Briskorn, Dirk; Choi, Byung-Cheon; Lee, Kangbok; Leung, Joseph; Pinedo, Michael

Working Paper

Inventory constrained scheduling on a single machine

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 640

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 640

Inventory Constrained Scheduling on a Single Machine

Dirk Briskorn[1,2,*], Byung-Cheon Choi[2], Kangbok Lee[2], Joseph Leung[3], Michael Pinedo[2]

October 2008

[1]: Christian-Albrechts-Universität zu Kiel
Institut für Betriebswirtschaftslehre
Olshausenstr. 40, 24098 Kiel, Germany
http://www.bwl.uni-kiel.de/bwlinstitute/Prod
briskorn@bwl.uni-kiel.de

[2]: Stern School of Business
New York University
44 West 4^{th} Street, New York, NY 10012, USA
http://www.stern.nyu.edu/~mpinedo
mpinedo@stern.nyu.edu, bchoi@stern.nyu.edu, klee3@stern.nyu.edu

[3]: Department of Computer Science
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA
http://web.njit.edu/~leung/
leung@cis.njit.edu

*: supported by a fellowship within the Postdoc-Program
of the German Academic Exchange Service (DAAD)

Abstract

This paper studies inventory constraints in a machine scheduling environment. Jobs can add and remove items of different types to the inventory and from the inventory, respectively. Jobs removing items cannot be processed if the required amount of items is not available. We first have a look at general models and determine the computational status of these problems. Since it turns out that general models are strongly NP-hard except for makespan minimization on one machine we have a look at special cases in the following. We determine the computational complexity of all considered special cases for objection functions $\sum C_j$ and $L_{\max}$ and several special cases for objective functions $\sum w_j C_j$ and $\sum U_j$.

Keywords: Machine scheduling, inventory constraints, computational complexity.

1 Introduction

In machine scheduling the usual setting gives the problem to schedule a set of jobs on a set of machines subject to a set of rules such that a given objective is optimized. One of the rules involved can be a precedence relation between two jobs. A regular precedence relation $i \prec j$ requires that job j's processing cannot be started before job i's processing is finished. The concept of precedence relation has been generalized to minimum and maximum time-lags. Minimum time-lag $l_{i,j}^{\min}$ and maximum time-lag $l_{i,j}^{\max}$ related to jobs i and j require that j is started not earlier and not later, respectively, than $l_{i,j}^{\min}$ and $l_{i,j}^{\max}$ time units after i is finished. Note that we obtain $i \prec j$ if we set $l_{i,j}^{\min} = 0$ and $l_{i,j}^{\max} = \infty$.

In this paper we introduce an other generalization of precedence constraints. We consider a set T of types. There is a storage for items of each type and an initial inventory is given. Now, a job might require (or consume) an amount of items to be started and might release an amount of items to the storage when it is finished. Naturally, we want to consider only schedules having non-negative inventory levels.

Inventory constraints as described above has been considered in project scheduling, see Bartels and Zimmermann [1], Neumann and Schwindt [8], Neumann et al. [9], and Schwindt and Trautmann [10] for example. In Neumann and Schwindt [8] it has been shown that inventory constraints are a generalization of both renewable and non-renewable resources. Thus, finding a minimum makespan project schedule considering (standard) precedence constraints and inventory constraints is a generalization of the well-known Resource Constrained Project Scheduling Problem which is known to be strongly NP-hard.

An obvious application of this model can be observed in cross-dock-terminals, see Boysen et al. [2] for example. Trucks arrive to drop or pick up, respecitvely, goods. Here, types of goods correspond to the types of items in our model. Trucks correspond to jobs. Each door of the terminal corresponds to machine each of which can handle one truck/job at a time. Here, the requirement of non-negativity of the inventory is trivially motivated by the real-world-application: no truck can be processed if he is supposed to pick up goods being not available at that time. We can think of various objectives here, in particular all of the common machine scheduling objectives.

An other application arises in financial investment projects. Let us assume we have a set of investment projects we want to undertake or have to undertake, respectively. Each project corresponds to a job and needs a certain amount of money to be started. After finishing the project we have a cash inflow which can be reinvested in an other project. Money corresponds to one type of items here. We can imagine further types such as project managers supervising a project during execution. Of course a project can only be started if the necessary amount

of managers is available. After finishing the project this number of managers will be available for further projects. An obvious objective for example would be to maximize the (discounted) cash inflow of projects carried out in a given time horizon (or – equivalenty – minimizing the (discounted) cash inflow of projects carried out outside the horizon). An other objective we can easily represent by means of a standard machine scheduling objective function. Assuming we have to undertake each project we may want to minimize the needed makespan.
In this paper we consider inventory constraints in machine scheduling environment and investigate the computational complexity of various problems. The remainder is organized as follows. In Section 2 we fomalize the problem we have given the underlying idea of above. We furthermore extend the notation by Graham et al. [3] to represent the problems considered in the paper at hand. In Section 3 we give results regarding complexity results. In Section **??** we proof that for two variants only finding a feasible solution is strongly NP-hard. In Section 3.1 we consider the objectives related to flowtime and in Section 3.2 we consider due date related objectives.. For each problem special cases are considered also. In Section 4, finally, we give conclusions and an outlook an future work.

2 Problem Specification

We consider a set T of types and set J of jobs. Each job j has a processing time p_j and might have a due date d_j. Moreover, each job j has a consumption of $r^c_{j,t}$ units of items of type $t \in T$ at the beginning of its processing which means that the inventory of type t is decreased by $r^c_{j,t}$ when j is started. When finished each job j releases $r^r_{j,t}$ units of items of type t to the corresponsing inventory, thus the inventory level increases by $r^r_{j,t}$. Note, that if $r^c_{j,t} = r^r_{j,t}$ holds for all $j \in J$ and $t \in T$, then inventory constraints are equivalent with resource constraints where a certain amount of resource is occudpied during processing of a job and is released afterwards. However, $r^c_{j,t} \neq r^r_{j,t}$ may hold. For each type an initial inventory R_t is given.
On a single machine a schedule is specified by a sequence σ. Let $\sigma(l)$, $1 \leq l \leq n$, be the job being scheduled in the lth position and let $I_{l,t}$, $1 \leq l \leq n$, $t \in T$, be the inventory level of t after the job scheduled in position l is finished. Thus,

$$I_{l,t} = R_t + \sum_{1 \leq l' < l} \left(r^r_{\sigma(l'),t} - r^c_{\sigma(l'),t} \right).$$

Let $J^- = \left\{ j \mid \sum_{t \in T} r^c_{j,t} > 0 \right\}$ and $J^+ = \left\{ j \mid \sum_{t \in T} r^r_{j,t} > 0 \right\}$. A schedule σ is feasible if and only if all jobs in J are scheduled and at each time the inventory of each type is non-negative, that is if job $j \in J$ is scheduled in position l, then we have $I_{l-1,t} \geq r^c_{j,t}$ for each $t \in T$.
We extend the notation by Graham et al. [3] as follows to represent the concepts mentioned above. If inventory constraints are considered, then we have $inv \in \beta$. If the number of types is limited to k, then we have $inv(k) \in \beta$.
As described above we consider a common lower bound for the inventory level of each type by requiring the level to be non-negative. Naturally, we can consider an upper bound $\overline{I}_t$ as well. The motivation may be given by limited storage capacity. If storage capacity is limited, we have $\overline{inv} \in \beta$ or $\overline{inv}(k) \in \beta$, respectively.
Next, we outline the relation between inventory constraints and precedence constraints. We reduce an arbitrary problem having precedence constraints to the corresponsing problem having inventory constraints. Consider a problem $\alpha|\beta|\gamma$ where $prec \in \beta$. Let precedence be given by $E = \{(i,j)\}$ where (i,j) means that job j cannot be started before job i is finished. For an arbitrary instance P^p of $\alpha|\beta|\gamma$ we construct an instance P^i of $\alpha|\beta'|\gamma$, $\beta' = (\beta \setminus \{prec\}) \cup \{inv\}$

as follows. We do not change any parameters but drop prcedence constraints. Instead we give inventory constraints as follows:

$$
\begin{aligned}
T &= \{t_e \mid e \in E\} && (1)\\
R_t &= 0 \quad \forall i \in T && (2)\\
r^c_{j,t_e} &= 1 \text{ if } (i,j) \in E, i \in J && (3)\\
r^r_{j,t_e} &= 1 \text{ if } (j,i) \in E, i \in J && (4)
\end{aligned}
$$

For each precedence relation e a separate type t_e is given, see (1). Type t_e is consumed and released by j and i, $(i,j) = e$, exclusively, see (3) and (4). Due to (2) i must proceed j.

3 Results

This Section is outlined as follows. First, we state that finding a feasible solution for $1|\overline{inv}(1)|\gamma$ or $1|inv(2)|\gamma$ is strongly NP-hard.

Theorem 1. *Finding a solution to* $1|\overline{inv}(1)|\gamma$ *is strongly NP-hard.*

Theorem 2. *Finding a solution to* $1|inv(2)|\gamma$ *is strongly NP-hard.*

Both Theorems are proven by reduction from 3-PARTITION. The formal proofs can be found in Appendices A and B. Consequently, in the following we restrict ourselves to the cases with $inv(1) \in \beta$ where each job either consumes items or releases items that is

$$\sum_{t \in T} r^r_{j,t} \sum_{t \in T} r^c_{j,t} = 0 \quad \forall j \in J.$$

We then represent the modification of the inventory level (of the only type) by job j by r_j ($r_j = r^r_{j,1}$ if $r^r_{j,1} > 0$, $r_j = -r^c_{j,1}$ otherwise). Furthermore, we write R, I_l, and $\overline{I}$ instead of R_t, $I_{l,t}$, and $\overline{I}_t$, respectively, for notational convenience. We consider objective functions $\sum C_j$ and $\sum w_j C_j$ in Section 3.1 as well as $L_{\max}$ and $\sum U_j$ in Section 3.2.
The resulting problems turn out to be strongly NP-hard for both easier objective functions, $\sum C_j$ and $L_{\max}$. We give proofs for that and afterwards focus on special cases derived by letting certain parameters be equal throughout J^- or J^+.
In order to have a short notation we introduce binary parameters $F_p(J')$, $F_w(J')$, $F_d(J')$, and $F_r(J')$ indicating whether parameter p, w, d, and r, respectively, of all jobs in J' are fixed to the same value (parameter's value equals 1) or not (parameter's value equals 0). Additionally, we denote the value of fixed parameters by p^+, p^-, w^+, w^-, d^+, d^-, r^+, and r^-, respectively, if the values apply for J^+ or J^-. That is,

$$
\begin{aligned}
F_p(J^+) = 1 \Leftrightarrow p_j = p^+ \quad \forall j \in J^+ \quad &\text{and} \quad F_p(J^-) = 1 \Leftrightarrow p_j = p^- \quad \forall j \in J^-,\\
F_w(J^+) = 1 \Leftrightarrow w_j = w^+ \quad \forall j \in J^+ \quad &\text{and} \quad F_w(J^-) = 1 \Leftrightarrow w_j = w^- \quad \forall j \in J^-,\\
F_d(J^+) = 1 \Leftrightarrow d_j = d^+ \quad \forall j \in J^+ \quad &\text{and} \quad F_d(J^-) = 1 \Leftrightarrow d_j = d^- \quad \forall j \in J^-, \text{ and}\\
F_r(J^+) = 1 \Leftrightarrow r_j = r^+ \quad \forall j \in J^+ \quad &\text{and} \quad F_r(J^-) = 1 \Leftrightarrow r_j = r^- \quad \forall j \in J^-.
\end{aligned}
$$

In particular in cross dock terminals problems with certain parameters having equal values may have straightforward applications since sizes (capacities) and handling speeds e.g. are often standardized.

