NOTICE: this is the author’s version of a work that was accepted for publication in European Journal of Operational Research. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in European Journal of Operational Research, Volume 228, Issue 1, (2013) DOI 10.1016/j.ejor.2012.12.028.

Fast Approximation Schemes for Boolean Programming and Scheduling Problems Related to Positive Convex Half-Product

Hans Kellerer[a], Vitaly Strusevich[b,*]

[a]*Institut für Statistik und Operations Research, Universität Graz, Universitätsstraße 15, A-8010, Graz, Austria*
[b]*School of Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, U.K.*

Abstract

We address a version of the Half-Product Problem and its restricted variant with a linear knapsack constraint. For these minimization problems of Boolean programming, we focus on the development of fully polynomial-time approximation schemes with running times that depend quadratically on the number of variables. Applications to various single machine scheduling problems are reported: minimizing the total weighted flow time with controllable processing times, minimizing the makespan with controllable release dates, minimizing the total weighted flow time for two models of scheduling with rejection.

Keywords: Scheduling, half-product, quadratic knapsack, scheduling with rejection, scheduling with controllable processing times, FPTAS.

1. Introduction

The topic of designing approximation schemes for scheduling problems with min-sum objective functions has recently drawn considerable attention. While for many problems of this range purpose-built approximation schemes have been developed, a general framework has been identified based on reformulation of the original scheduling problems in terms of minimization problems of quadratic Boolean programming. The Half-Product Problem and the closely related Symmetric Quadratic Knapsack Problem appear to be among the most suitable models, see recent reviews by Kacem et al. (2011) and Kellerer and Strusevich (2012).

This paper studies a version of the Half-Product Problem and its modification with the knapsack constraint, establishes conditions under which the

*Corresponding author
Email addresses: `hans.kellerer@uni-graz.at` (Hans Kellerer), `V.Strusevich@greenwich.ac.uk` (Vitaly Strusevich)

problems admit fast fully polynomial-time approximation schemes, describes the relevant algorithms and discusses their scheduling applications.

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; in fact, without loss of generality, we may assume that all γ_j are non-negative, since otherwise for a negative γ_j we may set $x_j = 0$ without increasing the value of $H(\mathbf{x})$. Problems of quadratic Boolean programming similar to (1) were introduced in 1990s as mathematical models for various scheduling problems by Kubiak (1995) and Jurisch et al. (1997). This function 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 $a_j = b_j$ for all $j = 1, 2, \ldots, n$, 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

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

where all coefficients α_j are positive integers, and β_j, μ_j, ν_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*. Notice that Problem PosHP is a special case of Problem HPAdd.

In all Half-Product 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 also study a more restricted version of Problem PosHP, in which an additional knapsack constraint is introduced, i.e., the problem

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

which we call the *Positive Half-Product Knapsack Problem* and denote by *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. 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 of our problem 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

$$\begin{aligned}
\text{Minimize} \quad & S(\mathbf{x}) = \sum_{1\le i<j\le n} \alpha_i\beta_j x_i x_j + \sum_{1\le i<j\le n}\alpha_i\beta_j(1-x_i)(1-x_j) \\
& \qquad + \sum_{j=1}^{n}\mu_j x_j + \sum_{j=1}^{n}\nu_j(1-x_j) + K \\
\text{Subject to} \quad & \sum_{j=1}^{n}\alpha_j x_j \le A \\
& x_j \in \{0,1\},\ j=1,2,\ldots,n.
\end{aligned} \tag{5}$$

Following Kellerer and Strusevich (2010a, 2010b), 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 $(1 - x_j)$. A comprehensive review of the results on Problem SQK and its scheduling applications is given by Kellerer and Strusevich (2012).

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

Since this paper focuses on the development of approximation algorithms and schemes, below we recall the definitions of the relevant notions. For a problem of minimizing a function $Z(\mathbf{x})$, where $\mathbf{x}$ is a collection of decision variables, let $\mathbf{x}^*$ denote the vector that delivers the minimum to the function $Z(\mathbf{x})$; we call $\mathbf{x}^*$ an *optimal solution* of the corresponding problem. A polynomial-time algorithm that finds a feasible solution $\mathbf{x}'$ such that $Z(\mathbf{x}')$ is at most $\rho \ge 1$ times the optimal value $Z(\mathbf{x}^*)$ is called a $\rho-$*approximation* algorithm; the value

Problem Acronym	**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)	$\sum_{j=1}^{n} \alpha_j x_j \leq A$
SQK	$S(\mathbf{x})$ (5)	$\sum_{j=1}^{n} \alpha_j x_j \leq A$

Table 1: Notation for Boolean programming problems under consideration

of ρ is called a *worst-case ratio bound*. For a problem of Boolean programming of minimizing a function $Z(\mathbf{x})$, which may take negative and positive values, a vector $\mathbf{x}'$ is called an $\varepsilon-$*approximate solution* if for a given positive ε the inequality $Z(\mathbf{x}') - Z(\mathbf{x}^*) \leq \varepsilon |Z(\mathbf{x}^*)|$ 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 $1/\varepsilon$.

A detailed review on the design of FPTASs for problems relevant to this study is given by Kellerer and Strusevich (2012). Badics and Boros (1998) give the first FPTAS for Problem HP but its running time is $O(n^2 \log \sum \alpha_j/\varepsilon)$ is not strongly polynomial. The first FPTAS for Problem HP that requires strongly polynomial time $O(n^2/\varepsilon)$ is due to Erel and Ghosh (2008).

