Noname manuscript No.
(will be inserted by the editor)

Scheduling of inventory releasing jobs to minimize a regular objective function of delivery times

Márton Drótos · Tamás Kis

Received: date / Accepted: date

Abstract In this note we provide new complexity and algorithmic results for scheduling inventory releasing jobs, a new class of single machine scheduling problems proposed recently by Boysen et al. We focus on tardiness related criteria, while known results are concerned with inventory levels between fixed delivery points. Our interest is motivated by the fact that deciding whether a feasible schedule exists is NP-hard in the strong sense, provided that all delivery deadlines are fixed, and there are no restrictions on the amount of products released by the jobs, nor on the job processing times. We will establish NP-hardness results, or provide polynomial or pseudo-polynomial time algorithms for various special cases, and describe a fully polynomial approximation scheme for one of the variants with the maximum tardiness criterion.

Keywords Machine scheduling · Inventory · Job tardiness · Computational complexity · Approximation scheme

1 Introduction

In this paper we study new variants of the problem of scheduling inventory releasing jobs, recently proposed by Boysen et al. [1]. The problem has been motivated by Just-In-Time manufacturing, where a set of inventory releasing jobs have to be sequenced on a single machine (or server) in order to meet a set of delivery requests. However, as Boysen et al. observed, with fixed deadlines it is NP-hard in the strong sense to decide whether a feasible solution exists even if there is only one product type, but the job sizes and the quantities of products released are arbitrary. Therefore, Boysen et al. consider special cases where the existence of feasible schedules can be decided in polynomial time, and they aim at minimizing stock level related criteria.

In this paper we focus on the problem of meeting the due-dates of deliveries, i.e., late delivery is permitted, and we seek a schedule which minimizes a regular function of the delivery completion times. An optimal or suboptimal solution can be used to set hard deadlines, and then in a second round one can apply the methods of Boysen et al. to minimize the stock levels. In a combined approach, one may consider the two types of criteria together in a multi-criteria optimization framework, but this is out of the scope of the present paper.

As a concrete application, consider a manufacturing workshop assembling and welding metal pieces together to satisfy customer orders. Each product consists of several pieces cut out from steel slabs. After some pieces are cut out from a steel slab, the remaining part becomes waste. This preparatory work transforms raw material into an inventory of various components. Cutting operations are inventory releasing jobs that may produce metal pieces (intermediate products) of several forms and sizes from the same metal sheet to economize on steel slabs. The due dates and inventory levels are dictated by an MRP system that computes the material requirements of the customer orders for the final assemblies over time. The scheduling models studied in this paper can handle a single cutting machine. Since pieces of various types may be cut out from the same steel slab, a cutting operation cannot be replaced by several cutting operations, and since the various components cannot replace one-another, this problem does not fit the framework of Boysen et al. [1], where a job always produces only one product type.

M. Drótos · T. Kis
Computer and Automation Research Institute, Hungarian Academy of Sciences, H1111 Budapest, Kende str. 13–17, Hungary
Tel.: +36 1 2796156; Fax: +36 1 4667503
E-mail: marton.drotos@sztaki.hu, tamas.kis@sztaki.hu

Related work

The problem of minimizing the inventory levels while satisfying all the external demands on time is studied in [1]. In that paper, the delivery requests have strict deadlines, and special cases where the existence of a feasible schedule is decidable in polynomial time are considered with the objective of minimizing some stock level related criteria. Polynomial algorithms or NP-hardness proofs are provided for several special cases.

The opposite problem, in which jobs consume non-renewable resources has been studied e.g. in [5,6,8]. In these models a job may be started only if the requirements of the job do not exceed the available quantities from each non-renewable resource. If a job is started, the available quantities of the non-renewable resources are instantly decreased by the requirements of the job. Moreover, non-renewable resources are supplied over time at given time points. For the special case when there are only non-renewable resource- and precedence constraints, Carlier and Rinnooy Kan [5] gave polynomial solution methods. If, in addition, the jobs have to be sequenced on a machine, the problem has been shown to be NP-hard [6]. In [8], the complexity of various sequencing problems on a single machine is studied subject to non-renewable resource constraints. The authors provide NP-hardness proofs, and approximation algorithms for the hard problems, or polynomial time algorithms for the tractable ones.

In a more general setting jobs may produce and consume a common set of materials, and a basic question is whether a feasible sequence of producers and consumers exists. This problem has been shown NP-hard in the strong sense by Kellerer et al. [9]. In the same paper, the authors consider the minimization of maximum stock level and propose three different approximation algorithms with relative error 2, 8/5, 3/2, respectively. Some numerical results complement the theoretical findings. This line of work has been taken up by Briskorn et al. [2], where several variants are studied, and either an NP-hardness proof is presented, or a polynomial time algorithm is devised. In [3], an exact branch-and-bound based method is developed and numerical results are provided for the general problem with 5 to 20 jobs.

Finally, Neumann and Schwindt [10] study project scheduling problems with inventory constraints. They analyze the feasible region of job starting times, and propose to add temporary constraints between pairs of jobs to resolve resource conflicts in a branch-and-bound algorithm.

2 Formal problem statement

There is a set of jobs J and a set of different product types S with $|J| = n^J$, and $|S| = n^S$. Each job $j \in J$ has a processing time $p_j > 0$, and produces an amount of $\tilde{c}^s_j \geq 0$ from product $s \in S$. It is permitted that the same job produces a positive amount from several distinct product types. The inventory level of each product is 0 initially. The inventory of products is consumed by a set of deliveries R with $|R| = n^R$. Each delivery R_r has a due-date d_r, and specifies a quantity of $\tilde{e}^s_r$ for each product type $s \in S$ to be withdrawn from the corresponding inventory. Like in the case of jobs, a delivery may specify positive requests for several product types simultaneously. It is assumed that $d_1 \leq d_2 \leq \cdots \leq d_{n^R}$. A delivery can be served only if the inventory level of each product type is not below the requested quantity. Hence, a delivery can be *tardy*. The delivery requests must be served in increasing due-date order, and in case of ties, their order is fixed in advance. The indexing of delivery requests indicates the order in which they have to be served.