3.1 Flowtime Related Objectives

For a given instance P of $1|inv(1)|\sum C_j$ we define the symmetric instance P'. P' has the same number of jobs as P and the jth job of P' has processing time and inventory modification denoted as p'_j and r'_j, respectively.

1. $p'_j = \max_{j' \in J} p_{j'} + 1 - p_j$
2. $r'_j = -r_j$
3. $R' = R + \sum_{j \in J} r_j$

Lemma 1. *The optimal sequence of jobs to an instance P of problem $1|inv(1)|\sum C_j$ is the reverse optimal sequence of corresponding jobs of symmetric instance P'.*

Proof. Consider symmetric sequences σ and σ' as solutions to P and P'. It is easy to see that σ is feasible to P if and only if σ' is feasible to P'. Now, let

$$A = \frac{n(n+1)}{2}\left(\max_{j \in J} p_j + 1\right) \text{ and } B = (n+1)\sum_{j \in J}^{n} p_j.$$

Obviously, A and B are constant values.

$$\begin{aligned}
\sum_{k=1}^{n}(n+1-k)p'_{\sigma'(k)} &= \sum_{k=1}^{n}(n+1-k)\left(\max_{j \in J} p_j + 1 - p_{\sigma(n+1-k)}\right) \\
&= \frac{n(n+1)}{2}\left(\max_{j \in J} p_j + 1\right) - \sum_{k=1}^{n}(n+1-k)p_{\sigma(n+1-k)} \\
&= A - \sum_{k=1}^{n}(n+1)p_{\sigma(n+1-k)} + \sum_{k=1}^{n} k p_{\sigma(n+1-k)} \\
&= A - B + \sum_{k=1}^{n}(n+1-k)p_{\sigma(k)}
\end{aligned}$$

Hence, the objective values of σ and σ' differ only by a constant and, therefore, σ is optimal to P if and only if σ' is optimal to P'. □

For a given instance P of $1|inv(1)|\sum w_j C_j$ we define the symmetric instance P'. Again, P' has the same number of jobs as P and the jth job of P' has processing time and inventory modification denoted as p'_j and r'_j, respectively.

1. $p'_j = w_j$
2. $w'_j = p_j$
3. $r'_j = -r_j$
4. $R' = R + \sum_{j \in J} r_j$

Lemma 2. *The optimal sequence of jobs to an instance P of problem $1|inv(1)|\sum w_j C_j$ is the reverse optimal sequence of corresponding jobs of symmetric instance P'.*

Proof. Consider symmetric sequences σ and σ' as solutions to P and P'. It is easy to see that σ is feasible to P if and only if σ' is feasible to P'. Optimality follows from the well known symmetry property of $1||\sum w_jC_j$. □

As mentioned above we look at special cases specified by equal values for certain parameters of jobs in J^+ or J^-. Lemmas 1 and 2 enable us to project results for one special case to an other special case since if P has identical processing times, weights, or inventory modification for jobs in J^+, then P' has identical processing times, weights, or inventory modification, respectively, for jobs in J^- (and vice versa).

Theorem 3. $1|inv(1)|\sum C_j$ *is strongly NP-hard even if all jobs in* J^- *are identical.*

The proof is inspired by the proof for NP-hardness of $1|r_j|\sum C_j$ by Lenstra et al. [6]. Here, we just give the basic idea of the proof. The proof itself can be found in Appendix C.
We construct an instance of $1|inv(1)|\sum C_j$ with three classes of jobs J^a , J^b, and J^-. Jobs in J^a represent number a_j from the instance of 3-PARTITION by having $r_j = a_j + p$ and $p_j = a_j + p$ for each $j \in J^a$ where is p is constant. Jobs in J^- are used to form a partition of J^a.

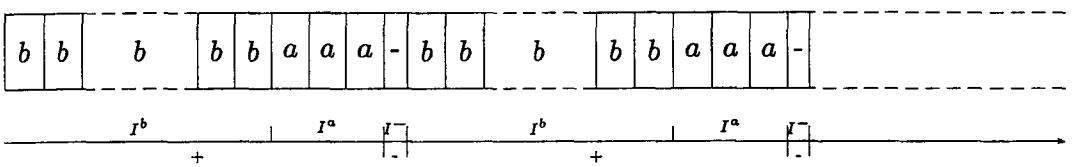

Figure 1: Schedule Structure

Figure 1 sketches the structure of a schedule of the instance of $1|inv(1)|\sum C_j$ if the answer to the instance of 3-PARTITION is yes. We have sections consisiting of several jobs of J^b, three jobs of J^a, and one job of J^-. In each section all jobs of J^b are followed by all jobs of J^a. In each section the job of J^- is the concluding job. The main issue is to fix jobs of J^- to specific positions. We arrange that each job of J^- cannot be positioned earlier than a specific position by tuning consumption and release of items accordingly. A specific number of jobs of J^b serve as block in each section which must be completely processed after the consuming job of the previous block. The processing time of jobs of J^b must be chosen large enough such that a switch of a job of J^- and a consecutive job of J^b leads to an objective value exceeding a given threshold.

Theorem 4. $1|inv(1)|\sum C_j$ *is strongly NP-hard even if all jobs in* J^+ *are identical.*

Proof. The Theorem follows from Theorem 3 and Lemma 1. □

Theorem 5. $1|inv(1)|\sum w_jC_j$ *is strongly NP-hard*

- *even if all jobs in* J^+ *are identical and* $F_w(J^-) = 1$,
- *even if all jobs in* J^- *are identical and* $F_w(J^+) = 1$,
- *even if all jobs in* J^+ *are identical and* $F_p(J^-) = 1$, *and*
- *even if all jobs in* J^- *are identical and* $F_p(J^+) = 1$.

Proof. The Theorem follows from Theorems 3 and 4 and Lemma 2. □

Theorem 6. *If* $F_p(J^+) + F_w(J^+) + F_r(J^+) \geq 2$ *and* $F_p(J^-) + F_w(J^-) + F_r(J^-) \geq 2$, *then* $1|inv(1)|\sum w_j C_j$ *is solvable in* $O(n \log n)$.

Proof. First we show that there is an optimal solution such that J^+ is ordered according the individual parameter (in SPT order if $F_w(J^+) = F_r(J^+) = 1$, in non-increasing order of weights if $F_p(J^+) = F_r(J^+) = 1$, and in non-increasing order of inventory modification if $F_p(J^+) = F_w(J^+) = 1$). Consider a solution having at least two jobs of J^+ not in the proposed order. Then, there are at least two consecutive jobs j_1^+ and j_2^+ in J^+ not in the proposed order in the current solution. It is easy to see that switching j_1^+ and j_2^+ cannot violate the inventory constraints. So, if $F_p(J^+) = F_w(J^+) = 1$, then we can switch j_1^+ and j_2^+. If $F_p(J^+) = F_r(J^+) = 1$, then we can switch j_1^+ and j_2^+ since this does not affect completions times of other jobs and sum of weights to a position l cannot increase for each $1 \leq l \leq n$. If $F_w(J^+) = F_r(J^+) = 1$, then we can switch j_1^+ and j_2^+ since the weights of the job in each position does not change and no job can be delayed.
The same arguments apply to J^-. Note that each of these orders has jobs in WSPT order. Let $J^+ = \{1, \ldots, n^+\}$ and $J^- = \{n^+ + 1, \ldots, n^+ + n^-\}$ be ordered (according to the criteria described above).

Algorithm 1

1. $l = 0$, $j^+ = 1$, $j^- = n^+ + 1$
2. $I_0 = R$
3. repeat until $j^+ = n^+ + 1$ or $j^- = n^+ + n^- + 1$
   (a) if $p_{j^-}/w_{j^-} < p_{j^+}/w_{j^+}$ and $I_l \geq -r_{j^-}$, then choose j^- and set $I_{l+1} = I_l + r_{j^-}$ and $j^- = j^- + 1$
   (b) else choose j^+ and set $I_{l+1} = I_l + r_{j^+}$ and $j^+ = j^+ + 1$
   (c) $l = l + 1$
4. schedule remaining jobs of J^+ or J^- at the end

This algorithm gives an optimal schedule. Let σ and σ^* be the result of our algorithm and an optimal schedule (obeying partial orders for J^+ and J^- as described above), respectively. Let k and k^* be the first position where we have $\sigma(k) \neq \sigma^*(k)$ and the position of $\sigma(k)$ in σ^*, respectively. Due to the order of J^+ and J^- we must have $r_{\sigma(k)} r_{\sigma^*(k)} < 0$.
If $\sigma(k) \in J^+$, then since our algorithm chooses $\sigma(k)$ instead of $\sigma^*(k)$ scheduling $\sigma^*(k)$ in position k of σ is infeasible or

$$\frac{p_{\sigma(k)}}{w_{\sigma(k)}} \leq \frac{p_{\sigma^*(k)}}{w_{\sigma^*(k)}}.$$

In the first case σ^* is infeasible. In the second case, we can move $\sigma(k)$ in position k of σ^* and delay jobs in positions k to $k^* - 1$ by one position. Note that only jobs out of J^- will be delayed and furthermore jobs in both subsets are in WSPT order. So, we argue by pairwise exchange that we can bring $\sigma(k)$ in position k of σ^* without increasing the objective value. Furthermore, this move is feasible since I_l, $k \leq l \leq k^*$ is increased by $r_{\sigma(k)} - r_{\sigma^*(l)} > 0$.
If $r_{\sigma(k)} < 0$, then since our algorithm chooses $\sigma(k)$ instead of $\sigma^*(k)$ we must have

$$\frac{p_{\sigma(k)}}{w_{\sigma(k)}} < \frac{p_{\sigma^*(k)}}{w_{\sigma^*(k)}}.$$

Then, we can move $\sigma(k)$ at position k in σ^* as in the above case. Again, the move is feasible and cannot worsen σ^*. □

Summarizing the results provided by Theorems 3 to 6, we can state that

- $1|inv(1)|\sum C_j$ is unary NP-hard,
- $1|inv(1)|\sum w_jC_j$ is polynomially solvable if $F_p(J^+) + F_w(J^+) + F_r(J^+) \geq 2$ and $F_p(J^-) + F_w(J^-) + F_r(J^-) \geq 2$.

Note that $1|inv(1)|\sum C_j$ is polynomially solvable if $F_p(J^+) + F_r(J^+) \geq 1$ and $F_p(J^-) + F_r(J^-) \geq 1$. In the following we give a polynomial time algorithm for a special case which does not completely fit in our scheme of fixing certain parameter of jobs in J^+ and J^- to specific values.