The algorithms that behave as an FPTAS for Problem HP to minimize a half-product $H(\mathbf{x})$ of the form (1) do not necessarily deliver an $\varepsilon-$approximate solution for the problem of minimizing the function $F(\mathbf{x})$ of the form (2), although both problems have the same optimal solution $\mathbf{x}^*$ and for any vector $\mathbf{x}$ the equality $F(\mathbf{x}) - F(\mathbf{x}^*) = H(\mathbf{x}) - H(\mathbf{x}^*)$ holds. This is due to the fact that $H(\mathbf{x}^*) < 0$ and it is possible that $|F(\mathbf{x}^*)| = |H(\mathbf{x}^*) + K| < |H(\mathbf{x}^*)|$. Starting from the pioneering work by Badics and Boros (1998), the matter of designing an FPTAS for the Half-product problem with an additive constants, including adapting an FPTAS for Problem HP, has initiated many publications, see, e.g., Janiak et al. (2005), Kubiak (2005), Erel and Ghosh (2008) and Kellerer and Strusevich (2012). In particular, addressing this issue Janiak et al. (2005) introduce the Positive Half-Product, i.e., Problem PosHP. However, the FPTAS that they develop in their paper for Problem PosHP still requires $O(n^2 \log \sum \alpha_j/\varepsilon)$ time. Erel and Ghosh (2008) present several conditions under which an FPTAS for Problem HP behaves as an FPTAS for Problem HPAdd; see, e.g., Lemma 2 in Section 2 of this paper.

As follows from the survey by Kellerer and Strusevich (2012), in all known applications of quadratic Boolean programming problems related to the Half-Product the objective function is convex. That is why we study Problem PosHPK and its relaxed version, Problem PosHP without a knapsack constraint, provided that the objective function $P(\mathbf{x})$ is convex. For each of these problems with a convex objective function, this paper delivers an FPTAS that requires $O(n^2/\varepsilon)$ time. This running time is much smaller than $O(n^4/\varepsilon^2)$ established by Kellerer and Strusevich (2010b) for Problem SQK (under additional assumptions that include the convexity of an objective function) or $O\left(n^4 \log\log n + n^4/\varepsilon^2\right)$ provided by Xu (2011) for an arbitrary Problem SQK. The developed FPTAS can be applied to several scheduling problems, resulting in improved approximation schemes for their solution.

2. What Is Needed to Design an FPTAS

In this section, we describe the ingredients that are needed in order to design an FPTAS for Problem PosHP and Problem PosHPK.

We start our consideration with Problem PosHPK of the form (4). As is often the case, the resulting FPTAS is obtained by modifying a dynamic programming (DP) algorithm for the problem. Such an algorithm is given below. Notice that the algorithm is rather straightforward and manipulates with the objective function (3) in a different way, compared to Janiak et al. (2005) who adapt a decomposition result by Badics and Boros (1998).

Define

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

and suppose that the values $x_1, x_2, \dots, x_k$ have been assigned. Our 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;

$y_k := \sum_{j=1}^{k} \alpha_j x_j$, the total weight of the items put into the knapsack.

We now give a formal statement and implementation details 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 DP

Step1. 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

$$(k-1, Z_{k-1}, y_{k-1}) \tag{6}$$

into the states of the form

$$(k, Z_k, y_k) \tag{7}$$

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 $y_{k-1} + \alpha_k \leq A$ holds. If feasible, the assignment $x_k = 1$ changes a state (6) to a state of the form (7), where

$$Z_k = Z_{k-1} + \beta_k y_{k-1} + \mu_k, \; y_k = y_{k-1} + \alpha_k. \tag{8}$$

(b) Define $x_k = 0$, which is always feasible. This assignment changes a state of the form (6) into the state of the form (7) such that

$$Z_k = Z_{k-1} + \nu_k; \; y_k = y_{k-1}. \tag{9}$$

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 $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ that leads to Z_n^*. Output $\mathbf{x}^*$ and $P(\mathbf{x}^*) = Z_n^*$.

Algorithm DP essentially uses the representation of the objective function in the form (3). Structurally, the algorithm is similar to the DP algorithms for various versions of the knapsack problems with quadratic objective functions that can be found in Kellerer and Strusevich (2006, 2010a, 2010b), and its running time does not exceed $O(nA)$.

Algorithm DP is the *first* ingredient we need to design an FPTAS for Problem PosHPK. As the *second* ingredient, we require an upper bound P^{UB}, such that for all instances of the Problem PosHPK the inequality $P^{UB}/P(\mathbf{x}^*) \leq R$ holds. Since $P(\mathbf{x}^*) \geq \frac{1}{R}P^{UB}$, it follows that $P_{LB} = \frac{1}{R}P^{UB}$ is a lower bound on $P(\mathbf{x}^*)$.

With both ingredients at hand, we can convert Algorithm DP into an approximation scheme. The running time of such an scheme will depend on the ratio R. To reduce the number of computed function values, we round the objective function values to multiples of a selected small number. Moreover, we disregard the generated states with the same rounded objective function value and keep only one, with the smallest current weight of the knapsack.

