



seit 1558

Friedrich-Schiller-Universität Jena

## **Jena Research Papers in Business and Economics**

### **Scheduling of inventory releasing jobs to satisfy time-varying demand**

*Nils Boysen, Stefan Bock, Malte Fliedner*

02/2011

*Jenaer Schriften zur Wirtschaftswissenschaft*

**Working and Discussion Paper Series  
School of Economics and Business Administration  
Friedrich-Schiller-University Jena**

ISSN 1864-3108

**Publisher:**

Wirtschaftswissenschaftliche Fakultät  
Friedrich-Schiller-Universität Jena  
Carl-Zeiß-Str. 3, D-07743 Jena  
[www.jbe.uni-jena.de](http://www.jbe.uni-jena.de)

**Editor:**

*Prof. Dr. Hans-Walter Lorenz*  
[h.w.lorenz@wiwi.uni-jena.de](mailto:h.w.lorenz@wiwi.uni-jena.de)  
*Prof. Dr. Armin Scholl*  
[armin.scholl@wiwi.uni-jena.de](mailto:armin.scholl@wiwi.uni-jena.de)

[www.jbe.uni-jena.de](http://www.jbe.uni-jena.de)

# Scheduling of inventory releasing jobs to satisfy time-varying demand

Nils Boysen<sup>a</sup>, Stefan Bock<sup>b</sup>, Malte Fliedner<sup>a</sup>

<sup>a</sup>*Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3, D-07743 Jena, Germany, {nils.boysen,malte.fliedner}@uni-jena.de*

<sup>b</sup>*Bergische Universität Wuppertal, Lehrstuhl für Wirtschaftsinformatik und Operations Research, Gaußstraße 20, D-42097 Wuppertal, Germany, sbock@winfor.de*

## Abstract

This paper studies a new class of single-machine scheduling problems, that are faced by Just-in-Time-suppliers satisfying a given demand. In these models the processing of jobs leads to a release of a predefined number of product units into inventory. Consumption is triggered by predetermined time-varying, and product-specific demand requests. While all demands have to be fulfilled, the objective is to minimize the resulting product inventory. We investigate different subproblems of this general setting with regard to their computational complexity. For more restricted problem versions (equal processing times and equal number of released products) strongly polynomial time algorithms are presented. In contrast to this, NP-hardness in the strong sense is proven for more general problem versions (varying processing times or varying number of released products). Moreover, for the most general version, even finding a feasible solution is shown to be strongly NP-hard.

*Keywords:* Machine scheduling; Inventory; Time-varying demand; Computational complexity

## 1 Introduction

This paper considers a new class of scheduling problems, that is motivated by real-world settings where a single server (or machine) has to ensure an efficient Just-in-Time (JIT) supply of some predetermined process defining specific demand events over time. The completion of each job releases a number of items of a specific product (type) and released items are subsequently consumed by predetermined (product-specific and time-varying) demand requests that occur during the considered planning horizon. While all demand requests have to be fulfilled, the minimization of resulting inventory is pursued, which is represented either by total or maximum inventory levels. Clearly, this problem setting is relevant in a large variety of existing industrial production

and logistic processes:

*Production scheduling:* Consider a production stage where different batches of predetermined sizes are produced. These batches are consumed by a succeeding cost-intensive and time-critical stage. Therefore, its production schedule is assumed to be determined in a preceding step and has to be ensured by a supply of needed modules or parts. A typical example for such a setting is a mixed-model assembly line, on which, e.g., cars are manufactured. Usually, the sequence of car models that have to be produced on the line is fixed days before production starts and communicated to part suppliers. Consequently, a predetermined deterministic and time-varying demand for different parts arises, which depends on the specification of car models in the predetermined production sequence. This demand has to be satisfied JIT either by succeeding in-house production stages or external suppliers. Hence, for instance, our problem setting arises when batches of different seat types are produced on a single (feeder) assembly line. Clearly, no station of the final assembly line may ever run out of parts, as this might cause a line stoppage with hundreds of assembly workers being idle. Moreover, in accordance with JIT-principles, our model pursues the minimization of resulting total and maximum inventory that has to be stored in direct vicinity of the line.

*Logistics scheduling:* Consider a typical one-warehouse-multiple-retailer setting, where given and time-varying retailer demands are to be satisfied by successive point-to-point deliveries of a single truck (server). Each trip from the warehouse to a retailer including the necessary tour back to the warehouse builds a job supplying a specific amount of products. Clearly, in such a setting the assumptions of deterministic demands and a JIT-objective seem especially suited in a build-to-order supply chain. Another example is a cross docking terminal, where the outbound side operates according to a fixed-schedule policy like it is typically applied in Less-than-Truck-Load (LTL) shipping (see Boysen and Fliedner, 2010). Here, a set of inbound trucks that are loaded with different products needs to be processed at a given number of receiving doors. Moreover, outbound trucks need to be loaded with a given number of different products at predetermined departure times. Consequently, by applying the described problem setting, the inbound trucks are mapped as the jobs, while the outbound trucks define the time-varying demand events. The single server corresponds to a single receiving door at which inbound trucks are unloaded. By fulfilling given demands late deliveries to final customers are avoided and the minimization of resulting inventory in the cross dock reduces congestions of forklifts inside the terminal.

In what follows, we formulate the decision problem as a single-machine scheduling problem where processed jobs release a predefined number of items of a corresponding product type. These items are consumed by external time-varying demand requests that are predetermined with regard to order quantity and time of consumption. By combining this basic setting with typical JIT-objective functions and additional assumptions on model restrictions, we obtain a

set of different subproblems whose complexity status is analyzed in the following. The remainder of the paper is organized as follows. In Section 2 the problem is more formally introduced. Specifically, subproblems are defined and related research is summarized. Subsequently, different problem settings are investigated in separate sections with regard to their computational complexity. For more restricted problem versions polynomial time algorithms are proposed and the resulting running time is derived. For the most general problem setting it is proven that even finding feasible solutions is strongly NP-hard. Finally, Section 7 concludes the paper with a brief summary and a description of future research fields.

## 2 Problem specification and related research

We consider a set  $P$  of products (or product types) and a set  $J$  of jobs with  $|P| = n_P$  and  $|J| = n_J$ , respectively. Processing a job  $j \in J$  requires time  $p_j$  and, with its completion, releases  $a_j$  units of product type  $\delta_j \in P$  to the corresponding inventory. Over a given planning horizon, product inventory is consumed by a set  $R$  of  $|R| = n_R$  external demand requests. The parameter  $d_r$  denotes the number of units of product type  $\rho_r \in P$  which needs to be at least in stock at time  $\tau_r$  when demand request  $r \in R$  is due. Since each job and demand request is associated with a product (type)  $p$ , sets  $J$  and  $R$  can be decomposed into  $|P|$  disjoint subsets  $J_p = \{j \in J : \delta_j = p\}$  and  $R_p = \{r \in R : \rho_r = p\}$ , respectively. We assume that inventory is released not before a job is completely processed, whereas any subsequent job can be processed as soon as the prior job finishes. We further assume that the machine constitutes the unique bottleneck in the upstream supply process of products and therefore enforces a no-wait policy (see notation of Graham et al., 1979), so that jobs need to be processed without delay. Note that while this problem formulation does not explicitly consider return times, it can nonetheless be used to model a point-to-point delivery system (see Section 1), since return times are usually constant for all jobs with the same product type/destination. Varying return times per product type can be readily considered by adding the return time to the processing time of all jobs of the product type and delaying its demand requests accordingly. Furthermore, safety stocks and initial inventory levels can be considered by modifying the demand parameters of demand events.

