} \le \varepsilon\,|H(U_*)|/n$ holds.

To convert Algorithm DPHP into an FPTAS for problem $\min\{H(U)\,|\alpha(U) \le A, U \subseteq N\}$ we only need to replace Step 2(b) by another storage mechanism:

(i) Divide the interval $[LB_{k+1}, 0]$ into subintervals of width Δ_{k+1}.
(ii) From all states $(U_{k+1}, \alpha(U_{k+1}), H(U_{k+1}))$ generated in Step 2(a) with $H(U_{k+1})$ in the same subinterval, retain the one with the smallest $\alpha(U_{k+1})$.

Notice that the number of subintervals created in each iteration is $O\left(n/\varepsilon\right)$. Since for each subinterval at most one state is kept with the function value in that subinterval, the total number of states kept in each iteration in $O\left(n/\varepsilon\right)$. The resulting algorithm outputs a set U_ε such that $H\left(U_\varepsilon\right) - H\left(U_*\right) \le \varepsilon\left|H\left(U_*\right)\right|$ and requires $O\left(n^2/\varepsilon\right)$ time, i.e., behaves as an FPTAS for problem $\min\left\{H\left(U\right)\left|\alpha\left(U\right) \le A, U \subseteq N\right.\right\}$, as well as for the less restricted Problem HP. Notice that time $O\left(n^2/\varepsilon\right)$ is the fastest possible for an FPTAS for these problems, since computing the objective function for fixed values of decision variables (or, equivalently, a given set U) requires $O\left(n^2\right)$ time.

Kubiak (2005) introduces another type of the half-product function that is defined in terms of the exclusive OR operation (see Sect. 3.9) as follows

$$H_{\alpha,\beta}\left(\mathbf{x}\right) = -\sum_{1\le j<i\le n} \alpha_i\beta_j x_i \oplus x_j. \tag{30}$$

This function is called the *symmetric half-product*, since for any vector $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ with positive components the equalities $H_{\alpha,\beta}\left(\mathbf{x}\right) = H_{\alpha-e,\beta}\left(\mathbf{x}\right) + H_{e,\beta}\left(\mathbf{x}\right)$ and $H_{\alpha,\beta}\left(\mathbf{x}\right) = H_{\alpha,\beta-e}\left(\mathbf{x}\right) + H_{\alpha,e}\left(\mathbf{x}\right)$ hold. For scheduling applications, the *ordered* symmetric half-products are of special interest, in which either the components of the vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ are non-decreasing or the components of the vector $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ are non-increasing. In either case, Kubiak (2005) shows that applying a dynamic programming algorithm to the instance of the problem with appropriately rounded components of the ordered vector results into an FPTAS that requires $O(n^2/\varepsilon)$ time.

It is pointed by Badics and Boros (1998) that algorithms that behave as an FPTAS for the problem $\min\left\{H\left(U\right)\left|U \subseteq N\right.\right\}$, i.e., Problem HP, do not necessarily deliver an $\varepsilon-$approximate solution for the problem $\min\left\{F\left(U\right)\left|U \subseteq N\right.\right\}$, i.e., Problem HPAdd with $F\left(U\right) = H\left(U\right) + K$. In other words, the inequality $H\left(U_\varepsilon\right) - H\left(U_*\right) \le \varepsilon\left|H\left(U_*\right)\right|$ does not imply $F\left(U_\varepsilon\right) - F\left(U_*\right) \le \varepsilon\left|F\left(U_*\right)\right|$. This is due to the fact that $H\left(U_*\right) < 0$ and it is possible that $\left|F\left(U_*\right)\right| = \left|H\left(U_*\right) + K\right| < \left|F\left(U_*\right)\right|$, despite the fact that both Problems HP and HPAdd have the same optimal solution U_* and for any set U the equality $F\left(U\right) - F\left(U_*\right) = H\left(U\right) - H\left(U_*\right)$ holds.

A systematic discussion of these issues is provided by Kubiak (2005) and Janiak et al. (2005). Kubiak (2005) proves the following statement.

Theorem 1 [Kubiak (2005)] *Let U_ε be a solution delivered by an FPTAS for problem $\min\left\{H\left(U\right)\left|U \subseteq N\right.\right\}$. If $\left|H\left(U_*\right)/F\left(U_*\right)\right| \le \alpha$ for some positive $\alpha > 0$, then $F\left(U_\varepsilon\right) - F\left(U_*\right) \le \varepsilon\alpha F\left(U_*\right)$.*

If the condition of Theorem 1 holds for an α that is bounded from above by a polynomial of the length of the input of Problem HPAdd, then an FPTAS by Erel and Ghosh applied to Problem HP with $\varepsilon' = \varepsilon/\alpha$ gives a solution $F\left(U_\varepsilon\right) - F\left(U_*\right) \le \varepsilon\alpha F\left(U_*\right)$, i.e., behaves as an FPTAS for minimizing the function $F\left(U\right)$ that requires $O\left(n^2\alpha/\varepsilon\right)$ time.

For various scheduling applications, there are examples for which α is either a constant or a polynomial of n; see Sect. 4.2. On the other hand, Janiak et al. (2005) demonstrate for Problem HPAdd related to problem $1\left|p_j = \overline{p}_j - y_j\right| \sum_{j\in N} w_jC_j + \sum_{j\in N} v_jy_j$ (see Sect. 3.6) that $|H(\mathbf{x}^*)/F(\mathbf{x}^*)| > F(n)$ for any positive rational function of n.

Erel and Ghosh (2008) develop another approach to Problem HPAdd. Suppose that a lower bound F_{LB} and an upper bound F^{UB} on the optimal value of the function F are available, i.e., $F_{LB} \le F\left(U_*\right) \le F^{UB}$.

Theorem 2 [Erel and Ghosh (2008)] *For Problem HPAdd, let F_{LB} and F^{UB} denote a lower bound and an upper bound, respectively, on the optimal value of the objective function, i.e.,* *for* $\min\{F(U)\,|U \subseteq N\}$ *the inequalities* $F_{LB} \le F(U_*) \le F^{UB}$ *hold. The problem admits an approximation algorithm that delivers a solution* U_0 *such that* $F(U_0) - F_{LB} \le \varepsilon F_{LB}$ *in* $O(\beta n^2/\varepsilon)$ *time, where* $\beta \ge F^{UB}/F_{LB}$.

For the algorithm that is guaranteed by Theorem 2 to be an FPTAS it is required that β should be bounded from above by a polynomial of the length of the input.

If for some initial lower and upper bounds the ratio $\beta \ge F^{UB}/F_{LB}$ is not bounded by a polynomial, then Erel and Ghosh explain how to tighten the bounds by a procedure similar to binary search.

Theorem 3 [Erel and Ghosh (2008)] *Under the conditions of Theorem 2, Problem HPAdd admits a general FPTAS that requires* $O(n^2 \log \beta/\varepsilon)$ *time.*

In particular, if for problem $\min\{F(U)\,|U \subseteq N\}$ the initial F^{UB} is equal to K, and the initial F_{LB} is set to be equal to $1/\varepsilon$, then the general FPTAS guaranteed by Theorem 3 requires $O\left(n^2 \log(K\varepsilon)/\varepsilon\right)$ time. Since normally we may assume that $\varepsilon < 1$, this gives the running time of $O\left(n^2 \log(K)/\varepsilon\right)$.

4.2 Scheduling applications

We start with the problem that is probably most studied in connection with the half-product minimization, namely problem $1||V$ of minimizing the completion time variance, see Section 3.5. The first FPTAS for problem $1||V$ is given by De et al. (1992). It takes $O\left(n^3/\varepsilon\right)$ time and does not involve any reformulation in terms of quadratic Boolean programming. Badics and Boros (1998) adapt their FPTAS for Problem HP to problem $1||V$ but obtain an algorithm that requires $O(n^3 \log p(N)/\varepsilon)$.

The best time of an FPTAS for problem $1||V$ known so far is $O(n^2/\varepsilon)$ and is achieved in several papers, all based on a quadratic Boolean reformulations of the problem:

(i) Kubiak et al. (2002) use a reformulation of problem $1||V$ in terms of a function similar to (30);
(ii) Kubiak (2005) reformulates the problem as a symmetric ordered half-product;
(iii) it is shown by Kubiak et al. (2002) that for the half-product formulation of problem $1||V$ the inequality $|H(\mathbf{x}^*)/F(\mathbf{x}^*)| \le 3$ holds, so that Theorem 1 with $\alpha = 3$ implies that the FPTAS of Erel and Ghosh (2008) gives an $\varepsilon-$approximate solution to problem $1||V$ in $O(n^2/\varepsilon)$ time.

For problem $1||WV$ to minimize the weighted completion time variance with agreeable due dates Cai (1995) gives an approximation scheme that requires $O\left(Wn^2/(w_{\min}\varepsilon)\right)$, where W is the sum of all weights defined by (15) and $w_{\min}$ is the smallest weight. Notice that if applied to problem $1||V$ to minimize the completion time variance, this scheme is an FPTAS that requires $O\left(n^3/\varepsilon\right)$ time, since $W = n$ and $w_{\min} = 1$; this corresponds to best running time for the problem known at the time, see De et al. (1992).

Assume that in problem $1||WV$ the jobs are numbered in accordance with (29). The first FPTAS for problem $1||WV$ is due to Woeginger (1999) and requires $O(n^5 \log^5(\max\{p_n, w_1, n, 1/\varepsilon\})/\varepsilon^5)$ time. An improved algorithm by Cheng and Kubiak (2005) is based on an adaptation of the algorithm by Badics and Boros (1998) and requires $O(n^4 \log(\max\{p_n, w_1, n\})/\varepsilon)$ time.

Erel and Ghosh (2008) report on several improvements regarding the running time of an FPTAS for problem $1||WV$. They claim that their general FPTAS can be adapted in a similar way as it is done by Cheng and Kubiak (2005) with respect to the FPTAS by Badics and Boros

(1998), and this results in a scheme that runs in $O(n^2 \log(\max\{p_n, w_1\})/\varepsilon)$ time. Further, it is known from Cheng and Kubiak (2005) that the constant term K in the formulation of the problem in the form $F(\mathbf{x}) = H(\mathbf{x}) + K$ can be seen as an upper bound F^{UB} on the optimal value of the function, while there exists a lower bound $F_{LB} \leq F(\mathbf{x}^*)$ such that $K \leq 4n^2 F_{LB}$. Thus, $\beta = 4n^2$, and the algorithm guaranteed by Theorem 2 is an FPTAS that needs $O(\beta n^2/\varepsilon) = O(n^4/\varepsilon)$ time, which is strongly polynomial in the length of the problem input. Moreover, for an FPTAS that is guaranteed by Theorem 3 the running time reduces to $O(n^2 \log \beta/\varepsilon) = O(n^2 \log n/\varepsilon)$.

Consider now problem $1|d_j = d, p(N) > d|\sum w_j(E_j + T_j)$ of minimizing the total weighted earliness and tardiness about a common non-restrictive due date; see Sect. 3.3. Hall and Posner (1991) show that the problem is solvable by a dynamic programming algorithm. We may assume that

$$n \leq \max_{j \in N} \{p_j, w_j\}, \tag{31}$$

since otherwise the DP algorithm will require polynomial time.

The first FPTAS for this problem is due to Kovalyov and Kubiak (1999), the running time is $O(n^2 \log^3 (\max\{p_j, w_j, n, 1/\varepsilon\}/\varepsilon^2)$, or under the assumption (31), is $O(n^2 \log^3 (\max\{p_j, w_j, 1/\varepsilon\}/\varepsilon^2)$; their approach does not involve a half-product reformulation. Kubiak (2005) reformulates the problem as an ordered symmetric half-product. He uses the problem as one of the examples for which a representation $F(\mathbf{x}) = H(\mathbf{x}) + K$ is possible, but $|H(\mathbf{x}^*)/F(\mathbf{x}^*)|$ can be arbitrary large, so that there is no direct conversion of an FPTAS for Problem HP to an FPTAS for this scheduling problem based on Theorem 1. Erel and Ghosh (2008) show that for their general FPTAS from Theorem 3 the running time of $O\left(n^2 \log(K)/\varepsilon\right)$ time under the assumption (31) becomes $O(n^2 \log\left(\max\{p_j, w_j\}/\varepsilon\right)$, since here $K = \sum_{i=1}^{n} w_i \sum_{j=1}^{i} p_j \leq n^2 \max\{p_j, w_j\}$.

Problem $1|Cumu, MP(\Phi)|C_{\max}$ formulated in Sect. 3.2 also admits a reformulation as Problem HPAdd. Let the set N of jobs be partitioned into two subsets N_1 and N_2. Consider a schedule in which the jobs of set N_1 are assigned to the first group, while the jobs of set N_2 are scheduled in the second group, after the MP. For such a schedule, let $F_\Phi(N_1, N_2)$ and $F_0(N_1, N_2)$ denote the values of the makespan in problems $1|Cumu, MP(\Phi)|C_{\max}$ and $1|Cumu, MP(0)|C_{\max}$, respectively. Further, let $N_1^*(\Phi)$ and $N_2^*(\Phi)$ denote the sets that form a partition associated with a schedule that is optimal for $1|Cumu, MP(\Phi)|C_{\max}$, $\Phi \geq 0$.

As seen from (23), a partition that defines an optimal schedule for problem $1|Cumu, MP(0)|C_{\max}$ is such that $p(N_1)^2 + p(N_2)^2$ is as small as possible. Finding such a partition $N_1^*(0)$ and $N_2^*(0)$ reduces to the subset-sum problem which admits a very fast FPTAS by Kellerer et al. (2003) that requires no more than $O(\min\{n/\varepsilon, n + 1/\varepsilon^2 \log(1/\varepsilon)\})$ time. The found sets N_1^ε and N_2^ε are such that $p(N_1^\varepsilon)^2 + p(N_1^\varepsilon)^2 \leq (1+\varepsilon)\, p(N_1^*(0))^2 + p(N_2^*(0))^2$, but because of the additive constant in (23) the value $F_0(N_1^\varepsilon, N_2^\varepsilon)$ may be larger than $(1+\varepsilon)\, F_0(N_1^*(0), N_2^*(0))$. Still, Kellerer et al. (2013) show that an FPTAS for the subset-sum problem can be converted into an FPTAS for problem $1|Cumu, MP(0)|C_{\max}$.

For $\Phi > 0$ and an arbitrary partition $N = N_1 \cup N_2$, it is proved by Kellerer et al. (2013) that

$$F_\Phi(N_1, N_2) \leq \left(1 + \frac{\Phi}{2}\right) F_0(N_1, N_2).$$

This allows expressing the lower and upper bounds on the optimal makespan in terms of an $\varepsilon-$approximate solution to problem $1|Cumu, MP(0)|C_{\max}$, so that the ratio $\beta = F^{UB}/F_{LB}$

of the bounds is at most $\left(1+\frac{\Phi}{2}\right)(1+\varepsilon)$, and the algorithm guaranteed by Theorem 2 behaves as an FPTAS for problem $1\,|Cumu, MP(\Phi)|\,C_{\max}$ that runs in $O(n^2/\varepsilon)$ time.

The half-product formulation of problem $P2\,||\,C_{\max}$ from Sect. 3.9 does not contain a constant term. Thus, the fastest FPTAS results from a direct application of the scheme by Erel and Ghosh (2008) which requires $O(n^2/\varepsilon)$ time. It should be mentioned that this problem is not the best to handle via quadratic programming. In fact, it is equivalent to the subset-sum problem for which there is a very fast FPTAS by Kellerer et al. (2003) that requires no more than $O\left(\min\left\{n/\varepsilon, n+1/\varepsilon^2\log(1/\varepsilon)\right\}\right)$ time.

For problem $P2\,||\sum w_jC_j$ it is shown by Kubiak (2005) that for the half-product formulation the inequality $|H(\mathbf{x}^*)/F(\mathbf{x}^*)|\le 2$ holds. Thus, due to Theorem 1 with $\alpha=2$ it follows that the scheme by Erel and Ghosh (2008) requires $O(n^2/\varepsilon)$ time. Notice that Sahni (1976) gives an FPTAS of the same running time derived from different principles.

5 Positive half-product: approximation and scheduling applications

In this section, we describe an approach to designing an FPTAS for Problem PosHP of minimizing function (3) and its knapsack-constrained variant, Problem PosHPK. This approach is developed by Kellerer and Strusevich (2013) and results into the fastest possible FPTASs for each of these problems, with the running time $O\left(n^2/\varepsilon\right)$. However, this is achieved under the assumption that the objective function is convex. Notice that for all known scheduling applications of this result, the objective function is convex, i.e., the convexity assumption does not affect applicability of this approach. Recall that for function (3) to be convex, it is sufficient that (11) holds; see Sect. 2.2. Below we mainly focus on Problem PosHPK, which is more general than the unrestricted Problem PosHP.