Algorithm EpsPosHPK

Step 1. Given an upper bound P^{UB} such that $P^{UB}/P(\mathbf{x}^*) \leq R$, define $P_{LB} := \frac{1}{R}P^{UB}$. For an arbitrary $\varepsilon > 0$, define $\delta := \frac{\varepsilon}{n}P_{LB}$. Define $v := \lceil P^{UB}/\delta \rceil = \lceil Rn/\varepsilon \rceil$. Split the interval $[0, P^{UB}]$ into subintervals $I_1, I_2, \ldots, I_v$ of length δ each, except possibly I_v which may be shorter.

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

(a) In line with Algorithm DP, 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 $\tilde{Z}_k \leq F^{UB}$, using the relations (8) and (9).

(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 $\mathbf{x}^{\varepsilon} = (x_1^{\varepsilon}, x_2^{\varepsilon}, \dots, x_n^{\varepsilon})$ that leads to Z^{ε}. Output $\mathbf{x}^{\varepsilon}$ and $P(\mathbf{x}^{\varepsilon})$ as an approximate solution of Problem PosHPK.

Below we briefly analyze the behavior of the algorithm.

Lemma 1. *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 $R = P^{UB}/P_{LB}$.*

Proof: Define $v := \lceil P^{UB}/\delta \rceil = \lceil Rn/\varepsilon \rceil$. In Step 2, moving from iteration $k-1$ to k, Algorithm EpsPosHPK creates at most $2v$ states of the form $(k, \tilde{Z}_k, \tilde{y}_k)$ from at most v kept states of the form $(k-1, Z_{k-1}, y_{k-1})$ and keeps at most v of them. Thus, for each k, Step 2 takes $O(v)$ time. Thus, the overall running time of Algorithm EpsPosHPK is $O(nv) = O(Rn^2/\varepsilon)$.

Let

$$(0, Z_0^*, y_0^*), (0, Z_1^*, y_1^*), \cdots, (0, Z_n^*, y_n^*)$$

denote the path of the states created by Algorithm DP that leads to the optimal solution of Problem PosHPK, i.e., $P(\mathbf{x}^*) = Z_n^*$.

In order to prove that Algorithm EpsPosHPK behaves as an FPTAS, we use induction to demonstrate that for each k, $1 \leq k \leq n$, the inequalities

$$Z_k \leq Z_k^* + k\delta; \tag{10}$$

$$y_k \leq y_k^* \tag{11}$$

hold.

For $k = 1$ Algorithm EpsPosHPK creates two states $(1, K + \mu_1, \alpha_1)$ and $(1, K + \nu_1, 0)$ which coincide with those created by the exact Algorithm DP.

Assume that (10) and (11) hold for all k, $1 \leq k \leq u-1 \leq n$, and prove they also hold for $k = u$. Take a state $(u-1, Z_{u-1}, y_{u-1})$ and create a new state $(u, \tilde{Z}_u, \tilde{y}_u)$. Of all states with the $Z-$values that belong to an interval I_q of length δ, Algorithm EpsPosHPK stores only one state, with the least filled knapsack. Thus, either $Z_u = \tilde{Z}_u$ and $y_u = \tilde{y}_u$ or

$$Z_u \leq \tilde{Z}_u + \delta, \; y_u \leq \tilde{y}_u. \tag{12}$$

Suppose that in the optimal path created by Algorithm DP the state (u, Z_u^*, y_u^*) has been found from the state $(u-1, Z_{u-1}^*, y_{u-1}^*)$ by applying the formula (8). Then consider $(u, \tilde{Z}_u, \tilde{y}_u)$ found in Algorithm EpsPosHPK by applying the formula (8) to state $(u-1, Z_{u-1}, y_{u-1})$, so that $\tilde{Z}_u = Z_{u-1} + \beta_u y_{u-1} + \mu_u$. We use (10) and (11) with $k = u-1$ to deduce

$$\tilde{Z}_u \leq Z_{u-1}^* + (u-1)\delta + \beta_u y_{u-1}^* + \mu_u = Z_u^* + (u-1)\delta,$$

and (10) with $k = u$ follows from (12). Besides,

$$y_u \le \tilde{y}_u = y_{u-1} + \alpha_u \le y^*_{u-1} + \alpha_u = y^*_u,$$

which proves (11) with $k = u$.

Suppose now that in the optimal path created by Algorithm DP the state (u, Z^*_u, y^*_u) has been found from the state $\left(u-1, Z^*_{u-1}, y^*_{u-1}\right)$ by applying the formula (9). Then consider $(u, \tilde{Z}_u, \tilde{y}_u)$ found in Algorithm EpsPosHPK by applying the formula (9) to state $(u-1, Z_{u-1}, y_{u-1})$, so that $\tilde{Z}_u = Z_{u-1} + \nu_u$. We use (10) and (11) with $k = u-1$ to deduce

$$\tilde{Z}_u \le Z^*_{u-1} + (u-1)\,\delta + \nu_u = Z^*_u + (u-1)\,\delta,$$

and (10) with $k = u$ follows from (12). Besides, $y_u \le \tilde{y}_u = y_{u-1} \le y^*_{u-1} = y^*_u$, which proves (11) with $k = u$.

To complete the proof, notice that $P\left(\mathbf{x}^{\varepsilon}\right)$ is equal to the smallest found value Z_n, and due to (10) for $k = n$ we have that $Z_n \le Z^*_n + n\delta \le P\left(\mathbf{x}^*\right) + \varepsilon P_{LB} \le (1+\varepsilon)\; P\left(\mathbf{x}^*\right)$.