In what follows, a schedule is unambiguously determined by a single job sequence  $\sigma$ , where  $\sigma(l)$  defines the job at position  $l$  ( $1 \leq l \leq n_J$ ) while the derived variable  $C_{\sigma(l)} = \sum_{k=1}^l p_{\sigma(k)}$ ,  $\forall l = 1, \dots, n_J$  denotes its completion time. Based on these definitions, the total supply  $A_p(t)$  and the total demand  $D_p(t)$  of units of product  $p$  until period  $t$  can be calculated as follows:

$$A_p(t) = \sum_{j \in J_p | C_j \leq t} a_j \quad \forall p \in P; t \geq 0 \quad (1)$$

$$D_p(t) = \sum_{r \in R_p | \tau_r \leq t} d_r \quad \forall p \in P; t \geq 0 \quad (2)$$

Based on these values, the inventory  $I_p(t)$  for product  $p$  at any point in time  $t \geq 0$  is given by  $I_p(t) = A_p(t) - D_p(t)$ . Clearly, these values solely depend on the determined sequence of jobs  $\sigma$ . Without loss of generality, we impose that  $\sum_{j \in J_p} a_j = \sum_{r \in R_p} d_r$ ,  $\forall p \in P$ . Hence, it can be concluded that there exists a period  $T$  for which it holds that  $I_p(t) = 0$ ,  $\forall t > T$ . In what follows, we refer to this period as the end of the planning horizon. We further assume that all demand requests are to be satisfied, i.e.,  $I_p(t) \geq 0$ ,  $\forall p \in P; t \geq 0$ . Therefore, we conclude that  $T = \max_{r \in R} \{\tau_r\}$ .

Since any excess inventory is considered to be waste according to the JIT-philosophy and thus to be reduced to a minimum, we pursue the following two objective functions. Specifically, we consider minimizing the weighted sum of inventories  $\sum I = \sum_{p \in P} w_p \cdot \sum_{t=1}^T I_p(t)$  as well as the minimization of the weighted maximum inventory  $I^{\max} = \max_{p \in P; t=1, \dots, T} \{w_p \cdot I_p(t)\}$ . While the total inventory addresses the overall efficiency of a found schedule, the maximum inventory objectives also considers its applicability. Since in many assembly systems available space in direct vicinity of the working places is strongly limited, maximum inventory objectives may ensure feasibility.

In order to determine optimal schedules according to the introduced objectives, it is sufficient to focus on those periods in which the inventory changes, i.e., periods of set  $\Theta = \{C_j : j \in J\} \cup \{\tau_r | r \in R\}$ . Clearly, since the objective function values and all restrictions can be checked in polynomial time, the decision versions of both scheduling problems are in  $\mathcal{NP}$ .

In order to simplify the following complexity analysis, we introduce the parameter  $n$  denoting the input length of an instance of our problem. Hence, it holds that  $n_J + n_R + n_P \in O(n)$ . Moreover, we assume that demand requests are sorted in ascending order according to their demand periods  $\tau_r$ . Consequently, parameter values  $A_p(t)$ ,  $D_p(t)$  and thus  $I_p(t)$  with  $p \in P$  and  $t \in \Theta$  can be iteratively determined in linear time  $O(n)$ .

In what follows, we analyze several subproblems by assuming individual processing times (i.e., individual  $p_j$ -values) as well as the case of uniform processing times (i.e.,  $p_j = p$ ,  $\forall j \in J$ ). Furthermore, we assume that each job releases an individual number of products (i.e., individual  $c_j$ -values) as well as the case that all jobs that deliver the same product  $p$  release an identical number of units (i.e.,  $c_j = c_p$ ,  $\forall p \in P; j \in J_p$ ).

Thus, by a full-factorial combination of these parameter settings and the two objective functions, we obtain altogether eight subproblems. We specify these subproblems by extending the classic Graham-notation:

- $[1|no - wait; p_j = p; c_p | I^{\max}]$  and  $[1|no - wait; p_j = p; c_p | \sum I]$
- $[1|no - wait; c_p | I^{\max}]$  and  $[1|no - wait; c_p | \sum I]$
- $[1|no - wait; p_j = p; c_j | I^{\max}]$  and  $[1|no - wait; p_j = p; c_j | \sum I]$
- $[1|no - wait; c_j | I^{\max}]$  and  $[1|no - wait; c_j | \sum I]$

In what follows, we provide upper and lower bounds on the computational complexity for all these subproblems.

To the best of the authors knowledge, existing research on scheduling with inventory releasing jobs does not consider the case of external demand. Former studies assume that demand is caused by (a subset of) jobs to be scheduled (together with inventory releasing jobs) either on a single machine (e.g., see Monma, 1980; Kellerer et al., 1998; Briskorn et al., 2010) or in flow shop settings (e.g., see Kim and Park, 2000; Bülbül et al., 2004). However, the majority of these research works pursues time oriented objective functions and, therefore, neglect resulting excess inventory. Similar problems are investigated in the field of truck scheduling at a cross docking terminal. Here, inbound trucks build up inventory consumed by outbound trucks (jobs) to be processed at dock doors (Boysen and Fliedner, 2010). Recently, Boysen and Bock (2011) presented a related albeit more complex real-world scheduling problem, where a mixed-model assembly line is to be supplied with parts via forklifts. Their work is motivated by supply processes conducted at a warehouse of a major German car manufacturer.

### 3 Equal processing times and equal number of delivered products

In what follows, we consider the simplified problem version, where all jobs have uniform processing time, i.e., it holds that  $p_j = p, \forall j \in J$ . Moreover, we assume an equal number of delivered items of each product:  $[1|no - wait; p_j = p; c_p|\gamma]$ . We shall begin our analysis of this problem with its feasibility variant.

#### 3.1 Determining a feasible schedule

Since all jobs have the uniform processing time  $\pi$ , we can conclude that for any feasible solution sequence  $\sigma$  all job completion times are out of set  $\Theta = \{C_j : j \in J\} = \{k \cdot \pi : k = 1, \dots, n_J\}$ . Consequently, all demand requests that occur in period  $t$ , with  $k \cdot \pi < t < (k + 1) \cdot \pi$  have to be satisfied at period  $k \cdot \pi$  or the problem instance is infeasible. As further all jobs in the set  $J_p$  contribute the same amount of inventory, any two jobs of the same set can interchange positions in the solution sequence without affecting the objective function value. On the basis of these insights, we can deduce due dates on the completion of each  $j \in J$  in the following manner:

Consider each product  $p \in P$  successively. Let  $|J_p| = n_p$  and  $i = 1, \dots, n_p$  be an index referring to the  $i$ -th job of an arbitrary ordering of set  $J_p$ . The due date  $dd_{p,i}$  for the job with index  $i$  of set  $J_p$  is hence:

$$dd_{p,i} = \max_{k=1, \dots, n_J} \{k \cdot \pi | (i - 1) \cdot c_p \geq D_p((k \cdot \pi) - 1)\}, \quad \forall p \in P; \forall i = 1, \dots, n_p \quad (3)$$

A scheduling of the  $i$ -th job that delivers product  $p$  and is completed at time  $t = k \cdot \pi$  is feasible, if the following two conditions are met: First, no material shortage in any prior period occurs.