Theorem 7. *If all jobs in $j \in J^-$ are identical, $r_j = -l_jr^-$, $l_j \in \mathbb{N}$, for $j \in J^+$, and $R + \sum_{j\in J} r_j = 0$, then $1|inv(1)|\sum w_jC_j$ is solvable in $O(n \log n)$.*

Proof. Clearly, the subset of jobs J^- can be scheduled in arbitrary order. Let

$$J^- = \{1, \ldots, |J^-|\}$$

and let

$$J_1^+ = \left\{ j \mid j \in J^+, \frac{p_j}{w_j} \leq \frac{p^-}{w^-} \right\} = \left\{ |J^-| + 1, \ldots, |J^-| + |J_1^+| \right\}$$

be ordered in WSPT order. Furthermore, let

$$J_2^+ = J^+ \setminus J_1^+ = \left\{ |J^-| + |J_1^+| + 1, \ldots, |J| \right\}$$

be ordered such that

$$\frac{p_j + \frac{r_j}{|r^-|}p^-}{w_j + \frac{r_j}{|r^-|}w^-} < \frac{p_{j+1} + \frac{r_{j+1}}{|r^-|}p^-}{w_{j+1} + \frac{r_{j+1}}{|r^-|}w^-} \qquad \forall j \in \left\{ |J_1^+| + 1, \ldots, |J^+| - 1 \right\}.$$

Algorithm 2

1. schedule J_1^+ in slots 1 to $|J_1^+|$
2. schedule jobs 1 to $l = \left(R + \sum_{j\in J_1^+} r_j\right)/|r^-|$ of J^- in slots $|J_1^+| + 1$ to $|J_1^+| + l$
3. $j^+ = |J^-| + |J_1^+| + 1$, $k = |J_1^+| + l + 1$
4. repeat until $j^+ = |J| + 1$
   (a) schedule j^+ in slot k
   (b) schedule jobs $l + 1$ to $l + r_{j^+}/|r^-|$ in slots $k + 1$ to $k + r_{j^+}/|r^-|$
   (c) $l = l + r_{j^+}/|r^-|$, $k = k + r_{j^+}/|r^-| + 1$

This algorithm gives an optimal schedule. Let σ and σ^* be the result of our algorithm and an optimal schedule, respectively. Obviously, we can reorder jobs of J^- in σ^* in ascending order of index. Let k and k^* be the first position where we have $\sigma(k) \neq \sigma^*(k)$ and the position of $\sigma(k)$ in σ^*, respectively. Due to the order of J^- we cannot have $r_{\sigma(k)} < 0$ and $r_{\sigma^*(k)} < 0$.
It is obvious that we can schedule J_1^+ and

$$\frac{R + \sum_{j \in J_1^+} r_j}{|r^-|}$$

jobs of J^- at the beginning (by pairwise exchange). Note that after position

$$|J_1^+| + \frac{R + \sum_{j \in J_1^+} r_j}{|r^-|}$$

the inventory equals zero.
Each job $j \in J_2^+$ is followed immediately by exactly $r_j/|r^-|$ jobs of J^-. Suppose there are only $k < r_j/|r^-|$ jobs of J^- between the first job $j \in J_2^+$ and the next job j' of J_2^+. By pairwise interchange of j' and $r_j/|r^-| - k$ following jobs of J^- we can increase the distance between j and j' to $r_j/|r^-| + 1$. This cannot increase the objective value since $p_{j'}/w_{j'} > p^-/w^-$. Then, at the start of job j' the inventory level is zero. Now, it is easy to see that the same argument applies for all other jobs of J_2^+.
Correctness of the algorithm follows by interchange argument considering job $j \in J_2^+$ and $r_j/|r^-|$ jobs out of J^- as an atomic unit. □

3.2 Due Date Related Objectives

This Section deals with the objectives to minimize the maximum lateness and the number of tardy jobs. Section 3.2.1 focuses on maximum lateness. However, a pseudopolynomial algorithm for minimum number of tardy jobs is given (Theorem 10) which obviously solves maximum lateness in pseudopolynomial time also. Section 3.2.2 is exclusively restricted to number of tardy jobs. For the formal proofs of NP-hardness we refer to Appendices D to D.

3.2.1 Maximum Lateness

Theorem 8. $1|inv(1)|L_{\max}$ *is strongly NP-hard even if* $F_p(J^-) = F_r(J^-) = F_d(J^+) = 1$.

Theorem 9. $1|inv(1)|L_{\max}$ *is NP-hard even if jobs of* J^- *are identical and* $F_d(J^+) = 1$.

Theorem 10. *If* $F_d(J^-) = 1$, *then* $1|inv(1)|\sum U_j$ *can be solved in* $O\left(n^2 \left(\sum_{j \in J^+} p_j\right)^5 \left(\sum_{j \in J^+} r_j\right)^2\right)$.

Proof. Let $J^{-,e} \subseteq J^-$ denote the subset of jobs of J^- being early. First, we observe that we can schedule all jobs of $J^{-,e}$ in a row. Consider the last job out of $J^{-,e}$. Moving all other jobs of $J^{-,e}$ right before it (and, therefore, scheduling some jobs of J^+ earlier) cannot violate the inventory constraints (because delaying consuming jobs is always allowed) and cannot worsen the objective value (because due dates of consuming jobs are identical). Jobs of $J^- \setminus J^{-,e}$ can be scheduled in row at the end of the schedule.
Additionally, we can divide J^+ into two subsets J_1^+ and J_2^+ preceeding and following $J^{-,e}$, respectively. Both subsets can be assumed to be scheduled by the algorithm by Moore [7]. This means J_1^+ (J_2^+) consists of a set of early jobs $J_1^{+,e}$ ($J_2^{+,e}$) followed by a set of tardy jobs

$J_1^{+,l}$ ($J_2^{+,l}$), as illustrated in Figure 2. $J_1^{+,e}$, $J_2^{+,e}$, $J_1^{+,l}$, and $J_2^{+,l}$ can be assumed to be in EDD order.
Assume jobs are numbered such that $J^+ = \{1, \ldots, |J^+|\}$ (ordered in EDD) and $J^- = \{|J^+|+1, \ldots, |J^+| + |J^-|\}$. Binary variables

$$x_{j,l,p,r} \quad \forall j \in J^-, 1 \leq l \leq j - |J^+|, p \in \left\{1, \ldots, \sum_{j \in J^-} p_j\right\}, r \in \left\{1, \ldots, \sum_{j \in J^-} |r_j|\right\},$$

being equal to 1 if and only if there is a subset of jobs $J' \subseteq \{|J^+|+1, \ldots, j\}$, $|J'| = l$, such that

$$\sum_{j \in J'} p_j = p \text{ and } \sum_{j \in J'} r_j = -r.$$

Clearly,

$$\begin{aligned} x_{|J^+|+1,1,p,r} &= 1 \text{ if and only if } p_{|J^+|+1} = p \text{ and } r_{|J^+|+1} = -r \text{ and} \\ x_{j,l,p,r} &= 1, j > |J^+| + 1 \text{ if and only if } x_{j-1,l,p,r} = 1 \text{ or } x_{j-1,l-1,p-p_j,r-r_j} = 1. \end{aligned}$$

Using this recursive relation we can compute all $x_{j,l,p,r}$ in $O(n^2 \sum |r_j| \sum p_j)$.
We consider the subproblem where the number of early jobs of J^- is required to be at least e^-. For given number $e^- = |J^{-,e}|$ of early jobs in J^- we do the following. We find all variables

$$x_{|J^+|+|J^-|,e^-,p,r} = 1, p \in \left\{1, \ldots, \sum_{j \in J^-} p_j\right\}, r \in \left\{1, \ldots, \sum_{j \in J^-} r_j\right\}.$$

The number of these variables cannot be larger than $\sum |r_j| \sum p_j$. Since we can schedule the corresponding subset in a row we consider a single job j^- specified by $p_{j^-} = p$, $r_{j^-} = r$, and $d_{j^-} = d^-$. Job j^- will represent $J^{-,e}$ in the following.
For given j^- we consider the problem to find the optimal schedule when j^- is required to be not tardy in the following.
First, we show that there is an optimal schedule with

$$C_{j^-} \in \left\{d^- - \max_{j \in J^+} p_j + 1, \ldots, d^-\right\}.$$

Obviously, C_{j^-} must not be larger than d^- since otherwise j^- is tardy. Suppose there is an optimal schedule with j^- not tardy and $C_{j^-} \leq d^- - \max_{j \in J^+} p_j$. We distinguish two cases.

- If $J_2^+ = \emptyset$, then we obviously can delay j^- by $d^- - C_{j^-}$ time units without violating inventory constraints. Still, j^- is not tardy.
- If $J_2^+ \neq \emptyset$, then we can choose the first job in J_2^+ and move it before j^-. Still, j^- cannot be late and no other job has been delayed. By repeating this procedure, we can reach

$$C_{j^-} \in \left\{d^- - \max_{j \in J^+} p_j + 1, \ldots, d^-\right\}.$$

For each j^- and each possible completion time

$$C_{j^-} \in \left\{ d^- - \max_{j \in J^+} p_j + 1, \ldots, d^- \right\}$$

(and, hence, no more than $\sum |r_j| \sum p_j \max p_j$ times) we consider the problem to find the optimal schedule if i^- is fixed to this specific completion time. To do this, we construct a graph $G = (V, E, c)$ as follows.

$$\begin{aligned}
V &= \Big\{ (j, P_1^e, P_1^l, P_2^e, R_2) \mid j \in J^+, P_1^e, P_1^l, P_2^e \in \{1 \ldots, \sum p_j\}, \\
&\quad R_2 \in \{1 \ldots, \sum |r_j|\} \Big\} \\
E &= E_1^e \cup E_2^e \cup E_1^l \cup E_2^l \\
E_1^e &= \big\{ ((j, P_1^e, P_1^l, P_2^e, R_2), (j+1, P_1^e + p_{j+1}, P_1^l, P_2^e, R_2)) \mid \\
&\quad P_1^e + p_{j+1} + P_1^l \leq C_{j^-} - p_{j^-}, P_1^e + p_{j+1} \leq d_{j+1} \big\} \\
E_1^l &= \big\{ ((j, P_1^e, P_1^l, P_2^e, R_2), (j+1, P_1^e, P_1^l + p_{j+1}, P_2^e, R_2)) \mid \\
&\quad P_1^e + P_1^l + p_{j+1} \leq C_{j^-} - p_{j^-} \big\} \\
E_2^e &= \Big\{ ((j, P_1^e, P_1^l, P_2^e, R_2), (j+1, P_1^e, P_1^l, P_2^e + p_{j+1}, R_2 + r_{j+1})) \mid \\
&\quad C_{j^-} + P_2^e + p_{j+1} \leq d_{j+1}, R_2 + r_{j+1} \leq R + \sum_{j' \in J^+} r_{j'} + r_{j^-} \Big\} \\
E_2^l &= \Big\{ ((j, P_1^e, P_1^l, P_2^e, R_2), (j+1, P_1^e, P_1^l, P_2^e, R_2 + r_{j+1})) \mid \\
&\quad R_2 + r_{j+1} \leq R + \sum_{j' \in J^+} r_{j'} + r_{j^-} \Big\} \\
c(e) &= 0 \qquad \forall e \in E_1^e \cup E_2^e \\
c(e) &= 1 \qquad \forall e \in E_1^l \cup E_2^l
\end{aligned}$$

Node $(j, P_1^e, P_1^l, P_2^e, R_2)$ represents a partial solution having jobs 1 to j assigned to $J_1^{+,e}$, $J_2^{+,e}$, $J_1^{+,l}$, and $J_2^{+,l}$ such that

$$\sum_{j' \in J_1^{+,e}} p_{j'} = P_1^e, \sum_{j' \in J_1^{+,l}} p_{j'} = P_1^l, \sum_{j' \in J_2^{+,e}} p_{j'} = P_2^e, \sum_{j' \in J_2^{+,e} \cup J_2^{+,l}} r_{j'} = R_2.$$