■

Notice that an FPTAS by Kellerer and Strusevich (2010a, 2010b) that is designed to handle a more general Problem SQK is much more complicated, it is based on two versions of the DP algorithm, primal and dual, and uses intervals of variable length to reduce the number of stored states. The running time of an FPTAS by Kellerer and Strusevich (2010a, 2010b) can be expressed as $O\left(Rn^4/\varepsilon^2\right)$. A simpler structure and a faster time of Algorithm EpsPosHPK is due to the special shape of the objective function (3), in which all coefficients are non-negative and the quadratic terms of the form $(1-x_i)\,(1-x_j)$ are absent.

Algorithm EpsPosHPK works also for a less restricted Problem PosHP with no knapsack constraint, it suffices in Problem PosHPK to take a sufficiently large value A on the weight of the knapsack, e.g., $A = \sum_{j=1}^{n} \alpha_j$.

On the other hand, there is an alternative way of obtaining an FPTAS for Problem PosHP. Recall that Erel and Ghosh (2008) describe an algorithm, which we call *Algorithm EG*, that behaves as an FPTAS for Problem HPAdd with an objective function $F\left(\mathbf{x}\right)$ of the form (2), provided that lower and upper bounds on the optimal value of the objective function are available, so that $F_{LB} \le F\left(\mathbf{x}^*\right) \le F^{UB}$. Erel and Ghosh (2008) prove the following statement, formulated in their paper as Theorem 5.

Lemma 2. *[cf. Erel and Ghosh (2008)] Let* $\mathbf{x}^{\varepsilon}$ *be a vector found by Algorithm EG applied to Problem HPAdd to minimize a function* $F\left(\mathbf{x}\right)$ *of the form (2), with a lower bound* F_{LB} *and an upper bound* F^{UB} *on the optimal value. Then* $F\left(\mathbf{x}^{\varepsilon}\right) - F\left(\mathbf{x}^*\right) \le \varepsilon F_{LB}$ *and the running time of the algorithm is* $O(R'n^2/\varepsilon)$ *time, where* $R' = F^{UB}/F_{LB}$.

Lemmas 1 and 2 are very similar in nature and both rely on what we call the second ingredient, the presence of the upper bound on the optimal value of the function. It follows that Problem PosHPK and Problem PosHP admit an

FPTAS, provided that the ratios R and R' are bounded by a polynomial of the length of the input. In particular, if both R and R' are constants and can be found in at most $O(n^2)$ time, then each of the corresponding FPTAS's requires $O\left(n^2/\varepsilon\right)$ time.

As demonstrated in the following section, a $O(n^2)$-time constant ratio approximation algorithm can be developed for each Problem PosHP and Problem PosHPK, provided that the objective is convex.

3. Constant Ratio Approximation Algorithms

Our development of constant ratio approximation algorithms for Problem PosHP and Problem PosHPK consists of two phases. First, we solve the continuous relaxation of these problems. Second, we appropriately round the continuous solution to obtain a Boolean vector of decision variables and show that the value of the objective function is a constant factor away from the optimum. Notice, that in the first phase we need to assume that the objective function is convex.

A similar two-stage approach is applied in Kellerer and Strusevich (2010b), where an FPTAS for Problem SQK of the form (5) with a convex quadratic function is developed.

Solving of the continuous relaxations of Problem PosHP and Problem PosHPK with convex functions is done by reducing them to a problem of minimizing a quadratic cost flow function in a network of a special structure. The same approach is used in Kellerer and Strusevich (2010b) for Problem SQK with a convex objective. Since this approach involves a non-trivial reduction, in order to make this paper self-contained we have decided to provide the reader with its brief description, see Section 3.1.

The rounding procedure in Kellerer and Strusevich (2010b) is a constant ratio approximation algorithm for Problem SQK. There are two reasons why we cannot use it directly for the purposes of this paper:

- its running time is $O(n^3)$, which would not lead to an FPTAS of $O(n^2/\varepsilon)$ running time;
- the constant ratio is only guaranteed under the additional constraints

$$\nu_j \geq \alpha_j \beta_j,\ j = 1, \dots, n; \tag{13}$$

  see Theorem 2 of Kellerer and Strusevich (2010b).

Thus, the rounding procedure in Section 3.2 relies on different reasoning and algorithmic techniques, e.g., the use of a constant ratio approximation algorithm for a linear integer knapsack minimization problem.

3.1. Solving Continuous Relaxation

For a problem of Boolean programming, introduce 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 $0 \le x_j \le 1$, $j = 1, 2, \ldots, n$. For Problem PosHP and Problem PosHPK, denote their continuous relaxations by *Problem PosHPr* and *Problem PosHPKr*, respectively. Denote the vector that solves a relaxation problem 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.

Recall that our ultimate goal is to develop an FPTAS that for each Problem PosHP and Problem PosHPK requires $O(n^2/\varepsilon)$ time. Thus, for each Problem PosHPKr and Problem PosHPKr we need an algorithm that solves the corresponding problem in at most $O(n^2)$ time.