Second, there is enough inventory in the current period  $k \cdot \pi$ . Note that this determination of due dates presupposes that the jobs of each set  $J_p$  are scheduled in an order preserving way, i.e., in sequence of their numbering in  $J_p$ . Since in any solution two jobs of the same set  $J_p$  can always swap positions without affecting the objective value, such an ordering can be readily generated for any sequence  $\sigma$ . Furthermore, due dates of Equation 3 can efficiently be generated in linear time, provided that demand events are ordered with regard to their date of occurrence by Algorithm 1.

---

**Algorithm 1** Determination of due dates

---

**Input:** an instance of  $[1|no - wait; p_j = p; c_p|\gamma]$

**Output:** returns due dates  $dd_{p,i}$  for all  $p \in P$  and  $i \in J_p$

```

Initialize parameters  $D_p \leftarrow 0, i_p \leftarrow 0 \quad \forall p \in P$ 
for  $r \in R$  do
     $D_{\rho_r} \leftarrow D_{\rho_r} + d_{\rho_r}$ 
    while  $D_{\rho_r} > i_{\rho_r} \cdot c_{\rho_r}$  do
         $i_{\rho_r} \leftarrow i_{\rho_r} + 1$ 
         $dd_{\rho_r, i_{\rho_r}} \leftarrow \lfloor \frac{\tau_r}{\pi} \rfloor \cdot \pi$ 
    end while
end for
Return  $dd$ 

```

---

Algorithm 1 iteratively inspects demand events and jobs, thereby updating cumulated demand  $D_p$  and supply  $i_p \cdot c_p$  per product  $p$ . Since each demand event and job is visited only once, the running time results to  $O(n_J + n_R) = O(n)$ .

After computing all due dates, we obtain a scheduling problem with integer due dates where all jobs have the uniform processing time  $\pi$ . Moreover, since all due dates are also multiples of  $\pi$ , this problem can be transformed to a unit task time scheduling problem with integer due dates, which can be solved by the algorithm of Frederickson (1983) in time  $O(n)$ . Thus, we obtain a solution procedure that decides the feasibility problem of  $[1|no - wait; p_j = p; c_p|\gamma]$  in time  $O(n)$ .

### 3.2 Minimizing the sum of inventory

Building up on the insights of the prior section, we shall show how the optimization problem  $[1|no - wait; p_j = p; c_p|\sum I]$  is solved to optimality in time  $O(n^3)$ . Since each job can be scheduled at  $n_J$  distinct periods ( $|\Theta| = n_J$ ), we can compute the contribution to the objective function value for scheduling any job at a period  $k \cdot \pi, \forall k = 1, \dots, n_J$  in time  $O(n^3)$ . Let  $i = 1, \dots, n_p$  denote an arbitrary ordering of set  $J_p$ , then the contribution  $I_k^{p_i}$  of assigning the  $i$ -th job of set  $J_p$  to position  $k$  results to:

$$I_k^{pi} = \sum_{t=k \cdot \pi}^T \min \{c_p; \max \{0; i \cdot c_p - D_p(t)\}\}, \forall p \in P; \forall i = 1, \dots, n_p; \forall k = 1, \dots, n_J |_{k \cdot \pi \leq dd_{p,i}}. \quad (4)$$

After processing the  $i$ -th job of type  $p$  a total number of  $i \cdot c_p$  units have been released and cumulative demand of  $D_p(t)$  is consumed up to period  $t$ . Consequently, the difference of both values provides us with the current inventory level at  $t$ . However, a single job occurrence cannot contribute more than  $c_p$  units and less than zero inventory per period. This is ensured by nested min and max-functions. Furthermore, feasibility is ensured by setting  $I_k^{pi} = M$ ,  $\forall p \in P; \forall i = 1, \dots, n_p; \forall k = 1, \dots, n_J |_{k \cdot \pi > dd_{p,i}}$ , since  $M$  defines a sufficiently large number that prohibits a late job processing.

Clearly, based on these equations, for each job occurrence the one feasible period, which causes minimum inventory, can be easily identified. However, since multiple products may compete for identical periods, a coordination problem arises. It can be solved by an assignment problem. Specifically, in this problem job occurrences are to be assigned to periods, where contribution  $I_k^{pi}$  provides the respective assignment costs. Note that, due to the definition of the assignment costs, each optimal solution keeps the sequence of jobs in each set  $J_p$ . The assignment problem can be solved in  $O(n^3)$ , e.g. by the Hungarian method. Therefore,  $[1|no - wait; p_j = p; c_p | \sum I]$  can be solved to optimality in time  $O(n^3)$ .

*Example:* Consider the data given in Table 1 and  $\pi = 1$ . The resulting assignment graph is depicted in Figure 1, where infeasible arcs are left out. The optimal assignment (depicted by bold faced arcs) amounts to an objective value of  $Z = 10$ .

$p$	$D_p(1)$	$D_p(2)$	$D_p(3)$	$D_p(4)$	$D_p(5)$	$n_p$	$c_p$
1	3	4	5	5	10	2	5
2	0	2	3	6	9	3	3

Table 1: Example data

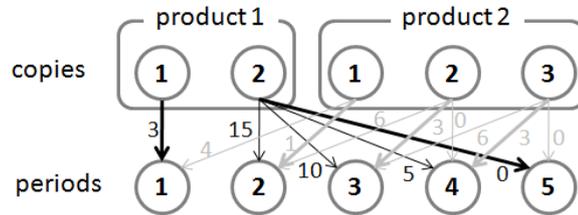


Figure 1: Assignment graph for the given example

### 3.3 Minimizing maximum inventory

In this subsection we consider the problem variant [1|*no-wait*;  $p_j = p$ ;  $c_p | I^{\max}$ ]. If the maximum inventory has to be minimized, an assignment problem cannot be applied anymore, since costs of each assignment also depend on the other locations. Therefore, we decompose the problem into a set of feasibility problems where a maximum inventory level  $\bar{I}$  has to be obeyed. Since inventory levels can only be maximal in a period in which a job releases inventory and are otherwise only reduced by demand requests, it is sufficient to focus on the  $n_J$  delivery periods. For any given  $\bar{I}$  the set  $\Gamma_{pi}$  of feasible periods for an assignment of the  $i$ -th job of product  $p$  is determined by:

$$\Gamma_{pi} = \{k = i, \dots, n_J \mid (i-1)c_p \geq D_p((k \cdot \pi) - 1) \wedge i \cdot c_p - D_p(k \cdot \pi) \leq \bar{I}\}, \quad (5)$$

$$\forall p \in P; \forall i = 1, \dots, n_p.$$

By imposing a maximum inventory level  $\bar{I}$ , we extend the constraints of the feasibility problem. Specifically, in addition to a due date that results from the predetermined demand requests, each job obtains a release date. By enforcing a later processing of jobs, this release date ensures that the prescribed maximum inventory level is kept. We can efficiently determine and store the set of feasible periods for each job  $i$  of set  $J_p$  by a pair of release dates  $rd_{p,i}$  and due dates  $dd_{p,i}$ . Given a maximum inventory level, release dates can be readily calculated in linear time  $O(n)$  by Algorithm 2.