Edges in E_1^e and E_2^e represent the assignment of job $j+1$ to $J_1^{+,e}$ and $J_2^{+,e}$. This is possible only if job $j+1$ is early when scheduled at the end of the corresponding set. Furthermore, assignment of $j+1$ to $J_1^{+,e}$ is allowed only if the total processing time of J_1^+ fits before j^-. Assignment of $j+1$ to $J_2^{+,e}$ is allowed only if $j+1$ is not needed before j^- to reach an inventory of $|r_{j^-}|$.
Edges in E_1^l and E_2^l represent the assignment of job $j+1$ to $J_1^{+,l}$ and $J_2^{+,l}$. Assignment of $j+1$ to $J_1^{+,l}$ is not allowed if the total processing time of J_1^+ does not fit before j^-. Assignment of $j+1$ to $J_2^{+,l}$ is allowed only if $j+1$ is not needed before j^- to reach an inventory of $|r_{j^-}|$.
The length of an edge representing a job being scheduled tardy (early) is 1 (0) since this assignment contributes 1 (0) to the objective value.
Clearly,

$$|V| = |E| \in O\left(n\left(\sum_{i\in J^+} p_j\right)^3 \sum_{j\in J^+} r_j\right)$$

and, hence, we can find a shortest path from an artificial source $(0,0,0,0,0)$ to a node $(|J^+|, P_1^e, P_1^l, P_2^e, R_2)$ in

$$O\left(n\left(\sum_{i\in J^+} p_j\right)^3 \sum_{j\in J^+} r_j\right).$$

Considering that we have to find a shortest path for each number of positive variables and each possible completion time we have overall computational complexity of $O\left(n^2\left(\sum_{i\in J^+} p_j\right)^5\left(\sum_{j\in J^+} r_j\right)^2\right)$. □

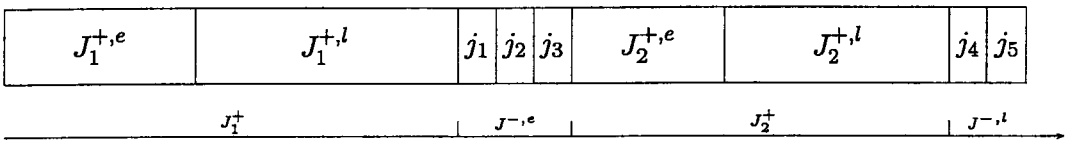

Figure 2: Schedule Structure

Figure 2 provides the scheme of schedules as considered in the proof of Theorem 10. Set J^- of jobs is divided into two blocks $J^{-,e}$ and $J^{-,l}$ where $J^{-,e}$ are guaranteed to be early (possibly jobs in $J^{-,l}$ can be early also). $J^{-,l}$ i scheduled last while $J^{-,e}$ is scheduled to be completed within $[d^- - \max_{j\in J^+} p_j + 1, \ldots, d^-]$. In Figure 2 $J^{-,e}$ consist of $e^- = 3$ jobs j_1, j_2, and j_3. Set J^+ of jobs is divided into two blocks J_1^+ and J_2^+ preceding and following $J^{-,e}$, respectively. Both blocks are subdivided into two subsets containing early and late jobs, respectively. Each subset can be assumend to be in EDD order.

Lemma 3. *There is an optimal solution to* $1|inv(1)|L_{\max}$ *having the subset* J^- *ordered in non-decreasing due dates.*

Proof. Note that the makespan is fixed. Let σ be an optimal solution where set J^- is not ordered in EDD order. Then, let j be the last job of J^- being preceded by a job $j' \in J^-$ with $d_{j'} > d_j$. Let k and k' be positions of j and j', respectively, in σ. Then, moving j' in position k and moving jobs in positions $k'+1, \ldots, k$ to positions $k', \ldots, k-1$ is feasible (since J_l, $k' \le l < k$, is increased by $-r_{j'} > 0$) and cannot worsen the solution (since $d_{j'} > d_j$). □

Theorem 11. *If* $F_p(J^+) + F_r(J^+) \ge 1$, *then* $1|inv(1)|L_{\max}$ *can be solved in* $O(n^3)$.

Proof. Let due dates d'_j, $j \in J$ be given. Furthermore, let J^- be ordered in reversed EDD order and let J_l^- be the lth job in J^-. Then, the following algorithm finds a schedule σ such that no job is tardy in $O(n^2)$ if such a schedule exists.

Algorithm 3

1. $I_n = R + \sum_{j\in J} r_j$
2. $T = \sum_{j\in J} p_j$

3. $J^{av} = \{j \mid j \in J^+, d_j \geq T\}$

4. $k = n$, $l = 1$

5. repeat until $k = 0$

   (a) if $d_{J_l^-} \geq C$, then

      i. $\sigma(k) = J_l^-$

      ii. $l = l + 1$

      iii. $I_{k-1} = I_k - r_{\sigma(k)}$

      iv. $J^{av} = J^{av} \cup \{j \mid j \in J^+, d_j \in [T - p_{\sigma(k)}, T[\}$

      v. $T = T - p_{\sigma(k)}$

   (b) else if $J^{av} = \emptyset$ or $\min_{j \in J^{av}} \{r_j\} > I_k$, then STOP

   (c) else if $\min_{j \in J^{av}} \{r_j\} \leq I_k$

      i. if $F_p(J^+)$, then $j^* = \arg\min_{j \in J^{av}} \{r_j\}$

      ii. else if $F_r(J^+)$, then $j^* = \arg\max_{j \in J^{av}} \{p_j\}$

      iii. $\sigma(k) = j^*$

      iv. $I_{k-1} = I_k - r_{\sigma(k)}$

      v. $J^{av} = (J^{av} \setminus \{j^*\}) \cup \{j \mid j \in J^+, d_j \in [T - p_{\sigma(k)}, T[\}$

      vi. $T = T - p_{\sigma(k)}$

   (d) $k = k - 1$

Assume there is a sequence σ having no tardy job and the algorithm aborted without finding a schedule. Then, there is a feasible schedule σ' where jobs J^- are ordered in EDD order, see Lemma 3. Let k be the last position where σ' and the partial schedule σ^p constructed by our algorithm differ. If no job was found for position k by our algorithm, then σ' must be infeasible and, consequently, so is σ. Note that $r_{\sigma^p(k)} > 0$ or $r_{\sigma'(k)} > 0$, otherwise, due to EDD order of J^-. If a job was assigned to position k by our algorithm, then let k' denote the position of $\sigma^p(k)$ in σ'.
If $r_{\sigma^p(k)} < 0$, then we can move $\sigma^p(k)$ at position k of σ' while moving jobs in positions $k'+1$ to k in σ' to positions k' to $k-1$. Note that no job of J^- (except for $\sigma^p(k)$) is moved due to the unique order of J^- in σ' and σ^p. Obviously, σ' stays feasible.
If $r_{\sigma^p(k)} > 0$ and $r_{\sigma'(k)} > 0$, we have $\sigma^p(k) \in J^{av}$ and $\sigma'(k) \in J^{av}$ in iteration $n-k+1$ of our algorithm (when $\sigma^p(k)$ is assigned to σ^p). Then, $r_{\sigma^p(k)} \leq r_{\sigma'(k)}$ if $F_p(J^+)$ and $p_{\sigma^p(k)} \geq p_{\sigma'(k)}$ if $F_r(J^+)$ due to the choice of j^*. Hence, exchanging $\sigma^p(k)$ and $\sigma'(k)$ in σ' cannot violate inventory constraints and cannot delay a job except for $\sigma^p(k)$. However, $\sigma^p(k)$ can not be tardy after the switch because $\sigma^p(k) \in J^{av}$.
If $r_{\sigma^p(k)} > 0$ and $r_{\sigma'(k)} < 0$, then $\sigma'(k)$ must be an infeasible choice for σ' because otherwise the algorithm would have chosen $\sigma'(k)$ as well. This induces correctness of the proposed algorithm.
Note that the objective value is bounded from above by $\overline{L}_{\max} = \sum_{j \in J} p_j$ and from below by $\underline{L}_{\max} = -\max_{j \in J} d_j$. Using binary search on $[\underline{L}_{\max}, \overline{L}_{\max}]$ we can find the optimal solution by employing the algorithm above in $O(n^3)$. □

Remark 1. *If $F_p(J^+) + F_r(J^+) + F_d(J^+) \geq 2$, then $1|inv(1)|L_{\max}$ can be solved in $O(n \log n)$ since J^+ can be assumed to be sorted according to the individual parameter.*

Summarizing the results provided by Theorems 8 to 11, we can state that

- $1|inv(1)|L_{\max}$ is unary NP-hard,
- $1|inv(1)|L_{\max}$ is binary NP-hard even if jobs of J^- are identical and $F_d(J^+) = 1$, and
- $1|inv(1)|L_{\max}$ is polynomially solvable if $F_p(J^+) + F_r(J^+) \geq 1$.

3.2.2 Number of Tardy Jobs

Theorem 12. *$1|inv(1)|\sum U_j$ is NP-hard even if $F_d(J^-) = 1$ and all jobs of J^+ are identical.*

Theorem 13. *If $F_d(J^+) + F_p(J^-) = 2$ and $F_p(J^+) + F_r(J^+) \geq 1$, then $1|inv(1)|\sum U_j$ can be solved in $O(n^2 \log n)$.*

Proof. It is easy to see that J^+ can be ordered according non-increasing inventory modification if $F_p(J^+)$ and in SPT order if $F_r(J^+)$. We consider the subproblem where the number of early jobs of J^+ is required to be at least e^+.
Jobs of J^- are structured in an optimal schedule as follows. The subset $J^{-,l}$ of tardy jobs of J^- can be scheduled at end in a row in arbitrary order. The set $J^{-,e} = J^- \setminus J^{-,l}$ of early jobs of J^- can be assumed to be scheduled in EDD order (by the interchange argument of Lawler [5]). $J^{-,e}$ is divided into $J_1^{-,e}$ and $J_2^{-,e}$ preceding and following the the e^+th job of J^+. Furthermore, we can see that

$$|J_1^{-,e}| \leq e_1^- = \left\lfloor \frac{d^+ - \sum_{j \in J^+, j \leq e^+} p_j}{p^-} \right\rfloor.$$

Let $J^+ = \{1, \ldots, |J^+|\}$ be ordered as described above and let $J^+ = \{|J^+| + 1, \ldots, |J^+| + |J^-|\}$ be in EDD order. Each job in $J_1^{-,e}$ can be assumed to be scheduled as early as possible while preserving EDD order, non-negative inventories, and job e^+ being early. This property follows from a simple interchange argument (delaying early jobs of J^+ is allowed if inventory remains non-negative and job e^+ remains early). Thus, $J_1^{-,e}$ is further subdivided by jobs $1, \ldots, e^+ - 1$ which are scheduled such that (i) inventory is non-negative and (ii) job e^+ is early. Each job in $J_2^{-,e}$ can be assumed to be scheduled as early as possible while preserving EDD order and non-negative inventories. Thus, $J_2^{-,e}$ is further subdivided by jobs $e^+ + 1, \ldots, |J^+|$, see Figure 3.