5.1 Approximation scheme: general principles

According to Kellerer and Strusevich (2013), there are two main prerequisites that are required for designing an FPTAS for the problems under consideration.

The *first* of these prerequisites is a DP algorithm that finds an exact solution in pseudopolynomial time. The DP algorithm presented below is very similar to Algorithm DPHP for Problem HP given in Sect. 4.1, although written in terms of the Boolean programming notation, rather than the set-function notation. Define

$$A_k=\sum_{j=1}^{k}\alpha_j,\ k=1,2,\ldots,n.$$

and suppose that the values $x_1, x_2, \ldots, x_k$ have been assigned. The DP algorithm deals with partial solutions associated with states of the form

$$(k, Z_k, y_k),$$

where

k is the number of the assigned variables;
Z_k is the current value of the objective function;
$y_k := \sum_{j=1}^{k}\alpha_jx_j$; for Problem PosHPK y_k denotes the total weight of the items currently put into the knapsack.

We now give a formal statement of the DP algorithm. Notice that

$$\sum_{1\le i<j\le n} \alpha_i \beta_j x_i x_j = \sum_{j=2}^{n} \beta_j x_j \sum_{i=1}^{j-1} \alpha_i x_i .$$

Algorithm DP1

Step 1 Start with the initial state $(0, Z_0, y_0) = (0, K, 0)$. Compute the values $A_k = \sum_{j=1}^{k} \alpha_j, k = 1, 2, \dots, n$.

Step 2 For all k from 1 to n make transitions from each stored state of the form

$$(k-1, Z_{k-1}, y_{k-1}) \tag{32}$$

into the states of the form

$$(k, Z_k, y_k) \tag{33}$$

by assigning the next variable x_k.

(a) Define $x_k = 1$, provided that item k fits into the knapsack, i.e., if the inequality $y_{k-1} + \alpha_k \le A$ holds. If feasible, the assignment $x_k = 1$ changes a state (32) to a state of the form (33), where

$$Z_k = Z_{k-1} + \beta_k y_{k-1} + \mu_k,\ \ y_k = y_{k-1} + \alpha_k. \tag{34}$$

(b) Define $x_k = 0$, which is always feasible. This assignment changes a state of the form (32) into the state of the form (33) such that

$$Z_k = Z_{k-1} + \nu_k;\ \ y_k = y_{k-1}. \tag{35}$$

Step 3 Find Z_n^*, the smallest value of Z_n among all found states of the form (n, Z_n, y_n). Perform backtracking and find vector $\mathbf{x}^* = \left(x_1^*, x_2^*, \dots, x_n^*\right)$ that leads to Z_n^*. Output $\mathbf{x}^*$ and $P(\mathbf{x}^*) = Z_n^*$.

Algorithm DP1 can be implemented in $O(nA)$ time. This algorithm will serve as a template for DP algorithms for several other problems considered in this paper; see Sects. 6.1 and 8.

In order to convert Algorithm DP1 into a fast FPTAS, the *second* prerequisite is needed, which is a polynomial time algorithm that finds an upper bound P^{UB}, such that for all instances of the Problem PosHPK the inequality $P^{UB}/P(\mathbf{x}^*) \le R$ holds, where R is a constant.

Below we describe an approximation scheme that delivers the required performance, provided that both mentioned prerequisites (a DP algorithm and a bounded ratio approximation algorithm) are available. The scheme follows the steps of Algorithm DP1, and in order to reduce the number of computed function values, out of all generated states with close objective function values the scheme keeps only one state, that with the smallest current weight of the knapsack. Notice that the inequality $P(\mathbf{x}^*) \ge \frac{1}{R} P^{UB}$ implies that $P_{LB} = \frac{1}{R} P^{UB}$ is a lower bound on $P(\mathbf{x}^*)$.

Algorithm EpsPosHPK

Step 1 Given an upper bound P^{UB} such that $P^{UB}/P(\mathbf{x}^*) \le R$, define a lower bound $P_{LB} := \frac{1}{R} P^{UB}$. For an arbitrary $\varepsilon > 0$, define $\delta := \frac{\varepsilon}{n} P_{LB}$. Split the interval $\left[0, P^{UB}\right]$ into subintervals $I_1, I_2, \dots$ of length δ each.

Step 2 Store the initial state $(0, Z_0, y_0)$ with $Z_0 = K$ and $y_0 = 0$. For each k, $1 \le k \le n$, do the following:

(a) In line with Algorithm DP1, move from a stored state $(k-1, Z_{k-1}, y_{k-1})$ to at most two states of the form $(k, \tilde{Z}_k, \tilde{y}_k)$, where $\tilde{Z}_k \leq P^{UB}$, using the relations (34) and (35).

(b) For each interval I_q, find the state (k, Z_k, y_k) such that Z_k belongs to I_q and $y_k \leq \tilde{y}_k$ for all states $(k, \tilde{Z}_k, \tilde{y}_k)$ with $\tilde{Z}_k$ from I_q. Keep only state (k, Z_k, y_k) and remove all other states $(k, \tilde{Z}_k, \tilde{y}_k)$ with $\tilde{Z}_k$ from I_q.

Step 3 Determine Z^{ε} as the smallest value of Z_n among the states (n, Z_n, y_n). Perform backtracking and find the vector $\mathbf{x}^{\varepsilon} = \left(x_1^{\varepsilon}, x_2^{\varepsilon}, \ldots, x_n^{\varepsilon}\right)$ that leads to Z^{ε}. Output $\mathbf{x}^{\varepsilon}$ and $P\left(\mathbf{x}^{\varepsilon}\right)$ as an approximate solution of Problem PosHPK.

Define $v := \left\lceil P^{UB}/\delta \right\rceil = \lceil Rn/\varepsilon \rceil$. In Step 2, moving from iteration $k-1$ to k, Algorithm EpsPosHPK creates at most $2v$ states from at most v kept states, and moves to the next iteration with at most k stored states. Thus, for each k, Step 2 takes $O(v)$ time, and the overall running time of Algorithm EpsPosHPK is $O(nv) = O\left(Rn^2/\varepsilon\right)$. The following statement summarizes the behaviour of the algorithm.

Theorem 4 *Let $\mathbf{x}^{\varepsilon}$ be a vector found by Algorithm EpsPosHPK applied to Problem PosHPK to minimize a function $P(\mathbf{x})$ of the form (3), with a lower bound P_{LB} and an upper bound P^{UB} on the optimal value. Then $P(\mathbf{x}^{\varepsilon}) - P(\mathbf{x}^*) \leq \varepsilon P_{LB}$ and the running time of the algorithm is $O(Rn^2/\varepsilon)$ time, where $R = P^{UB}/P_{LB}$.*

5.2 Continuous relaxation and constant-ratio approximation

In this subsection, we explain how to obtain the second prerequisite needed for the FPTAS. To find the required upper bound P^{UB} the following approach is used by Kellerer and Strusevich (2013). For Problem PosHPK, let Problem PosHPKr be its continuous relaxation, i.e., the problem obtained from the original Boolean formulation by relaxing the integrality constraints and replacing the condition $x_j \in \{0, 1\}$ by $0 \leq x_j \leq 1$, $j = 1, 2, \ldots, n$. For Problem PosHP, the continuous relaxation is denoted by Problem PosHPr. To derive the corresponding upper bound P^{UB}, Kellerer and Strusevich (2013) first solve the continuous relaxation and then perform an appropriate rounding to obtain a heuristic Boolean solution to the original problem. In order to make sure that the continuous relaxation can be solved in polynomial time, the assumption on the convexity of the objective function is adopted.

Introduce the continuous relaxation of Problem PosHPK under the numbering (11), obtained by replacing the integrality condition $x_j \in \{0, 1\}$ by $0 \leq x_j \leq 1$ for each $j = 1, 2, \ldots, n$. The resulting Problem PosHPKr, belongs to a general area of Quadratic Programming in which it is required to optimize a quadratic function subject to linear constraints. See Hochbaum (2005, 2008) for detailed reviews of quadratic optimization with integer and continuous variables.

The problem of convex quadratic programming is known to be solvable in polynomial time by a modified ellipsoid algorithm due to Kozlov et al. (1979). The fastest known algorithm for this problem is given by Monteiro and Adler (1989) and requires $O(n^3 P)$ time, where n is the number of variables and P is the total length of the input coefficients. However, it is still unknown whether the problem admits a strongly polynomial algorithm, even if the number of linear constraints is fixed or even equal to one, as in the continuous relaxation of the Quadratic Knapsack Problem (Problem QK). Recall that the continuous relaxation of Problem QK to minimize a *separable concave quadratic* function under a single linear constraint is NP-hard, as proved by Moré and Vavasis (1991).

For problems with n continuous decision variables, if the objective function is convex and *separable*, i.e., is the sum of (not necessarily quadratic) convex functions, each depending on one decision variable only, then the problem with linear constraints admits a polynomial-time algorithm developed by Hochbaum and Shantikumar (1990). Problem QK to minimize a *separable convex quadratic* function is solvable in strongly polynomial time. Bretthauer and Shetty (1997) give multiple references to various algorithms for this problem. The best known algorithm is due to Brucker (1984), and it requires only $O(n)$ time. Moreover, an extension of Problem QK to the problem of minimizing a separable differentiable convex (non-necessarily quadratic) function subject to a fixed number of linear constraints also admits an $O(n)$-time algorithm developed by Berman et al. (1993).

Recently, Romeijn et al. (2007) have given an $O(n^3)-$time algorithm that *maximizes a non-separable convex* function of a special structure subject to two knapsack constraints. They specifically point out that their problem is different from Problem QK in possible applications and solution techniques.

In order to design an FPTAS that for each Problem PosHP and Problem PosHPK requires $O(n^2/\varepsilon)$ time, Problem PosHPKr and Problem PosHPKr must be solved in most $O(n^2)$ time.

Following Kellerer and Strusevich (2013), below we outline a possible approach to solving the continuous relaxations. Assume that function (3) is convex. Using the fact that for a Boolean variable $x_j = x_j^2,\ j \in N$, rewrite as

$$\begin{aligned} P(\mathbf{x}) &= \sum_{1 \le i < j \le n} \alpha_i \beta_j x_i x_j + \sum_{j=1}^{n} \mu_j x_j + \sum_{j=1}^{n} \nu_j \left(1 - x_j\right) + K \\ &= \sum_{1 \le i \le j \le n} \alpha_i \beta_j x_i x_j - \sum_{j=1}^{n} \left(\nu_j - \alpha_j \beta_j - \mu_j\right) x_j + \sum_{j=1}^{n} \nu_j + K. \end{aligned}$$

Introduce new decision variables $\chi_j = \alpha_j x_j$, $j = 1, 2, \ldots, n$, and rewrite the continuous relaxation of Problem PosHPK as

$$\begin{aligned} &\text{Minimize}\ \ P(\mathbf{x}) = \sum_{i=1}^{n} c_i \chi_i \sum_{j=1}^{i} \chi_j - \textstyle\sum_{j=1}^{n} \gamma_j \chi_j + K' \\ &\text{Subject to}\ \textstyle\sum_{j=1}^{n} \chi_j \le A \\ &\qquad\qquad\quad 0 \le \chi_j \le \alpha_j,\ \ j = 1, 2, \ldots, n; \end{aligned}$$

where $K' = \sum_{j=1}^{n} \nu_j + K$, $c_j = \beta_j / \alpha_j$ and $\gamma_j = \left(\nu_j - \alpha_j \beta_j - \mu_j\right) / \alpha_j$. We can reformulate the objective function in an almost separable form

$$\sum_{i=1}^{n} c_i \chi_i \sum_{j=1}^{i} \chi_j = \frac{1}{2} \sum_{i=1}^{n} c_i \chi_i^2 + \frac{1}{2} \sum_{i=1}^{n-1} (c_i - c_{i+1}) \left(\sum_{j=1}^{i} \chi_j \right)^2 + \frac{1}{2} c_n \left(\sum_{i=1}^{n} \chi_i \right)^2;$$

the proof of a similar equality can be found in Kellerer and Strusevich (2010b).

Introduce the network G with the set V of vertices and set E of arcs. Set V consists of a single source v_s, a single sink v_t, the vertices $w_n, w_{n-1}, \ldots, w_2$ and the vertices $t_n, t_{n-1}, \ldots, t_1$. Set E consists of the following arcs: (v_s, w_n) of capacity A, (w_j, t_j) of capacity α_j and (w_j, w_{j-1}) of capacity $\sum_{i=1}^{j-1} \alpha_i$ for $j = n, n-1, \ldots, 3$; (w_2, t_2) and (w_2, t_1) of capacity α_2 and α_1, respectively; besides, for each j, $1 \le j \le n$, vertex t_j is connected to the sink by the arc $\left(t_j, v_t\right)$ of capacity α_j. Let f be a flow on an arc, then the cost of that flow is defined as $\frac{1}{2} c_n f^2$ for arc (v_s, w_n), as $\frac{1}{2}\left(c_{j-1} - c_j\right) f^2$ for each arc (w_j, w_{j-1}) where $j = n, n-1, \ldots, 3$; as $\frac{1}{2} c_j f^2 - \gamma'_j f$ for the arc that enters vertex $t_j,\ j = 2, 3, \ldots, n$; as

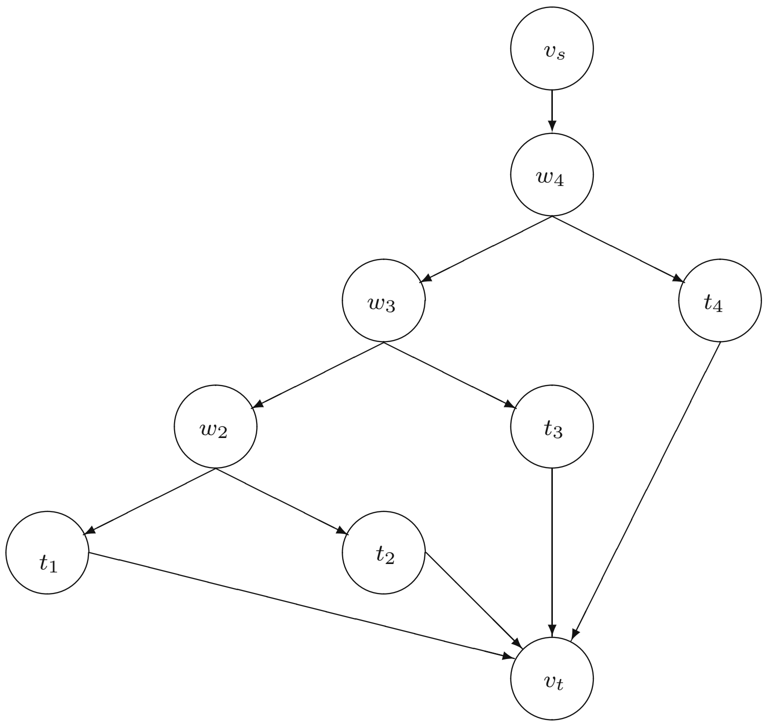

Fig. 1 A example of a series-parallel network for $n = 4$: a rooted tree with the leaves connected to the sink v_t

$(c_1 - \frac{1}{2}c_2) f^2 - \gamma_1' f$ for arc (w_2, t_1), while the cost of the flow on each arc that enters the sink is zero. It is clear that the minimum cost of the flow in the constructed network corresponds to the minimum value of $Z - K'$, while the flow on the arc that enters vertex t_j is equal to the corresponding value of the decision variable χ_j.

For illustration, consider the example below for $n = 4$. The network is shown in Fig. 1, while its parameters are given in Table 2.

Tamir (1993) presents an algorithm that minimizes a quadratic convex flow cost function on a series-parallel network with a single source and sink. In our case, network G satisfies the required condition, since it is a rooted tree with a single source and sink. Another point that makes our problem a special case of the one solved by Tamir is that his model is parametric, with the sum of all decision variables bounded by a running parameter that in his paper is denoted by q. For this parametric problem, each decision variable is defined as a piecewise-linear function of q. In our case, we only need an output of Tamir's algorithm for $q = A$, where A is either $\sum_{j \in N} \alpha_j$ (for Problem PosHPr) or the right-hand side of the knapsack constraint (for Problem PosHPKr).

In general, for a network with the set of vertices V and the set of arcs E, the running time of Tamir's algorithm is $O(|V||E| + |E| \log |E|)$, and it takes extra $O(\log |E|)$ time to output the solution for a particular value of q. Since for our network G, we have that $|V| = O(n)$ and $|E| = O(n)$, we conclude that the following statement holds.