---

**Algorithm 2** Determination of release dates

---

**Input:** an instance of [1|*no-wait*;  $p_j = p$ ;  $c_p | I^{\max}$ ] and a maximum inventory level  $\bar{I}$

**Output:** returns release dates  $rd_{pi}$  for all  $p \in P$  and  $i \in J_p$

```

Initialize parameters  $D_p \leftarrow 0, i_p \leftarrow 0 \quad \forall p \in P$ 
for  $p \in P$  do
  while  $(i_p + 1) \cdot c_p \leq \bar{I}$  do
     $i_p \leftarrow i_p + 1$ 
     $rd_{p,i_p} \leftarrow 0$ 
  end while
end for
for  $r \in R$  do
   $D_{\rho_r} \leftarrow D_{\rho_r} + d_{\rho_r}$ 
  while  $(i_{\rho_r} + 1) \cdot c_{\rho_r} - D_{\rho_r} \leq \bar{I}$  do
     $i_{\rho_r} \leftarrow i_{\rho_r} + 1$ 
     $rd_{\rho_r, i_{\rho_r}} \leftarrow (\lceil \frac{\tau_r}{\pi} \rceil - 1) \cdot \pi$ 
  end while
end for
Return  $rd$ 

```

---

The transformation into the corresponding unit scheduling problem is again conducted by iteratively considering demand requests and successively updating demand  $D_p$  and supply  $(i_p + 1) \cdot c_p$ . If the difference is not higher than the maximum inventory level the release date is fixed

to largest multiple of  $\pi$  just below the due date of the demand event. If the maximum inventory is instead violated, the release date of the respective job has to be set to a later point in time and the next demand event is selected. Due dates can be calculated by using Algorithm 1. Since each demand request and each job is considered only once and all update steps are performed in constant time, we obtain a total running time of  $O(n_J + n_R) = O(n)$ . Moreover, all release dates, due dates and job processing times are multiples of  $\pi$ , so that the resulting problem can be again expressed by a unit task time scheduling problem and solved by the procedure of Frederickson (1983) in linear time  $O(n)$ .

The min-max inventory level can be identified by a binary search within an interval of inventory levels ranging from an upper bound  $UB$  to a lower bound  $LB$ . A simple upper bound is based on the simple cognition, that in a worst scenario all jobs that release a specific product are scheduled in direct succession starting with the first period. Depending on the demand requests, a maximum inventory must occur in one of these periods. Therefore, we conclude that

$$UB = \max_{p \in P; k=1, \dots, n_p} \{k \cdot c_p - D_p(k \cdot \pi)\}. \quad (6)$$

In a best scenario, however, the  $n_p$  product deliveries occur in the  $n_p$  periods with highest demand for the respective product (and current inventory is zero). Thus, it holds that

$$LB = \max\{0, \max_{p \in P} \{c_p - n_p \cdot \max_{r \in R_p} \{d_r\}\}\}, \quad (7)$$

with function  $k$ -max identifying the  $k$ -highest value.

By deriving an upper bound on the difference of these two values, we obtain  $UB - LB \in O(n \cdot c_p - 0) = O(n \cdot 2^n)$ , since  $c_p$  is a parameter of the problem instance. The search can be further sped up if only those values between  $UB$  and  $LB$  are inspected that can result from the objective function value of any given instance. Since there are  $n_R$  delivery periods at which inventory can be maximal and the number of products and deliveries per product are bounded by  $O(n)$ , the number of possible objective values is in  $O(n^2)$ . If these values are generated on the fly, the search space can be examined by binary search in  $O(\log n)$ . Since the resulting feasibility problems are solved in linear time, the time complexity of the algorithm is in  $O(n \cdot \log n)$ .

*Example (cont.):* For the data given in Table 1 a binary search is to be executed between  $LB = 2$  and  $UB = 6$ . Figure 2 provides a bipartite graph presentation for  $\bar{I} = 3$ , where an edge indicates that a job can be assigned to the respective period.  $\bar{I} = 3$  is the smallest maximum inventory level for which a perfect matching (bold faced) exists and is, thus, the optimal solution value.

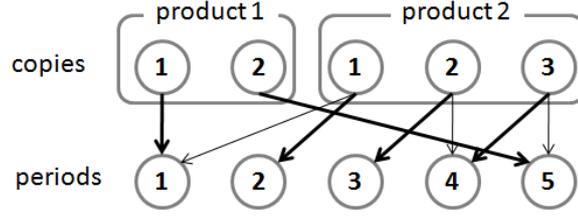


Figure 2: Bipartite graph for  $\bar{T} = 3$

## 4 Varying processing times and equal number of delivered products

In what follows, we generalize the problem setting by allowing varying processing times of the jobs to be scheduled. Again, we shall begin our analysis with the feasibility variant of the problem.

### 4.1 Determining a feasible schedule

While processing times of jobs ( $p_j$ ) are considered to be variable, the number of units that are delivered per product type remains constant, i.e., it still holds that  $c_j = c_p, \forall p \in P; j \in J_p$ . Based on these assumptions, a feasible schedule can be determined in polynomial time by applying Algorithm 3.

---

**Algorithm 3** Finding a feasible schedule for  $[1|no - wait; c_p|\gamma]$

---

**Input:** an instance of  $[1|no - wait; c_p|\gamma]$  and an empty sequence  $\sigma$  with length  $|\sigma| = 0$

**Output:** returns a feasible sequence  $\sigma$ ; if the instance is infeasible, the empty sequence  $\sigma = ()$  is returned

Order all jobs in sets  $J_p$  so that  $p_{\langle 1 \rangle} \leq p_{\langle 2 \rangle} \leq \dots \leq p_{\langle |J_p| \rangle} \forall p \in P$ .

**while**  $|\sigma| < n_J$  **do**

$r' \leftarrow \operatorname{argmin}\{\tau_r | r \in R, I_{\rho_r}(\tau_r) < 0\}$

**while**  $I_{\rho_{r'}}(\tau_{r'}) < 0$  **do**

**if**  $|J_{\rho_{r'}}| > 0$  **then**

            Remove job  $j'$  from set  $J_{\rho_{r'}}$  with smallest  $p_{j'}$  and append it to  $\sigma$

**else**

$\sigma \leftarrow ()$ ; Return  $\sigma$

**end if**

**end while**

**end while**

Return  $\sigma$

---

The correctness of this procedure can be established by the following insights: The algorithm generates a schedule where jobs of each product type are processed in sequence of non-decreasing processing times. Thus, we always attain the maximum release speed per time unit for each

product. Moreover, demand requests are satisfied according to their urgency, i.e., the algorithm follows the earliest due date rule (EDD). Hence, if there exists a feasible schedule to a given instance, Algorithm 3 generates one. Clearly, since the sorting step is most time-consuming, Algorithm 3 generates a feasible schedule (if there exists any) in time  $O(n \cdot \log n)$ .

## 4.2 Minimizing the sum of inventory

In this subsection we analyze the optimization variant of the problem that pursues the minimization of the sum of inventory ( $\gamma = \sum I$ ). Specifically, we can derive the following proposition.