The idea is outlined below. Assume that function (3) is convex. Using the fact that for a Boolean variable $x_j = x_j^2$, $j \in N$, rewrite

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

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

$$\begin{array}{ll} \text{Minimize} & P(\chi) = \sum\limits_{i=1}^{n} c_i\chi_i \sum\limits_{j=1}^{i} \chi_j - \sum_{j=1}^{n} \gamma_j\chi_j + K' \\ \text{Subject to} & \sum_{j=1}^{n} \chi_j \le A \\ & 0 \le \chi_j \le \alpha_j,\ j = 1, 2, \ldots, n; \end{array}$$

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

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

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 (t_j, v_t) 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 $j = n, n-1, \dots, 3$; as $\frac{1}{2}c_j f^2 - \gamma'_j f$ for the arc that enters vertex t_j, $j = 2, 3, \dots, n$; as $(c_1 - \frac{1}{2}c_2)f^2 - \gamma'_1 f$ 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. See Kellerer and Strusevich (2010b, 2012) for a numerical example of the network.

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

Notice that function $P(\mathbf{x})$ of the from (3) is convex, provided that

$$\frac{\alpha_1}{\beta_1} \le \frac{\alpha_2}{\beta_2} \le \dots \le \frac{\alpha_n}{\beta_n}. \tag{14}$$

This follows from the fact that matrix

$$G = \begin{bmatrix} \alpha_1\beta_1 & \alpha_1\beta_2 & \cdots & \alpha_1\beta_n \\ \alpha_1\beta_2 & \alpha_2\beta_2 & \cdots & \alpha_2\beta_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1\beta_n & \alpha_2\beta_n & \cdots & \alpha_n\beta_n \end{bmatrix}$$

is positive semi-definite, as proved by Skutella (2001). As will be seen in Section 4, the condition (14) is an analogue of the famous WSPT rule, widely used in scheduling theory.

3.2. Rounding Algorithms

In this subsection, for each Problem PosHP and Problem PosHPK with the objective function of the form (3), the components of vector $\mathbf{x}^C$ can be appropriately rounded, so that for the resulting Boolean vector $\mathbf{x}^H = \left(x_1^H, x_2^H, \dots, x_n^H\right)$, the inequality $P\left(\mathbf{x}^H\right) \le \rho P\left(\mathbf{x}^*\right)$ holds, where ρ is a constant. Notice that our goal is not to design an algorithm with a particularly small worst-case ratio ρ; we are rather interested in the fact that a fast constant-ratio rounding algorithm can be developed for each problem.

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

$$\frac{1}{\lambda^2} = \frac{1}{1-\lambda} = \frac{3+\sqrt{5}}{2} = 2.618\ldots > 1.618... = \frac{1}{\lambda}. \tag{15}$$

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.

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

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

$$\sum_{j\in I} c_j y_j^H \leq 2 \sum_{j\in I} c_j y_j^*.$$

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

Step 3. Introduce the following auxiliary linear knapsack problem

$$\begin{array}{ll} \text{Minimize} & \sum_{j\in N_2} \nu_j y_j \\ \text{Subject to} & \sum_{j\in N_2} \alpha_j y_j \geq \sum_{j\in N_2} \alpha_j - A \\ & y_j \in \{0,1\},\ j \in N_2. \end{array} \tag{16}$$

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 (16).

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.

Theorem 2. *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*

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

Proof: It is clear that for a given vector $\mathbf{x}^C$, Algorithm PosHPKConstR requires $O(n \log n)$ time, since its most time consuming component is Step 3, and Algorithm GR takes $O(n \log n)$ time.

Notice that a solution represented by vector $\mathbf{x}^H$ is knapsack-feasible since $\sum_{j \in N_2} \alpha_j y_j^H \ge \sum_{j \in N_2} \alpha_j - A$ and

$$\begin{aligned}\sum_{j \in N} \alpha_j x_j^H &= \sum_{j \in N_1} \alpha_j x_j^H + \sum_{j \in N_2} \alpha_j x_j^H = \sum_{j \in N_2} \alpha_j x_j^H = \sum_{j \in N_2} \alpha_j (1 - y_j^H) \\ &= \sum_{j \in N_2} \alpha_j - \sum_{j \in N_2} \alpha_j y_j^H \le A.\end{aligned}$$

For vector $\mathbf{x}^H$, let us estimate the contributions to the quadratic part of the objective function. Take an arbitrary pair of items i and j, with $i < j$. If either $i \in N_1$ or $j \in N_1$ then

$$\alpha_i \beta_j x_i^H x_j^H = 0 \le \frac{1}{\lambda^2} \alpha_i \beta_j x_i^C x_j^C.$$

If both i and j are from set N_2 then $x_i^C > \lambda$ and $x_j^C > \lambda$, so that

$$\alpha_i \beta_j x_i^H x_j^H \le \alpha_i \beta_j \le \frac{1}{\lambda^2} \alpha_i \beta_j x_i^C x_j^C.$$

Thus,

$$\sum_{1 \le i < j \le n} \alpha_i \beta_j x_i^H x_j^H \le \frac{1}{\lambda^2} \sum_{1 \le i < j \le n} \alpha_i \beta_j x_i^C x_j^C. \tag{17}$$