Algorithm 4

1. $j^+ = 1$, $j^- = |J^+| + 1$, $J^{-,e} = \emptyset$, $I_0 = R$, $T = n_1^e = 0$, $k = 1$, $k' = n$
2. for $n_1^e \leq e_1^-$ and $j^- \leq |J^+| + |J^-|$
   (a) if $I_{k-1} \geq |r_{j^-}|$ and $T + p^- \leq d_{j^-}$
      i. schedule j^- in slot k
      ii. $J^{-,e} = J^{-,e} \cup \{j^-\}$
      iii. $I_k = I_{k-1} + r_{j^-}$
      iv. $k = k + 1$, $n_1^e = n_1^e + 1$, $T = T + p^-$, $j^- = j^- + 1$
   (b) else $I_{k-1} \geq |r_{j^-}|$ and $T + p^- > d_{j^-}$
      i. $J^{-,e} = J^{-,e} \cup \{j^-\}$

ii. $j^* = \arg\min_{j' \in J^{-,e}} \{r_{j'}\}$

iii. schedule j^* in slot k'

iv. move jobs in $J^{-,e}$ following j^* in the slot of the previous job in $J^{-,e}$

v. adapt I_p, $1 \leq p \leq l-1$

vi. $k' = k' - 1$, $j^- = j^- + 1$

(c) else if $I_{k-1} < |r_{j^-}|$

i. schedule j^+ in slot k

ii. $I_k = I_{k-1} + r_{j^+}$

iii. $k = k+1$, $T = T + p_{j^+}$, $j^+ = j^+ + 1$

3. schedule jobs j^+ to e^+ next; set $k = k + e^+ + 1 - j^+$, $j^+ = e^+ + 1$

4. for $j^- \leq |J^+| + |J^-|$

(a) 2a

(b) 2b

(c) 2c

5. schedule jobs j^+ to $|J^+|$ of J^+ in slots $k' - |J^+| + j^+ - 1$ to k'

The algorithm above gives an optimal schedule when at least e^+ early jobs of J^+ are required. It guarantees e^+ early jobs of J^+ due to the stopping criteria of Loop 2 and Step 3. Note that Step 3 does nothing if during the execution of Loop 2 at least e^+ jobs have been scheduled early.

Obviously, for each position l the number of jobs of J^+ being scheduled in positions 1 to l is minimum subject to inventory constraints and a minimum number of k early jobs of J^+ (note that the order of jobs in J^+ is fixed). In turn, this means for each position l the number of jobs of J^- being scheduled in positions 1 to l is maximum subject to inventory constraints and a minimum number of k early jobs of J^+.

First, we show that there is an optimal schedule having job j^* being chosen in Step 2(b)ii tardy. Consider the first job J^-_p, $1 \leq p \leq |J^-|$ being chosen tardy and let $J^-_{j^-}$ be the job which should be scheduled next. Since the number of jobs in J^+ being scheduled before is minimum there must be tardy job among the the first j^- jobs of J^-. Let σ^* and σ be an optimal schedule and the schedule provided by our algorithm, respectively. Suppose J^-_p is not tardy in σ^*. Then at least one other job J^-_q, $1 \leq q \leq |J^-|$, $q \neq p$, is scheduled tardy. Since $r_{J^-_p} \leq r_{J^-_q}$ we can exchange J^-_p and J^-_q in σ^* without violating inventory constraints. If $d_{J^-_p} \geq d_{J^-_q}$, then J^-_p is early. If $d_{J^-_p} < d_{J^-_q}$, then reordering of the jobs scheduled early may be necessary. However, it is easy to see we can feasibly reorder them in EDD order.

From the fact the we feasibly choose tardy jobs and maximize the number of jobs of J^- scheduled up to each position, correctness of the algorithm follows. Solving the problem described above takes $O(n \log n)$ (see Moore [7]) for each $1 \leq k \leq n$ and, thus, overall complexity is $O(n^2 \log n)$. □

Theorem 14. *If $F_d(J^+) = 1$ and $F_p(J^+) + F_r(J^+) \geq 1$, then $1|inv(1)|\sum U_j$ can be solved in $O\left(n^2 \left(\sum r_j\right)^2 \left(\sum p_j\right)^3\right)$.*

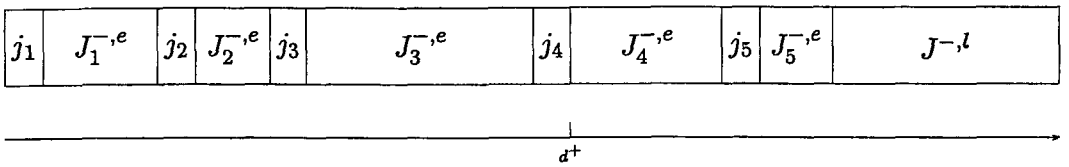

Figure 3: Schedule Structure

Proof. It is easy to see that J^+ can be assumed to be ordered in non-increasing inventory modification if $F_p(J^+) = 1$ and in SPT order if $F_\tau(J^+) = 1$. We can restrict ourselves to schedules being structured as in the proof of Theorem 13, see Figure 3.
Let $J^+ = \{1, \ldots, |J^+|\}$ be ordered as described above and let $J^- = \{|J^+|+1, \ldots, |J^+|+|J^-|\}$ be in EDD order.
We consider the subproblem where the number of early jobs in J^+ is required to be at least e^+. Note that job e^+ can be finished in each point of time in

$$\left[\sum_{j \le e^+} p_j, d^+\right]$$

and, hence, in no more than

$$d^+ \le \sum_{j \in J} p_j$$

periods. For each $1 \le e^+ \le |J^+|$ and each

$$C_{e^+} \in [\sum_{j \le e^+} p_j, d^+]$$

(and, therefore, no more than $n \sum p_j$ times) we define graph $G = (V, E, c)$ as follows. Let

$$P^+ = \sum_{j'=1}^{e^+} p_{j'} \text{ and } R^+ = R + \sum_{j'=1}^{e^+-1} r_{j'}.$$

$$\begin{aligned}
V &= \left\{(j, P_1^e, P_2^e, R_1^e, R_2^e) \mid j \in J^-, n_1^e, n^e \le j, P_1^e, P_2^e \le \sum p_j, R_1^e, R_2^e \le \sum r_j\right\} \\
E &= E_1^e \cup E_2^e \cup E^l \\
E_1^e &= \{((j, P_1^e, P_2^e, R_1^e, R_2^e), (j+1, P_1^e + p_{j+1}, P_2^e, R_1^e + r_{j+1}, R_2^e)) \mid \\
&\quad j < n, P_1^e + p_{j+1} \le C_{e^+} - P^+, -R_1^e - r_{j+1} \le R^+, C_1(j+1, P_1^e, R_1^e) \le d_{j+1}\} \\
E_2^e &= \{((j, P_1^e, P_2^e, R_1^e, R_2^e), (j+1, P_1^e, P_2^e + p_{j+1}, R_1^e, R_2^e + r_{j+1})) \mid \\
&\quad j < n, C_2(j+1, P_2^e, R_1^e, R_2^e) \le d_{j+1}\} \\
E^l &= \{((j, P_1^e, R_1^e, R_2^e), (j+1, P_1^e, R_1^e, R_2^e)) \mid j < n\} \\
c(e) &= 0 \quad \forall e \in E_1^e \cup E_2^e \\
c(e) &= 1 \quad \forall e \in E^l
\end{aligned}$$

Node $(j, P_1^e, P_1^l, P_2^e, R_2)$ represents a partial solution having jobs $|J^+|+1$ to $|J^+|+j$ assigned to $J_1^{-,e}$, $J_2^{-,e}$, and $J^{-,l}$ such that

$$\sum_{j' \in J_1^{-,e}} p_{j'} = P_1^e, \sum_{j' \in J_2^{-,e}} p_{j'} = P_2^e, \sum_{j' \in J_1^{-,e}} r_{j'} = R_1^e, \sum_{j' \in J_2^{-,e}} r_{j'} = R_2^e.$$

Edges in E_1^e and E_2^e represent the assignment of job $j+1$ to $J_1^{-,e}$ and $J_2^{-,e}$. Assignment of job $j+1$ to $J_1^{-,e}$ is possible only if e^+ still can be finished by C_{e+} and the inventory is not decreased below zero. Furthermore, job $j+1$ must be early when scheduled after those jobs assigned to $J_1^{-,e}$ earlier. Function C_1 gives the completion time and is specified below. Assignment of job $j+1$ to $J_2^{-,e}$ is possible only if job $j+1$ is early when scheduled after those jobs assigned to $J_2^{-,e}$ earlier. Function C_2 gives the corresponding completion time.
Edges in E^l represent the assignment of job $j+1$ to $J^{-,l}$ which is always possible. The length of an edges representing a job being scheduled tardy (early) is 1 (0) since this assignment contributes 1 (0) to the objective value. Completion time functions C_1 and C_2 sind defined as follows.

$$\begin{aligned} C_1(j, P_1^e, R_1^e) &= P_1^e + p_j + \sum_{j' \in J^-, j' \leq s(|R_1^e|)} p_{j'} \\ C_2(j, P_2^e, R_1^e, R_2^e) &= C_{e+} + P_2^e + p_j + \sum_{j' \in J^-, j' \leq s(|R_1^e|+|R_2^e|)} p_{j'} \\ s(R) &= \min\{j \in J^+ \mid \sum_{j' \in J^+, j' \leq j} r_{j'} \geq R\} \end{aligned}$$

The completion time C_j of job $j \in J_1^{-,e}$ is the sum of all preceding jobs plus its own processing time. Preceding jobs are jobs already assigned to $J_1^{-,e}$ before. Additionally, $s(R)$ is the first job in J^+ such that jobs 1 to $s(R)$ provide an inventory level such that j can be processed (after jobs assigned to $J_1^{-,e}$ earlier have been processed before). Jobs 1 to $s(R)$ must be processed before j also and, therefore, their processing times contribute to the completion time of j. Completion time C_j of job $j \in J_2^{-,e}$ is defined analogously taking into account that the completion time of e^+ is fixed.
Clearly, we have

$$|V| = |E| \in O\left(n \left(\sum r_j\right)^2 \left(\sum p_j\right)^2\right)$$

and we can find a shortest path from an artificial source $(0,0,0,0,0)$ to a node $(|J^+| + |J^-|, P_1^e, P_1^l, P_2^e, R_2)$ in

$$O\left(n \left(\sum r_j\right)^2 \left(\sum p_j\right)^2\right).$$

Then, solving the problem for each

$$1 \leq e^1 \leq |J^+| \text{ and } C_{e+} \in [\sum_{j \leq e^+} p_j, d^+]$$

takes $O\left(n^2 \left(\sum r_j\right)^2 \left(\sum p_j\right)^3\right)$. □

Theorem 15. *If $F_d(J^-) = F_p(J^+) = 1$ and $F_p(J^-) + F_r(J^-) \geq 1$, then $1|inv(1)|\sum U_j$ can be solved in $O(n^6)$.*

Proof. It is easy to see that J^- can be ordered according non-increasing inventory modification if $F_p(J^-)$ and in SPT order if $F_r(J^-)$. We consider the subproblem where the number of early jobs of J^- is required to be at least e^-. We can see that with same arguments we can restrict ourselves to the same structure of schedules as in the proof of Theorem 10 (see Figure 2) and, therefore, we consider subsets of jobs J_1^+, J_2^+, $J_1^{+,e}$, $J_1^{+,l}$, $J_2^{+,e}$, $J_2^{+,l}$, and $J^{-,e}$ (as defined in the proof of Theorem 10) in the following. Furthermore, we can fix

$$|J_1^+| = \min\left\{\left\lfloor \frac{d^- - \sum_{j\in J^{-,e}} p_j}{p^+}\right\rfloor, |J^+|\right\} = l_p.$$