**Proposition:** Problem  $[1|no - wait; c_p|\sum I]$  is strongly NP-hard for  $|P| \geq 1$ .

**Proof:** The proof follows from a pseudo-polynomial reduction of 3-Partition. Since 3-partition is known to be NP-hard in the strong sense (see Garey and Johnson, 1979), we have shown that problem  $[1|no - wait; c_p|\sum I]$  is also strongly NP-hard.

Therefore, we consider an instance  $\Phi$  of 3-Partition that is defined by an integer  $B \in \mathbb{Z}^+$  and a set  $A$  with  $|A| = 3q$ . The set  $A$  contains integers  $B/4 < a_j < B/2, \forall j = 1, \dots, 3q$ . An instance asks whether there exists a partition of the set  $A$  into  $q$  disjoint subsets  $\{A_1, A_2, \dots, A_q\}$  such that  $\sum_{i \in A_j} = B, \forall j = 1, \dots, q$ .

We construct a corresponding instance  $\Psi$  to the problem variant  $[1|no - wait; c_p|\sum I]$ . For this purpose, we introduce natural numbers  $K \geq B + 1$  and  $M \geq q \cdot B + 1$ . The instance  $\Psi$  consists of two sets of jobs. The first set comprises  $3q$  slow jobs with indices  $j = 1, \dots, 3q$ , which are characterized by  $\delta_j = 1 \wedge c_j = 1 \wedge p_j = a_j + K$ . Additionally, we have  $Mq$  fast jobs with indices  $j = 3q + 1, \dots, 3q + Mq$  in the second set that possess the parameter setting  $\delta_j = 1 \wedge c_j = 1 \wedge p_j = 1$ . Finally, there are two groups of demand requests. The first group comprises  $q$  requests with indices  $l = 1, \dots, q$  and demand  $d_l = 3$ . These requests, which we denote as 3-demands, occur in the periods  $\tau_l = (B + 3K) \cdot l + (l - 1) \cdot M, \forall l = 1, \dots, q$ . Moreover, the second group comprises  $Mq$  additional 1-requests. These requests with indices  $l = q + 1, \dots, q \cdot (M + 1)$  and demand  $d_l = 1, \forall l = q + 1, \dots, q \cdot (M + 1)$  occur in the periods  $\tau_{q+(l-1) \cdot M+k} = (B + 3K) \cdot l + (l - 1) \cdot M + k, \forall l = 1, \dots, q, \forall k = 1, \dots, M$ .

We ask whether there exists a solution for  $\Psi$  with objective function value  $Z \leq q \cdot (B + 3K)$ .

First of all, it can be stated that in a no-wait schedule 1-demands have to be satisfied solely by fast jobs. This results from the fast iteration of the 1-demands and the significant duration of the slow jobs ( $p_j = a_j + K > B + 1$ ). Consequently, in each solution there are always three slow jobs in advance of each 3-demand job and no slow job during the periods  $(B + 3K) \cdot l + (l - 1) \cdot M + 1, \dots, (B + 3K + M) \cdot l$ . This structure is depicted in Figure 3.

In what follows, we show the equivalence of both problems. We assume there is a solution to 3-Partition. Then, we generate a solution to Problem  $[1|no - wait; c_p|\sum I]$  by scheduling the slow jobs that correspond to each of the  $q$  subsets of the solution to 3-Partition just prior to a 3-demand. In between, we only process fast jobs. Therefore, due to the fast repetition of

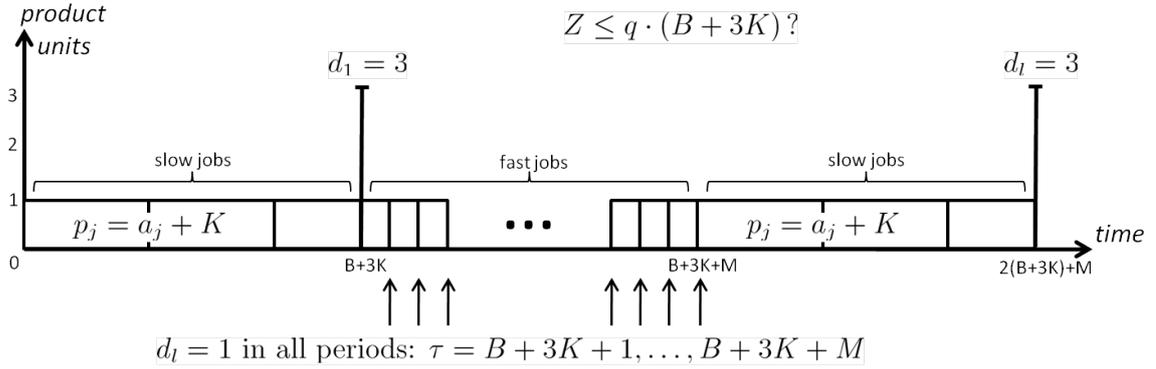


Figure 3: Schematic schedule structure for  $[1|no-wait; c_p|\sum I]$

demand requests, inventory exclusively occurs in advance of each 3-demand. We focus on these subsequences. Prior to any 3-demand  $j$ , we obtain a job-sequence  $3 \cdot (j-1) + 1$ ,  $3 \cdot (j-1) + 2$ , and  $3 \cdot (j-1) + 3$  with processing times  $a_{3 \cdot (j-1) + 1} + K$ ,  $a_{3 \cdot (j-1) + 2} + K$ , and  $a_{3 \cdot (j-1) + 3} + K$ . Since in the resulting schedule the cost contributions are  $C_j = a_{3 \cdot (j-1) + 2} + 2 \cdot a_{3 \cdot (j-1) + 3} + 3 \cdot K$ , we always process jobs of each 3-pair according to non decreasing processing time, i.e., it holds that  $a_{3 \cdot (j-1) + 1} \geq a_{3 \cdot (j-1) + 2} \geq a_{3 \cdot (j-1) + 3}$ . Hence, in a worst case scenario, all jobs are of equal duration, so that a maximum cost contribution of  $B/3 + 2 \cdot B/3 + 3 \cdot K = B + 3 \cdot K$  is obtained per 3-demand. In total, we have  $q$  3-demands, so that  $Z \leq q \cdot (B + 3K)$  is fulfilled.

On the other hand, we assume a solution of Problem  $[1|no-wait; c_p|\sum I]$  with an objective function value  $Z \leq q \cdot (B + 3K)$ . We consider three elements  $3 \cdot (j-1) + 1$ ,  $3 \cdot (j-1) + 2$ , and  $3 \cdot (j-1) + 3$  prior to any 3-demand  $j$ . Therefore, these elements have processing times  $a_{3 \cdot (j-1) + 1} + K$ ,  $a_{3 \cdot (j-1) + 2} + K$ , and  $a_{3 \cdot (j-1) + 3} + K$ . Clearly, if it holds that  $a_{3 \cdot (j-1) + 1} + a_{3 \cdot (j-1) + 2} + a_{3 \cdot (j-1) + 3} = B + 3K$  we directly obtain a solution to 3-Partition. Otherwise, there exists at least one 3-demand  $j$  with  $a_{3 \cdot (j-1) + 1} + a_{3 \cdot (j-1) + 2} + a_{3 \cdot (j-1) + 3} + 3 \cdot K < B + 3K$ . In this case, at least one fast jobs starts earlier and we have an increased inventory level by (at least) one unit for all subsequent  $M$  fast jobs. Consequently, we obtain an increase of the objective function value by  $M$ . In order to minimize inventory, the slow jobs are scheduled in sequence of non-increasing processing times, i.e.,  $a_{3 \cdot (j-1) + 1} \geq a_{3 \cdot (j-1) + 2} \geq a_{3 \cdot (j-1) + 3}$ . Hence, in a best case scenario the second two jobs have minimal duration which leads to a lower bound of  $B/4 + K + 2 \cdot (B/4 + K) = 3B/4 + 3K$  per 3-demand  $j$ . Thus, we obtain a total lower bound of  $q \cdot (3B/4 + 3K) + M > q \cdot (3B/4 + 3K) + q \cdot B > q \cdot (B + 3K)$ . Since we know that  $Z \leq q \cdot (B + 3K)$ , such a 3-demand  $j$  does not exist. Consequently, all slow job prior to a 3-demand exactly sum up to a duration of  $B + 3K$  and the proposition holds.  $\square$