Let π be a sequence of the n^J jobs, i.e., $\pi(u) \in J$ is the job in position u ($1 \leq u \leq n^J$), and $\pi(u) \neq \pi(v)$ for $u \neq v$. The jobs in π are executed without idle times. The total amount of product type $s \in S$ after completing the first h jobs in π is $\sum_{u=1}^{h} \tilde{c}^s_{\pi(u)}$, whereas the total requested quantity over the first r delivery requests is $e^s_r = \sum_{k=1}^{r} \tilde{e}^s_k$. Let h^π_r be the minimum number of jobs needed to reach the required inventory level by delivery r in job sequence π, i.e., $h^\pi_r := \min\left\{h \mid \sum_{u=1}^{h} \tilde{c}^s_{\pi(u)} \geq e^s_r, \text{ for all } s \in S\right\}$. The *completion time* of delivery r is $C^\pi_r := \sum_{u=1}^{h^\pi_r} p_{\pi(u)}$ (there are no idle time between the jobs). If $C^\pi_r \leq d_r$, then the delivery is *on time*, otherwise it is *tardy*. The *lateness* and *tardiness* of delivery r are defined as $L^\pi_r = C^\pi_r - d_r$ and $T^\pi_r := \max(0, L^\pi_r)$, respectively. The *completion time of job* $\pi(h)$ in a sequence π is $\tilde{C}^\pi_h = \sum_{k=1}^{h} p_{\pi(k)}$. The *total amount of product type s* produced up to time t in schedule π is $A^s_\pi(t) = \sum_{h|\tilde{C}^\pi_h \leq t} \tilde{c}^s_{\pi(h)}$. We will consider regular (non-decreasing) objective functions γ in the delivery completion times, e.g., $\max_r T_r$, $\max_r L_r$, $\sum_r w_r T_r$, or $\sum_r w_r L_r$. Let $\gamma(\pi)$ denote the objective function value of a schedule π. Since π and the number of delivery requests are finite, γ is well defined.

Without loss of generality, we may assume that the due-date of the last delivery satisfies $d_{n^R} \geq \sum_j p_j$, and $e^s_{n^R} = \sum_j \tilde{c}^s_j$ for all $s \in S$. Namely, if $d_{n^R} < \sum_j p_j$, we can add a new delivery request to R, and set its due-date to $\sum_j p_j$. On the other hand, by that time, all the products are delivered, hence we lose no solution by assuming $e^s_{n^R} = \sum_{j \in J} \tilde{c}^s_j$ for $s \in S$.

An illustrative example is given in Fig. 1.

Our results

Our model extends that of Boysen et al. by permitting that a delivery request concerns the joint delivery of several products, and that a single job may release products of several types to the respective product inventories at the same time. We will study various special cases of the problem with regular objective function of the delivery dates. For two deliveries, $n^R = 2$, and for arbitrary different product types (n^S

		$n^S = 1$		$n^S \geq 2$, const.		$n^S = *$	
		$\tilde{c}_j^s = \tilde{c}^s$	$\tilde{c}_j^s = *$	$\tilde{c}_j^s = \tilde{c}^s$	$\tilde{c}_j^s = *$	$\tilde{c}_j^s = \tilde{c}^s$	$\tilde{c}_j^s = *$
$n^R \geq 2$, const.	$p_j = 1$	P (2)	P (1)	P (2)	o.NP (3, 8)	P (2)	s.NP (5)
	$p_j = p$	P (2)	P (1)	P (2)	o.NP (3, 8)	P (2)	s.NP (5)
	$p_j = *$	P (2)	o.NP (7, 8)	P (2)	o.NP (3, 8)	P (2)	s.NP (5)
$n^R = *$	$p_j = 1$	P (2)	P (1)	P (2)	s.NP (6)	P (2)	s.NP (5)
	$p_j = p$	P (2)	P (1)	P (2)	s.NP (6)	P (2)	s.NP (5)
	$p_j = *$	P (2)	s.NP (4)	P (2)	s.NP (6)	P (2)	s.NP (5)

Table 1 Overview of the complexity results for $1||\{\gamma, \gamma^T\}$. Each cell of the table corresponds to a variant of the problem with additional restrictions; a star $(*)$ indicates that the problem parameter can be arbitrary positive integer value. The complexity can be polynomial (P), NP-hard in the ordinary sense (o.NP), or NP-hard in the strong sense (s.NP). The objective function is any regular function γ in the delivery completion times for those problems of polynomial time complexity, and it is any regular function γ^T in the delivery tardiness times for all NP-hard problems in the table. Numbers after the complexity class (in parenthesis) refer to the corresponding theorem(s).

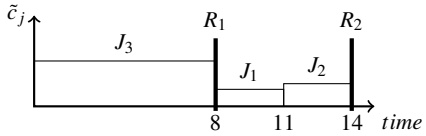
Parameters:

Param.	Value
n^J	3
n^S	1
n^R	2

Job	p_j	$\tilde{c}_j$
J_1	3	3
J_2	3	4
J_3	8	8

Delivery	d_r	e_r
R_1	7	6
R_2	14	15

Schedule $\pi(J_3, J_1, J_2)$:



Total production of $\pi(J_3, J_1, J_2)$:

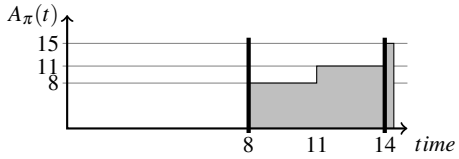


Fig. 1 Example of the scheduling problem. The boxes in the schedule represent the jobs; the length and height of job j is proportional to p_j and $\tilde{c}_j$, respectively. The depicted schedule has a maximal tardiness of 1 time unit.

arbitrary), we show that it is NP-hard in the strong sense to decide whether a solution with maximum tardiness of 0 exists. Notice that when $n^S = 1$, and n^R is part of the input, then the same decision problem is known to be NP-hard in the strong sense, see Proposition 7 in [1]. We will also consider the special cases with $p_j = 1$ (unit processing times), or $p_j = p$ (equal processing times), and $\tilde{c}_j^s = \tilde{c}^s$ (equal production quantities per product type). On the other hand, if $n^S \geq 1$ and $n^R \geq 2$ are fixed constant, we show that the problem is solvable in pseudo-polynomial time. The dynamic program also yields an FPTAS for the $n^S = 1$, $n^R = 2$ special case by using standard techniques along with some additional tricks. Our complexity results are summarized in Table 1.