Now we estimate the contributions to the linear part. For $j \in N_1$, we obtain

$$\begin{aligned}\mu_j x_j^H &= 0 \le \frac{1}{\lambda} \mu_j x_j^C < \frac{1}{1-\lambda} \mu_j x_j^C; \\ \nu_j (1 - x_j^H) &= \nu_j \le \frac{1}{1-\lambda} \nu_j \left(1 - x_j^C\right),\end{aligned}$$

so that

$$\sum_{j \in N_1} \mu_j x_j^H + \sum_{j \in N_1} \gamma_j (1 - x_j^H) \le \frac{1}{1-\lambda} \left(\sum_{j \in N_1} \mu_j x_j^C + \sum_{j \in N_1} \gamma_j \left(1 - x_j^C\right) \right)$$

For each $j \in N_2$ we have that

$$\mu_j x_j^H \le \mu_j \le \frac{1}{\lambda} \mu_j x_j^C < \frac{1}{1-\lambda} \mu_j x_j^C.$$

Besides,

$$\sum_{j\in N_2}\gamma_j(1-x_j^H)=\sum_{j\in N_2}\gamma_j y_j^H\le 2\sum_{j\in N_2}\gamma_j y_j^*,$$

where y_j^* is a component of an optimal solution vector for problem (16) in Step 3 of the algorithm. By definition, $\sum_{j\in N_2}\gamma_j y_j^*\le\sum_{j\in N_2}\gamma_j y_j$ for any feasible binary vector with the components y_j. For vector $\mathbf{x}^*$ that is optimal for Problem PosHPK, we deduce

$$\sum_{j\in N}\alpha_j x_j^*=\sum_{j\in N_1}\alpha_j x_j^*+\sum_{j\in N_2}\alpha_j x_j^*\le A,$$

so that

$$\sum_{j\in N_2}\alpha_j(1-x_j^*)\ge\sum_{j\in N_2}\alpha_j-A+\sum_{j\in N_1}\alpha_j x_j^*\ge\sum_{j\in N_2}\alpha_j-A.$$

This implies that the vector with the components $1-x_j^*$, $j\in N_2$, is feasible for problem (16) and therefore

$$\sum_{j\in N_2}\gamma_j y_j^*\le\sum_{j\in N_2}\gamma_j(1-x_j^*).$$

Thus,

$$\sum_{j\in N_2}\mu_j x_j^H+\sum_{j\in N_2}\gamma_j(1-x_j^H)\le\frac{1}{1-\lambda}\sum_{j\in N_2}\mu_j x_j^C+2\sum_{j\in N_2}\gamma_j(1-x_j^*).$$

Using (15), we deduce

$$\begin{aligned}
P(\mathbf{x}^H) &= \sum_{1\le i<j\le n}\alpha_i\beta_j x_i^H x_j^H+\sum_{j\in N_1}\mu_j x_j^H+\sum_{j\in N_1}\gamma_j(1-x_j^H)+\sum_{j\in N_2}\mu_j x_j^H\\
&\quad+\sum_{j\in N_2}\gamma_j(1-x_j^H)+K\\
&\le \frac{1}{\lambda^2}\sum_{1\le i<j\le n}\alpha_i\beta_j x_i^C x_j^C+\frac{1}{1-\lambda}\left(\sum_{j\in N}\mu_j x_j^C+\sum_{j\in N_1}\gamma_j\left(1-x_j^C\right)\right)\\
&\quad+2\sum_{j\in N_2}\gamma_j(1-x_j^*)+K\\
&= \frac{3+\sqrt{5}}{2}\left(\sum_{1\le i<j\le n}\alpha_i\beta_j x_i^C x_j^C+\sum_{j\in N}\mu_j x_j^C+\sum_{j\in N_1}\gamma_j\left(1-x_j^C\right)\right)\\
&\quad+2\sum_{j\in N_2}\gamma_j(1-x_j^*)+K\\
&\le \frac{3+\sqrt{5}}{2}P(\mathbf{x}^C)+2P(\mathbf{x}^*)\le\frac{7+\sqrt{5}}{2}P(\mathbf{x}^*),
\end{aligned}$$

as required. ∎

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 of Sections 2 and 3 as the following statement.

Theorem 3. *Each of 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 demonstrate the implications of Theorem 3 for various scheduling applications.

4. Scheduling Applications of Problems PosHP and PosHPK

In this section, we present a number of scheduling problems that can be formulated in terms of either Problem PosHP or Problem PosHPK. We show that for each of these problems an $\varepsilon-$approximate solution can be found in $O(n^2/\varepsilon)$ time, thereby improving most of the known results. A detailed review of the related issues is contained in Kellerer and Strusevich (2012). Also notice that the presented applications do not follow from the previously known FPTASs, e.g., for Problem SQK. In particular, this is because for all scheduling applications given below the conditions (13) need not hold.

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 $F(S)$ that depends on the completion times $C_j(S)$. For all problems under consideration S^* denotes an optimal schedule, i.e., $F(S^*) \leq F(S)$ 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

$$\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}. \tag{18}$$

We call the sequence of jobs numbered in accordance with (18) 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).

In our working presented below we often use the fact for a Boolean variable $x_j^2 = x_j$.

4.1. 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.

Consider 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 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. Extending standard scheduling notation, we denote this problem by $1\left|p_j = \overline{p}_j - y_j\right| \sum_{j\in N} w_j C_j + \sum_{j\in N} v_j y_j$.

For this problem,Vickson (1980) proves what is known as the “All-or-None” property: in an optimal schedule each job is either fully compressed, i.e., $p_j = \underline{p}_j$ or fully decompressed, i.e., $p_j = \overline{p}_j$.