Theorem 5 *Each Problem PosHPr and Problem PosHPKr with a convex objective function can be solved in $O(n^2)$ time.*

Table 2 Parameters of the network

Arc	Capacity	Cost for flow f
(v_s, w_4)	A	$\frac{1}{2}c_4 f^2$
(w_4, w_3)	$\alpha_1 + \alpha_2 + \alpha_3$	$\frac{1}{2}(c_3 - c_4) f^2$
(w_4, t_4)	α_4	$\frac{1}{2}c_4 f^2 - \gamma_4' f$
(w_3, w_2)	$\alpha_1 + \alpha_2$	$\frac{1}{2}(c_2 - c_3) f^2$
(w_3, t_3)	α_3	$\frac{1}{2}c_3 f^2 - \gamma_3' f$
(w_2, t_2)	α_2	$\frac{1}{2}c_2 f^2 - \gamma_2' f$
(w_2, t_1)	α_1	$(c_1 - \frac{1}{2}c_2) f^2 - \gamma_1' f$
(t_1, v_t)	α_1	0
(t_2, v_t)	α_2	0
(t_3, v_t)	α_3	0
(t_4, v_t)	α_4	0

Denote the vector that solves a Problem PosHPKr (or problem PosHPr) by $\mathbf{x}^C = \left(x_1^C, x_2^C, \ldots, x_n^C\right)$. Obviously, for each Problem PosHP and Problem PosHPK, the inequality $P\left(\mathbf{x}^C\right) \le P\left(\mathbf{x}^*\right)$ holds. The components of vector $\mathbf{x}^C$ can be appropriately rounded, so that for the resulting Boolean vector $\mathbf{x}^H = \left(x_1^H, x_2^H, \ldots, x_n^H\right)$, the inequality $P\left(\mathbf{x}^H\right) \le R\,P\left(\mathbf{x}^*\right)$ holds, where R is a constant. Notice that the actual value of R does not have be particularly small; what is needed a fast a fast constant-ratio rounding algorithm.

In the corresponding rounding algorithms the number $\lambda = \frac{1}{2}\sqrt{5} - \frac{1}{2} = 0.618\,03$ plays an important role. This number is the positive root of the equation $x^2 = 1 - x$. Notice that

$$\frac{1}{\lambda^2} = \frac{1}{1-\lambda} = \frac{3+\sqrt{5}}{2} = 2.618\cdots > 1.618\cdots = \frac{1}{\lambda}.$$

The algorithm below uses an approximation algorithm for a linear knapsack minimization problem. Consider the minimization linear knapsack problem with the set of items I.

$$\begin{array}{ll} \text{Minimize} & \sum_{j \in I} c_j y_j \\ \text{Subject to} & \sum_{j \in I} q_j y_j \ge Q \\ & y_j \in \{0, 1\}, \ j \in I. \end{array}$$

Let $\mathbf{y}^*$ be an optimal solution vector. Csirik et al. (1991) give an $O(n \log n)$ algorithm, which they call Algorithm GR, that finds a vector $\mathbf{y}^H$ such that

$$\sum_{j \in I} c_j y_j^H \le \frac{3}{2} \sum_{j \in I} c_j y_j^*.$$

There are other algorithms known for the problem that also provide a constant ratio; see, e.g., Güntzer and Jungnickel (2000).

Algorithm PosHPKConstR

Step 1 Input vector $\mathbf{x}^C = \left(x_1^C, x_2^C, \ldots, x_n^C\right)$ that solves Problem PosHPKr. Define $\lambda := \frac{1}{2}\sqrt{5} - \frac{1}{2}$.

Step 2 Define $N_1 := \left\{ j \in N | x_j^C \le \lambda \right\}$ and $N_2 := N \backslash N_1$.

Step 3 Introduce the following auxiliary linear knapsack problem

$$\begin{array}{ll}\text{Minimize} & \sum_{j\in N_2} v_j y_j \\ \text{Subject to} & \sum_{j\in N_2} \alpha_j y_j \geq \sum_{j\in N_2} \alpha_j - A \\ & y_j \in \{0,1\},\ j\in N_2.\end{array} \tag{36}$$

Run Algorithm GR by Csirik et al. (1991) to find a vector $\mathbf{y}^H$ with components y_j^H, $j \in N_2$, which delivers an approximate solution to problem (36).

Step 4 Output vector $\mathbf{x}^H$ with components $x_j^H = 0$ for $j \in N_1$ and $x_j^H = 1 - y_j^H$ for $j \in N_2$ and the value $Z(\mathbf{x}^H)$. Stop.

The following statement addresses the performance of the rounding algorithm.

Theorem 6 *Let $\mathbf{x}^*$ be a vector that delivers an optimal solution to Problem PosHPK with a convex objective function. Algorithm PosHPKConstR requires $O(n \log n)$ time and finds a vector $\mathbf{x}^H$ such that*

$$\frac{P(\mathbf{x}^H)}{P(\mathbf{x}^*)} \leq \frac{7+\sqrt{5}}{2} = 4.618\ldots$$

Algorithm PosHPKConstR can easily be simplified to handle Problem PosHP with no knapsack constraint. It suffices to skip Step 3 all together and set $x_j^H = 1$ for $j \in N_2$. The modified rounding algorithm requires only $O(n)$ time and finds a vector $\mathbf{x}^H$ such that

$$\frac{P(\mathbf{x}^H)}{P(\mathbf{x}^*)} \leq \frac{3+\sqrt{5}}{2} = 2.618\ldots$$

Thus, we can summarize the results reviewed Sects. 5.1 and 5.2 as the following statement.

Theorem 7 *Problem PosHP and Problem PosHPK with a convex objective of the form* (3) *admits an FPTAS that requires $O(n^2/\varepsilon)$ time.*

In the following section, we discuss the implications of Theorem 7 for various scheduling applications.

5.3 Scheduling Applications

In Sect. 3, several scheduling problems are shown to reduce to Problems PosHP and PosHPK. As demonstrated by Kellerer and Strusevich (2013) , each of these problems admits an FPTAS that runs in $O\left(n^2/\varepsilon\right)$ time. Notice that for all these problems, in an optimal schedule the jobs are sequenced in accordance with a certain permutation that is a obtained by a form of the WSPT rule (14), which is similar to (11). This fact implies convexity of the corresponding function, which is an assumption in Theorem 7.

Problem $1\left|p_j = \overline{p}_j - y_j\right| \sum_{j\in N} w_j C_j + \sum_{j\in N} v_j y_j$ formulated in Sect. 3.6 is an example of a scheduling problem for which a reformulation in the form $F(\mathbf{x}) = H(\mathbf{x}) + K$ is possible, but $|H(\mathbf{x}^*)/F(\mathbf{x}^*)|$ can be arbitrary large; see Janiak et al. (2005) and Sect. 4.1. In fact, exactly this scheduling problem has motivated Janiak et al. to introduce the positive half-product. By reducing problem $1\left|p_j = \overline{p}_j - y_j\right| \sum_{j\in N} w_j C_j + \sum_{j\in N} v_j y_j$ to Problem PosHP, Janiak et al. (2005) report an FPTAS that requires either $O\left(n^2 \log\left(\sum \overline{p}_j\right)/\varepsilon\right)$ or $O\left(n^2 \log\left(\sum w_j\right)/\varepsilon\right)$ time. Theorem 3 guarantees that the general FPTAS by Erel and Ghosh (2008) of the running time $O\left(n^2 \log\left(K\varepsilon\right)/\varepsilon\right)$ takes $O(n^2 \log\left(\max\left\{p_j, w_j, n\right\}\right)/\varepsilon)$ time, since here $K = \sum_{j=1}^n \overline{p}_j v_j \leq n \max\left\{\overline{p}_j^2, v_j^2\right\}$.

Recall that in a job sequence that is optimal for problem $1\left|p_j = \overline{p}_j - y_j\right| \sum_{j\in N} w_j C_j + \sum_{j\in N} v_j y_j$ some jobs will have zero processing times, while the other jobs will be sequenced in non-decreasing order of $\overline{p}_j / w_j$. This means that the problem reduces to Problem PosHP with a convex objective function, so that Theorem 7 applies and the problem admits an FPTAS that runs in $O\left(n^2/\varepsilon\right)$ time.

Problem $1\left|r_j \in \left[\underline{r}, \bar{r}\right]\right| C_{\max} + \sum_{j\in N} v_j y_j$ from Sect. 3.8 reduces to Problem PosHPK; see Shakhlevich and Strusevich (2006) and Sect. 3.8. Interpreting the results of Janiak et al. (2005), this implies that the problem admits an FPTAS that runs either in $O\left(n^2 \log\left(p(N)\right)/\varepsilon\right)$ or $O\left(n^2 \log\left(\sum v_j\right)/\varepsilon\right)$ time. Since here $K = p(N) + \overline{r}$, Theorem 3 guarantees that the general FPTAS by Erel and Ghosh (2008) will require $O(n^2 \log\left(p(N) + \overline{r}\right)/\varepsilon)$. The convexity of the objective function follows from the fact that in an optimal sequence the early jobs will be sequenced in non-increasing order of p_j/v_j. Due to Theorem 7, the problem admits an FPTAS that runs in $O\left(n^2/\varepsilon\right)$ time.

For problem $1\left|rej\right| \sum_{j\in N_A} w_j C_j + \sum_{j\in N_R} v_j$ from Sect. 3.7, Engles et al. (2003) present an FPTAS that requires $O\left(n^2 \log\left(\sum p_j\right)/\varepsilon\right)$ time. Their reasoning does not use a link between this problem and quadratic Boolean programming. As demonstrated in Kellerer and Strusevich (2013) (see also Sect. 3.7), the objective function problems $1\left|rej\right| \sum_{j\in N_A} w_j C_j + \sum_{j\in N_R} v_j$ and $1\left|rej, C_j \le d\right| \sum_{j\in N_A} w_j C_j + \sum_{j\in N_R} v_j$ is a positive half-product function of the form (3). In both problems, an optimal sequence of the accepted jobs is formed in accordance with (14), which implies that the objective function is convex. Due to Theorem 7, each problem $1\left|rej\right| \sum_{j\in N_A} w_j C_j + \sum_{j\in N_R} v_j$ and $1\left|rej, C_j \le d\right| \sum_{j\in N_A} w_j C_j + \sum_{j\in N_R} v_j$ admits an FPTAS that runs in $O\left(n^2/\varepsilon\right)$ time.

6 Symmetric quadratic knapsack: approximation and scheduling applications

In this section, we discuss an FPTAS that is capable of handling instances of Problem SQK that are related to scheduling applications listed in Sect. 3. Full technical details for the development of such an FPTAS are given in Kellerer and Strusevich (2010a, b). Below we only stress the differences between their approach and the approach to designing an FPTAS for Problem PosHPK outlined in Section 5.

6.1 Approximation scheme

To design an FPTAS for Problem SQK the same two prerequisites are needed as for Problem PosHPK: a DP algorithm and a constant-ratio approximation algorithm.

We start with DP algorithms for solving Problem SQK given in the form (6). These algorithms first appeared in Kellerer and Strusevich (2010a), and under some additional conditions they can be converted into an FPTAS.

Unlike for Problem PosHPK, here we need two versions of the DP algorithm. One of these versions is a minor modification of Algorithm DP1 from Sect. 5.1 and uses the same states of the form (k, Z_k, y_k). These states will be called the *primal* states, and the DP algorithm that manipulates the primal sates will be called the *primal* algorithm.

For our purposes, we also need another form of the DP algorithm that manipulates the states of the dual form $(k, Z_k, \widetilde{y}_k)$, where k and Z_k have the same meaning as above, while $\widetilde{y}_k = A_k - y_k$. It is clear that $\widetilde{y}_k$ is the total weight of the considered items that have not been put into the knapsack.

The primal DP algorithm, which we call Algorithm PDP can be easily deduced from Algorithm DP1. All what is needed is to modify Step 2 by replacing the recursive formula (34) by the formula

$$Z_{k+1} = Z_k + \beta_{k+1} y_k + \mu_{k+1}, \; y_{k+1} = y_k + \alpha_{k+1}, \tag{37}$$

and the recursive formula (35) by

$$Z_{k+1} = Z_k + \beta_{k+1} \left(A_k - y_k\right) + \nu_{k+1}; \; y_{k+1} = y_k. \tag{38}$$

The corresponding dual DP algorithm, which we will refer to as Algorithm DDP, also starts with the state $(0, K, 0)$, but manipulates the dual states. We skip its formal description, since is it very similar to that of Algorithm PDP. It suffices to say that in iteration k, given a state

$$\left(k, Z_k, \widetilde{y}_k\right),$$

Algorithm DDP transforms it into a state

$$\left(k+1, Z_{k+1}, \widetilde{y}_{k+1}\right),$$

where for $x_{k+1} = 1$ we define

$$Z_{k+1} = Z_k + \beta_{k+1}(A_k - \widetilde{y}_k) + \mu_{k+1}, \; \widetilde{y}_{k+1} = \widetilde{y}_k, \tag{39}$$

provided that $A_k - \widetilde{y}_k \le A$, while for $x_{k+1} = 0$ we define

$$Z_{k+1} = Z_k + \beta_{k+1}\widetilde{y}_k + \nu_{k+1}, \; \widetilde{y}_{k+1} = \widetilde{y}_k + \alpha_{k+1}. \tag{40}$$

Algorithm PDP and Algorithm DDP can be implemented efficiently to run in $O(nA)$ time. Let Z^* denote the optimal value of the objective function (6).

Assume now the second prerequisite has also been obtained, i.e., for Problem SQK, an upper bound Z^{UB} such that $Z^{UB}/Z^* \le \rho$ can be found in polynomial time, where ρ is a positive constant. As in Sect. 5, given Z^{UB}, we derive that

$$Z_{LB} = \frac{1}{\rho} Z^{UB}$$

is a lower bound on Z^*.

The FPTAS for Problem SQK appears to be more elaborate compared to the FPTAS for Problem PosHPK, where the latter is obtained by a rather straightforward conversion of a DP algorithm.

It is worth mentioning that Woeginger (2000) proves that if a dynamic programming algorithm for some optimization problem possesses a certain structure, then it can be converted into an FPTAS. Notice that the method of Woeginger is not applicable to Problem SQK. The reason is that variable y_k in the recursion for Z_{k+1} given in (38) has a negative coefficient. In particular, the objective function of Problem SQK is not what is called cc-benevolent in Woeginger (2000).

Below we explain that Condition C.1(i) of the scheme given in Woeginger (2000) is not satisfied. For the DP algorithm for Problem SQK we consider the state vectors $S = (s_1, s_2) = (Z_k, y_k)$ and $S' = (s_1', s_2') = (Z_k', y_k')$. Moreover, we are given a *degree-vector* $D = (d_1, d_2)$, with d_1, d_2 positive integers. For a real number $\Delta > 1$, vector S is said to be $[D, \Delta]$ *-close* to vector S', if

$$\Delta^{-d_\ell} \cdot s_\ell \le s_\ell' \le \Delta^{d_\ell} \cdot s_\ell, \quad \ell = 1, 2.$$

Condition C.1(i) implies that if S is $[D, \Delta]$-close to S' and $y'_k \leq y_k$, then also the inequalities

$$\Delta^{-d_1} Z_{k+1} \leq Z'_k + \beta_{k+1} \left(A_k - y'_k + \nu_{k+1} \right) \leq \Delta^{d_1} Z_{k+1} \tag{41}$$

hold for $Z_{k+1} = Z_k + \beta_{k+1} (A_k - y_k) + \nu_{k+1}$. Notice that the inequality $y'_k \leq y_k$ is due to the fact that y_k can be considered as critical coordinate.

Choosing $Z_k = Z'_k = \nu_{k+1} = 0$, the inequalities (41) reduce for $\ell = 1$ to $A_k - y'_k \leq \Delta^{d_1} (A_k - y_k)$ for $\Delta^{-d_1} \cdot y_k \leq y'_k \leq y_k$. It can be easily seen that this cannot be true if A_k is close enough to y_k. Thus, designing an FPTAS for Problem SKQ cannot be done within the standard scheme and requires special actions, as outlined below.

The FPTAS for Problem SKQ has been developed in Kellerer and Strusevich (2010a, b). It is based on both DP algorithms, the primal and the dual. To reduce the number of computed function values, we round the computed values up to a multiple of a chosen small number. To reduce the number of states stored after each iteration we split the range of possible $y-$values (and $\widetilde{y}-$values) into subintervals of a variable length and for each of the resulting subintervals we keep at most two $y-$values (and at most two $\widetilde{y}-$values) related to the same value of the function.

Algorithm EpsSQK

Step 1 Given an instance of Problem SQK, find an upper bound Z^{UB} on the optimal value Z^* of the objective function, such that $Z^{UB}/Z^* \leq \rho$.