### 4.3 Minimizing maximum inventory

In what follows, we pursue the minimizing of the maximum inventory (i.e.,  $\gamma = I^{\max}$ ). For this problem variant we prove the following complexity result:

**Proposition:** Problem  $[1|no - wait; c_p|I^{\max}]$  is strongly NP-hard even with a single product.

**Proof:** The proof is provided by a polynomial reduction from 3-Partition. Consider an instance  $\Phi$  of 3-Partition as defined above. The corresponding instance  $\Psi$  of our scheduling problem consists of  $3q$  jobs and is constructed as follows:  $\delta_j = 1 \wedge c_j = 1 \wedge p_j = a_j, \forall j = 1, \dots, 3q$ . Additionally there are  $|R| = q$  demand requests with  $d_r = 3, \forall r \in R$  that occur at time points  $\tau_k = k \cdot B, \forall k = 1, \dots, q$ .

We ask whether there is a solution for  $\Psi$  with  $Z \leq 2$ .

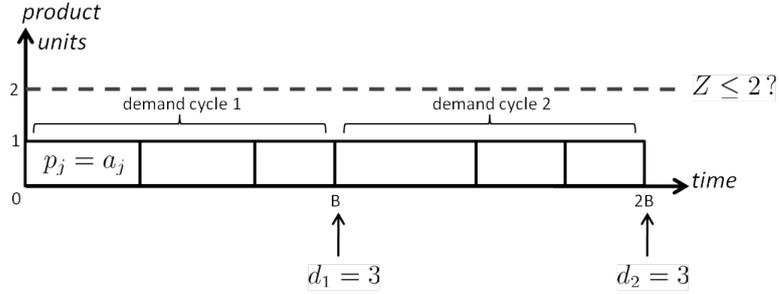


Figure 4: Schematic schedule structure for  $[1|no - wait; c_p|I^{\max}]$

We refer to a demand request and the three previously scheduled jobs satisfying this demand as a demand cycle. Clearly, any solution of 3-Partition can be easily transformed into a solution of our scheduling problem by scheduling the corresponding jobs of any set having three elements in a demand cycle (scheduled in an arbitrary sequence). Since in each demand cycle the final demand request and the third job completion occur at the same point in time, maximum inventory level is 2.

The proof of the opposite direction is based on a simple induction argument. We consider an optimal schedule of  $\Psi$  whose structure is depicted in Figure 4. Obviously, in the first demand cycle exactly three jobs are required in order to satisfy the subsequent demand request. This results from the specific restrictions of integer values in 3-Partition. Furthermore, the sum of processing times of these three jobs must be equal to  $B$ . Otherwise, either demand cannot be satisfied or  $Z \leq 2$  cannot hold. This, however, induces equal prerequisites for the second demand cycle. Consequently, we can apply this argumentation iteratively until the final demand cycle is reached. This completes the proof.  $\square$

## 5 Equal processing times and varying number of delivered product units

In this section, equal processing times of jobs ( $p_j = p, \forall j \in J$ ) are considered whereas the number of units delivered per product type ( $c_j$ ) may vary.

### 5.1 Determining a feasible schedule

Again, we shall begin our analysis with the feasibility variant of the problem. Algorithm 4 defines how to determine a feasible schedule in polynomial time ( $O(n \log n)$ ).

---

**Algorithm 4** Finding a feasible schedule for  $[1|no - wait; p_j = p; c_j|\gamma]$

---

**Input:** an instance of  $[1|no - wait; p_j = p; c_j|\gamma]$  and an empty sequence  $\sigma$  with length  $|\sigma| = 0$

**Output:** returns a feasible sequence  $\sigma$ ; if the instance is infeasible the empty sequence  $\sigma = ()$  is returned

Order jobs in sets  $J_p$  so that  $c_{\langle 1 \rangle} \geq c_{\langle 2 \rangle} \geq \dots \geq c_{\langle |J_p| \rangle} \forall p \in P$ .

**while**  $|\sigma| < n_J$  **do**

$r' \leftarrow \operatorname{argmin}\{\tau_r | r \in R, I_{\rho_r}(\tau_r) < 0\}$

**while**  $I_{\rho_{r'}}(\tau_{r'}) < 0$  **do**

**if**  $|J_{\rho_{r'}}| > 0$  **then**

            Remove job  $j'$  from set  $J_{\rho_{r'}}$  with largest  $c_{j'}$  and append it to  $\sigma$

**else**

$\sigma \leftarrow ()$ ; Return  $\sigma$

**end if**

**end while**

**end while**

Return  $\sigma$

---

The correctness of the Algorithm 4 follows analogously to the correctness of Algorithm 3. Again, we schedule all jobs of each product in a sequence that maximizes the number of items that are released per time unit. Consequently, by scheduling the products according to the EDD-rule, it can be concluded that if the given instance is solvable the procedure provides a feasible solution.

### 5.2 Minimizing the sum of inventory

In this subsection we consider the optimization variant of problem  $[1|no - wait; p_j = p; c_j|\gamma]$  with  $\gamma = \sum I$ . Again, we prove that this problem is NP-hard in the strong sense.

**Proposition:** Problem  $[1|no - wait; p_j = p; c_j|\sum I]$  is strongly NP-hard even if  $|P| = 1$ .

**Proof:** We provide a pseudo-polynomial reduction of 3-Partition. Therefore, we shall transform an instance  $\Phi$  of this NP-hard problem. The corresponding instance  $\Psi$  of  $[1|no - wait; p_j =$

$p; c_j | \sum I$ ] comprises two types of jobs: Type one jobs with indices  $j = 1, \dots, 3q$  correspond to the integer values of 3-Partition and are defined as follows:  $\delta_j = 1 \wedge c_j = a_j \wedge p_j = 1$ . Additionally, jobs of the second type with indices  $j = 3q + 1, \dots, 3q + q \cdot (qB + 1)$  release a considerably higher number of product units. Specifically, for these jobs it holds that  $\delta_j = 1 \wedge c_j = qB + 1 \wedge p_j = 1$ .