We focus on a special case of the problem in which $\underline{p}_j = 0$. The resulting problem is 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 (18) for minimizing $\sum_{j\in N} w_j C_j$ on a single machine, 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) use problem $1\left|p_j = \overline{p}_j - y_j\right| \sum_{j\in N} w_j C_j + \sum_{j\in N} v_j y_j$ as an example of a scheduling problem for which reformulation in the form $F(\mathbf{x}) =$

$H(\mathbf{x})+K$ is possible, but $|H(\mathbf{x}^*)/F(\mathbf{x}^*)|$ can be arbitrary large, so that an FPTAS available for the corresponding Problem HP does not behave as an FPTAS for Problem HPAdd. In fact, namely problem $1\left|p_j=\overline{p}_j-y_j\right|\sum_{j\in N}w_jC_j+\sum_{j\in N}v_jy_j$ has motivated Janiak et al. to introduce the positive half-product. Following Janiak et al. (2005), introduce the Boolean decision variables

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

and renumber the jobs in non-decreasing order of $\overline{p}_j/w_j$.

Then job j completes at

$$C_j=\sum_{i=1}^{j}p_ix_i, \tag{19}$$

so that the objective function can be written as

$$\begin{aligned}\sum_{j=1}^{n}w_jC_j+\sum_{j=1}^{n}v_jy_j &= \sum_{j=1}^{n}w_jx_j\sum_{i=1}^{j}\overline{p}_ix_i+\sum_{j=1}^{n}\overline{p}_jv_j\left(1-x_j\right)\\ &= \sum_{1\le i<j\le n}^{n}\overline{p}_ix_ix_j+\sum_{j=1}^{n}\overline{p}_jw_jx_j+\sum_{j=1}^{n}\overline{p}_jv_j\left(1-x_j\right),\end{aligned}$$

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

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

It is shown in Janiak et al. (2005) that problem $1\left|p_j=\overline{p}_j-y_j\right|\sum_{j\in N}w_jC_j+\sum_{j\in N}v_jy_j$ admits 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. Erel and Ghosh (2008) give a faster FPTAS of the running time $O(n^2\log\left(\max\left\{p_j,w_j,n\right\}\right)/\varepsilon)$.

Due to the numbering of the jobs, the condition (14) holds, so that the objective function is convex. Thus, a direct application of Theorem 3 guarantees that problem $1\left|p_j=\overline{p}_j-y_j\right|\sum_{j\in N}w_jC_j+\sum_{j\in N}v_jy_j$ admits an FPTAS that requires $O(n^2/\varepsilon)$ time.

4.2. 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 for an optimal schedule S^* the makespan, i.e., the maximum completion time, is given by

$$C_{\max}(S^*)=\max_{1\le u\le n}\left\{r_u+\sum_{j=u}^{n}p_j\right\}. \tag{20}$$

Below, 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 $\bar{r}$.

Adapting the WSPT rule (18), we deduce that in an optimal sequence the early jobs will be sequenced in non-decreasing order of v_j/p_j. Renumber the jobs in this order, and introduce the Boolean decision variables

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

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

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

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

$$\alpha_j = v_j, \ \beta_j = p_j, \ \mu_j = p_j v_j, \ \nu_j = p_j, \ K = \bar{r}.$$

Interpreting the results of Janiak et al. (2005), this implies that the problem admits an FPTAS that runs either in $O\left(n^2 \log\left(\sum p_j\right)/\varepsilon\right)$ or $O\left(n^2 \log\left(\sum v_j\right)/\varepsilon\right)$ time.

Due to the numbering of the jobs, the condition (14) holds, so that the objective function is convex. Thus, a direct application of Theorem 3 guarantees that problem $1 \left| r_j \in [\underline{r}, \bar{r}] \right| C_{\max} + \sum_{j \in N} v_j y_j$ admits an FPTAS that requires $O(n^2/\varepsilon)$ time.

4.3. Scheduling with Rejection

Consider the following model of scheduling with rejection introduced by Engels 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 $1 \left| rej \right| \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j$.

In practice rejection decisions are often taken when the processing capabilities will not allow the completion of all jobs by a given deadline. Below, we introduce a restricted version of the problem not studied by Engels et al. (2003), in which all accepted jobs must be completed by a given time d. We denote this problem by $1 \left| rej, C_j \leq D \right| \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j$.

Engels et al. (2003) show that problem $1 \left| rej \right| \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j$ is NP-hard in the ordinary sense and 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.

Below we show that the objective function in each of the problems $1 \left| rej \right| \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j$ and $1 \left| rej, C_j \leq d \right| \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j$ is in fact a convex positive Half-Product, so that Theorem 3 is applicable.

As before, it follows from the optimality of the WSPT rule for minimizing the total weighted completion time $\sum_{j \in N} w_j C_j$ on a single machine that in an optimal sequence the accepted jobs will be sequenced in accordance with (18). 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 (19), and the objective function can be written as

$$\begin{aligned} \sum_{j \in N_A} w_j C_j + \sum_{j \in N_R} v_j &= \sum_{j=1}^{n} w_j x_j \sum_{i=1}^{j} p_i x_i + \sum_{j=1}^{n} v_j \left(1 - x_j\right) \\ &= \sum_{1 \leq i < j \leq n}^{n} p_i w_j x_i x_j + \sum_{j=1}^{n} p_j w_j x_j + \sum_{j=1}^{n} v_j \left(1 - x_j\right), \end{aligned}$$

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

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

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

$$\sum_{j=1}^{n} p_j x_j \leq d,$$

so that the problem can be seen as Problem PosHPK.