We conclude this section with a summary of notation used throughout the paper.

Notation and Terminology

J	=	set of jobs
S	=	set of product types
R	=	set of delivery requests
n^J	=	number of jobs
n^S	=	number of products; products are indexed by s
n^R	=	number of delivery requests; deliveries are indexed by r
J_j	=	job j
p_j	=	processing time of job j
$\tilde{c}_j^s$	=	amount of product s released by job j
e_r^s	=	total amount of product s to be delivered up to delivery r
d_r	=	due date of delivery r, $d_0 = 0$
π	=	sequence of jobs
C_r^π	=	completion time of delivery r in schedule π
T_r^π	=	tardiness of delivery r
L_r^π	=	lateness of delivery r
$\tilde{C}_h^\pi$	=	completion time of job $\pi(h)$ in schedule π
$A_\pi^s(t)$	=	total amount of product type s produced up to time t in schedule π

Index s may be omitted if $n^S = 1$.

We will use the term *slot* r to refer to the time interval $[C_{r-1}, \ldots, C_r]$ (even when the completion time of the deliveries may not be fixed yet). Note that although this problem is defined as a sequencing problem, the sequence of the jobs between two consecutive deliveries does not affect the objective function value. This means that it is enough to give an assignment of the jobs to the slots defined by the deliveries, then those jobs that share the same slot can be sequenced arbitrarily.

3 Complexity results

In the following results γ is any regular function of the delivery completion times. Recall that the restriction $p_j = p$ means that all jobs have a common processing time, and $\tilde{c}^s_j = \tilde{c}^s$ indicates that all jobs produce a quantity of $\tilde{c}^s$ from each product type $s \in S$.

Theorem 1 *The problem $1|p_j = p, n^S = 1|\gamma$ can be solved in $O(n^J \log n^J)$ time.*

Proof We show that ordering the jobs in non-increasing order of the $\tilde{c}_j$ values gives an optimal schedule. Suppose that there is an optimal schedule π, where for two adjacent jobs $\pi(l)$ and $\pi(l+1)$, $\tilde{c}_{\pi(l)} < \tilde{c}_{\pi(l+1)}$. Let π' be the schedule obtained by swapping the jobs $\pi(l)$ and $\pi(l+1)$. Since $\tilde{c}_{\pi(l)} < \tilde{c}_{\pi(l+1)}$, swapping the two jobs ensures that for any time t, $A_{\pi'}(t) \geq A_{\pi}(t)$. Hence, for every $r \in R$, the completion time of delivery r in the two schedules satisfy $C^{\pi'}_r \leq C^{\pi}_r$. Hence, $\gamma(\pi') \leq \gamma(\pi)$. □

Theorem 2 *The problem $1|\tilde{c}^s_j = \tilde{c}^s|\gamma$ can be solved in $O(n^J \log n^J)$ time.*

Proof An exchange argument similar to that in the proof of Theorem 1 shows that ordering the jobs in non-decreasing processing time order (SPT rule) gives an optimal schedule. □

Let γ^T be any regular function of the tardiness of deliveries, such that $\gamma^T(\pi) = 0$ if and only if $T^{\pi}_r = 0$ for all $r \in R$.

In order to prove the NP-hardness of $1|p_j = 1, n^S = 2, n^R = 2|\gamma^T$, we will need the NP-hard E-kKP problem [4], which is defined next:

Definition 1 The Exact k-item Knapsack Problem (E-kKP) is the following: given a finite set U, for each $u \in U$ two sizes $v_1(u), v_2(u) \in \mathbb{Z}^+$, and positive integers B_1, B_2 and k, is there a subset $V \subseteq U$ such that $\sum_{u\in V} v_1(u) \geq B_1$, $\sum_{u\in V} v_2(u) \leq B_2$ and $|V| = k$?

For our purposes we define the minimization variant as follows:

Definition 2 The Exact k-item Minimum Knapsack Problem (E-kMKP) is the following: given a finite set U, for each $u \in U$ two sizes $v_1(u), v_2(u) \in \mathbb{Z}^+$, and positive integers B_1, B_2 and k, is there a subset $V \subseteq U$ such that $\sum_{u\in V} v_1(u) \leq B_1$, $\sum_{u\in V} v_2(u) \leq B_2$ and $|V| = k$?

The E-kKP and E-kMKP are equivalent problems, which can be seen as follows[1]. Suppose we are given an instance of E-kKP, the corresponding instance of E-kMKP contains the same data, except the $v_1(u)$, $u \in U$, and B_1, which are redefined for E-kMKP from the data of E-kKP as follows: $v^{\min}_1(u) = M - v_1(u)$, $u \in U$, and $B^{\min}_1 = kM - B_1$, where $M = \max_{u\in U} v_1(u)$. Observe that $\sum_{u\in V} v_1(u) \geq B_1$ for some $V \subset U$ with $|V| = k$ if and only if $\sum_{u\in V} v^{\min}_1(u) \leq B^{\min}_1$.

[1] To use E-kKP for proving the NP-hardness of E-kMKP was suggested by a reviewer.

