Article:
Condotta, A, Knust, S, Meier, D et al. (1 more author) (2012) Tabu search and lower bounds for a combined production-transportation problem. Computers and Operations Research, 40 (3). 886 - 900. ISSN 0305-0548

https://doi.org/10.1016/j.cor.2012.08.017

Tabu Search and Lower Bounds for a Combined Production-Transportation Problem

Alessandro Condotta[1], Sigrid Knust[2], Dimitri Meier[2], Natalia V. Shakhlevich[1]
[1] *School of Computing, University of Leeds, Leeds LS2 9JT, U.K.*
condotta@comp.leeds.ac.uk; N.Shakhlevich@leeds.ac.uk
[2] *University of Osnabrück, Institute of Computer Science, 49069 Osnabrück, Germany*
sigrid.knust@informatik.uni-osnabrueck.de

March 19, 2012

Abstract

In this paper we consider a combined production-transportation problem, where n jobs have to be processed on a single machine at a production site before they are delivered to a customer. At the production stage, for each job a release date is given; at the transportation stage, job delivery should be completed not later than a given due date. The transportation is done by m identical vehicles with limited capacity. It takes a constant time to deliver a batch of jobs to the customer. The objective is to find a feasible schedule minimizing the maximum lateness.

After formulating the considered problem as a mixed integer linear program, we propose different methods to calculate lower bounds. Then we describe a tabu search algorithm which enumerates promising partial solutions for the production stage. Each partial solution is complemented with an optimal transportation schedule (calculated in polynomial time) achieving a coordinated solution to the combined production-transportation problem. Finally, we present results of computational experiments on randomly generated data.

Key words: scheduling, transportation, batching, tabu search, lower bounds

AMS Classification: 90B35 Scheduling

1 Introduction

Traditional scheduling research deals with problems of sequencing jobs on processing machines without taking into account transportation issues. Recent trends in scheduling involve extended scheduling models where more practical constraints are included. In particular, in a typical supply chain system materials and resources are available at some release dates at the manufacturer's site; the manufacturer should process them in accordance with technological constraints; finally, finished goods should be delivered to a customer by given due dates. In the context of a supply chain, scheduling of production cannot be done in isolation from scheduling of transportation since a coordinated solution to the integrated problem may improve the performance of the whole supply chain.

The first stage of our model - *production* - deals with a single production facility and a set of n jobs, which should be processed one at a time. The jobs may have different processing

requirements in terms of the production time and may be available at different release dates. The second stage - *transportation* - deals with the delivery of n finished goods to a customer using m transportation vehicles which have the same delivery characteristics: equal transportation times from the production site to the customer and equal capacities, i.e., the maximum number of jobs which can be transported by a vehicle in one batch. The objective is to define a production and a transportation schedule so that the jobs are delivered to the customer by their due dates. In a more general setting, it is required to minimize the maximum lateness among the jobs, see Section 2 for a formal definition.

The results related to the problem under consideration are scattered among a broad range of publications on (A) *production-transportation* and (B) *generalized flow-shop models* with the second stage involving m identical batching machines.

(A): The literature on production-transportation is vast ranging in multiple parameters. With several survey papers available, e.g., [5, 8, 13, 26], we refer the reader to the most recent review by Chen [6]. The results related to our study fall in the category of "batch delivery by direct shipping" discussed in Section 5.1 of the review. Eliminating the non-relevant models with cost factors (for which it is usually assumed that the number of vehicles is sufficiently large, $m = n$, [11, 12, 13, 25]), the models with resource availability constraints and those with min-sum criteria, we review here the most closely related papers [16] and [28] with min-max criteria. Their main outcomes can be summarized as follows. If all jobs are available simultaneously, then the production-transportation problem with the makespan objective is solvable in $O(n \log n)$ time [16], while the following two generalizations are NP-hard: the version studied in [16] with two machines at the production site operating as a flow-shop and the version studied in [28] with an additional transportation stage from a supplier to the production site which precedes the introduced production-transportation model. No results are available for the production-transportation model with arbitrary release dates and due dates.

(B): In the flow-shop model with two machines, each job should be processed on the first machine and then on the second one. If the second machine operates in a batching mode (i.e. several jobs can be processed simultaneously) and the batch processing time is a constant independent of the jobs included in the batch, the corresponding flow-shop model is equivalent to our production-transportation model with one transportation vehicle. To the best of our knowledge, the flow-shop problem with a batching machine and unequal release dates and due dates was not addressed in the literature. The studied models deal with the following special cases:

- If all jobs have equal release dates and equal due dates, then the corresponding problem is solvable in $O(n \log n)$ time [1]. Notice that, as discussed in part (A), an algorithm with the same complexity is also known for the more general case with multiple batching machines (multiple transportation vehicles) [16].
- If the jobs have arbitrary release dates but equal due dates, then the corresponding problem is NP-hard in the strong sense [27] and, as shown in the same paper, it can be solved by a 2-approximation algorithm. Notice that the NP-hardness result is also proved in a later paper [22], which uses the production-transportation terminology (A); the same paper presents also a 5/3-approximation algorithm. The most recent approximation algorithm has a worst-case ratio of 3/2 [21].