Clearly, $|J_1^+| \leq l_p$ since otherwise jobs of $J^{-,e}$ are tardy or we have machine idle time before J^+. If $|J_1^+| < l_p$ we must have $|J_2^+| \neq \emptyset$. Hence, we can switch $J^{-,e}$ and the first $l_p - |J_1^+|$ jobs of J_2^+. The solution cannot be worsened since all jobs of $J^{-,e}$ are early and no other job has been delayed. Furthermore, inventory constraints cannot be violated since only jobs of J^- have been delayed. Obviously, from $|J_1^+| = l_p$ we obtain $|J_2^+| = |J^+| - l_p$.
Let jobs of J^+ be numbered in EDD. We define a graph $G = (V, E, c)$ where $c : e \to (u, r)$, $u \in \{0, 1\}$, $r \in \{0, \ldots, M\}$ for each $e \in E$.

$$\begin{aligned}
V &= \{(j, n_1^e, n_1^l, n_2^e) \mid j \in J^+, n_1^e, n_1^l, n_2^e \leq j\}\\
E &= E_1^e \cup E_2^e \cup E_1^l \cup E_2^l\\
E_1^e &= \{((j, n_1^e, n_1^l, n_2^e), (j+1, n_1^e+1, n_1^l, n_2^e)) \mid j < n, n_1^e + 1 + n_1^l \leq l_p,\\
&\quad d_{j+1} \geq (n_1^e + 1)p^+\}\\
E_1^l &= \{((j, n_1^e, n_1^l, n_2^e), (j+1, n_1^e, n_1^l+1, n_2^e)) \mid j < n, n_1^e + 1 + n_1^l \leq l_p\}\\
E_2^e &= \Big\{((j, n_1^e, n_1^l, n_2^e), (j+1, n_1^e, n_1^l, n_2^e+1)) \mid j < n, j + 1 - n_1^e - n_1^l \leq |J^+| - l_p,\\
&\quad d_{j+1} \geq l_p p^+ + \sum_{j\in J^{-,e}} p_j + (n_2^e + 1)p^+\Big\}\\
E_2^l &= \{((j, n_1^e, n_1^l, n_2^e), (j+1, n_1^e, n_1^l, n_2^e)) \mid j < n, n_2^e + 1 + n_2^l \leq |J^+| - l_p\}\\
c(e) &= (0, M - r_{i+1}) \quad \forall e \in E_1^e\\
c(e) &= (1, M - r_{i+1}) \quad \forall e \in E_1^l\\
c(e) &= (0, 0) \quad \forall e \in E_2^e\\
c(e) &= (1, 0) \quad \forall e \in E_2^l
\end{aligned}$$

Node (j, n_1^e, n_1^l, n_2^e) represents a partial solution such that jobs 1 to j are assigned to $J_1^{+,e}$, $J_2^{+,e}$, $J_1^{+,l}$, and $J_2^{+,l}$ such that

$$|J_1^{+,e}| = n_1^e, |J_2^{+,e}| = n_2^e, |J_1^{+,l}| = n_1^e, |J_2^{+,l}| = j - n_1^e - n_2^e - n_1^l.$$

Edges in E_1^e and E_2^e represent the assignment of job $j+1$ to $J_1^{+,e}$ and $J_2^{+,e}$, respectively. This is possible only if job $j+1$ is early when scheduled at the end of the corresponding set. Furthermore, assignment of $j+1$ to $J_1^{+,e}$ and $J_2^{+,e}$ is allowed only if the total number of jobs in J_1^+ and J_2^+, respectively, is not exceeded.
Edges in E_1^l and E_2^l represent the assignment of job $j+1$ to $J_1^{+,l}$ and $J_2^{+,l}$. Assignment of $j+1$ to $J_1^{+,l}$ and $J_2^{+,l}$ is allowed only if the total number of jobs in J_1^+ and J_2^+, respectively, is not exceeded.
If $e \in E_1^l \cup E_2^l$ and $e \in E_1^e \cup E_2^e$ we have $u(e) = 1$ and $u(e) = 0$ since this assignment contributes 1 and 0, respectively, to the objective value.

If $e \in E_1^e \cup E_1^l$ we have $r(e) = M - r_{j+1}$. Since the number of jobs in $e \in E_1^e \cup E_1^l$ in the solution is fixed we get total

$$\sum_{e \in P} d(e) = l_p M - \sum_{j \in J_1^+} r_j$$

for each path P. Note that for the solution to be feasible we must have

$$\sum_{j \in J_1^+} r_j \geq - \sum_{j \in J^{-,e}} r_j.$$

Clearly, $|V| = |E| \in O(n^4)$. We find a restricted shortest path (regarding the first component of c) where a path is a restricted path if and only if the sum of the second cost components of all edges in the path sum up to no more than a prespecified upper bound

$$\overline{d} = l_p M - \sum_{j \in J^{-,e}} r_j.$$

According to Hassin [4], we can solve the restricted shortest path problem in $O(\overline{u}|E|)$ if the shortest restricted path is known to be not longer than $\overline{u}$. In our case $\overline{u} = n$ and hence computational complexity for finding the restricted shortest path is $O(n^5)$.Then, solving the problem described above for each $1 \leq k \leq n$ takes $O(n^6)$. □

Theorem 16. *If $F_p(J^-) = F_r(J^-) = F_p(J^+) = F_r(J^+) = 1$, then $1|inv(1)|\sum U_j$ can be solved in $O(n^2 \log n)$.*

Proof. Let $J^+ = \{1, \ldots, |J^+|\}$ and $J^- = \{|J^+|+1, \ldots, |J^+|+|J^-|\}$ be in EDD order. We consider the subproblem where the number of early jobs of J^+ and J^- is required to be at least e^+ and e^-, respectively. First, we develop a property of optimal schedules for this subproblem that there is a feasible schedule for this subproblem having a structure as follows.

Property 1. *There is an optimal schedule such that we have subsets of early and tardy jobs, respectively, of J^+ and J^- as follows.*

$$\begin{aligned}
J^{+,e} &= \{|J^+| - e^+ + 1, \ldots, |J^+|\} \\
J^{+,l} &= \{1, \ldots, |J^+| - e^+\} \\
J^{-,e} &= \{|J^+| + |J^{-,l}| - e^- + 1, \ldots, |J^+| + |J^-|\} \\
J^{-,l} &= \{|J^+| + 1, \ldots, |J^+| + |J^-| - e^-\},
\end{aligned}$$

that is for each subset the e^+ and e^-, respectively, jobs having the largest due dates are chosen to be early. Furthermore,, we can assume orders of jobs of J^+ and J^- to be

$$(|J^+| - e^+ + 1, \ldots, |J^+|, 1, \ldots, |J^+| - e^+) \text{ and}$$
$$\left(|J^+| + |J^{-,l}| - e^- + 1, \ldots, |J^+| + |J^-|, |J^+| + 1, \ldots, |J^+| + |J^-| - e^-\right).$$

Suppose we have a solution σ where $|J_\sigma^{-,l}| = |J^-| - e^-$ and $J_\sigma^{-,l} \neq J^{-,l}$. Let $j \in J^{-,l}$ be the first job that is not in $J_\sigma^{-,l}$. Then, there is at least one job of $j' \in J^{-,e} \cap J_\sigma^{-,l}$. Note that $d_{j'} \geq d_j$ and, hence, we can exchange j and j' in σ without violating any constraint. Furthermore, there is an feasible solution to the subproblem having jobs $1, \ldots, |J^+| - e^+$ last in J^+ and jobs $|J^+| - e^+ + 1, \ldots, |J^+|$ in EDD order by the same arguments as above. Moreover, it is easy to see that $J^{+,e}$, $J^{+,l}$, $J^{-,e}$, and $J^{-,l}$ can be ordered in EDD order.

Summarizing, orders for both subsets J^+ and J^- of jobs can be assumed to be fixed for given e^+ and e^-. Let both subsets be given in the corresponding order and numbered accordingly for given e^+ and e^- in the following. Let J_l^+ and J_k^- denote the lth and kth job in J^+ and J^-, respectively.

Algorithm 5

1. $j^+ = 1$, $j^- = |J^+| + 1$, $I = R$, $T = 0$
2. for $j^+ \leq |J^+|$ and $j^- \leq |J^+| + |J^-|$
   (a) if $I \geq |r^-|$ and $d_{j^-} < d_{j^+}$ and $d_{j^-} \geq T + p^-$
      i. schedule j^- next
      ii. $I = I + r^-$
      iii. $j^- = j^- + 1$
   (b) else if $(I < |r^-|$ or $d_{j^-} \geq d_{j^+})$ and $d_{j^+} \geq T + p_{j^+}$
      i. schedule j^+ next
      ii. $I = I + r^+$
      iii. $j^+ = j^+ + 1$
   (c) else STOP
3. schedule remaining jobs in J^- or J^+ and check feasiblity (STOP in case of infeasibility)

The algorithm gives a feasible solution if there is one in $O(n)$. Assume there is a feasible sequence σ and the algorithm aborted without finding a schedule. Let k be the first position where σ and the partial schedule σ^p constructed by our algorithm differ. If no job was found for position k by our algorithm, then σ must be infeasible. Note that $r_{\sigma^p(k)} r_{\sigma(k)} < 0$ due to Property 1 otherwise. If a job was assigned to position k by our algorithm, then let k' denote the position of $\sigma^p(k)$ in σ.
If $r_{\sigma^p(k)} < 0$, then $d_{\sigma^p(k)} < d_{\sigma(k)}$. If, additionally, $\sigma(k'-1) \geq |J^+| - e^+$, then $\sigma(k'-1)$ must be early and $d_{\sigma(k)} \leq \ldots \leq d_{\sigma(k'-1)}$. Then, moving $\sigma^p(k)$ in position k of σ and delaying jobs in positions k to $k'-1$ by one position cannot make any job tardy. Moreover, since $\sigma^p(k)$ is a feasible choice regarding inventory constraints we cannot violate inventory constraints in σ. By a similar argument if $\sigma(k'-1) < |J^+| - e^+$, then only jobs $1, \ldots, |J^+| - e^+$ can become tardy by this move. Obeviously, this is feasible to the subproblem. If $r_{\sigma^p(k)} \geq 0$, then the arguments are analogous to the case above.
Next, we consider the subproblem to find a schedule where the number of early jobs is required to be at least $0 \leq e \leq n$, respectively. We can solve this subproblem be solving the subproblem above for each $0 \leq e^+ \leq e$ and $e^- = e - e^+$ which takes $O(n^2)$. Employing binary search on the domain of $0 \leq e \leq n$ we can solve the overall problem in $O(n^2 \log n)$. □

Theorem 17. *If $F_d(J^-) = F_r(J^-) = F_d(J^+) = F_r(J^+) = 1$, then $1|inv(1)|\sum U_j$ can be solved in $O(n^2)$.*

We give only a sketch of the proof in the following since it employs techniques used intensively in preceding proofs.

Proof. Both subsets J^+ and J^- can be assumend to be ordered in SPT order. Varying the number of early jobs in J^- we can find the optimal solution in $O(n^2)$ □

Summarizing the results provided by Theorems 8, 9, 10 and Theorems 12 to 17, we can state that

- $1|inv(1)|\sum U_j$ is unary NP-hard even if $F_p(J^-) = 1$, $F_r(J^-) = 1$, and $F_d(J^+) = 1$,
- $1|inv(1)|\sum U_j$ is binary NP-hard even if jobs of J^- are identical and $F_d(J^+) = 1$,
- $1|inv(1)|\sum U_j$ is binary NP-hard even if $F_p(J^+)+F_r(J^+) \geq 1$ and $F_d(J^+)+F_d(J^-) \geq 1$,
- $1|inv(1)|\sum U_j$ is NP-hard even if $F_p(J^+) + F_r(J^+) \geq 1$,
- $1|inv(1)|\sum U_j$ is polynomially solvable if $F_p(J^+) = 1$, $F_d(J^-) = 1$, and $F_p(J^-) + F_r(J^-) \geq 1$,
- $1|inv(1)|\sum U_j$ is polynomially solvable if $F_p(J^-) = 1$, $F_d(J^+) = 1$, and $F_p(J^+) + F_r(J^+) \geq 1$,
- $1|inv(1)|\sum U_j$ is polynomially solvable if $F_p(J^+) = 1$, $F_r(J^+) = 1$, $F_p(J^-) = 1$, and $F_r(J^-) = 1$, and
- $1|inv(1)|\sum U_j$ is polynomially solvable if $F_d(J^+) = 1$, $F_r(J^+) = 1$, $F_d(J^-) = 1$, and $F_r(J^-) = 1$.