Finally, demand requests are defined as follows: The first  $q$  requests are denoted as small demands. They have the indices  $l = 1, \dots, q$  and a demand of  $d_l = B$ . These small demands occur in the periods  $\tau_l = (qB + 1) \cdot (l - 1) + 3 \cdot l, \forall l = 1, \dots, q$ . The second group of demand requests are denoted as large demands and have the indices  $l = q + 1, \dots, q + q \cdot (qB + 1)$ . These large demand requests occur in the periods  $\tau_{q+l \cdot (qB+1)+k} = (qB + 1) \cdot (l - 1) + 3 \cdot l + k, \forall l = 1, \dots, q, \forall k = 1, \dots, qB + 1$  and have a demand of  $d_l = qB + 1, \forall l = q + 1, \dots, q + q \cdot (qB + 1)$ . We ask whether there is a solution for  $\Psi$  with  $Z \leq qB$ .

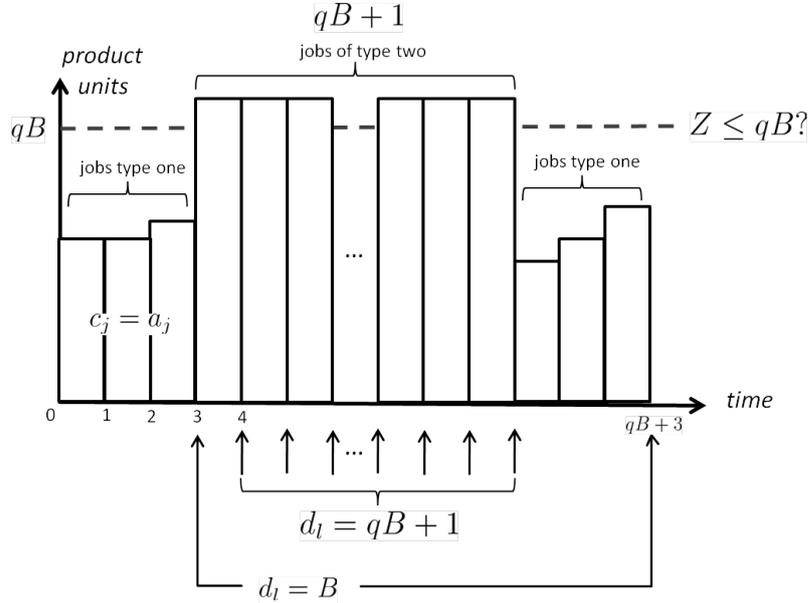


Figure 5: Schematic schedule structure for  $[1|no-wait; p_j = p; c_j | \sum I]$

First, we assume that there is a solution of 3-Partition. According to this solution, for each set of three elements with a total value of  $B$  the corresponding type one jobs are processed prior to a small demand. In order to obtain a minimal contribution to the objective function value, we schedule these jobs in sequence on non-decreasing  $c$ -values (i.e., released units). Moreover, type two jobs satisfy the large demands. Clearly, inventory occurs only prior to the small demand requests. Specifically, we obtain a maximum contribution if all type one jobs release an identical number of items. Consequently, we derive the upper bound  $2 \cdot B/3 + B/3 = B$ . Therefore, our solution has an objective function value  $Z \leq q \cdot B$ .

We consider the opposite direction and assume that there is a solution to  $[1|no-wait; p_j = p; c_j | \sum I]$  with  $Z \leq q \cdot B$ . We consider the structure of this solution. Because of the no-wait

property, any small demand request is to be supplied by exactly three type one jobs. These jobs are scheduled in the periods directly preceding the respective demand request. We denote this setting that occurs prior to each small demand request as a demand cycle. Clearly, a demand cycle that does not exactly deliver  $B$  product units leads to a scenario where at least one product unit is taken over from a previous demand cycle or is taken to the subsequent demand cycle. In both cases, however, at least one unit is kept in stock for at least  $qB + 1$  time units. Since we know that  $Z \leq q \cdot B$ , these scenarios are not possible in  $\Psi$ .

Consequently, since all demands have to be satisfied, in each demand cycle the small demand request is satisfied by three type one jobs that deliver exactly  $B$  product units. Clearly, such a solution for  $\Psi$  can simply be transformed into a solution to 3-Partition. This is done by unifying the integer values represented by the type one jobs in a demand cycle into a 3-pair.  $\square$

### 5.3 Minimizing maximum inventory

In this subsection we consider the optimization variant  $[1|no - wait; p_j = p; c_j|I^{max}]$ . For this variant we derive the following proposition.

**Proposition:** Problem  $[1|no - wait; p_j = p; c_j|I^{max}]$  is strongly NP-hard even for  $|P| = 1$ .

**Proof:** We conduct the proof again by a polynomial reduction of 3-Partition.

Therefore, consider an instance  $\Phi$  of 3-Partition as defined above. The corresponding instance  $\Psi$  of our scheduling problem comprises two job sets. The jobs of the first set with job indices  $j = 1, \dots, 3q$  correspond to the integer values of 3-Partition and are defined as follows:  $\delta_j = 1 \wedge c_j = a_j \wedge p_j = 1, \forall j = 1, \dots, 3q$ . The jobs of the second set have indices  $j = 3q + 1, \dots, 4q$  and release a considerably larger number of product units. Specifically, it holds that  $\delta_j = 1 \wedge c_j = B + 1 \wedge p_j = 1, \forall j = 3q + 1, \dots, 4q$ .

Moreover, we define  $q$  demand requests with indices  $l = 1, \dots, q$ . Specifically, we define  $d_l = 2B + 1, \forall l = 1, \dots, q$  and  $\tau_l = 4 \cdot l, \forall l = 1, \dots, q$ . We ask whether there is a solution for  $\Psi$ , with  $Z \leq B$ .

We shall begin with a feasible solution to 3-Partition. We schedule the jobs of type one in groups of three jobs as defined in the solution to 3-Partition. Subsequently, prior to every demand request, a job of type two is scheduled. Hence, the three jobs of type one together release  $B$  units. Since the completion of the job of type two coincides with the demand request, the maximum inventory level is  $B$ .

We consider the opposite direction. Hence, we shall begin with a solution of our scheduling problem where it holds  $Z \leq B$ . Clearly, any type two job leads to a maximum inventory of  $Z = B + 1$  if it is not scheduled in a demand period. Consequently, since we assume that  $Z \leq B$ , type two jobs are scheduled in the periods  $t = 4 \cdot l - 1, \forall l = 1, \dots, q$ . Moreover, due to the no-wait property of the schedule, each time span between type two jobs is to be filled with three type one jobs. We denote this total sequence of four jobs as a demand cycle. The delivery of less

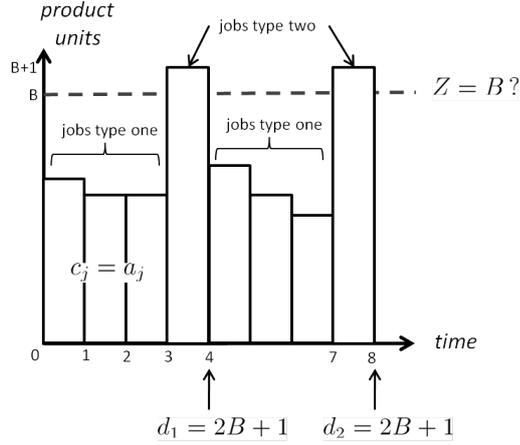


Figure 6: Schematic schedule structure for  $[1|no - wait; p_j = p; c_j | I^{\max}]$

than  $B$  products by type one jobs in a demand cycle is only possible if inventory was taken over from a previous cycle. This, however, cannot apply to our schedule, since any excess unit would produce a higher maximum inventory than  $B$  in the previous demand cycle. Consequently, in any demand cycle the number of products delivered by type one jobs amounts to  $B$  units, so that the corresponding integers can be unified to a 3-pair in 3-Partition and the proposition holds. The resulting structure of the generated schedule is depicted in Figure 6.  $\square$

## 6 Varying processing times and varying number of delivered product items

Finally, we consider the most general case where processing times as well as the number of units that are delivered per job may vary. In this case, it turns out that even finding a feasible solution is complex.