Step 2 Given an arbitrary $\varepsilon > 0$, define $Z_{LB} = \frac{1}{\rho} Z^{UB}$ and

$$\delta = \frac{\varepsilon Z_{LB}}{2n}.$$

Step 3 Let there be $h \leq n$ distinct values among $\beta_j,\ j = 1, 2, \ldots, n$. Sort these values in decreasing order, i.e., determine a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(h))$ such that

$$\beta_{\pi(1)} > \beta_{\pi(2)} > \cdots > \beta_{\pi(h)}.$$

Split the interval $\left[0, \frac{Z^{UB}}{\beta_{\pi(h)}}\right]$ into h intervals

$$I_1 = \left[0, \frac{Z^{UB}}{\beta_{\pi(1)}}\right], I_2 = \left[\frac{Z^{UB}}{\beta_{\pi(1)}}, \frac{Z^{UB}}{\beta_{\pi(2)}}\right], \ldots, I_h = \left[\frac{Z^{UB}}{\beta_{\pi(h-1)}}, \frac{Z^{UB}}{\beta_{\pi(h)}}\right].$$

Additionally, split each interval I_j into subintervals I_j^r of length $\delta/\beta_{\pi(j)}$ (it may turn out that the last of the subintervals of an interval I_j is strictly shorter than $\delta/\beta_{\pi(j)}$).

Step 4 Store the initial state $(0, K, 0)$. For each k, $0 \leq k \leq n-1$, do the following:

(a) According to Algorithm PDP, move from a stored primal state (k, Z_k, y_k) to at most two primal states of the form $(k+1, Z_{k+1}, y_{k+1})$, where $Z_{k+1} \leq Z^{UB}$, using the relations (37) and (38), each time rounding up the updated value of Z_{k+1} to the next multiple of δ. For each selection of states with the same value of Z_{k+1} and a subinterval I_j^r, determine the value $y_{k+1}^{\min}$ as the smallest value of y_{k+1} that belongs to I_j^r and the value $y_{k+1}^{\max}$ as the largest value of y_{k+1} that belongs to I_j^r. If these values exist and are distinct, then out of all states $(k+1, Z_{k+1}, y_{k+1})$ with the same value of Z_{k+1} for $y_{k+1} \in \left[y_{k+1}^{\min}, y_{k+1}^{\max}\right]$ store only two states $(k+1, Z_{k+1}, y_{k+1}^{\min})$ and $(k+1, Z_{k+1}, y_{k+1}^{\max})$.

(b) According to Algorithm DDP, move from a stored dual state $(k, Z_k, \widetilde{y}_k)$ to at most two dual states of the form $(k+1, Z_{k+1}, \widetilde{y}_{k+1})$, where $Z_{k+1} \leq Z^{UB}$, using the relations (39) and (40), each time rounding up the updated value of Z_{k+1} to the next multiple of δ. For each selection of states with the same value of Z_{k+1} and a subinterval I_j^r, determine the value $\widetilde{y}_{k+1}^{\min}$ as the smallest value of $\widetilde{y}_{k+1}$ that belongs to I_j^r and the value $\widetilde{y}_{k+1}^{\max}$ as the largest value of $\widetilde{y}_{k+1}$ that belongs to I_j^r. If these values exist and are distinct then out of all states $(k+1, Z_{k+1}, \widetilde{y}_{k+1})$ with the same value of Z_{k+1} for $\widetilde{y}_{k+1} \in \left[\widetilde{y}_{k+1}^{\min}, \widetilde{y}_{k+1}^{\max}\right]$ store only two states $(k+1, Z_{k+1}, \widetilde{y}_{k+1}^{\min})$ and $(k+1, Z_{k+1}, \widetilde{y}_{k+1}^{\max})$.

(c) For each primal state $(k+1, Z_{k+1}, y_{k+1})$ stored in Step 4(a) of this iteration additionally store the dual state $(k+1, Z_{k+1}, \widetilde{y}_{k+1})$, where $\widetilde{y}_{k+1} = A_{k+1} - y_{k+1}$, unless it coincides with one of the states stored in Step 4(b) of this iteration.

(d) For each dual state $(k+1, Z_{k+1}, \widetilde{y}_{k+1})$ stored in Step 4(b) of this iteration additionally store the primal state $(k+1, Z_{k+1}, y_{k+1})$, where $y_{k+1} = A_{k+1} - \widetilde{y}_{k+1}$, unless it coincides with one of the states stored in Step 4(a) of this iteration.

Step 4 Among all values Z_n found in Step 4 identify the smallest one associated with a feasible value $y_n \leq A$. With this value of Z_n, perform the backtracking to find the corresponding decision variables $x_j,\ j = 1, \ldots, n$. Compute the value of the objective function with the found x_j's, call this value Z^ε and accept it as an approximate value of the objective function.

To understand better the role of the intervals created in Step 3 of Algorithm EpsSQK, for each $k,\ 0 \leq k \leq n-1$, define

$$B(k) := \max\{\beta_{k+1}, \beta_{k+2}, \ldots, \beta_n\}. \tag{42}$$

The following statement holds.

Lemma 2 [Kellerer and Strusevich (2010a)] *Assume that the primal dynamic programming Algorithm PDP is applied to Problem SQK and finds a chain of states*

$$(0, K, 0), (1, Z_1^*, y_1^*), \ldots, (n, Z_n^*, y_n^*)$$

leading to the optimal value $Z^* = Z_n^*$. Then for each $k,\ 0 \leq k \leq n-1$, for $B(k)$ defined by (42) either

$$B(k)y_k^* \leq Z^*,$$

or

$$B(k)\widetilde{y}_k^* \leq Z^*,$$

where $\widetilde{y}_k^* = A_k - y_k^*$.

The following statement studies the behavior of Step 4 of Algorithm EpsSQK.

Lemma 3 [Kellerer and Strusevich (2010a)] *Assume that the primal dynamic programming Algorithm PDP is applied to Problem SQK and finds a chain of primal states*

$$(0, K, 0), (1, Z_1^*, y_1^*), \ldots, (n, Z_n^*, y_n^*)$$

leading to the optimal value $Z^* = Z_n^*$. Let

$$(0, K, 0), (1, Z_1^*, \widetilde{y}_1^*), \ldots, (n, Z_n^*, \widetilde{y}_n^*)$$

be the corresponding chain of dual states. Then for each k, $1 \leq k \leq n$, Algorithm EpsSQK finds *(i) a pair of primal states* (k, Z'_k, y'_k) *and* (k, Z''_k, y''_k) *such that*

$$Z'_k \le Z^*_k + 2k\delta;\ \ Z''_k \le Z^*_k + 2k\delta \tag{43}$$

and

$$y'_k \le y^*_k \le y''_k,\ y''_k - y^*_k \le \frac{\delta}{B(k)},\ y^*_k - y'_k \le \frac{\delta}{B(k)};$$

(ii) a pair of dual states $(k, Z'_k, \tilde{y}'_k)$ *and* $(k, Z''_k, \tilde{y}''_k)$ *such that* (43) *holds and*

$$\tilde{y}'_k \le \tilde{y}^*_k \le \tilde{y}''_k,\ \tilde{y}''_k - \tilde{y}^*_k \le \frac{\delta}{B(k)},\ \tilde{y}^*_k - \tilde{y}'_k \le \frac{\delta}{B(k)},$$

where $B(k)$ *is defined by* (42).

Based on Lemmas 2 and 3, the main theorem can be proved.

Theorem 8 [Kellerer and Strusevich (2010a)] *Let* Z^* *denote an optimal value of the objective function for Problem SQK. Given a positive* ε*, Algorithm EpsSQK outputs a value* Z^ε *such that*

$$Z^\varepsilon - Z^* \le \varepsilon Z^*$$

and requires $O\left(T(n) + \frac{n^4}{\varepsilon^2}\right)$ *time, where* $T(n)$ *denotes the time needed for finding an upper bound* Z^{UB} *such that* $Z^{UB}/Z^* \le \rho$.

For scheduling applications a version of Problem SQK in which all $\beta_j = 1$ can be of certain importance (this corresponds to the case of equal weights of jobs). In this case Algorithm EpsSQK is less time-consuming. In Step 3 of the algorithm it is sufficient to split the interval $[0, Z^{UB}]$ into subintervals of equal length δ. Thus, in each iteration for at most Z^{UB}/δ values of the objective function and each of Z^{UB}/δ subintervals we store at most two states, i.e., the total number of states created and stored in each iteration is $O\left(\left(\frac{Z^{UB}}{\delta}\right)^2\right)$, i.e., factor n less than in the general case of arbitrary β_j. This reduces the overall running time of Algorithm EpsSQK for the case that all $\beta_j = 1$ to $O\left(T(n) + \frac{n^3}{\varepsilon^2}\right)$.

The running time estimate in Theorem 8 is valid, provided that the second prerequisite is available, i.e., a polynomial-time algorithm exists that finds a heuristic solution with the objective function value Z^{UB}, which does not exceed ρZ^* for a constant ρ. Notice that Algorithm EpsSQK remains an FPTAS if the ratio ρ in $Z^{UB}/Z^* \le \rho$ is not a constant but polynomially depends on n . More precisely, if $\rho = O(n^c)$ for a positive c, then the running time of Algorithm EpsSQK is $O\left(T(n) + \frac{n^{2c+4}}{\varepsilon^2}\right)$.

Below, we present a constant-ratio approximation algorithm for Problem SQK under specific additional conditions. As in Sect. 4, one of these assumptions is the convexity of the objective function, which can be guaranteed by the numbering of items given by (11). The algorithm below is based on rounding a solution to a continuous relaxation of Problem SQK.

Introduce Problem SQKr, a continuous relaxation of Problem SQK, obtained by replacing the integrality condition $x_j \in \{0, 1\}$ by $0 \le x_j \le 1$ for each $j = 1, 2, \dots, n$. Using transformations similar to those described in Sect. 5.2 for Problem PosHP, we can rewrite (6) as

$$\begin{aligned} &\text{Minimize } Z = 2\sum_{i=1}^{n} c_i \chi_i \sum_{j=1}^{i} \chi_j - \textstyle\sum_{j=1}^{n} \gamma'_j \chi_j + K' \\ &\text{subject to } \textstyle\sum_{j=1}^{n} \chi_j \le A \\ &\qquad\qquad 0 \le \chi_j \le \alpha_j,\ j = 1, 2, \dots, n, \end{aligned}$$

where $c_j = \beta_j/\alpha_j$, $\gamma'_j = \gamma_j/\alpha_j$ and $\chi_j = \alpha_j x_j$, $j = 1, 2, \ldots, n$. In turn,

$$2\sum_{i=1}^{n} c_i \chi_i \sum_{j=1}^{i} \chi_j = \sum_{i=1}^{n} c_i \chi_i^2 + \sum_{i=1}^{n-1} (c_i - c_{i+1}) \left(\sum_{j=1}^{i} \chi_j\right)^2 + c_n \left(\sum_{i=1}^{n} \chi_i\right)^2,$$

so that, as in Sect. 5.2, Problem SQKr can be solved by an algorithm by Tamir (1993) in $O\left(n^2\right)$ time.

Kellerer and Strusevich (2010b) present an algorithm that under an additional condition behaves as a constant-ratio approximation algorithm for the original Problem SQK with Boolean decision variables. The algorithm is based on an appropriate rounding of the solution to the continuous relaxation. In the analysis of the algorithm the value of the objective function for the found optimal solution to Problem SQKr serves as a lower bound on the optimal value for the original Problem SQK.

Informally, the algorithm works as follows. Start with a solution to Problem SQKr and round down to zero those components that are less than a specially chosen value. To determine the other variables, a continuous linear knapsack problem has to be solved to obtain a solution with at most one fractional component. Then the value for that component is fixed to 1, thereby reducing the dimension of the problem, and the process is repeated.

Algorithm Round

Step 1 Define Problem P_n as the given Problem SQK of the form (6). Set $H := N$, $F := \varnothing$ and $h := n$.

Step 2 Solve the continuous relaxation of problem P_h and find the corresponding solution vector with components x_j^{CR}, where $j \in H$.

Step 3 Define the sets $I_1 := \{j \in H | x_j^{CR} \le \eta\}$ and $I_2 := \{j \in H | x_j^{CR} > \eta\}$, where η is chosen to be the smaller root of the equation $\eta = (1-\eta)^2$, i.e., $\eta = \frac{3-\sqrt{5}}{2} \approx 0.381\,97$.

Step 4 Set $x_j^{(h)} := 1$ for $j \in F$ and $x_j^{(h)} := 0$ for $j \in I_1$.

Step 5 For each $j \in I_2$, define $\pi_j := \nu_j + \beta_j \sum_{i<j, i \in I_1} \alpha_i + \alpha_j \sum_{i>j, i \in I_1} \beta_i$. Introduce the following auxiliary continuous linear knapsack problem

$$\begin{aligned} &\text{Minimize } \textstyle\sum_{j \in I_2} \pi_j (1 - x_j) \\ &\text{subject to } \textstyle\sum_{j \in I_2} \alpha_j x_j \le A \\ &\qquad\qquad 0 \le x_j \le 1,\ j \in I_2 \end{aligned}$$

and solve it in the following greedy way.

(a) Denote $u = |I_2|$. Find a sequence $\sigma = (\sigma(1), \ldots, \sigma(u))$ of items of set I_2 sorted in non-increasing order of the cost-weight ratios π_j/α_j.

(b) Find the index ℓ such that

$$\sum_{k=1}^{\ell-1} \alpha_{\sigma(k)} \le A; \ \sum_{k=1}^{\ell} \alpha_{\sigma(k)} > A.$$

An optimal solution to the auxiliary problem above is determined by a vector with components $x_{\sigma(k)}^{\Delta} = 1$ for $1 \le k < \ell$, $x_{\sigma(k)}^{\Delta} = 0$ for $\ell + 1 \le k \le u$, while the value $x_{\sigma(\ell)}^{\Delta}$ is chosen to satisfy

$$\sum_{k=1}^{\ell-1} \alpha_{\sigma(k)} + \alpha_{\sigma(\ell)} x_{\sigma(\ell)}^{\Delta} = A.$$

Step 6 Set $x_j^{(h)} := 0$ for $j \in I_1$. Define $x_j^{(h)} := x_j^{\Delta}$, $j \in I_2$, $j \neq \sigma(\ell)$ and set $x_{\sigma(\ell)}^{(h)} := \left\lfloor x_{\sigma(\ell)}^{\Delta} \right\rfloor$. Compute the value of the objective function Z of the original Problem SQK with $x_j = x_j^{(h)}$, $j \in N$; store this value as $Z^{(h)}$.

Step 7 If either $x_{\sigma(\ell)}^{(h)} = 1$ or $h = 1$, go to Step 9; otherwise, go to Step 8.

Step 8 Update $H := H \backslash \{\sigma(\ell)\}$, $F := F \cup \{\sigma(\ell)\}$. In problem P_h, replace variable $x_{\sigma(\ell)}$ by 1 and call the resulting problem P_{h-1} . To derive problem P_{h-1} from problem P_h, we set $A := A - \alpha_{\sigma(\ell)}$; keeping the variables other than $x_{\sigma(\ell)}$ in the same order as in problem P_h, renumber them by $1, \dots, h-1$; set $h := h-1$ and define $\mu_i := \mu_i + \alpha_i \sum_{j=i+1}^{h} \beta_j$ for $i = 1, \dots, h$. Go to Step 2.

Step 9 Output the solution that corresponds to the smallest of the found values $Z^{(h)}$.

The following statement holds.

Theorem 9 [Kellerer and Strusevich (2010b)] *For Problem SQK of the form* (6) *for which* (11) *holds and additionally*

$$\nu_j \geq \alpha_j \beta_j, \ j \in N \tag{44}$$

Algorithm Round requires $O(n^3)$ *time and is an approximation algorithm with a ratio of at most* $2 + \frac{3}{\eta} = \frac{3\sqrt{5}+13}{2} \approx 9.8541$.

To summarize our discussion of this section, we formulate the main statement regarding a possibility of solving Problem SQK by an FPTAS.

Theorem 10 *For Problem SQK of the form* (6) *for which the conditions* (11) *and* (44) *hold, Algorithm EpsSQK is an FPTAS that requires* $O\left(\frac{n^4}{\varepsilon^2}\right)$ *time.*

Notice that the additional conditions (11) and (44) hold for all known scheduling applications of Problem SQK.

6.2 Alternative FPTAS

In this subsection, we describe an approach by Xu (2012) who shows that Algorithm EpsSQK can be formulated as a strongly polynomial FPTAS without additional assumptions, such as convexity. Recall that due to Theorem 8 the running time of EpsSQK is in $O\left(T(n) + \frac{n^4}{\varepsilon^2}\right)$, where $T(n)$ denotes the time needed for finding an upper bound Z^{UB} such that $Z^{UB}/Z^* \leq \rho$. If Z^{UB}/Z^* is not bounded by a constant ρ, the running time is given by $O\left(T(n) + \frac{Z^{UB}}{Z^{LB}} \frac{n^4}{\varepsilon^2}\right)$. Hence for obtaining a strongly polynomial FPTAS, it is sufficient to develop a strongly polynomial time algorithm for computing a lower bound Z^{LB} and an upper bound Z^{UB} on the optimal objective value of problem SQK, such that the ratio of the bounds is bounded by a polynomial in the size of the instance.