4 Conclusions and Outlook

In this paper we introduce inventory constraints to be embedded in the well-known machine scheduling framework. Inventory constraints can be seen as a generalization of precedence constraints. We consider different objective functions and special cases where one or more parameter of subsets J^+ or J^- of jobs are identical. These special cases are motivated by real world scheduling problems arising in cross-dock terminals, for example.
The general problems turn out to be strongly NP-hard even for $L_{\max}$ and $\sum C_j$. However, for several of the special cases polynomial algorithms are developed. Furthermore, we provide two pseudo polynomial time algorithms. Although we could determine computational complexity of many special cases some remain open so far. Table 1 lists minimum and maximum open problems for objective functions $\sum w_j C_j$ and $\sum U_j$. Note that no problems for $\sum C_j$ and $L_{\max}$ are left open.
For the future we identify several promising fields of research in this area. First, computational complexity of the open cases should be determined. Moreover, objective functions $\sum_{w_j U_j}$, $\sum_{T_j}$, and $\sum_{w_j T_j}$ are not considered so far. Second, we propose to consider problems with more than one machine. Third, a setting where $|T| > 1$ but no job is involved with more than one type seems to be interesting. The question to answer first for this setting is whether we can extend algorithms developed in the paper at hand to handle this case. Last but not least it is important to develop efficient heuristics or approximation algorithms for the problems that are strongly NP-hard. Here, it seems to be promising to employ algorithms for special cases as building blocks.

Prob		$F_p(J^+)$	$F_w(J^+)$	$F_d(J^+)$	$F_r(J^+)$	$F_p(J^-)$	$F_w(J^-)$	$F_d(J^-)$	$F_r(J^-)$
$1\|inv(1)\|\sum w_jC_j$	min open	0	0		1	1	1		1
		1	1		1	0	0		1
	max open	0	0		1	0	0		1
		0	0		1	1	1		0
		1	1		0	0	0		1
$1\|inv(1)\|\sum U_j$	min open	0		0	1	1		1	1
		1		0	0	1		0	1
		1		0	1	1		0	0
		1		1	1	0		0	1
	max open	0		0	1	0		0	1
		1		0	0	0		0	1
		0		0	1	1		0	0
		1		0	0	1		0	0

Table 1: Open Problems

References

[1] J.-H. Bartels and J. Zimmermann. Scheduling tests in automotive R&D projects. *European Journal of Operational Research*, In Press, corrected proof, available online 7 November 2007.

[2] N. Boysen, M. Fliedner, and A. Scholl. Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum*, to appear.

[3] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimisation and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:236–287, 1979.

[4] R. Hassin. Approximation Schemes for the Restricted Shortest Path Problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[5] E. Lawler. Optimal Sequencing of a Single Machine subject to Precedence Constraints. *Management Science*, 19:544–546, 1973.

[6] J. Lenstra, A. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343362, 1977.

[7] J. M. Moore. An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs. *Management Science*, 15(1):102–109, 1968.

[8] K. Neumann and C. Schwindt. Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56:513–533, 2002.

[9] K. Neumann, C. Schwindt, and N. Trautmann. Scheduling of continuous and discontinuous material flows with intermediate storage restrictions. *European Journal of Operational Research*, 165:495–509, 2005.

[10] C. Schwindt and N. Trautmann. Batch scheduling in process industries: An application of resource-constrained project scheduling. *OR Spectrum*, 22(4):501–524, 2000.

A NP-Hardness of Feasibility of $1|\overline{inv}(1)|\gamma$

We only give the reduction scheme from 3-PARTITION. It is easy to see that there is a feasible schedule if and only if the answer to the instance of 3-PARTITION is yes.

Proof. Consider an instance P of 3-PARTITION defined by $B \in \mathbb{Z}^+$, set A, $|A| = 3m$, $B/4 < a_j < B/2$ for each $j \in \{1, \ldots, 3m\}$. We construct an instance P' of $1|\overline{inv}(1)|\gamma$ as follows:

- $n = 4m$
- $p_j = 1$
- $r_j = -a_j$ if $j \in \{1, \ldots, 3m\}$, $r_j = B$ otherwise
- $R = 0$

- $\overline{I} = B$

The idea is that jobs $3m+1$ to $4m$ partition the remaining jobs 1 to $3m$ into groups of three jobs such that each group has overall consumption of B. □

B NP-Hardness of Feasibility of $1|inv(2)|\gamma$

Again we only give the reduction scheme from 3-PARTITION.

Proof. Consider an instance P of 3-PARTITION defined by $B \in \mathbb{Z}^+$, set A, $|A| = 3m$, $B/4 < a_j < B/2$ for each $j \in \{1, \ldots, 3m\}$. We construct an instance P' of $1|inv(2)|\gamma$ as follows:

- $n = 4m + 1$
- $p_j = 1$
- $r^c_{j,1} = a_j$, $r^r_{j,1} = 0$, $r^c_{j,2} = 0$, and $r^r_{j,2} = a_j$ if $j \in \{1, \ldots, 3m\}$
- $r^c_{3m+1,1} = 0$, $r^r_{3m+1,1} = B$, $r^c_{3m+1,2} = 0$, and $r^r_{3m+1,2} = 0$
- $r^c_{j,1} = 0$, $r^r_{j,1} = B$, $r^c_{j,2} = B$, and $r^r_{j,2} = 0$ if $j \in \{3m+2, \ldots, 4m\}$
- $r^c_{4m+1,1} = 0$, $r^r_{4m+1,1} = 0$, $r^c_{4m+1,2} = B$, and $r^r_{4m+1,2} = 0$
- $R_1 = R_2 = 0$

Again, the idea is that jobs $3m+1$ to $4m+1$ partition the remaining jobs 1 to $3m$ into groups of three jobs such that each group has overall consumption of type 1 of B and overall release of type 2 of B. □

C NP-Hardness of $1|inv(1)|\sum C_j$

Proof. Consider an instance P of 3-PARTITION defined by $B \in \mathbb{Z}^+$, set A, $|A| = 3m$, $B/4 < a_j < B/2$ for each $j \in \{1, \ldots, 3m\}$. We construct an instance P' of $1|inv(1)|\sum C_j$ as follows:

- $n^a = 3m$
- $n^b = 2mB - 2$
- $n^- = m$
- $n = n^a + mn^b + n^-$
- $J^a = \{1, \ldots, n^a\}$, $J^b = \{n^a + n^- + 1, \ldots, n^a + n^- + mn^b\}$, $J^+ = J^a \cup J^b$, $J^- = \{n^a + 1, \ldots, n^a + n^-\}$, $J = J^+ \cup J^-$
- $p = \max\{2n^b Bm^2, (n^b + 3)(B+1)^{\frac{m(m-1)}{2}} + 3mB + 1\}$
- $p_j = p + a_j$ if $j \in J^a$, $p_j = p$ if $j \in J^b$, $p_j = 1$ if $j \in J^-$

- $r_j = p + a_j$ if $j \in J^a$, $r_j = p$ if $j \in J^b$, $r_j = -(n^b+3)p - B$ if $j \in J^-$

First, we derive a threshold for the objective value from a schedule having a structure such that we have m sections of n^b jobs out of J^b, three jobs out of J^a, and one job out of J^- in each section. Figure 1 illustrates this class of schedules. Suppose that the partition of jobs of J^a into sections implies a yes-answer to the instance of 3-PARTITION, then we can find an upper bound for the objective value. We develop upper bounds for the contributions γ^a, γ^b, and γ^- of subsets J^a, J^b, and J^- to the objective value $\overline{\gamma}$ in such a schedule. In each section jobs out of J^b are scheduled first and the job out of J^- last (see Figure 1). Let $j(i,l)$ denote the lth job in section i out of J^a for $1 \le i \le m$ and $1 \le l \le 3$.

$$\begin{aligned}
\gamma^a &= \sum_{i=1}^{m} \left(3\,(i-1)\left((n^b+3)p+B+1\right) + 3n^b p + 6p + 3a_{j(i,1)} + 2a_{j(i,2)} + a_{j(i,3)}\right) \\
&\le 3\left((n^b+3)p+B+1\right)\frac{m(m-1)}{2} + mp(3n^b+6) + \sum_{i=1}^{m}\left(3\frac{B}{3} + 2\frac{B}{3} + \frac{B}{3}\right) \\
&= 3\left((n^b+3)p+B+1\right)\frac{m(m-1)}{2} + 3mp(n^b+2) + 2mB
\end{aligned}$$

Note that

$$3a_{j(i,1)} + 2a_{j(i,2)} + a_{j(i,3)} \le 3\frac{B}{3} + 2\frac{B}{3} + \frac{B}{3}$$

since we can assume jobs out of J^+ in each section to be ordered in SPT order.

$$\begin{aligned}
\gamma^b &= \sum_{i=1}^{m}\left(n^b\,(i-1)\left((n^b+3)p+B+1\right) + \sum_{k=1}^{n^b} kp\right) \\
&= n^b\left((n^b+3)p+B+1\right)\frac{m(m-1)}{2} + mp\frac{n^b(n^b+1)}{2}
\end{aligned}$$

$$\begin{aligned}
\gamma^a+\gamma^b &= (n^b+3)^2 p\frac{m(m-1)}{2} + mp\left(\frac{(n^b)^2}{2} + 3.5n^b + 6\right) + \\
&\quad (n^b+3)(B+1)\frac{m(m-1)}{2} + 2mB \\
&= \frac{(n^b+3)^2pm^2}{2} - \frac{(n^b+3)^2pm}{2} + mp\left(\frac{(n^b)^2}{2} + 3.5n^b + 6\right) + \\
&\quad (n^b+3)(B+1)\frac{m(m-1)}{2} + 2mB \\
&= \frac{(n^b+3)^2pm^2}{2} + p\frac{m(n^b+3)}{2} + (n^b+3)(B+1)\frac{m(m-1)}{2} + 2mB
\end{aligned}$$

$$\begin{aligned}
\gamma^- &= \sum_{i=1}^{m} i\left((n^b+3)p+B+1\right) \\
&= \left((n^b+3)p+B\right)\frac{m(m+1)}{2}
\end{aligned}$$

$$\overline{\gamma} \leq \gamma^a + \gamma^b + \gamma^-$$

In the following we will show that there is a schedule to P' having total completion time not larger than $\overline{\gamma}$ if and only if the answer to P is yes.
If the answer to P is yes, then we can find a feasible solution to P' by scheduling m sections. We choose n^b arbitrary jobs of J^b to be scheduled first in each section, the partition gives m subsets of jobs of J^a to be scheduled in different section (we arrange them in SPT order in each section), and we choose an arbitrary job of J^- as last job of each section. Doing the math like above we obtain a total flow time of no more than $\overline{\gamma}$.
Suppose now we have a schedule having total completion time not larger than $\overline{\gamma}$. We show that this schedule must be structured as above and, moreover, induces a 3-partition for P (by the partition of J^a to subsets being scheduled in each section). First, it is obvious that we can schedule J^b and J^- in arbitrary order. We can assume that they are numbered in scheduled order.