Theorem 3 *The problem $1|p_j = 1, n^S = 2, n^R = 2|\gamma^T$ is NP-hard in the ordinary sense.*

Proof Consider an instance $\mathscr{I}$ of E-kMKP. From this instance, we construct an instance of the scheduling problem as follows:

Param.	Value	Param.	Value
n^J	$\lvert U\rvert$	e^1_1	$\sum_{u\in U} v_1(u) - B_1$
n^S	2	e^2_1	$\sum_{u\in U} v_2(u) - B_2$
n^R	2	e^s_2	$\sum_j \tilde{c}^s_j$
p_j	1	d_1	$\lvert U\rvert - m$
$\tilde{c}^1_j$	$v_1(u)$	d_2	$\lvert U\rvert$
$\tilde{c}^2_j$	$v_2(u)$		

and the question is if there exists a schedule π with $\gamma^T(\pi) = 0$.

The idea of the reduction is that by limiting the amount of the products released after the first delivery, we ensure the release of enough products before it, while the tardiness is controlled by fixing the number of jobs in each of the two slots.

First suppose that $\mathscr{I}$ admits a feasible solution V. Then the schedule

$$\pi(\{J_u \mid u \in U \setminus V\}, \{J_u \mid u \in V\})$$

satisfies $\gamma^T(\pi) = 0$, because $C_1 \leq d_1$, since

$$A^1_\pi(d_1) = A^1_\pi(|U| - m) = \sum_{u\in U\setminus V} v_1(u) = \sum_{u\in U} v_1(u) - \sum_{u\in V} v_1(u) \geq \sum_{u\in U} v_1(u) - B_1,$$

and similarly $A^2_\pi(d_1) \geq \sum_{u\in U} v_2(u) - B_2$.

In the opposite direction, if there is a solution of the scheduling problem with no tardiness, then at least m jobs are performed after the first delivery with a total release of at most B_1 and B_2 products, respectively. Hence, any subset of size m of these jobs define an appropriate set V in $\mathscr{I}$. □

Theorem 4 *The problem $1|n^S = 1|\gamma^T$ is NP-hard in the strong sense.*

The proof of this result is identical to that of Proposition 7 in [1].

Theorem 5 *The problem $1|n^R = 2, p_j = 1|\gamma^T$ is NP-hard in the strong sense.*

Proof Consider an instance $\mathscr{I}$ of X3C: given set X with $|X| = 3q$ and a collection C of 3-element subsets of X, find a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'.

From the above instance, we construct an instance of the scheduling problem as follows:

Param.	Value	Param.	Value
n^J	$\|C\|$	e_1^s	1
n^S	$3q$	e_2^s	$\sum_j \tilde{c}_j^s$
n^R	2	d_1	q
p_j	1	d_2	$\|C\|$
$\tilde{c}_j^s$	$\begin{cases} 1 & \text{if } s \in X_j \\ 0 & \text{otherwise} \end{cases}$		

and the question is if there exists a schedule π with $\gamma^T(\pi) = 0$.

If there exists an exact cover $C' = \{X_i : i = 1 \dots q\}$, then

$$\pi(J_1, \dots, J_q, J_{q+1}, \dots, J_{n^J})$$

is clearly feasible, and has an objective function value of $\gamma^T(\pi) = 0$, since no delivery is tardy, i.e., $T_r^\pi = 0$ for all $r \in R$.

If there exists a schedule π with no tardiness, then $C_1^\pi = d_1 = q$, since it is impossible to produce an amount of $\sum_{s\in S} e_1^s = 3q$ products in less than q time. Furthermore, $\sum_{s\in S} A_\pi^s(q)$ is exactly $3q$, and all these products are different because of the definition of the first delivery. This means that the subsets $X_{\pi(1)}, \dots, X_{\pi(q)} \in C$ form an exact cover of X. □

Theorem 6 *The problem* $1|n^S = 2, p_j = 1|\gamma^T$ *is NP-hard in the strong sense.*

Proof [2] Consider an instance $\mathscr{I}$ of 3-PARTITION: given integers $a_1, a_2, \dots, a_{3m}, B$ such that $B/4 < a_j < B/2$ and $\sum_{j=1}^{3m} a_j = mB$, find partition $\mathscr{A} = (A_1 : A_2 : \cdots : A_m)$ such that

- $\sum_{j\in A_r} a_j = B, r = 1, 2, \dots, m$ and
- $A_1 \cup A_2 \cup \cdots \cup A_m = \{a_1, a_2, \dots, a_{3m}\}$ and $A_{r_1} \cap A_{r_2} = \emptyset$ for $r_1 \neq r_2$

From the above instance, we construct an instance of the scheduling problem as follows:

Param.	Value	Param.	Value
n^J	$3m$	$\tilde{c}_j^1$	$mB - a_j$
n^S	2	$\tilde{c}_j^2$	a_j
n^R	m	e_r^1	$r(3mB - B)$
p_j	1	e_r^2	rB
		d_r	$3r$

[2] The proof presented here is a simplified version of our original proof and was suggested by a reviewer.

and the question is if there exists a schedule π with $\gamma^T(\pi) = 0$.

If there exists a partition $\mathscr{A} = (A_1 : A_2 : \cdots : A_m)$ for $\mathscr{I}$, then after re-indexing the elements such that $A_r = \{a_{3r-2}, a_{3r-1}, a_{3r}\}$, the schedule

$$\pi(J_1, J_2, J_3, \dots, J_{3r-2}, J_{3r-1}, J_{3r}, \dots, J_{3m-2}, J_{3m-1}, J_{3m})$$

satisfies $C_r^\pi \leq d_r$ for all $r \in R$. Hence, $\gamma^T(\pi) = 0$.

Conversely, suppose there exists a schedule π with $C_r^\pi \leq d_r$ for all $r \in R$, meaning that $A_\pi^1(d_r) \geq r(3mB - B) = e_r^1$ and $A_\pi^2(d_r) \geq rB = e_r^2$. We prove by induction on r that π has such a structure that by the first delivery request, and between any two consecutive delivery requests, there are exactly three jobs that release a total of $3mB - B$ from product 1, and B from product 2, which implies that a feasible schedule yields a feasible solution for the 3-PARTITION problem.