Let Λ denote the set which contains zero and all possible coefficients of the objective function of Problem SQK given in the form (6), i.e.,

$$\Lambda = \{0\} \cup \{\alpha_i \beta_j | 1 \leq i < j \leq n\} \cup \{\mu_j | 1 \leq j \leq n\} \cup \{\nu_j | 1 \leq j \leq n\}.$$

Define λ^* as the minimum value of $\lambda \in \Lambda$ such that there is a Boolean vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ for which the following inequalities hold.

$$\alpha_i \beta_j x_i x_j \le \lambda, \quad 1 \le i < j \le n, \tag{45}$$
$$\alpha_i \beta_j (1 - x_i)(1 - x_j) \le \lambda, \quad 1 \le i < j \le n, \tag{46}$$
$$\mu_j x_j \le \lambda, \quad 1 \le j \le n, \tag{47}$$
$$\nu_j \left(1 - x_j\right) \le \lambda, \quad 1 \le j \le n, \tag{48}$$
$$\sum_{j=1}^{n} \alpha_j x_j \le A. \tag{49}$$

Set $\lambda_{\max} = \max\{\lambda | \lambda \in \Lambda\}$. For $\lambda = \lambda_{\max}$ the zero vector satisfies all inequalities (45)–(49). Hence, Λ is non-empty and λ^* is well-defined.

Consider the optimal solution vector $\mathbf{x}^*$ to Problem SQK. Let λ' denote the maximum among the values of $\alpha_i \beta_j x_i^* x_j^*$, $\alpha_i \beta_j (1 - x_i^*)(1 - x_j^*)$ for $1 \le i < j \le n$, and $\mu_j x_j^*$, $\nu_j \left(1 - x_j^*\right)$ for $1 \le j \le n$. Obviously, $\mathbf{x}^*$ satisfies (45)–(49) for $\lambda = \lambda'$. Since $0 \in \Lambda$ we get $\lambda^* + K \le Z^*$.

Consider now any Boolean vector $\mathbf{x}$ that satisfies (45)–(49) for $\lambda = \lambda^*$. Due to (49), $\mathbf{x}$ is a feasible solution to Problem SQK. We have

$$\sum_{1 \le i < j \le n} \alpha_i \beta_j x_i x_j + \sum_{1 \le i < j \le n} \alpha_i \beta_j (1 - x_i)(1 - x_j)$$
$$\le \lambda^* \sum_{1 \le i < j \le n} \left(x_i x_j + (1 - x_i)(1 - x_j)\right) \le \frac{n(n-1)}{2} \lambda^*,$$

and

$$\sum_{j=1}^{n} \mu_j x_j + \sum_{j=1}^{n} \nu_j (1 - x_j) \le \lambda^* \sum_{j=1}^{n} \left(x_j + (1 - x_j)\right) = n\lambda^*.$$

These two inequalities imply $Z^* \le \left(\frac{n(n-1)}{2} + n\right) \lambda^* + K$. Hence, the following statement holds.

Lemma 4 *For Problem SQK, the values $Z^{LB} = \lambda^* + K$ and $Z^{UB} = \left(\frac{n(n-1)}{2} + n\right) \lambda^* + K$ are a lower bound and an upper bound on the optimal value Z^* of the objective function, respectively.*

In order to compute the lower and upper bounds on Z^* suggested by Lemma 4, we need to compute λ^*. If we can test in reasonable running time for a given λ whether it is *feasible*, i.e. whether there is an $\mathbf{x}$ which satisfies (45)–(49), then the value λ^* can be found by using binary search on the values of Λ.

First, we establish a sufficient and necessary condition for a Boolean vector $\mathbf{x}$ to satisfy (45) and (46). Define a graph $G = (V, E)$ with vertices $V = \{1, \dots, n\}$. There is an edge between i and j if $\alpha_i \beta_j > \lambda$. Since (45) and (46) are equivalent to $x_i + x_j = 1$, $\mathbf{x}$ satisfies (45) and (46) if and only if $\mathbf{x}$ corresponds to a 2-coloring of the vertices of G with colors zero and one. But $\mathbf{x}$ corresponds to a 2-coloring if and only if there is a bipartition of G with the vertices with $x_j = 0$ in one set and the vertices with $x_j = 1$ in the other set. For each connected component $G' = (V', E')$ of G the bipartition (V_1', V_2') of V' is unique. Thus, either (i) $x_j = 1$ for $j \in V_1'$ and $x_j = 0$ for $j \in V_2'$ or (ii) $x_j = 0$ for $j \in V_1'$ and $x_j = 1$ for $j \in V_2'$, must hold.

Hence, the procedure is as follows: Compute the connected components of G. For each connected component G' of G check whether there exists a bipartition of G'. Then, examine whether at least one the two possibilities (i) and (ii) satisfy (47) and (48). If both possibilities satisfy (47) and (48), then choose the one with smaller value of $\sum_{j \in V'} \alpha_j x_j$. This guarantees that $\sum_{j=1}^{n} \alpha_j x_j$ is minimized, i.e., we find values x_j which satisfy (49) if they exist.

The running time of this algorithm is dominated by the sorting of the $O(n^2)$ elements of Λ. This gives $T(n) = O(n^2 \log n)$. Since by Lemma 4, $Z^{UB}/Z^{LB} \leq n^2$, we get a running time of $O(\frac{n^6}{\varepsilon^2})$ for Algorithm EpsSQK. By applying the Bound Improvement Procedure of Tanaev et al. (1998) to Z^{LB} and Z^{UB}, it is possible to find new lower and upper bounds Z^{LB_1} and Z^{UB_1} with $\frac{Z^{UB_1}}{Z^{LB_1}} \leq 3$. The total running time of the Bound Improvement Procedure is $O(n^4 \log \log n)$. Thus, the following statement holds.

Theorem 11 [Xu (2012)] *There is an FPTAS for Problem SQK which requires* $O\left(n^4 \log \log n + \frac{n^4}{\varepsilon^2}\right)$ *time.*

6.3 Scheduling applications

In this subsection, we review approximation results for several scheduling problems listed in Section 3. This will include known constant-ratio approximation algorithms, FPTASs derived by adapting an FPTAS designed for Problem SQK, as well as FPTASs that are developed based on alternative principles.

6.3.1 Scheduling with machine availability constraints

Kellerer and Strusevich (2010a) show how the FPTAS outlined in Sect. 6.1 can be adapted to the problems $1|h(1), N-res|\sum w_j C_j$ and $1|h(1), Res|\sum w_j C_j$ of minimizing the total weighted completion time on a single machine with a single machine non-availability interval. Both problems can be reformulated in terms of Problem SQK; see Sect. 3.1.

Recall that in order to be able to apply the FPTAS, for each problem we need a polynomial time approximation algorithm with a constant ratio. These problems admit the algorithms with a required performance developed by pure scheduling reasoning.

For example, for problem $1|h(1), Res|\sum w_j C_j$ under the resumable scenario Wang et al. (2005) give a 2−approximation algorithm that requires $O(n^2)$ time. An improved algorithm for this problem is due to Megow and Verschae (2009). The authors notice that an algorithm that sequences the jobs in the WSPT order is a $\delta-$approximation algorithm, where $\delta = t/s$, the ratio of the two endpoints of the non-availability interval. On the other hand, they show that if the jobs that are sequenced to complete before the non-availability interval by a known FPTAS, the resulting algorithm behaves as a $\left(1 + \frac{1}{\delta} + \varepsilon\right) -$ approximation algorithm. Thus, the value of the objective function for the better of the two schedules is at most $\Phi + \varepsilon$ times the optimum, where $\Phi = \left(1 + \sqrt{5}\right)/2$ is the so-called golden ratio, i.e., the larger solution to the equation $\delta = 1 + 1/\delta$. The running time of the algorithm is $O(n \log n + n/\varepsilon)$.

For problem $1|h(1), N-res|\sum w_j C_j$, Lee (1996) shows that scheduling the jobs in the WSPT order leads to an arbitrarily bad schedule. This is also confirmed by Kacem and Chu (2008) for a modified WSPT rule. Kellerer et al. (2009) show that based on a $\rho-$approximation algorithm for problem $1|h(1), Res|\sum w_j C_j$ it is possible to design a $(2\rho)-$approximation algorithm for problem $1|h(1), N-res|\sum w_j C_j$. They also use a formulation of problem $1|h(1), N-res|\sum w_j C_j$ as Problem SQK and an FPTAS for a linear knapsack problem to obtain a $(2+\varepsilon)-$approximation algorithm for the problem that runs in $O(n \log n + n/\varepsilon)$ time. The same performance can be achieved by applying the approach of Megow and Verschae (2009) not to the resumable but to the non-resumable version of the problem. Besides, a 2−approximation algorithm that requires $O(n^2)$ time is developed by Kacem (2008).

Recall that the unweighted problem $1|h(1), N-res|\sum C_j$ is NP-hard. Lee and Liman (1992) by refining the analysis by Adiri et al. (1989) demonstrate that for problem $1|h(1), N-res|\sum C_j$ an algorithm that finds schedule S_{SPT} in which the jobs are sequenced in the SPT order (i.e., in non-decreasing order of their processing times) is a $(9/7)$ −approximation algorithm. An algorithm with a worst-case performance ratio of $20/17$ is given by Sadfi et al. (2005). A further improvement is done by Breit (2007).

The algorithms with constant performance guarantees are needed to compute an upper bound on the optimal value of the function for the problem under consideration. It is not important which particular approximation algorithm is taken for this purpose, as long as it guarantees a constant ratio. Let us agree that for problem $1|h(1), N-res|\sum w_j C_j$ we take $Z^{UB} = Z(S_H)$, where S_H is a heuristic schedule found by a 2− approximation algorithm by Kacem (2008); for problem $1|h(1), N-res|\sum C_j$ we take $Z^{UB} = Z(S_{SPT})$ and for problem $1|h(1), Res|\sum w_j C_j$ we take $Z^{UB} = Z(S_W)$, where S_W is a heuristic schedule found by a 2−approximation algorithm by Wang et al. (2005).

We are now ready to apply Algorithm EpsSQK to obtain an FPTAS for problem $1|h(1), N-res|\sum w_j C_j$. The DP algorithms are applied to Problem SQK of the form (19). It follows that $Z_{LB} = \frac{1}{2} Z^{UB}$. Suppose that Algorithm EpsSQK finishes, having found a solution vector $x^\varepsilon = \left(x_1^\varepsilon, \dots, x_n^\varepsilon\right)$ where each $x_j^\varepsilon \in \{0, 1\}$. As an approximate solution of problem $1|h(1), N-res|\sum w_j C_j$ we take schedule S^ε in which the jobs of set $N_1 = \left\{j \in N | x_j^\varepsilon = 1\right\}$ are sequenced from time zero according to the WSPT rule and complete before time s, while the remaining jobs of set $N_2 = \left\{j \in N | x_j^\varepsilon = 0\right\}$ are sequenced from time t according to the WSPT rule. It follows that $Z(S^\varepsilon)/Z(S^*) \le 1+\varepsilon$ and the running time of the resulting algorithm is $O(n^2 + n^4/\varepsilon^2) = O(n^4/\varepsilon^2)$.

Notice that for problem $1|h(1), N-res|\sum C_j$ with equal weights of jobs minimizing $\sum C_j$ is equivalent to solving Problem SQK with all $\beta_j = 1$. The running time of the resulting FPTAS for problem $1|h(1), N-res|\sum C_j$ is $O(n^3/\varepsilon^2)$.

Kacem and Mahjoub (2009) claim that their scheme for problem $1|h(1), N-res|\sum w_j C_j$ is an FPTAS that requires $O\left(n^2/\varepsilon^2\right)$ time. However, Epstein et al. (2010) express some doubt whether the analysis by Kacem and Mahjoub (2009) is complete and give their own FPTAS with the running time $O\left(n^3/\varepsilon^2\right)$. Marchetti-Spaccamela et al. (2008) outline another FPTAS for this problem, but they do not give all implementation details and do not estimate the running time.

In order to design an FPTAS for problem $1|h(1), Res|\sum w_j C_j$, an approach presented in Sect. 3.1 is used. We can select a job as crossover, find a schedule S_m for $m = n-1$ remaining jobs and then insert the chosen job into the derived schedule. Suppose that a job with the processing time p and the weight w is chosen as the crossover job, and the remaining jobs are taken in accordance with the WSPT rule (14) and renumbered by $1, 2, \dots, m$. Finding schedule S_m can be done by adapting a DP algorithms from Sect. 6.1. Define

$$n = m,\ A = s,$$

$$\alpha_j = p_j,\ \beta_j = w_j,\ \mu_j = w_j p_j,\ \nu_j = w_j \left(t + \sum_{i=1}^{j} p_i\right),\ j = 1, 2, \dots, m.$$

For the primal algorithm, a typical primal state after the values $x_1, x_2, \dots, x_k$ have been assigned is represented by a state of the form

$$(k, Z_k, y_k, V_k),$$

where k is the number of the assigned variables; Z_k is the current value of the objective function; $y_k := \sum_{j=1}^{k} p_j x_j$, the total processing time of the jobs processed before the non-availability interval $[s, t]$; $V_k := p \sum_{j=1}^{k} w_j (1 - x_j)$, the total weight of the jobs processed after the interval $[s, t]$ times the processing time of the crossover job.

In iteration k of the primal algorithm a move from a state (k, Z_k, y_k, V_k) to a state $(k+1, Z_{k+1}, y_{k+1}, V_{k+1})$ is done in accordance with

$$Z_{k+1} = Z_k + w_{k+1} y_k + \mu_{k+1},\ \ y_{k+1} = y_k + p_{k+1},\ \ V_{k+1} = V_k, \tag{50}$$

provided that $y_{k+1} \le s$, and with

$$Z_{k+1} = Z_k + w_{k+1}\left(A_k - y_k\right) + \nu_{k+1};\ \ y_{k+1} = y_k, V_{k+1} = V_k + w_{k+1} p, \tag{51}$$

where $A_k = \sum_{j=1}^{k} p_j$.

For each state generated in the last iteration of the primal algorithm, we find x and compute the value of the function Z by the formula (20). The smallest of the found $Z-$values corresponds to an optimal value of the total weighted completion time for problem $1|h(1), Res|\sum w_j C_j$ with a chosen crossover job. If the smallest value of $Z(p)$ is achieved for $x = 1$, then the corresponding schedule should be disregarded, since the overall optimal schedule belongs to a class of schedules with another crossover job.

The DP algorithm outlined above is used as a subroutine in the FPTAS. No removal of dominated states is allowed in this DP algorithm, i.e., all feasible combinations should be generated and stored, except those states (k, Z_k, y_k, V_k) for which either Z_k or V_k exceeds Z^{UB}. For a fixed crossover job such a DP algorithm requires $O\left(nA\left(Z^{UB}\right)^2\right)$ time, i.e., an overall optimal solution will be found in $O\left(n^2 A\left(Z^{UB}\right)^2\right)$ time. The dual version of the DP algorithm can be defined similarly.

Let S^* denote the optimal schedule that delivers the smallest value $Z(S^*)$ of the objective function, while $S(p)$ denote a feasible schedule with a fixed crossover job with the processing time p and weight w. Denote the smallest value of the function among all schedules with the chosen crossover job by $Z^*(p)$.

Define $Z^{UB} = Z(S_W)$ as an upper bound on $Z(S^*)$, where S_W is a schedule delivered by the algorithm by Wang et al. (2005). It follows that $Z_{LB} = \frac{1}{2} Z^{UB}$ is a lower bound on $Z(S^*)$, and therefore on $Z^*(p)$. In Step 2 of Algorithm EpsSQK redefine

$$\delta = \frac{\varepsilon Z_{LB}}{3m}.$$

Change Step 4(a) of the algorithm in the following way. Move from a stored primal state (k, Z_k, y_k, V_k) to at most two primal states of the form $(k+1, Z_{k+1}, y_{k+1}, V_{k+1})$, where $Z_{k+1} \le Z^{UB}$ and $V_{k+1} \le Z^{UB}$, using the relations (50) and (51), each time rounding up each of the updated values of Z_{k+1} and V_{k+1} to the next multiple of δ. For each selection of states related to the same pair (Z_{k+1}, V_{k+1}) and a subinterval I_j^r, determine the value $y_{k+1}^{\min}$ as the smallest value of y_{k+1} that belongs to I_j^r and the value $y_{k+1}^{\max}$ as the largest value of y_{k+1} that belongs to I_j^r. If these values exist and are distinct, then out of all states $(k+1, Z_{k+1}, y_{k+1}, V_{k+1})$ with the same values of $Z_{k+1},\ V_{k+1}$ for $y_{k+1} \in \left[y_{k+1}^{\min}, y_{k+1}^{\max}\right]$ store only two states $(k+1, Z_{k+1}, y_{k+1}^{\min}, V_{k+1})$ and $(k+1, Z_{k+1}, y_{k+1}^{\max}, V_{k+1})$.