Due to the numbering of the jobs, the condition (14) holds, so that the objective function is convex. Thus, a direct application of Theorem 3 guarantees that each problem $1\,|rej|\sum_{j\in N_A} w_jC_j+\sum_{j\in N_R} v_j$ and $1\,|rej, C_j \le D|\sum_{j\in N_A} w_jC_j+\sum_{j\in N_R} v_j$ admits an FPTAS that requires $O(n^2/\varepsilon)$ time.

4.4. When Positive Half-Product Formulations are Impossible

There are other scheduling problems that can be formulated, e.g., as Problem PosHP; see Erel and Ghosh (2008) and Kellerer and Strusevich (2012). We do not discuss such problems here, since they are known to admit an FPTAS that requires $O(n^2/\varepsilon)$ time. On the other hand, there are problems that can be reformulated in terms of Problem HPAdd or Problem SQK, but not in terms of minimization of a positive Half-product.

As an illustration, we take the problem of minimizing the total weighted earliness and tardiness on a single machine. 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 $\sum_{j\in N} w_j\left(E_j\left(S\right)+T_j\left(S\right)\right)$.

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. Below we only focus on the version of the problem with a *large* or *nonrestrictive* due date, i.e., we assume that $d\ge\sum p_j$. We denote this problem by $1|\,|\sum w_j(E_j+T_j)$. 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). As proved by Hall and Posner (1991), for problem $1|\,|\sum w_j(E_j+T_j)$ 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. Renumber the jobs in accordance with the WSPT rule (18). In an optimal schedule the early jobs are processed in the order opposite to this numbering, while the jobs that start either at or after the due date are processed in the order of their numbering; see Hall and Posner (1991).

Kellerer and Strusevich (2010b) show that problem $1|\,|\sum w_j(E_j+T_j)$ can be formulated as Problem HPAdd. Introduce Boolean decision variables

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

Then

$$\begin{aligned}
\sum_{j=1}^{n} w_j(E_j + T_j) &= \sum_{1\le i<j\le n} p_i w_j x_i x_j + \sum_{1\le i<j\le n} p_i w_j (1-x_i)(1-x_j) \\
&+ \sum_{j=1}^{n} p_j w_j (1-x_j) \\
&= 2\sum_{1\le i<j\le n} p_i w_j x_i x_j - \sum_{j=1}^{n}\left(p_j \sum_{i=j}^{n} w_i + w_j \sum_{i=1}^{j} p_i - p_j w_j\right) x_j \\
&+ \sum_{i=1}^{n}\left(p_i \sum_{j=i}^{n} w_j\right),
\end{aligned}$$

so that this function can be seen as a function of the form (2) with

$$\alpha_j = p_j,\ \beta_j = 2w_j,\ \gamma_j = p_j \sum_{i=j}^{n} w_i + w_j \sum_{i=1}^{j} p_i - p_j w_j,\ K = \sum_{i=1}^{n}\left(p_i \sum_{j=i}^{n} w_j\right).$$

However, it can be verified the function cannot be written in the form (3) with all non-negative coefficients. This means that Theorem 3 is not applicable, and the best known FPTASs for the problem remain those developed by Erel and Ghosh (2008) and Kellerer and Strusevich (2010b) that require $O(n^2 \log(K)/\varepsilon)$ and $O(n^4/\varepsilon^2)$ time, respectively.

5. Conclusion

In this paper we demonstrate that the quadratic Boolean programming problem known as the positive Half-Product admits an FPTAS that requires $O(n^2/\varepsilon)$ time, provided that the objective function is convex. Moreover, if an additional linear knapsack constraint is added, the running time of the resulting FPTAS does not increase.

Notice that the factor n^2 in the running time estimate cannot be reduced, since computing the objective function for a given set of decision variables takes $O(n^2)$ time. The proposed fast FPTASs are applied to several scheduling problems, for which no FPTAS with a strongly polynomial time has been known.

It is an attractive goal to continue a search for sufficient conditions under which Problem HPAdd or Problem SQK admits an FPTAS that runs faster than in general case. Another interesting topic is to study Boolean programming problems with a quadratic objective function more general than a half-product, as well as quadratically constrained problems.

Acknowledgment

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

References

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

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

Engels, 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.

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

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.

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

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., Kellerer, H., & Strusevich, V.A. (2011). Single machine scheduling with a common due date: total weighted tardiness problems. In A.R. Ahjoub (Ed.), *Progress in Combinatorial Optimization* (pp. 391–421). London: ISTE-Wiley.

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

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 Science, 21*, 357–383.

Kellerer, H., & Strusevich, V.A. (2012). The symmetric quadratic knapsack problem: approximation and scheduling applications. *4OR – A Quarterly Journal of Operations Research, 10*, 111-161.

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.

Martello, S., & Toth, P. (1990). *Knapsack Problems. Algorithms and Computer Implementation.* Chichester: Wiley.

Nowicki, E., & Zdrzałka, S. (1990). A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics, 26*, 271–287.

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 ACM, 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.

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.

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