The base case $r = 1$. Since $d_1 = 3$, and each job has processing time 1, there are at most three jobs scheduled before the first delivery request. Since the schedule is feasible by assumption, we have $A_\pi^2(d_1) \geq B$. This implies that there are exactly three, since $\tilde{c}_j^2 = a_j$ and $B/4 < a_j < B/2$. Let A_1 be the set of these three jobs. In addition, $A_\pi^1(d_1) \geq 3mB - B$ also holds. Since $\tilde{c}_j^1 = mB - a_j$, we have $3mB - B \leq A_\pi^1(d_1) = \sum_{j\in A_1} \tilde{c}_j^1 = \sum_{j\in A_1}(mB - a_j)$, which implies $\sum_{j\in A_1} a_j \leq B$. Consequently, $\sum_{j\in A_1} a_j = B$.

The inductive step. Suppose our claim is true for the first $r < m$ delivery requests. This implies $A_\pi^1(d_r) = r(3mB - B) = e_r^1$ and $A_\pi^2(d_r) = rB = e_r^2$. Therefore, since the schedule is feasible, in slot $r+1$ the jobs have to produce a quantity of at least $e_{r+1}^1 - e_r^1 = 3mB - B$ from product 1 and $e_{r+1}^2 - e_r^2 = B$ from product 2. We can use exactly the same argument as in the base case to show that there are precisely three jobs scheduled in slot $r+1$ that produce a quantity of $3mB - B$ from product 1 and B from product 2. This proves our claim. □

Theorem 7 *The problem* $1|n^S = 1, n^R = 2|\gamma^T$ *is NP-hard in the ordinary sense.*

Proof Consider an instance $\mathscr{I}$ of PARTITION: given a finite set A and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$, find a subset $A' \subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(A)$.

From the above instance, we construct an instance of the delivery problem as follows:

Param.	Value	Param.	Value
n^J	$\|A\|$	p_j	$s(a_j)$
n^S	1	$\tilde{c}_j$	$s(a_j)$
n^R	2	e_r	$r\frac{\sum_{a\in A} s(a)}{2}$
		d_r	$r\frac{\sum_{a\in A} s(a)}{2}$

and the question is if there exists a schedule π with $\gamma^T(\pi) = 0$.

Suppose $(A' : A - A')$ is a solution of $\mathscr{I}$. Then by reindexing the items we may assume that $A' = \{1,\dots,k\}$, and $A - A' = \{k+1,\dots,n^J\}$. The schedule

$$\pi(J_1,\dots,J_k,J_{k+1},\dots,J_{n^J})$$

is clearly feasible, and for all $r = 1,2$: $e_r = d_r$, hence $\gamma^T(\pi) = 0$.

Conversely, if there exists a schedule π with no tardiness, clearly all deliveries are performed exactly at their due dates (for any time t, $A_\pi(t) \le t$ because $p_j = \tilde{c}_j$). This means that for $r = 1,2 : C_r = r \cdot \frac{\sum_{a\in A} s(a)}{2}$, hence the jobs completing before and after d_1, respectively, define a feasible partition $(A' : A - A')$ in $\mathscr{I}$. □

Theorem 8 *The problem $1|n^S = const., n^R = const.|\gamma$ can be solved in pseudo-polynomial time.*

Proof The main idea of the following algorithm is that we enumerate all different assignments of jobs to (delivery) slots. Two assignments are *different*, if there exists a slot r such that the total processing time or the total production of some product of those jobs assigned to slot r differ in the two assignments. From an assignment a schedule can be easily built by sequencing those jobs assigned to the same slot arbitrarily and then by joining the pieces together. We call an assignment feasible only if the set of jobs assigned to the first r slots satisfies the demand e_r^s for each product $s \in S$, and delivery request $r \in R$.

We construct a directed graph as follows. Let a node $N(j,P_1,\dots,P_{n^R},\Delta_1^1,\dots,\Delta_{n^R}^{n^S})$ represent a partial (not necessarily feasible) solution where jobs $1,\dots,j$ are already assigned to slots, slot r has P_r total processing time, and the amount of product s produced in it is Δ_r^s. Let $N(0,\dots,0)$ be the start node. From each node $N(j,\dots),(j \ne n^J)$ exactly n^R edges are directed outwards, and these edges are labeled with the numbers $1,\dots,n^R$. An edge with label r from node $N(j,P_1,\dots,P_r,\dots,P_{n^R},\Delta_1^1,\dots,\Delta_r^s,\dots,\Delta_{n^R}^{n^S})$ is directed to the following node: $N(j+1,P_1,\dots,P_r + p_{j+1},\dots,P_{n^R},\Delta_1^1,\dots,\Delta_r^s + \tilde{c}_{j+1}^s,\dots,\Delta_{n^R}^{n^S})$, and represents the choice of assigning job $j+1$ to slot r, given the previous jobs are already assigned.

Nodes $N(n^J,\dots)$ represent all the different assignments of the jobs to slots, and are called *terminal nodes*. Notice that the same terminal node may represent several job sequences. On the one hand, there can be several directed paths in the graph from the start node to the same terminal node, and each path gives rise to a distinct assignment of jobs to slots. On the other hand, even a single path may yield several job sequences, as we can order the jobs in a slot arbitrarily. We call the set of job sequences that can be obtained in this way the *realizations* of the terminal node. A terminal node is *feasible* if $\forall s \in S, r \in R : \sum_{r'=1}^{r} \Delta_{r'}^s \ge e_r^s$ (all demands of delivery r are satisfied). In the sequel we consider only feasible terminal nodes. We define the *delivery completion times* of a feasible terminal node $N(n^J,P_1,\dots,P_r,\Delta_1^1,\dots,\Delta_{n^R}^{n^S})$ as $C_r^{N(n^J,\dots)} = \sum_{r'=1}^{r} P_{r'}$ for $r \in R$. Now let the job sequence π be a realization of this terminal node. Then we have $C_r^\pi \le C_r^{N(n^J,\dots)}$ for all $r \in R$, and we may have strict inequality for some r (when more jobs are assigned to slot r than it is necessary to meet the demands e_r^s for $s \in S$).