The description of Step 4(b) of Algorithm EpsSQK is altered similarly; here we manipulate the corresponding dual states. Step 5 is reformulated accordingly. The chosen crossover job is inserted into a schedule found by the modified FPTAS, and the best of all found schedules is accepted as an approximate solution.

For each choice of the crossover job, running a modified FPTAS that finds the values Z_m takes $O\left(n^5/\varepsilon^3\right)$ time. To find an approximate solution to the original problem $1|h(1), Res|\sum w_j C_j$, we need to run the FPTAS for each job selected as the crossover job, rejecting the solutions in which the selected job completes before the non-availability interval. Thus, the overall running time of the FPTAS is $O\left(n^6/\varepsilon^3\right)$.

Due to the equivalence between problem $1|h(1), Res|\sum w_j C_j$ and the problem with the floating maintenance period from Sect. 3.2, the described FPTAS is applicable to the latter problem, see Kellerer and Strusevich (2010a).

For problem$1|h(1), Res|\sum w_j C_j$ an FPTAS by Epstein et al. (2010) is not strongly polynomial, while an FPTAS by Marchetti-Spaccamela et al. (2008) is not accompanied by sufficient details and estimations of its running time.

6.3.2 Minimizing total weighted earliness and tardiness

As mentioned in Sect. 4.2, problem $1|d_j = d, p(N) > d|\sum w_j(E_j + T_j)$ with a large common due date admits an FPTAS via its reformulation in terms of Problem HPAdd. Thus, below we concentrate on problem $1|d_j = d, p(N) \le d|\sum w_j(E_j+T_j)$ with a small common due date. In our presentation we follow Kellerer and Strusevich (2010b). As discussed in Sect. 3.3, for this problem an optimal schedule may belong to one of the two classes: Class 1 with no straddling job and Class 2 with a straddling job. Recall that it is shown in Sect. 3.3 that for problem $1|d_j = d|\sum w_j(E_j+T_j)$ searching for the best schedule in Class 1 is equivalent to solving the quadratic knapsack problem (27). Besides, due to (28), that problem is a special case of Problem SQK.

It is rather straightforward to adapt a general FPTAS for Problem SQK to handle the case with no straddling job. For Problem SQK, let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector of the decision variables that deliver a feasible solution. Denote the corresponding value of the objective function by $Z(\mathbf{x})$. This solution can be converted into a schedule $S(\mathbf{x})$ that belongs to Class 1 and is feasible for problem $1|d_j = d, p(N) \le d|\sum w_j(E_j + T_j)$ with the value of the objective function $Z(S(\mathbf{x})) = Z(\mathbf{x})$. In schedule $S(\mathbf{x})$ the jobs $j \in N$ with $x_j = 1$ are assigned in the order that is opposite to their numbering by the WSPT rule to be processed as a block, without intermediate idle time, with the job with the smallest number completing exactly at time d. The remaining jobs are processed as a block starting at time d.

Notice that the accepted numbering (14) of jobs in problem $1|d_j = d|\sum w_j(E_j + T_j)$ implies that (11) holds for the corresponding Problem SQK. Additionally, due to (28) for Problem SQK we have that $\nu_j = \alpha_j \beta_j$ for each $j \in N$. Thus, all conditions of Theorem 9 are valid and Algorithm Round can be applied. Let $x^H = \left(x_1^H, x_2^H, \ldots, x_n^H\right)$ be the solution vector, which can be found in $O(n^3)$ time. As described above, this solution can be converted into a heuristic schedule $S_H^1 = S(x^H)$ that belongs to Class 1. If there exists an optimal schedule S^* with no straddling job, then Theorem 9 guarantees that $Z(S_H^1) \le \frac{3\sqrt{5}+13}{2} Z(S^*)$.

We can use the found value $Z(x^H)$ as an upper bound on Z^*, the optimal value of the objective function in Problem SQK. Algorithm EpsSQK is directly applied and the found solution is converted into a feasible schedule S_ε^1 as outlined above. Theorem 10 guarantees that $Z(S_\varepsilon^1) \le (1+\varepsilon)\, Z(S^*)$, and the required running time does not exceed $O\left(\frac{n^4}{\varepsilon^2}\right)$.

Assume now that for problem $1|d_j = d, p(N) \leq d|\sum w_j(E_j + T_j)$ there exists an optimal schedule S^* with a straddling job. First, we show that in the case under consideration Algorithm Round can be adapted to guarantee a smaller worst-case bound than that proved in Theorem 9.

Below we present an algorithm that finds a heuristic schedule S_H^2 such that $Z(S_H^2) \leq \frac{3+\sqrt{5}}{2} Z(S^*)$.

Algorithm ETStr

Step 1 Run Steps 1–6 of the first iteration of Algorithm Round. Let x_j^{Δ}, $j \in I_2$, be the components of the found solution to the continuous linear knapsack problem in Step 6.

Step 2 Define the vector $x^H = (x_1^H, \ldots, x_n^H)$, where $x_j^H = x_j^{\Delta}$ for $j \in I_2$ and $x_j^H = 0$, otherwise.

Step 3 Output schedule S_H^2 in which the jobs with $x_j^H = 1$ are processed starting at time zero in the order opposite to the WSPT sequence, followed by a job ℓ with $0 < x_\ell^H < 1$ as the straddling job, and finally the late jobs with $x_j^H = 0$ are scheduled in the WSPT order.

The running time of the algorithm is $O(n^2)$, since the continuous relaxation of Problem SQK has to be solved only once. It is proved by Kellerer and Strusevich (2010b) that Algorithm ETStr is in fact a $\frac{1}{\delta}$−approximation algorithm, i.e., $Z(S_H^2) \leq \frac{3+\sqrt{5}}{2} Z(S^*) = 2.618\ Z(S^*)$.

To find a feasible (but not necessarily the best) schedule that belongs to Class 2, we select each job as a possible straddling job, find a Class 1 schedule S_m of the remaining $m = n - 1$ jobs, and then try to insert the chosen job as straddling in such a way that the first early job starts at time zero.

To develop an FPTAS for the problem of finding the best schedule of Class 2, we need to modify the DP algorithms described for the standard Problem SQK in Sect. 6.1. Similarly to Sect. 6.3.1, let $S(p)$ denote a feasible schedule with a fixed crossover job with the processing time p and weight w.

Let W denote the total weight of all n jobs. For schedule S_m, define W_m to be the total weight of the early jobs. The value of the objective function of the resulting schedule $S(p)$ can be written as

$$Z(S(p)) = Z_m + F(W_m, x), \tag{52}$$

where $F(W_m, x)$ denotes the total increment of the objective function due to the insertion of the straddling job. It follows that

$$F(W_m, x) = W_m px + (W - W_m)\, p\, (1 - x)\,.$$

Thus, for each choice of the straddling job we need to find the best way of inserting the job into the schedule for the remaining jobs. The value of Z_m can be found either by a primary dynamic programming algorithm, similar to Algorithm PDP or by its dual counterpart. The only difference is that now for computing the overall value of the objective function Z we need to know the value of W_m, or rather pW_m that contributes into Z. This value can be found recursively during the iterations of the DP algorithm. The description of the resulting algorithm is similar to that of Algorithm PDP.

A typical primal state after the values $x_1, x_2, \ldots, x_k$ have been assigned is represented by a state of the form

$$(k, Z_k, y_k, V_k)\,,$$

where k, Z_k and y_k have the same meaning as in Algorithm PDP, while

$$V_k = p \sum_{j=1}^{k} w_j x_j$$

denotes the product of the total weight of the jobs processed before the due date and the processing time of the straddling job.

In iteration k of the primal algorithm a move from a state (k, Z_k, y_k, V_k) to a state $(k+1, Z_{k+1}, y_{k+1}, V_{k+1})$ is done in accordance with

$$Z_{k+1} = Z_k + w_{k+1} y_k,\ y_{k+1} = y_k + p_{k+1},\ V_{k+1} = V_k + w_{k+1} p,$$

provided that $y_{k+1} \le d$, and with

$$Z_{k+1} = Z_k + w_{k+1} \left(A_k - y_k\right) + w_{k+1} p_{k+1};\ y_{k+1} = y_k, V_{k+1} = V_k,$$

where $A_k = \sum_{j=1}^{k} p_j$.

We may assume $2W_m < W$; otherwise a schedule with the chosen straddling jobs cannot be optimal. This implies

$$Z(S(p)) \ge F(W_m, x) \ge W_m p x + W_m p (1-x) = W_m p,$$

i.e., $V_m = W_m p \le Z^{UB}$. As an upper bound Z^{UB} we may take the value $Z(S_H^2)$, where S_H^2 is a heuristic schedule found by Algorithm ETStr.

For each state generated in the last iteration of the primal algorithm, we find x and compute the value of the function Z by formula (52). The smallest of the found $Z-$values corresponds to an optimal value of the total weighted completion time for the original problem with a chosen straddling job.

The DP algorithm outlined above is used as a subroutine in the FPTAS, so that all feasible combinations should be generated and stored. For a fixed straddling job, such a DP algorithm requires $O\left(nd\,(Z^{UB})^2\right)$ time, i.e., an overall optimal solution will be found in $O\left(n^2 d\,(Z^{UB})^2\right)$ time. We skip the details of the dual version of our DP algorithm for finding the values of Z_m.

For problem $1|d_j = d| \sum w_j \left(E_j + T_j\right)$ with a chosen straddling job, Algorithm EpsSQK can be modified as follows. Let $Z^*(p, w)$ denote the smallest value of the objective function in the class of schedules with the chosen straddling job with processing time p and weight w.

For $Z^{UB} = Z(S_H^2)$ it follows that $Z_{LB} = \frac{3-\sqrt{5}}{2} Z^{UB}$ is a lower bound on $Z(S^*)$, and therefore on $Z^*(p, w)$. In Step 1 of Algorithm EpsSQK redefine $\delta = \frac{\varepsilon Z_{LB}}{4m}$. Change Step 4(a) of the algorithm in the following way. Move from a stored primal state (k, Z_k, y_k, V_k) to at most two primal states of the form $(k+1, Z_{k+1}, y_{k+1}, V_{k+1})$, each time rounding up each of the updated values of Z_{k+1} and V_{k+1} to the next multiple of δ. For each selection of states related to the same pair (Z_{k+1}, V_{k+1}) and a subinterval I_j^r, determine the value $y_{k+1}^{\min}$ as the smallest value of y_{k+1} that belongs to I_j^r and the value $y_{k+1}^{\max}$ as the largest value of y_{k+1} that belongs to I_j^r. If these values exist and are distinct, then out of all states $(k+1, Z_{k+1}, y_{k+1}, V_{k+1})$ with the same values of Z_{k+1} and V_{k+1} for $y_{k+1} \in \left[y_{k+1}^{\min}, y_{k+1}^{\max}\right]$ store only two states $(k+1, Z_{k+1}, y_{k+1}^{\min}, V_{k+1})$ and $(k+1, Z_{k+1}, y_{k+1}^{\max}, V_{k+1})$. The description of Step 4(b) of Algorithm EpsSQK is altered similarly; here we manipulate the corresponding dual states. Step 5 is reformulated accordingly.

For each choice of the straddling job, the described FPTAS takes $O\left(n^5/\varepsilon^3\right)$ time. To find an approximate solution to the original problem we need to run the FPTAS for each job selected as the straddling job, rejecting the solutions in which the selected job completes before the due date. Thus, the overall running time of the FPTAS is $O\left(n^6/\varepsilon^3\right)$.

For problem $1|d_j = d|\sum\left(E_j + T_j\right)$ with equal weights, we can take the value of the function delivered by the algorithm by Hoogeveen et al. (1994) as an upper bound Z^{UB}. Recall that $Z^{UB}/Z^* \leq 4/3$. For each choice of the straddling job, running a modified FPTAS that finds the values Z_m takes $O\left(n^4/\varepsilon^2\right)$ time. Thus, the overall running time of the FPTAS for problem $1|d_j = d|\sum\left(E_j + T_j\right)$ is $O\left(n^5/\varepsilon^2\right)$. See also Kacem et al. (2011) for a discussion of the problems from this subsection.

6.3.3 *Minimizing total weighted tardiness*

For the common due date problem $1|d_j = d\,|\sum w_j T_j$, Fathi and Nuttle (1990) provide a 2−approximation algorithm that requires $O(n^2)$ time. Kolliopoulos and Steiner (2006) give an approximation scheme for the problem with a fixed number of distinct due dates, however, the running time of their algorithm is pseudopolynomial since it is bounded by a polynomial of the largest weight.

As shown in Sect. 3.4, for problem $1|d_j = d\,|\sum w_j T_j$ we can select a job as straddling, find a schedule S_m for $m = n - 1$ remaining jobs and then insert the chosen job into the derived schedule. Kellerer and Strusevich (2006) explain how to adapt a general FPTAS for Problem SQK for handling problem $1|d_j = d\,|\sum w_j T_j$ to obtain an approximation scheme that requires the running time of $O\left(\left(n^6 \log W\right)/\varepsilon^3\right)$.

The achieved running time, although polynomial is not strongly polynomial and should be seen as impractical. Still, this algorithm has a certain theoretical importance as the first evidence that problem $1|d_j = d|\sum w_j T_j$ admits an FPTAS. Besides, an adaptation of the general FPTAS for Problem SQK to solving problem $1|d_j = d|\sum w_j T_j$ involves techniques that might be useful for handling other possible applications. For example, Karakostas et al. (2012) develop an FPTAS for the problem of minimizing the total weighted tardiness with a fixed number of distinct due dates, and their consideration is based on the ideas of Algorithm EpsWT above.

A much faster FPTAS for problem $1|d_j = d|\sum w_j T_j$ is designed by Kacem (2010). He gives a modified DP algorithm that scans the jobs in the order opposite to the WSPT, assigns the jobs form right to left starting from time $p(N)$. This algorithm is converted into an FPTAS that is based on splitting the solution space into equal subintervals and reducing the number of states stored in each interval. The running time of the FPTAS is $O(n^2/\varepsilon)$. See also Kacem et al. (2011) for a discussion of the problems from this subsection.

7 Differential approximation schemes

In this section, we discussed a less traditional measure of the quality of an approximate solution. Following Sarto Basso and Strusevich (2014), it is demonstrated that the existence of an FPTAS for Problem HP, with or without a knapsack constraint, implies the existence of so-called differential fully polynomial-time approximation schemes for the whole range of problems reviewed in this paper. Below we mainly use the set-function notation.

As stated in Sect. 4, finding an FPTAS for problem $\min\left\{F(U)\,|U \subseteq N\right\}$ with $F(U) = H(U) + K$ is not straightforward, even though problem $\min\left\{H(U)\,|U \subseteq N\right\}$ admits an FPTAS. This reflects a known phenomenon of non-stability of the traditional approximation measure under linear transformations. This phenomenon has brought numerous researchers to considering an alternative approximation measure for problems of combinatorial optimization. Rather than measuring the relative error of a heuristic solution, it could be more insightful to study the position of the heuristic solution within the interval of all its possible values. Such an approximation measure takes the name of *differential approximation*. Several dozens of papers present differential approximation results for traditional problems of combinatorial optimization; see the review Ausiello and Paschos (2007).

Below we formally give the required definitions, presented in terms of optimizing set-functions. Let $\mathcal{M}$ denote a set of feasible subsets $U \subseteq N$. For a given set-function $\varphi(U)$, let U^* and U_* be the set-maximizer and the set-minimizer, respectively, i.e., $\varphi(U_*) \le \varphi(U)$ and $\varphi(U^*) \ge \varphi(U)$ for all sets $U \in \mathcal{M}$. Consider the problem $\min\{\varphi(U)\,|U \in \mathcal{M}\}$ of minimizing $\varphi(U)$ over $\mathcal{M}$. An algorithm A that finds a set $U_A \in \mathcal{M}$ is called a *$\delta-$differential approximation* algorithm if

$$\varphi(U_A) \le \delta\varphi(U_*) + (1-\delta)\varphi\left(U^*\right). \tag{53}$$

Here $0 < \delta < 1$, and the value of an approximate solution is placed between the best and the worst objective function values. A good differential approximation algorithm finds a solution that is fairly close to the optimum, i.e., there is an interest in designing an algorithm that delivers a δ close to 1 in (53). For any small positive δ, a family of $(1-\delta)-$differential approximation algorithms is called a *Differential Fully Polynomial-Time Approximation Scheme (DFPTAS)* if its running time depends polynomially on the length of the problem's input and additionally is polynomial in $1/\delta$.