**Proposition:** Finding a feasible solution for  $[1|no - wait; c_j | \gamma]$  is strongly NP-hard even for  $|P| = 1$ .

**Proof:** We again show the proposition by a polynomial reduction of 3-Partition. For this purpose, consider an instance  $\Phi$  of 3-Partition as defined above. We construct an instance  $\Psi$  by introducing jobs with indices  $j = 1, \dots, 3q$  that corresponds to the integer values of 3-Partition. Specifically, these jobs are defined by the parameter values  $\delta_j = 1 \wedge p_j = c_j = a_j, \forall j = 1, \dots, 3q$ . Therefore, the values of the 3-Partition problem determines processing times of jobs as well as the number of released items.

Furthermore, we introduce  $q$  demand requests with  $d_l = B, \forall l = 1, \dots, q$  in the periods  $\tau_l = B \cdot l, \forall l = 1, \dots, q$ .

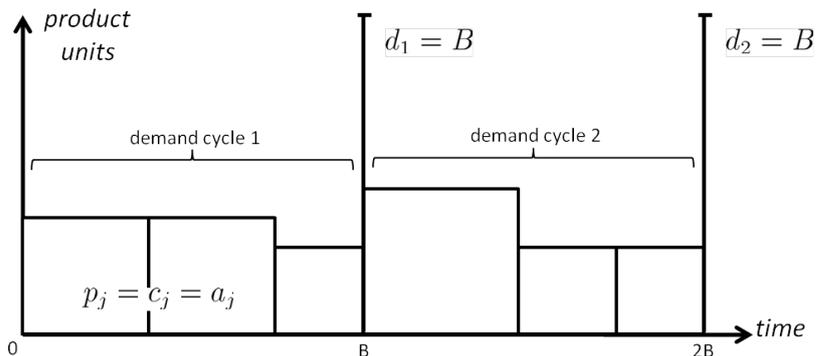


Figure 7: Schematic schedule structure for the feasibility version of  $[1|no-wait; c_j|\gamma]$

We shall begin the proof of equivalence of both problems with a feasible solution of 3-Partition. Clearly, this solution can be transformed into a feasible solution of our scheduling problem. For this purpose, the corresponding jobs of each group are scheduled prior to each demand request. Consequently, we obtain a total release of  $B$  product units after  $B$  time units. Consequently, all demands can be fulfilled in time and the generated solution is feasible.

We consider the opposite direction and shall start with a feasible solution to our scheduling problem. We consider the first demand request in this schedule and the jobs that supply the respective demand. We denote this setting as a demand cycle. We shall prove at first that exactly three jobs are required to fulfill the demand.

Due to the restrictions on the integers values  $B/4 < a_j < B/2, \forall j = 1, \dots, 3q$  of each 3-Partition instance, two jobs deliver an insufficient number of product items. Moreover, the processing of four jobs requires more than  $B$  time units. Therefore, these solutions are not feasible.

Analogously, three jobs will either deliver not enough product units if completed before  $B$  or will be late if more than  $B$  units are delivered. Consequently, in order to ensure feasibility, the first demand cycle will take exactly  $B$  time units and deliver  $B$  product units. This, however, implies that demand cycle two has the same prerequisites. By induction, we conclude that all subsequent demand cycles fulfill these requirements. Clearly, such a solution for  $\Psi$  can directly be transformed into a feasible solution for  $\Phi$ . This completes the proof.  $\square$

## 7 Conclusion

This paper considers a variety of scheduling problems that pursue an efficient supply of materials under JIT-restrictions. These problems are modeled as single-machine scheduling problems, where a set of jobs is processed, which release different products into inventory. Jobs are executed under a no-wait policy and have to meet hard demand restrictions. Specifically, external (product-specific and time-varying) demand requests occur during the planning horizon. According to the JIT-philosophy, it is the objective to satisfy the given demand requests while

minimizing resulting product inventory. This basic problem setting applies to many real-world applications in logistics and production.

By varying the objective function as well as the assumptions with regard to processing times and the number of units that are released per job, different subproblems are derived and analyzed according to their complexity. Table 2 summarizes all attained results.

subproblem	complexity	
	feasibility problem	optimization problem
$[1 no - wait; p_j = p; c_p I^{\max}]$	$O(n)$	$O(n \log n)$
$[1 no - wait; p_j = p; c_p \sum I]$	$O(n)$	$O(n^3)$
$[1 no - wait; c_p I^{\max}]$	$O(n \log n)$	strongly NP-hard
$[1 no - wait; c_p \sum I]$	$O(n \log n)$	strongly NP-hard
$[1 no - wait; p_j = p; c_j I^{\max}]$	$O(n \log n)$	strongly NP-hard
$[1 no - wait; p_j = p; c_j \sum I]$	$O(n \log n)$	strongly NP-hard
$[1 no - wait; c_j I^{\max}]$	strongly NP-hard	strongly NP-hard
$[1 no - wait; c_j \sum I]$	strongly NP-hard	strongly NP-hard

Table 2: Summary of complexity results.

While the results underline that the defined problem class possesses significant complexity, future research shall be conducted into two main directions.

First, in order to provide decision support for the problem variants that are proven to be NP-hard, new exact and heuristic solution procedures shall be generated. Second, the considered problem setting should be extended by integrating additional aspects of real-world JIT-processes. For instance, the consideration of parallel machines allows the mapping of scenarios where the supply is maintained by multiple sources.

## References

- [1] Boysen, N., Bock, S., 2011. Scheduling Just-in-Time Part Supply for Mixed-Model Assembly Lines. *European Journal of Operational Research* 211, 15–25.
- [2] Boysen, N., Fliedner, M., 2010. Cross Dock Scheduling: Classification, Literature Review and Research Agenda. *Omega* 38, 413–422.
- [3] Briskorn, D., Choi, B.-C., Lee, K., Leung, J., Pinedo, M., 2010. Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research* 207, 605–619.
- [4] Bülbül, K., Kaminsky, P., Yano, C., 2004. Flow shop scheduling with earliness, tardiness, and intermediate inventory holding cost. *Naval Research Logistics* 51, 407–445.

- [5] Frederickson, G, 1983. Scheduling Unit-Time Tasks with Integer Release Times and Deadlines. *Information Processing Letters* 16, 171–173.
- [6] Garey, M.R., Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco.
- [7] Kellerer, H., Kotov, V., Rendl, F., Woeginger, G.J., 1998. The stock size problem. *Operations Research* 46, S1–S12.
- [8] Park, M.-W., Kim, Y.-D., 2000. A branch and bound algorithm for a production scheduling problem in an assembly system under due date constraints. *European Journal of Operational Research* 123, 504–518.
- [9] Monma, C.L., 1980. Sequencing to minimize the maximum job cost. *Operations Research* 28, 942–951.