Clearly, any optimal job sequence π^* is represented by a feasible terminal node (just assign those jobs finishing by $C_1^{\pi^*}$ to the first slot, and those jobs finishing later than $C_{r-1}^{\pi^*}$, but not later than $C_r^{\pi^*}$ to slot r: the corresponding terminal node is clearly feasible and is reachable from the start node). For such a terminal node $N(n^J,P_1,\dots,P_r,\Delta_1^1,\dots,\Delta_{n^R}^{n^S})$, we always have $C_r^{\pi^*} = C_r^{N(n^J,\dots)}$ for each $r \in R$, so any optimal solution is represented by a terminal node of minimum objective function value $\gamma(C_1,\dots,C_{n^R})$, where $C_r = C_r^{N(n^J,\dots)}$. Therefore, to solve the problem it is enough to identify those feasible terminal nodes reachable from the start node, and the optimal solution will be represented by one with the lowest objective function value. Finally, an optimal solution can be recovered by following any path backward from an optimal terminal node to the unique start node.

Complexity of the algorithm:

- Nodes of the graph:
  $|V| \le n^J \left|\sum p_j\right|^{n^R} \left|\sum \tilde{c}_j^1\right|^{n^R} \dots \left|\sum \tilde{c}_j^{n^S}\right|^{n^R} + 1.$
- Edges of the graph: $|E| \le n^R|V|$.
- Checking the feasibility of all solutions takes $O(n^R n^S|V|) = O(|V|)$ time.
- Calculating the objective function for all feasible solutions takes $O(n^R|V|) = O(|V|)$ time.
- Building the graph takes $O(|V| + |E|)$ time, since for each node we can maintain two lists of edges: one for the inbound edges, and another for the outbound edges, and a node $N(j,P_1,\dots,P_{n^R},\Delta_1^1,\dots,\Delta_r^s,\dots,\Delta_{n^R}^{n^S})$ is uniquely identified by the parameters j, $P_1,\dots,P_{n^R}$, $\Delta_1^1,\dots,\Delta_r^s,\dots,\Delta_{n^R}^{n^S}$.
- Recovering an optimal solution takes $O(n^J)$ time by following any path backward from an optimal terminal node.

The construction of the graph and the solution of the problem are polynomial if the input is given in unary encoding, so this is indeed a pseudo-polynomial algorithm for the problem. □

4 An FPTAS for $1|n^S = 1, n^R = 2|T_{\max}$

In this section we describe a Fully Polynomial Time Approximation Scheme (for basic definitions on approximation schemes, see e.g. [11]) for $1|n^S = 1, n^R = 2|T_{\max}$, which is NP-hard in the ordinary sense by Theorem 7.

The idea of the FPTAS is that the feasible domain Λ of the problem is partitioned into some subdomains, and each subdomain λ is represented by a feasible solution. This approximation method is called as „structuring the output" in [11].

As the maximal tardiness can be zero, and deciding whether a schedule with no tardiness exists is NP-complete, it would be impossible to give an FPTAS to the problem unless P=NP. Hence the objective function is modified in the well-known manner: we will consider the shifted maximal tardiness of schedule $T^s_{\max} = \max_r(\max(C_r - d_r, 0) + p_{sum})$. This means that after each delivery a "lag time" is required. As a result, we always have $p_{sum} \le T^s_{\max} \le 2p_{sum}$.

We will round up the processing times and the amount of produced products to the nearest integer divisible by a constant K and L, respectively. The resulting values will be $\bar{p}_j = K\left\lceil \frac{p_j}{K} \right\rceil$ and $\bar{c}_j = L\left\lceil \frac{\tilde{c}_j}{L} \right\rceil$. We will use the directed acyclic graph described in the proof of Theorem 8 with additional weights on the edges. A node in this formulation will have the following structure: $\bar{N}(j, \bar{P}_1, \bar{P}_2, \bar{\Delta}_1, \bar{\Delta}_2)$, and the edge representing the assignment of job j to the first slot will have a weight $\bar{\Delta}_j - \Delta_j$. The weight of other edges is zero. The subdomain $\lambda_{\bar{N}(n^J, \bar{P}_1, \bar{P}_2, \bar{\Delta}_1, \bar{\Delta}_2)}$ contains all schedules that have $\bar{P}_1$ total rounded processing time in slot 1, $\bar{\Delta}_1$ total rounded produced amount in slot 1, etc. These schedules correspond to all paths leading from the start node to $\bar{N}(n^J, \bar{P}_1, \bar{P}_2, \bar{\Delta}_1, \bar{\Delta}_2)$.

A reachable terminal node of the rounded problem is transformed into a solution of the scheduling problem by replacing the rounded jobs with the original ones. As a solution σ of the scheduling problem is determined by a path from the start node to a terminal node, a terminal node in the rounded problem may represent multiple different solutions in the original problem (see Figure 2). The weight of σ, denoted by $w(\sigma)$, is the sum of edge weights of the path. Notice that if a path ends at $\bar{N}(n^J, \bar{P}_1, \bar{P}_2, \bar{\Delta}_1, \bar{\Delta}_2)$, then $\bar{\Delta}_1 - w(\sigma)$ is the total production of the product in the first slot in the corresponding non-rounded solution. Among the paths to a terminal node we select one of lowest weight. This will be called as the *representative* solution of the given node.

Definition 3 A terminal node in the rounded problem is called *feasible* if its representative non-rounded solution is feasible. Its objective function value is the objective function value of the non-rounded representative solution.

The algorithm searches all feasible terminal nodes in the rounded problem, and chooses the solution with the lowest maximal shifted tardiness. We will need some definitions for the different objective function values for a given subdomain λ.