See Ausiello et al. (2003) and Demange and Paschos (1996) for a detailed formal discussion on differential approximation and on comparisons with traditional approximability. It should be stressed that a problem that admits a good differential approximation algorithm does not necessarily admits an algorithm with a good traditional approximation ratio, and vice versa. The examples include the travelling salesman problem and the minimum graph vertex problem; see, e.g., Demange and Paschos (1996) and Kacem and Paschos (2013).

In particular, in general for problem $\min\{\varphi(U)\,|U \in \mathcal{M}\}$ the existence of an FPTAS does not imply the existence of a DFPTAS. However, for the problems of our interest such a conversion can easily be done.

Let $\varphi(U)$ be a set-function such that

$$\varphi(U_*) < 0 = \varphi(\varnothing) \le \varphi\left(U^*\right). \tag{54}$$

The following statement holds.

Lemma 5 *Let $\varphi(U)$ be a set-function that satisfies (54), and $\psi(U) = \varphi(U) + K$ for a constant K. If problem $\min\{\varphi(U)\,|U \in \mathcal{M}\}$ admits an algorithm that finds a set U_A such that $\varphi(U_A) - \varphi(U_*) \le \delta\,|\varphi(U_*)|$, then $\varphi(U_A) \le (1-\delta)\,\varphi(U_*) + \delta\varphi(U^*)$ and $\psi(U_A) \le (1-\delta)\,\psi(U_*) + \delta\psi(U^*)$ hold.*

Proof This lemma is proved in Sarto Basso and Strusevich (2014). If follows from $\varphi(U_A) - \varphi(U_*) \le \delta\,|\varphi(U_*)|$ that $\varphi(U_A) - \varphi(U_*) \le -\delta\varphi(U_*)$ and, therefore, $\varphi(U_A) \le (1-\delta)\,\varphi(U_*)$. Since $\varphi(U^*) \ge 0$ we have that for any $\delta > 0$ the inequality $\delta\varphi(U^*) \ge 0$ holds. Thus, we deduce $\varphi(U_A) \le (1-\delta)\,\varphi(U_*) + \delta\varphi(U^*)$. Adding a constant K to both sides yields $\psi(U_A) \le (1-\delta)\,\psi(U_*) + \delta\psi(U^*)$. □

Lemma 5 illustrates an advantage of using a differential approximation measure: adding a constant can be handled easily. It forms a basis of converting an FPTAS to a DFPTAS for the problems with the objectives related to the half-product.

Notice that for the half-product function (1) or (12) the assumption (54) holds. Apply Lemma 5 to problem $\min\{H(U)\,|U\in\mathcal{M}\}$, where either $\mathcal{M}=2^N$ or $\mathcal{M}=\{U|\alpha(U)\leq A,\ U\subseteq N\}$. Each problem $\min\{H(U)\,|U\subseteq N\}$ and $\min\{H(U)\,|\alpha(U)\leq A,\ U\subseteq N\}$ admits an FPTAS that requires $O\left(n^2/\varepsilon\right)$ time; see Sect. 4.1. Thus, for $\varphi(U)=H(U)$ the inequality $\varphi(U_A)-\varphi(U_*)\leq\delta\,|\varphi(U_*)|$ holds for $\delta=\varepsilon$, and an FPTAS for problem $\min\{H(U)\,|U\subseteq N\}$ and $\min\{H(U)\,|\alpha(U)\leq A,\ U\subseteq N\}$ behaves as an DFPTAS for the corresponding problem, even if the objective function changes to $H(U)+K$.

Theorem 12 *For set-functions* $H(U)$ *of the form* (12) *and* $F(U)=H(U)+K$ *each problem* $\min\{H(U)\,|U\subseteq N\}$, $\min\{H(U)\,|\alpha(U)\leq A,\ U\subseteq N\}$, $\min\{F(U)\,|U\subseteq N\}$ *and* $\min\{F(U)\,|\alpha(U)\leq A,\ U\subseteq N\}$ *admits a DFPTAS that requires* $O\left(n^2/\varepsilon\right)$ *time.*

Problem $1|h(1),N-res|\sum w_jC_j$ formulated in Sect. 3.1 is among the first scheduling problems that have been studied from the differential approximability point of view. It is proved in Kacem and Paschos (2013) that the problem admits an $O(n\log n)-$time differential $\delta-$approximation algorithm for $\delta=\frac{3-\sqrt{5}}{2}=0.381\,97$. Theorem 12 implies that not only this problem, but all scheduling problems formulated in Sect. 3 admit a DFPTAS. The running time of such a DFPTAS is always $O\left(n^2/\varepsilon\right)$, even for those scheduling problems for which the best known FPTAS requires more time than $O\left(n^2/\varepsilon\right)$.

8 Maximizing half-product related functions

In this section, we turn to the problem of maximizing functions, related to the half-product, with and without a knapsack constraint. These problems are of interest in their own right, as well as due to their possible applications. The results of this section are mainly contained in Kellerer et al. (2015).

We start with problem $\max\{Q(U)\,|U\subseteq N\}$ of maximizing a quadratic function (13) in which the coefficients of the quadratic terms are all non-negative. As stated in Sect. 2.3, function $Q(U)$ is supermodular. Solving problem $\max\{Q(U)\,|U\subseteq N\}$ is equivalent to solving problem $\min\{-Q(U)\,|U\subseteq N\}$, where function $-Q(U)$ obtained from $Q(U)$ by changing the signs of all coefficients is submodular.

It is well-known that minimizing a submodular function (or, equivalently, maximizing a supermodular function) can be done in polynomial time. See Iwata et al. (2001) and Schrijver (2000) for algorithms for minimizing an arbitrary submodular function. Due a special structure, the problem of maximizing a quadratic supermodular function $Q(U)$ can be solved much faster than in the general case, since it reduces to the problem of finding the maximum flow in a digraph with $O(n^2)$ nodes. This approach is originally due to Rhys (1970), and is outlined in Wolsey and Nemhauser (1988). It is demonstrated in Kellerer et al. (2015) that such an algorithm can be implemented in $O\left(n^3\right)$ time. The problem of maximizing the half-product function (12) is a special case of problem $\max\{Q(U)\,|U\subseteq N\}$ with non-negative coefficients of the quadratic terms. Thus, the following statement holds.

Theorem 13 *Each problem* $\max\{H(U)\,|U\subseteq N\}$ *and* $\max\{F(U)=H(U)+K|U\subseteq N\}$ *can be solved in* $O\left(n^3\right)$ *time.*

This result is in contrast with the fact that the minimization counterparts of these problems, i.e., problems $\min\{H(U)\,|U\subseteq N\}$ and $\min\{F(U)\,|U\subseteq N\}$, are NP-hard.

We now turn to problem $\max\{H(U)\,|\alpha(U)\leq A,\ U\subseteq N\}$ of maximizing a half-product function subject to a knapsack constraint. The latter problem belongs to the class of maximization integer programming problems known as *supermodular knapsack* problems; see, e.g., Gallo and Simeone (1988) and Rader and Woeginger (2002).

Kellerer et al. (2015) show that if there are negative coefficients γ_j in the linear part of the function that this NP-hard problem is non-approximable. In fact, they show that for problem $\max\{H(U)\,|\alpha(U) \le A,\ U \subseteq N\}$ it is NP-hard to decide whether $H(U^*) \ge 1$, where U^* is a set-maximizer. This leads in the following non-approximability result.

Theorem 14 *Unless $P = NP$, for any ρ, $0 < \rho < 1$ problem* $\max\{H(U)\,|\alpha(U) \le A,\ U \subseteq N\}$ *does not admit a polynomial-time algorithm which finds a feasible set S^H such that $h\left(S^H\right) > \rho h(S^*)$.*

In the remainder of this section, we focus on problem $\max\{G(U)\,|\alpha(U) \le A,\ U \subseteq N\}$, where

$$G(U) = \sum_{i,j\in U;\ i<j;} \alpha_i\beta_j + \gamma(U) + K$$

with all coefficients being non-negative; an equivalent expression for this function in terms of Boolean variables is

$$G(\mathbf{x}) = \sum_{1\le i<j\le n}^{n} \alpha_i\beta_j x_i x_j + \sum_{j=1}^{n}\gamma_j x_j + K. \tag{55}$$

As stated in Kellerer et al. (2015), problem $\max\{G(U)\,|\alpha(U) \le A,\ U \subseteq N\}$ can serve as a mathematical model of a prize collecting routing problem. Suppose that a vehicle, initially with no passengers, visits all or some of n destinations and picks up p_j people from destination j, $1 \le j \le n$. The vehicle is of a limited capacity, so that at any time no more than A people can be on board.

When the vehicle leaves destination j, every person on board that has been brought to this destination is paid a reward of w_j points, while each of p_j people picked up from the destination receives a different award of v_j points. It is required to maximize the total collected prize, provided that the vehicle never carries more than A passengers.

Without loss of generality, assume that the destinations are numbered in accordance with the WSPT rule (14). Adapting the results by Smith (1956) on a single machine scheduling problem to minimize the weighted sum of the completion times, it can be verified that the vehicle should visit destinations in the order that is opposite to their numbering.

Introduce the Boolean decision variables

$$x_j = \begin{cases} 1, & \text{if destination } j \text{ is visited} \\ 0, & \text{otherwise} \end{cases}.$$

The vehicle capacity constraint can be written as

$$\sum_{j=1}^{n} p_j x_j \le A.$$

The total profit can be written as

$$G(\mathbf{x}) = \sum_{1\le j\le n}^{n} w_j x_j \sum_{i=1}^{j-1} p_i x_i + \sum_{j=1}^{n} v_j p_j x_j = \sum_{1\le i<j\le n}^{n} p_i w_j x_i x_j + \sum_{j=1}^{n} v_j p_j x_j,$$

which corresponds to (55) with $\alpha_j = p_j$, $\beta_j = w_j$, $\gamma_j = v_j p_j$, $1 \le j \le n$, and $K = 0$. Thus, this version of the prize collecting problem reduces to $\max\{G(U)\,|\alpha(U) \le A,\ U \subseteq N\}$.

Recall that for a problem of maximizing a set-function $\varphi(U)$, a set U^{ε} is called an ε- *approximate solution* if for a given positive ε the inequality $\varphi(U^*) - \varphi(U^{\varepsilon}) \le \varepsilon\varphi(U^*)$ holds. A family of algorithms that for any given positive ε find an $\varepsilon-$ approximate solution is called a *Fully Polynomial-Time Approximation Scheme (FPTAS)*, provided that the running time depends polynomially on both the length of the input and $1/\varepsilon$.

Kellerer et al. (2015) show that problem $\max\{G(U)\,|\alpha(U) \le A,\ U \subseteq N\}$ admits an FPTAS which can be obtained by converting a DP algorithm that finds an optimal solution of the problem in pseudopolynomial time. Such an algorithm can be easily developed by an appropriate modification of Algorithm DP1. We only need to modify Step 2 by replacing the recursive formula (34) by the formula

$$Z_k = Z_{k-1} + \beta_k y_{k-1} + \gamma_k,\ \ y_k = y_{k-1} + \alpha_k. \tag{56}$$

and the recursive formula (35) by

$$Z_k = Z_{k-1};\ y_k = y_{k-1}. \tag{57}$$

Besides, in Step 3, we need to find Z_n^*, the largest value of Z_n among all found states of the form (n, Z_n, y_n).

The running time of the resulting DP algorithm, which we call Algorithm DPMax, does not exceed $O(nAZ^{UB})$, where Z^{UB} is an upper bound on the value $G(\mathbf{x}^*)$. Here and below an obvious upper bound

$$Z^{UB} = \sum_{1\le i<j\le n}^{n} \alpha_i\beta_j + \sum_{j=1}^{n}\gamma_j + K \tag{58}$$

is used. To convert Algorithm DPMax into an FPTAS, Kellerer et al. (2015) employ the geometric rounding technique to reduce the number of stored states.

Algorithm EpsGmax

Step 1 Compute Z^{UB} by (58). Introduce the intervals, whose endpoints form geometric sequences. For the $y-$values, introduce the intervals

$$[0,0], [1, 1+\varepsilon], \left[1+\varepsilon, (1+\varepsilon)^2\right], \left[(1+\varepsilon)^2, (1+\varepsilon)^3\right], \dots, \left[(1+\varepsilon)^{u-1}, (1+\varepsilon)^u\right],$$

where $\lceil (1+\varepsilon)^u \rceil = A$. Call these intervals I_ℓ, $\ell = 0, 1, \dots, u$. For the $Z-$values, introduce the intervals

$$[0,0], \left[1, (1+\varepsilon)^{\frac{1}{n}}\right], \left[(1+\varepsilon)^{\frac{1}{n}}, (1+\varepsilon)^{\frac{2}{n}}\right], \dots, \left[(1+\varepsilon)^{\frac{v-1}{n}}, (1+\varepsilon)^{\frac{v}{n}}\right],$$

where $\left\lceil (1+\varepsilon)^{\frac{v}{n}} \right\rceil = Z^{UB}$. Call these intervals J_r, $r = 0, 1, \dots, v$.

Step 2 Store the initial state $(0, Z_0, y_0)$ with $Z_0 = K$ and $y_0 = 0$. For each k, $1 \le k \le n$, do the following:

(a) In line with Algorithm DPMax, move from a stored state $(k-1, Z_{k-1}, y_{k-1})$ to at most two states of the form (k, Z_k, y_k), where $Z_k \le Z^{UB}$, using the relations (56) and (57).

(b) If the number of generated states (k, Z_k, y_k) with the Z-values in the same interval J_r and with the y-values in the same interval I_ℓ exceeds 2, then keep only states two states, with the largest and the smallest y-value.

Step 3 Determine Z^{ε} as the largest value of Z_n among the states (n, Z_n, y_n). Perform back-tracking and find the vector $\mathbf{x}^{\varepsilon} = \left(x_1^{\varepsilon}, x_2^{\varepsilon}, \dots, x_n^{\varepsilon}\right)$ that leads to Z^{ε}. Output $\mathbf{x}^{\varepsilon}$ and $G(\mathbf{x}^{\varepsilon})$ as an approximate solution of the problem.

The required performance of Algorithm EpsGmax is guaranteed by the following statement.

Lemma 6 *Let*

$$(0, K, 0), (1, Z_1^*, y_1^*), \ldots, (n, Z_n^*, y_n^*)$$

be the sequence of states that lead to the optimal solution with the function value Z_n^*. *For each* k, $0 \le k \le n$, *for a state* (k, Z_k^*, y_k^*) *there exists a state* (k, Z_k, y_k) *stored by Algorithm FPTAS such that*

(i) $y_k \le y_k^*$;
(ii) $(1+\varepsilon)\, y_k \ge y_k^*$;
(iii) $(1+\varepsilon)^{\frac{k}{n}}\, Z_k \ge Z_k^*$.

Algorithm EpsGmax delivers an approximate solution of a required quality and run in time that is polynomial (but not strongly polynomial).

Theorem 15 *Algorithm EpsGmax is an FPTAS for problem* $\max\{G(U) \,|\alpha(U) \le A,\ U \subseteq N\}$ *that requires* $O\left(\frac{n^2}{\varepsilon^2} \log A \log Z^{UB}\right)$ *time.*

9 Conclusion

The survey presents the known results on exact and approximation algorithms and schemes for solving Boolean optimization problems with quadratic objective functions related to the half-product. Further research in this direction may include extending conditions under which the problems of the outlined range admit an FPTAS or a constant-ratio approximation algorithm.

1. An FPTAS resented in Sect. 5.1 for Problems PosHP requires $O\left(n^2/\varepsilon\right)$ time, provided that the objective function is convex. Is it possible to develop an FPTAS for problem Problem HPAdd that runs in $O\left(n^2/\varepsilon\right)$ time, without any extra assumptions on the shape of the objective function such as convexity or a representation with positive coefficients?
2. Are there algorithmic techniques that would allow further reducing the running time of an FPTAS for Problem SQK and its scheduling applications?
3. The continuous relaxation of Problems PosHP and SQK can be solved in $O(n^2)$ time, provided that the objective function is non-separable convex; see Sects. 5.2 and 6.1. On the other hand, as proved in Moré and Vavasis (1991), the continuous quadratic knapsack problem is NP-hard if the objective function is separable concave. Does there exist a strongly polynomial-time algorithm that solves the continuous relaxation of the general quadratic knapsack problem with a convex function?
4. The search for possible applications of the corresponding Boolean programming problems is of interest. In particular, the existence of an FPTAS for the problem of minimizing the total weighted earliness and tardiness with asymmetric weights would resolve the status of the problem with respect to the unary encoding; so far the problem is not known to be solvable in pseudopolynomial time or to admit a polynomial-time approximation scheme. Are there applications of Problem HPAdd or Problem SQK to a problem area different from scheduling?