Property 2. *Job j, $3m+1 \leq j \leq 4m$, can not be finished before*

$$C_j \geq (j-3m)((n^b+3)p+B+1).$$

Clearly, jobs $3m+1,\ldots,j-1$ must be finished and inventory must be at least $(n^b+3)p+B$. This induces that before j can be started at least

$$k = (j-3m)((n^b+3)p+B)$$

units of items must have been released (which implies k units of processing time used by jobs in J^+ before the start of j). Furthermore, $j-3m-1$ jobs of J^- (having processing time 1 each) must be finished. Including processing time of j itself this sums up to $k+(j-3m)$.

Property 3. *Job j, $3m+1 \leq j \leq 4m$, cannot be started before at least $k = (j-3m)(n^b+3)$ jobs out of J^+ have been finished.*
Suppose only k', $k' < k$, jobs are finished before j. There are $j-3m-1$ jobs of J^- being finished before j. Therefore, k' jobs out of J^+ must have released

$$(j-3m)((n^b+3)p+B)$$

units of items. However, k' jobs must release less than

$$\begin{aligned}
(k-1)\left(p+\frac{B}{2}\right) &= k\left(p+\frac{B}{2}\right) - \left(p+\frac{B}{2}\right) \\
&< kp + k\frac{B}{2} - 2n^bBm^2 \\
&\leq kp \\
&< kp + B(j-3m) \\
&= (j-3m)((n^b+3)p+B)
\end{aligned}$$

units of items.

Property 4. *No more than $k = (j-3m)(n^b+3)$ jobs out of J^+ can be finished before job j, $3m+1 \leq j \leq 4m$, is started. Suppose at least $k+1$ jobs out of J^+ are finished before j. Then, with Property 2 we have*

$$\begin{aligned}\sum_{j\in J^-} C_j &\geq \gamma^- - (j-3m)((n^b+3)p+B+1)+(k+1)p \\ &= \gamma^- - (j-3m)(B+1)+p.\end{aligned}$$

Furthermore, since all jobs in J^+ have processing time of at least p we have

$$\sum_{j\in J^+} C_j \geq \sum_{k=1}^{m(n^b+3)} kp.$$

Now, we can see that

$$\begin{aligned}\sum_{j\in J} C_j &\geq \gamma^- - (j-3m)B + p + \sum_{k=1}^{m(n^b+3)} kp \\ &= \gamma^- - (j-3m)B + p + p\frac{m(n^b+3)(m(n^b+3)+1)}{2} \\ &= \gamma^- - (j-3m)B + p + p\frac{m^2(n^b+3)^2}{2} + p\frac{m(n^b+3)}{2} \\ &= \gamma^- - (j-3m)B + p + \gamma^a + \gamma^b - (n^b+3)(B+1)\frac{m(m-1)}{2} - 2mB \\ &\geq \gamma^a + \gamma^b + \gamma^- + p - (n^b+3)(B+1)\frac{m(m-1)}{2} - 3mB \\ &\geq \overline{\gamma}+1\end{aligned}$$

where the last inequality holds if $m > 1$. Thus, the solution can not have an objective value of no more than $\overline{\gamma}$ by lower bound consideration.

Due to Property 3 and 4, exactly n^b+3 jobs out of J^+ are scheduled between job j and $j+1$ for $j = 3m+1, ..., 4m-1$. Thus, if $C_j \leq \overline{\gamma}$, the schedule must have a sector structure.

Property 5. *For job j, $3m+1 \leq j \leq 4m$, we have*

$$C_j = (j-3m)((n^b+3)p+B+1).$$

Due to Property 2, j cannot be finished before C_j. Now assume that j is finished after C_j.Since all jobs are scheduled in $m\left((n^b+3)p+B+1\right)$ time units $j < 4m$ must hold. If $j < 4m$, then sector $j+1$ starts not before $(j-3m)\left((n^b+3)p+B+1\right)+1$ and, thus,

$$\begin{aligned}\sum_{j\in J^+} C_j &\geq \sum_{i=1}^{m}\left((n^b+3)(i-1)((n^b+3)p+B+1)+\sum_{k=1}^{n^b+3} kp\right)+(n^b+3) \\ &= (n^b+3)^2 p\frac{m(m-1)}{2} + (n^b+3)(B+1)\frac{m(m-1)}{2} + mp\frac{(n^b+3)(n^b+4)}{2} + (n^b+3) \\ &= (n^b+3)^2 p\frac{m^2}{2} + mp\frac{(n^b+3)}{2} + (n^b+3)(B+1)\frac{m(m-1)}{2} + (n^b+3) \\ &= \gamma^a + \gamma^b - 2mB + n^b + 3 \\ &\geq \gamma^a + \gamma^b + 1\end{aligned}$$

Since the lower bound for the contribution to the objective value by jobs of J^- is given by $\gamma^- + 1$, we obtain an overall lower bound larger than $\overline{\gamma} + 2$. Thus, the solution can not have an objective value of no more than $\overline{\gamma}$ by lower bound consideration.

Due to Property 5, it is obvious that there exactly three jobs out of J^a between each pair of consecutive jobs out of J^-. The corresponding values of a_j in each section sum up to B. This completes the proof. □

D NP-Hardness of $1|inv(1)|L_{\max}$

Proof. The reduction is from 3-PARTITION. Consider an instance P of 3-PARTITION defined by $B \in \mathbb{Z}^+$, set A, $|A| = 3m$, $B/4 < a_j < B/2$ for each $j \in \{1, \ldots, 3m\}$. We construct an instance P' of $1|inv(1)|L_{\max}$ as follows:

- $n = 4m$
- $J^+ = \{1, \ldots, 3m\}$, $J^- = \{3m+1, \ldots, 4m\}$
- $p_j = a_j$ if $j \in J^+$, $p_j = 1$ if $j \in J^-$
- $r_j = a_j$ if $j \in J^+$, $r_j = -B$ if $j \in J^-$
- $d_j = (B+1)m$ if $j \in J^+$, $d_j = (j-3m)(B+1)$ if $j \in J^-$
- $R = 0$

In the following we will show that there is a schedule to P' where each job is finished not later than its due date if and only if the answer to P is yes.
If the answer to P is yes, then we can construct a solution to P' having no tardy jobs by scheduling induced subsets of A and jobs of J^- (in order increasing due dates) alternatingly.
Suppose there is a solution to P' having no tardy job.
Since jobs of J^- only differ by their due dates we can assume that they are scheduled in order of increasing due dates.
Let $J_k \subset J^+$, $k \in \{1, \ldots, m\}$, be the set of jobs scheduled before job $3m + k$. Obviously,

$$\sum_{j \in J_k} r_j \geq kB \quad \forall k \in \{1, \ldots, m\}.$$

Furthermore,

$$\sum_{j \in J_k} r_j = \sum_{j \in J_k} p_j \leq d_{3m+k} - k = k(B+1) - k = kB.$$

The Theorem follows. □

E NP-Hardness of $1|inv(1)|L_{\max}$, jobs in J^- identical and $d_j = d^+$ for $j \in J^+$

Proof. The reduction is from PARTITION. Consider an instance P of PARTITION defined by set A, $|A| = 2m$, $a_j \in \mathbb{N}$ for each $j \in \{1, \ldots, 2m\}$. We construct an instance P' of $1|inv(1)|L_{\max}$ as follows:

- $n = 2m + 1$
- $J^+ = \{1, \ldots, 2m\}$, $J^- = \{j^-\}$
- $p_j = a_j$ if $j \in J^+$, $p_{j^-} = 1$ otherwise
- $r_j = a_j$ if $j \in J^+$, $r_{j^-} = -\left(\sum_{j \in J^+} r_j\right)/2$ otherwise
- $d_j = 1 + \sum_{1 \leq j' \leq 2m} a_{j'}$ if $j \in J^+$, $d_{j^-} = \left(\sum_{j \in J^+} p_j\right)/2 + 1$ otherwise
- $R = 0$

In the following we will show that there is a schedule to P' having no tardy job if and only if the answer to P is yes.
Suppose the answer to P is yes. Let A_1 and A_2 be the induced subsets of A. Then we can find a solution to P' having no tardy job by scheduling the subsets of jobs corresponding to A_1 first, scheduling job $2m + 1$ next and, scheduling A_2 last.
Suppose there is a solution to P' having no tardy job. Let J_1 be the subset of J^+ scheduled before job $2m + 1$. Obviously,

$$\sum_{j \in J_1} r_j \geq \frac{\sum_{1 \leq j' \leq 2m} a_{j'}}{2}.$$

Furthermore,

$$\sum_{j \in J_1} r_j = \sum_{j \in J_1} p_j \leq d_{2m+1} - 1 = \frac{\sum_{j \in J^+} p_j}{2} = \frac{\sum_{1 \leq j' \leq 2m} a_{j'}}{2}.$$

The Theorem follows. □

F NP-Hardness of $1|inv(1)|\sum U_j$, jobs in J^+ identical and $d_j = d^-$ for $j \in J^-$

Proof. The reduction is from PARTITION. Consider an instance P of PARTITION defined by set A, $|A| = 2m$, $a_j \in \mathbb{N}$ for each $j \in \{1, \ldots, 2m\}$. Let

$$a = \sum_{j \in \{1, \ldots, 2m\}} a_j.$$

We construct an instance P' of $1|inv(1)|\sum U_j$ as follows:

- $n = 2m + 2$
- $J^- = \{1, \ldots, 2m\}$, $J^+ = \{2m + 1, 2m + 2\}$
- $p_j = a + a_j$ if $j \in J^-$, $p_j = 2ma$ otherwise
- $r_j = a_j - a$ if $j \in J^-$, $r_j = (m - 1/2)\, a$ otherwise
- $d_j = (3m + 1/2)\, a$ if $j \in J^-$, $d_j = 2ma$ otherwise

- $R = 0$

We show that there is a solution to P' having no more than $m+1$ tardy jobs if and only the answer to P is yes.
Suppose there is a solution to P' having no more than $m+1$ tardy jobs. We can assume that job $2m+1$ is scheduled before job $2m+2$. Obviously, job $2m+1$ must be scheduled first and, therefore, it is not tardy. Scheduling job $2m+2$ next leads to exactly 1 non-tardy job since each job in $1, \ldots, 2m$ is late and so is job $2m+2$. Therefore, we can assume that the set $J' \subset \{1, \ldots, 2m\}$ of additional non-tardy jobs is scheduled between jobs $2m+1$ and $2m+2$.
Due to the due dates we have

$$\sum_{j \in J'} p_j \leq \left(m + \frac{1}{2}\right) a$$

and, hence, $|J'| \leq m$. Considering that the overall number of tardy jobs is no more than $m+1$ we have $|J'| = m$. From $|J'| = m$ and

$$\sum_{j \in J'} p_j \leq \left(m + \frac{1}{2}\right) a$$

it follows that

$$\sum_{j \in J'} a_j \leq \frac{a}{2}.$$

Furthermore, we have

$$-\sum_{j \in J'} r_j = ma - \sum_{j \in J'} a_j \leq \left(m - \frac{1}{2}\right) a \Leftrightarrow \sum_{j \in J'} a_j \geq \frac{a}{2}.$$

Overall, we have $\sum_{j \in J'} a_j = a/2$.
If the answer to P is yes, then obviously the partition gives J' and $J \setminus J'$. □