Definition 4 For a subdomain λ,

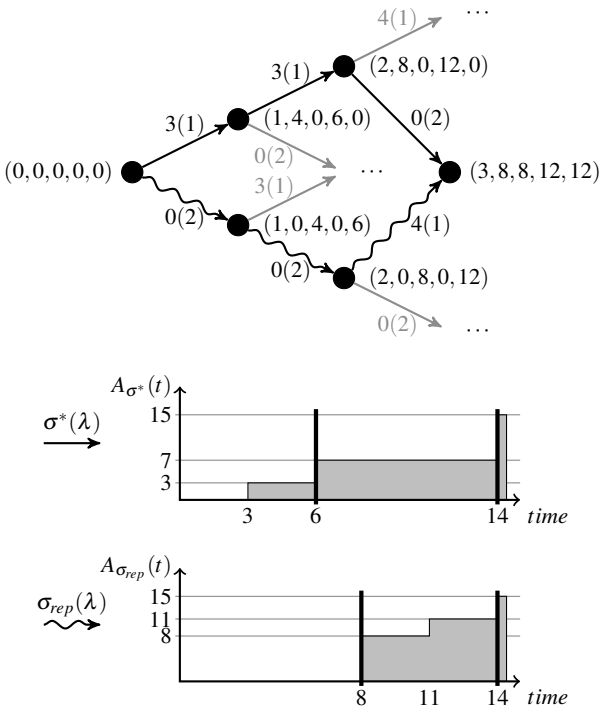


Fig. 2 Example of choosing the representative solution for a given rounded terminal node. The input parameters are the same as in Figure 1, and $K = 2$, $L = 6$. The label $w(r)$ on an edge has the meaning that this edge assigns the corresponding job to slot r, and this assignment has weight w. For ease of understanding, only the relevant part of the graph is displayed, and the nodes are organized into layers representing the assignments of the first, second…jobs.

- $T^s_{\max}(\lambda)$ is the objective function value of the representative solution $\sigma_{rep}(\lambda)$, considering non-rounded processing times
- $\bar{T}^s_{\max}(\lambda)$ is the objective function value of the representative solution $\sigma_{rep}(\lambda)$, considering rounded processing times
- ${T^s_{\max}}^*(\lambda)$ is the objective function value of the best schedule $\sigma^*(\lambda)$ in λ

We have to show that the algorithm described here is correct, i.e. all feasible solutions correspond to a feasible terminal node reachable from the source node on a path, and that the objective function value of a representative solution is close to the objective function value of any feasible solution that it represents.

Property 1 If a rounded terminal node is not feasible, then it does not represent any feasible solutions.

Proof Suppose we have an infeasible rounded terminal node $\bar{N}(n^J, \bar{P}_1, \bar{P}_2, \bar{\Delta}_1, \bar{\Delta}_2)$ with schedule $\sigma_{rep}(\lambda)$ and there exists a feasible solution $\sigma(\lambda)$ in subdomain λ. Then $w(\sigma(\lambda))$ cannot be smaller than $w(\sigma_{rep}(\lambda))$, hence $\bar{\Delta}_1 - w(\sigma_{rep}(\lambda)) \ge \bar{\Delta}_1 - w(\sigma(\lambda))$. Therefore, since $\sigma(\lambda)$ is feasible, a contradiction. □

Property 2 Any feasible solution of the original problem is represented by exactly one feasible representative solution.

Proof It is trivial that no solution can be represented by more than one representative solution. Furthermore, every possible assignment to the first and second slot of the rounded jobs is investigated in the rounded problem, so any feasible solution will be represented. According to Property 1, the corresponding representative solution will be feasible. □

Property 3 By setting $K = \frac{\varepsilon p_{\max}}{n^J}$, it holds that $\bar{T}^s_{\max}(\lambda) \leq (1+\varepsilon) {T^s_{\max}}^*(\lambda)$ for any $\lambda \in \Lambda$.

Proof First we observe that

$$\bar{T}^s_{\max}(\lambda) \leq {T^s_{\max}}^*(\lambda) + n^J K,$$

because if we replace each p_j in the optimal solution of λ with $\bar{p}_j$, the processing time of any job may increase by at most K time units, so any delivery may be delayed by at most $n^J K$ time units. Hence the difference of the objective function value of $\sigma_{rep}(\lambda)$ and $\sigma^*(\lambda)$ is

$$\begin{aligned} T^s_{\max}(\lambda) - {T^s_{\max}}^*(\lambda) &\leq \bar{T}^s_{\max}(\lambda) - {T^s_{\max}}^*(\lambda) \\ &\leq {T^s_{\max}}^*(\lambda) + n^J K - {T^s_{\max}}^*(\lambda) \\ &= n^J K = n^J \frac{\varepsilon p_{\max}}{n^J} \leq \varepsilon p_{sum} \\ &\leq \varepsilon {T^s_{\max}}^*(\lambda) \end{aligned}$$

□

Theorem 9 *There exists an FPTAS for the problem* $1|n^S = 1, n^R = 2|T^s_{\max}$.

Proof As the optimal solution has a feasible representative solution that is at most $(1+\varepsilon)$ times worse, and all feasible representative solutions are investigated, it is clear that the algorithm is $(1+\varepsilon)$-optimal. It has to be shown that it is polynomial in the input size and in $\frac{1}{\varepsilon}$.

If we set $K = \frac{\varepsilon p_{\max}}{n^J}$ and $L = \varepsilon c_{\max}$, then there are at most $\left\lceil \frac{n^J}{\varepsilon} \right\rceil$ and $\lceil \frac{1}{\varepsilon} \rceil$ different types of processing times and produced amounts of the product, respectively. This means $\bar{P}_r$ and $\bar{\Delta}_r$ can take at most $n^J \left\lceil \frac{n^J}{\varepsilon} \right\rceil$ and $n^J \lceil \frac{1}{\varepsilon} \rceil$ distinct values. The complexity can be upper bounded as follows:

- nodes of the graph: $|\bar{V}| \leq n^J \left(n^J \left\lceil \frac{n^J}{\varepsilon} \right\rceil \right)^2 \left(n^J \lceil \frac{1}{\varepsilon} \rceil \right)^2 + 1 = O\left(\left(\frac{1}{\varepsilon} \right)^4 (n^J)^7 \right)$
- edges of the graph: $|\bar{E}| \leq 2|\bar{V}|$
- finding the representative solutions takes $O(|\bar{V}| + |\bar{E}|)$ time
- checking the feasibility and the objective value of the representative solutions takes $O(|\bar{V}|)$ time

This means that the complete procedure takes at most $O(|\bar{V}|) = O\left(\left(\frac{1}{\varepsilon} \right)^4 (n^J)^7 \right)$ time. □

A natural question is whether the FPTAS could be generalized for other n^S and n^R values? The correctness of the algorithm relies on Property 1, so any similar algorithm for the more general case should select representative solutions with the same property. As it is demonstrated in Table 2 by a counter-example, the idea to select a path that has the lowest rounding error in lexicographic comparison cannot work, even for the $n^S = 1$, $n^R = 3$ case.