Acknowledgments This research was partly supported by the EPSRC funded Project EP/I018441/1 “Quadratic and Linear Knapsack Problems with Scheduling Applications”.

References

Adiri, I., Bruno, J., Frostig, E., & Rinnooy Kan, A. H. G. (1989). Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, *26*, 679–696.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research*, *52*, 229–242.

Ausiello, G., Bazgan, C., Demange, M., & Paschos, V.T. (2003). Completeness in differential approximation classes. In Rovan, B., Vojtáš, P. (Ed.), *Mathematical foundations of computer science 2003, 28th international symposium, lecture notes computer science* (Vol. 2747, pp. 179–188).

Ausiello, G., & Paschos, V. T. (2007). Differential ratio approximation. In T. F. Gonzalez (Ed.), *Handbook of approximation algorithms and metaheuristics, chapter 16*. London: Taylor and Francis.

Badics, T., & Boros, E. (1998). Minimization of half-products. *Mathematics of Operations Research*, *33*, 649–660.

Bagchi, U., Sullivan, R. S., & Chang, Y.-L. (1987). Minimizing mean squared deviation of completion times about a common due date. *Management Science*, *33*, 894–906.

Berman, P., Kovoor, N., & Pardalos, P. M. (1993). Algorithms for the least distance problem. In P. M. Pardalos (Ed.), *Complexity in numerical optimization* (pp. 33–56). Singapore: World Scientific.

Boros, E., & Hammer, P. (2002). Pseudo-Boolean optimization. *Discrete Applied Mathematics*, *123*, 155–225.

Breit, J. (2007). Improved approximation for non-preemptive single machine flow-time scheduling with an availability constraint. *European Journal of Operational Research*, *183*, 516–524.

Bretthauer, K. M., & Shetty, B. (1997). Quadratic resource allocation with generalized upper bounds. *Operations Research Letters*, *20*, 51–57.

Brucker, P. (1984). An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, *3*, 163–166.

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

Cheng, J., & Kubiak, W. (2005). A half-product based approximation scheme for agreeably weighted completion time variance. *European Journal of Operational Research*, *162*, 45–54.

Csirik, J., Frenk, J. B. G., Labbé, M., & Zhang, S. (1991). Heuristics for the 0–1 min-knapsack problem. *Acta Cybernetica*, *10*, 15–20.

De, P., Ghosh, J. B., & Wells, C. E. (1989). A note on the minimization of mean squared deviation of completion times about a common due date. *Management Science*, *35*, 1143–1147.

De, P., Ghosh, J. B., & Wells, C. E. (1992). On the minimization of completion time variance with bicriteria extension. *Operations Research*, *40*, 1148–1155.

Demange, M., & Paschos, V. T. (1996). On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoretical Computer Science*, *158*, 117–141.

Eilon, S., & Chowdhury, I. E. (1972). Minimizing time variance in the single machine problem. *Management Science*, *23*, 567–575.

Engles, D. W., Karger, D. R., Kolliopoulos, S. G., Sengupta, S., Uma, R. N., & Wein, J. (2003). Techniques for scheduling with rejection. *Journal of Algorithms*, *49*, 175–191.

Epstein, L., Levin, A., Marchetti-Spaccamela, A., Megow, N., Mestre, J., Skutella, M., & Stougie, L. (2010). Universal sequencing on a single machine. In: Eisenbrand, F., Shepherd, B. (Ed.), *IPCO 2010, lecture notes in computer science* (Vol. 6080, pp. 230–243).

Erel, E., & Ghosh, J. B. (2008). FPTAS for half-products minimization with scheduling applications. *Discrete Applied Mathematics*, *156*, 3046–3056.

Fathi, Y., & Nuttle, H. W. L. (1990). Heuristics for the common due date weighted tardiness problem. *IIE Transactions*, *22*, 215–225.

Foldes, S., & Hammer, P. (2005). Submodularity, supermodularity and higher order monotonicities of pseudo-Boolean functions. *Mathematics of Operations Research*, *30*, 453–461.

Gallo, G., & Simeone, B. (1988). On the supermodular knapsack problem. *Mathematical Programming*, *45*, 295–309.

Gordon, V. S., Potts, C. N., Strusevich, V. A., & Whitehead, J. D. (2008). Single machine scheduling models with deterioration and learning: Handling precedence constraints via priority generation. *Journal of Scheduling*, *11*, 357–370.

Güntzer, M. M., & Jungnickel, D. (2000). Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. *Operations Research Letters*, *26*, 55–66.

Hall, N. G., & Posner, M. E. (1991). Earliness–tardiness scheduling problems, I: Weighted deviation of completion times about a common due date. *Operations Research*, *39*, 836–846.

Hall, N. G., Kubiak, W., & Sethi, S. P. (1991). Earliness–tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research*, *39*, 847–856.

Hochbaum, D. S. (2005). Complexity and algorithms for convex network optimization and other nonlinear problems. *4OR—Quarterly Journal of Operations Research*, *3*, 171–216.

Hochbaum, D. S. (2008). Complexity and algorithms for nonlinear optimization problems. *Annals of Operations Research*, *153*, 257–296.

Hochbaum, D. S., & Shantikumar, J. G. (1990). Convex separable optimization is not much harder than linear optimization. *Journal of the Association for Computing Machinery*, *37*, 843–862.

Hoogeveen, J. A., Oosterhout, H., & van de Velde, S. L. (1994). New lower and upper bounds for scheduling around a small common due date. *Operations Research*, *42*, 102–110.

Hoogeveen, J. A., & van de Velde, S. L. (1991). Scheduling around a small common due date. *European Journal of Operational Research*, *55*, 237–242.

Hoogeveen, H., & Woeginger, G. J. (2002). Some comments on sequencing with controllable processing times. *Computing*, *68*, 181–192.

Iwata, S., Fleischer, L., & Fujishige, S. (2001). A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. *Journal of the Association for Computing Machinery*, *48*, 761–777.

Janiak, A., Kovalyov, M. Y., Kubiak, W., & Werner, F. (2005). Positive half-products and scheduling with controllable processing times. *European Journal of Operational Research*, *165*, 416–422.

Jurisch, B., Kubiak, W., & Józefowska, J. (1997). Algorithms for minclique scheduling problems. *Discrete Applied Mathematics*, *72*, 115–139.

Kacem, I. (2008). Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval. *Computers and Industrial Engineering*, *54*, 401–410.

Kacem, I. (2010). Fully polynomial-time approximation scheme for the weighted total tardiness minimization with a common due date. *Discrete Applied Mathematics*, *158*, 1035–1040.

Kacem, I., & Chu, C. (2008). Worst-case analysis of the WSPT and MWSPT rules for single machine scheduling with one planned setup period. *European Journal of Operational Research*, *187*, 1080–1089.

Kacem, I., Kellerer, H., & Strusevich, V. A. (2011). Single machine scheduling with a common due date: Total weighted tardiness problems. In A. R. Mahjoub (Ed.), *Progress in combinatorial optimization, chapter 13* (pp. 391–421). New York, London: Wiley-ISTE.

Kacem, I., & Mahjoub, A. R. (2009). Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval. *Computers and Industrial Engineering*, *56*, 1708–1712.

Kacem, I., & Paschos, V. T. (2013). Weighted completion time minimization on a single-machine with a fixed non-availability interval: Differential approximability. *Discrete Optimization*, *10*, 61–68.

Kanet, J. T. (1981). Minimizing variation of flow time in single machine systems. *Management Science*, *27*, 1453–1459.

Karakostas, G., Kolliopoulos, S.G., & Wang, J. (2012). An FPTAS for the minimum total weighted tardiness problem with a fixed number of distinct due dates. *ACM Transactions on Algorithms, 8*(4), Article 40. doi:10.1145/2344422.2344430.

Kellerer, H., Kubzin, M. A., & Strusevich, V. A. (2009). Two simple constant ratio approximation algorithms for minimizing the total weighted completion time on a single machine with a fixed non-availability interval. *European Journal of Operational Research*, *199*, 111–116.

Kellerer, H., Mansini, R., Pferschy, U., & Speranza, M. G. (2003). An efficient fully polynomial approximation scheme for the subset-sum problem. *Journal of Computer and System Sciences*, *66*, 349–370.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. Berlin: Springer.

Kellerer, H., Rustogi, K., & Strusevich, V. A. (2013). Approximation schemes for scheduling on a single machine subject to cumulative deterioration and maintenance. *Journal of Scheduling*, *16*, 675–683.

Kellerer, H., Sarto Basso, R., & Strusevich, V. A. (2015). Approximability issues for unconstrained and constrained maximization of half-product related functions. Report SORG-01-2015.

Kellerer, H., & Strusevich, V. A. (2006). A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. *Theoretical Computer Science*, *369*, 230–238.

Kellerer, H., & Strusevich, V. A. (2010a). Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica*, *57*, 769–795.

Kellerer, H., & Strusevich, V. A. (2010b). Minimizing total weighted earliness–tardiness on a single machine around a small common due date: An FPTAS using quadratic knapsack. *International Journal of Foundations of Computer Sciences*, *21*, 357–383.
Kellerer, H., & Strusevich, V. A. (2012). The symmetric quadratic knapsack problem: Approximation and scheduling applications. *4OR—Quarterly Journal of Operations Research*, *10*, 111–161.
Kellerer, H., & Strusevich, V. A. (2013). Fast approximation schemes for Boolean programming and scheduling problems related to positive convex half-product. *European Journal of Operational Research*, *228*, 24–32.
Kolliopoulos, S. V., & Steiner, G. (2006). Approximation algorithms for minimizing the total weighted tardiness on a single machine. *Theoretical Computer Science*, *355*, 261–273.
Kovalyov, M. Y., & Kubiak, W. (1999). A fully polynomial approximation scheme for weighted earliness–tardiness problem. *Operations Research*, *47*, 757–761.
Kozlov, M. K., Tarasov, S. P., & Hačijan, L. G. (1979). Polynomial solvability of convex quadratic programming. *Soviet Mathematics Doklady*, *20*, 1108–1111.
Kubiak, W. (1993). Completion time variance on a single machine is difficult. *Operations Research Letters*, *14*, 49–59.
Kubiak, W. (1995). New results on the completion time variance minimization. *Discrete Applied Mathematics*, *58*, 157–168.
Kubiak, W. (2005). Minimization of ordered, symmetric half-products. *Discrete Applied Mathematics*, *146*, 287–300.
Kubiak, W., Cheng, J., & Kovalyov, M. Y. (2002). Fast fully polynomial approximation schemes for minimizing completion time variance. *European Journal of Operational Research*, *137*, 303–309.
Kubzin, M. A., & Strusevich, V. A. (2005). Two-machine flow shop no-wait scheduling with machine maintenance. *4OR—Quarterly Journal of Operations Research*, *3*, 303–313.
Kubzin, M. A., & Strusevich, V. A. (2006). Planning machine maintenance in two-machine shop scheduling. *Operations Research*, *54*, 789–800.
Kuo, W.-H., & Yang, D.-L. (2006). Minimizing the makespan in a single machine scheduling problem with a time-based learning effect. *Information Processing Letters*, *97*, 64–67.
Lawler, E. L., & Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science*, *16*, 77–84.
Lee, C.-Y. (1996). Machine scheduling with an availability constraint. *Journal of Global Optimization*, *9*, 395–416.
Lee, C.-Y. (2004). Machine scheduling with availability constraints. In J. Y.-T. Leung (Ed.), *Handbook of scheduling: Algorithms, models and performance analysi* (pp. 22-1–22-13). London: Chapman & Hall/CRC.
Lee, C.-Y., & Liman, S. D. (1992). Single machine flow time scheduling with scheduled maintenance. *Acta Informatica*, *29*, 375–382.
Ma, Y., Chu, C., & Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers and Industrial Engineering*, *58*, 199–211.
Marchetti-Spaccamela, A., Megow, N., Skutella, M., & Stougie, L. (2008). *Robust sequencing on a single machine*. Matheon Preprint 533.
Martello, S., & Toth, P. (1990). *Knapsack problems. Algorithms and computer implementation*. Chichester: Wiley.
Merten, A. G., & Muller, M. E. (1972). Variance minimization in single machine sequencing problems. *Management Science*, *18*, 518–528.
Megow, N., & Verschae, J. (2009). *Short note on scheduling on a single machine with one non-availability period*. Matheon Preprint 557.
Monteiro, R. D. C., & Adler, I. (1989). Interior path following primal-dual algorithms. Part II: Convex quadratic programming. *Mathematical Programming*, *44*, 43–66.
Moré, J. J., & Vavasis, S. A. (1991). On the solution of concave knapsack problems. *Mathematical Programming*, *49*, 397–411.
Nemhauser, G. L., Wolsey, L. A., & Fischer, M. L. (1978). An analysis of approximations for maximizing submodular set-functions—I. *Mathematical Programming*, *14*, 265–294.
Nowicki, E., & Zdrzałka, S. (1990). A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, *26*, 271–287.
Pisinger, D. (2007). The quadratic knapsack problem—A survey. *Discrete Applied Mathematics*, *155*, 623–648.
Rader, D. J, Jr, & Woeginger, G. J. (2002). The quadratic 0–1 knapsack problem with series–parallel support. *Operations Research Letters*, *30*, 159–166.
Rhys, M. W. (1970). A selection problem of shared fixed costs and network flows. *Management Science*, *17*, 200–207.

Romeijn, H. E., Geunes, G., & Taafe, K. (2007). On a nonseparable convex maximization problem with continuous knapsack constraints. *Operations Research Letters*, *35*, 172–180.

Rustogi, K., & Strusevich, V. A. (2014). Combining time and position dependent effects on a single machine subject to rate-modifying activities. *Omega*, *42*, 166–178.

Rustogi, K., & Strusevich, V. A. (2015). Single machine scheduling with time-dependent linear deterioration and rate-modifying maintenance. *Journal of the Operational Research Society*, *66*, 500–515.

Sadfi, C., Penz, B., Rapin, C., Błażewicz, J., & Formanowicz, P. (2005). An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. *European Journal of Operational Research*, *161*, 3–10.

Sahni, S. K. (1976). Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery*, *23*, 116–127.

Sarto Basso, R., & Strusevich, V. A. (2014). Differential approximation schemes for half-product related functions and their scheduling applications. Report SORG-04-2014.

Schrijver, A. (2000). A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory B*, *80*, 346–355.

Shabtay, D., & Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, *155*, 1643–1666.

Shakhlevich, N. V., & Strusevich, V. A. (2006). Single machine scheduling with controllable release and processing parameters. *Discrete Applied Mathematics*, *154*, 2178–2199.

Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the Association for Computing Machinery*, *48*, 206–242.

Smith, W. E. (1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly*, *3*, 59–66.

Tamir, A. (1993). A strongly polynomial algorithm for minimum convex separable quadratic cost flow problems on two-terminal series-parallel networks. *Mathematical Programming*, *59*, 117–132.

Tanaev, V.S., Kovalyov, M.Y., & Shafransky, Y.M. (1998). Scheduling theory. Group technologies, Minsk, pp. 41–44 (in Russian).

Vickson, R. G. (1980). Two single machine sequencing problems involving controllable job processing time. *AII Transactions*, *12*, 258–262.

Wan, G., Yen, B. P.-C., & Li, C.-L. (2001). Single machine scheduling to minimize total compression plus weighted flow cost is NP-hard. *Information Processing Letters*, *79*, 273–280.

Wang, G., Sun, H., & Chu, C. (2005). Preemptive scheduling with availability constraints to minimize total weighted completion times. *Annals of Operations Research*, *133*, 183–192.

Woeginger, G. J. (1999). An approximation scheme for minimizing agreeably weighted variance on a single machine. *INFORMS Journal on Computing*, *11*, 211–216.

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

Wolsey, L. A., & Nemhauser, G. L. (1988). *Integer and combinatorial optimization*. New York: Wiley-Interscience.

Wu, C.-C., Yin, Y., & Cheng, S.-R. (2011). Some single-machine scheduling problems with a truncation learning effect. *Computers and Industrial Engineering*, *60*, 790–795.

Xu, Z. (2012). A strongly polynomial FPTAS for the symmetric quadratic knapsack problem. *European Journal of Operational Research*, *218*, 377–381.

Yang, S.-J., & Yang, D.-L. (2010). Minimizing the makespan single-machine scheduling with aging effects and variable maintenance activities. *Omega*, *38*, 528–533.

Yuan, J. (1992). The NP-hardness of the single machine common due date weighted tardiness problem. *Systems Science and Mathematical Sciences*, *5*, 328–333.

Zhao, C.-L., & Tang, H.-Y. (2010). Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. *Applied Mathematical Modeling*, *34*, 837–841.