Parameters:

Param.	Value
n^J	6
n^S	1
n^R	3
ε	$\frac{2}{15}$
L	2
p_j	p

Job	$\tilde{c}_j$	$\bar{c}_j$
J_1	3	4
J_2	5	6
J_3	8	8
J_4	12	12
J_5	1	2
J_6	15	16

Delivery	e_r
R_1	8
R_2	28
R_3	44

Solutions represented by $\bar{N}(6,10,20,18,\dots)$ *terminal node:*

r	1	2	3	
$\bar{\Delta}_r$ $(\sum_{r'=1}^{r} \bar{\Delta}_{r'})$	10 (10)	20 (30)	18 (48)	
$\sigma^*(\lambda)$	J_1, J_2	J_3, J_4	J_5, J_6	
Δ_r $(\sum_{r'=1}^{r} \Delta_{r'})$	8 (8)	20 (28)	16 (44)	feasible
$\bar{\Delta}_r - \Delta_r$	2	0	2	
$\sigma_{rep}(\lambda)$	J_3, J_5	J_1, J_6	J_2, J_4	
Δ_r $(\sum_{r'=1}^{r} \Delta_{r'})$	9 (9)	18 (27)	17 (44)	infeasible
$\bar{\Delta}_r - \Delta_r$	1	2	1	

Table 2 Counter-example for the $n^S = 1$, $n^R = 3$ case. While $\sigma_{rep}(\lambda)$ is infeasible, it has lexicographically less rounding error than $\sigma^*(\lambda)$, which is feasible.

In a general setting (n^S and n^R are arbitrary constants), selecting a representative solution for a given rounded terminal node with Property 1 is equivalent to the following problem: *given a DAG with a designated source and target node, find a path between them which obeys the following resource constraints: there are* $n^S(n^R - 1)$ *resources with given limits, each edge consumes a vector* $\{0 \dots L\}^{n^S(n^R-1)}$ *of these resources, and the total consumption of the path cannot be higher than the limit for any resource.* The limit on a resource (s,r) is the total allowable rounding error of product s during the first r periods. Furthermore, the consumption vectors on the edges have a special structure: for a product s, the first $r-1$ coordinates are zero, and the other coordinates have the value $\bar{c}^s_j - \tilde{c}^s_j$.

This is a special case of the Constrained (Shortest) Path problem, which is known to be NP-hard in the ordinary sense (problem [ND30] in [7]; for more details, see [12]). However, to our knowledge, there is no result for the com-

plexity of this special case. If it is polynomially solvable, then our approach gives an FPTAS for arbitrary fixed n^S and n^R values, but if it is NP-hard, then this dynamic programming formulation with this type of rounding reaches its limits at the $n^S = 1$, $n^R = 2$ case.

5 Final remarks

In this paper we have presented some complexity and algorithmic results for scheduling inventory releasing jobs with tardiness related criteria. Our results complement those of Briskorn et al. [2] and Boysen et al. [1].

There are a number of open questions, especially on the algorithmic side. We have an FPTAS for one ordinary NP-hard variant with a single product type and 2 deliveries. However, there are a number of other NP-hard variants, for which approximation algorithms or approximation schemes may be designed.

Acknowledgements The authors are indebted to the two anonymous referees for helpful suggestions for improving the paper. This work has been supported by the research grant "Digital, real-time enterprises and networks", OMFB-01638/2009. The research of Tamás Kis has been supported by the János Bólyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

References

1. Boysen, N., Bock, S., Fliedner. M.: Scheduling of inventory releasing jobs to satisfy time-varying demand: an analysis of complexity. *Journal of Scheduling*, to appear, doi:10.1007/s10951-012-0266-0
2. Briskorn, D., Choi, B-C., Lee, K., Leung, J., Pinedo, M.: Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, 207, 605–619, doi:10.1016/j.ejor.2010.05.036 (2010)
3. Briskorn, D., Jaehn, F., Pesch, E.: Exact algorithms for inventory constrained scheduling on a single machine. *Journal of scheduling*, to appear, doi:10.1007/s10951-011-0261-x
4. Caprara, A., Kellerer, H., Pferschy, U., Pisinger, D., Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123, 333–345 (2000)
5. Carlier, J., Rinnooy Kan, A. H. G.: Scheduling subject to nonrenewable resource constraints. *Operations Research Letters*, 1, 52–55 (1982)
6. Carlier, J.: Scheduling under financial constraints, in R. Slowiński, J. Weglarz (eds), *Advances in Project Scheduling*, Elsevier, Amsterdam, pp. 187–224 (1989)
7. Garey, M. R., & Johnson, D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco (1979)
8. Grigoriev, A., Holthuijsen, M., van de Klundert, J.: Basic Scheduling Problems with Raw Material Constraints. *Naval Research Logistics*, 52, 527–535, doi:10.1002/nav.20095 (2005)
9. Kellerer, H., Kotov, V., Rendl, F., Woeginger, G. J.: The Stock Size Problem. *Operations Research*, 46, S1–S12 (1998)
10. Neumann, K., & Schwindt, C.: Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56, 513–533 (2002)
11. Schuurman, P., & Woeginger, G. J.: Approximation Schemes – A Tutorial. In: Moehring, R., Potts, C., Schulz, A., Woeginger, G., Wolsey, L. (eds.), *Lectures on Scheduling* (to appear)
12. Ziegelmann, M., Constrained Shortest Paths and Related Problems. PhD thesis, Universität des Saarlandes (2001)