Summarizing we observe that there is a lack of research addressing the production-transportation problem in its general setting when the jobs have unequal release dates and unequal

due dates. Also, as stated in the review paper [6], "the majority of the existing work has been centered on clarifying complexity of some problems, most of which are, unfortunately, NP-hard... Therefore it is worthwhile to design fast heuristics or exact branch-and-bound algorithms for such problems". In this paper, we pursue this line of research by developing a tabu search algorithm for the general case of the production-transportation problem. It is hard to perform a fair comparison of our algorithm with published algorithms since they have been developed for models which differ too much from the one we consider. For example, the heuristics for a flow-shop model with batching machines often deal with batches of unequal size, see, e.g., [17, 18, 20, 23, 24]; the production-transportation papers often consider non-identical vehicles [29] and take into account the routing issues [9]. Due to this reason, we also pay special attention to lower bound calculations and estimate the quality of our tabu search algorithm with respect to the calculated lower bound values.

The remainder of this paper is organized as follows. After defining our problem formally and discussing known results for it in Section 2, a mixed integer linear programming formulation is given in Section 3. Section 4 is devoted to different lower bound calculations. In Section 5 a tabu search algorithm is presented. Computational results can be found in Section 6. We conclude the paper with some remarks in Section 7.

2 Problem formulation

In this paper, we consider a combined production-transportation problem, which can be described as follows. There are n jobs of a set $N = \{1, 2, \ldots, n\}$ which have to be processed at a production site before being delivered to a customer by transportation vehicles. The corresponding stages are called production and transportation, and the two operations of a job are called production and transportation operations, accordingly.

At the production stage, each job $j \in N$ becomes available for processing at its release date r_j and has to be processed for $p_j \geq 0$ time units. The production site operates as a single machine, i.e. it processes at most one job at any time. Additionally, due dates d_j for the jobs $j \in N$ are given by the customer with the meaning that job j should be delivered not later than time d_j. We assume that all input data are integer.

After job j has finished processing at the production stage, it becomes available for transportation to the customer. The delivery is performed by m identical vehicles with limited capacity where any vehicle can carry no more than b jobs at any time. It takes a constant time τ to deliver a batch of jobs to the customer. Additionally, we assume that the time for returning back to the customer is negligible. The overall performance of a schedule is measured in terms of the maximum lateness $L_{\max} := \max\{L_j \mid j \in N\}$, where $L_j := C_j - d_j$ is the lateness of job j and C_j denotes the time where the delivery of job j to the customer is completed.

Another objective function equivalent to $L_{\max}$ is the extended makespan $C^q_{\max} = \max\{C_j + q_j\}$ where the jobs have tails q_j instead of due dates d_j. A tail q_j means that after the completion time C_j of job j additionally q_j time units are needed before the job is finished (tails of different job can be executed simultaneously). If we set $q_j := D - d_j$ for all $j \in N$ with a constant $D \geq \max\limits_{j \in N} d_j$, it is easy to see that $C^q_j = C_j + q_j = C_j - d_j + D = L_j + D$ holds. Since D is a constant, minimizing $L_{\max}$ is equivalent to minimizing $C^q_{\max}$.

A feasible production-transportation schedule may be completely specified by

- a processing sequence of the jobs on the production machine, and
- a sequence of batches for the transportation stage.

From these two sequences a feasible (left-shifted) schedule, in which every operation starts as early as possible, may be calculated as follows. At the production stage, each job can start immediately after it is released and the previous job is completed, i.e. the starting time S_j^p of job j on the production machine is $S_j^p = \max\{r_j, S_i^p + p_i\}$, where i is the job directly processed before j. The completion time C_j^p of j on the production machine is $C_j^p = S_j^p + p_j$.

For the transportation stage we have a partitioning of all jobs into a sequence $(B_1, B_2, \ldots, B_\alpha)$ of batches where $\alpha \in \{\lceil \frac{n}{b} \rceil, \ldots, n\}$ denotes the number of used batches and the sequence of batches determines their starting order. Since the transportation time τ is constant and the objective function is regular, it is sufficient to consider schedules in which the assignment of the batches to the vehicles is done in a cyclic way such that vehicle $v \in \{1, \ldots, m\}$ processes batches B_v, B_{v+m}, B_{v+2m}, etc. In such a schedule the starting time S_{B_k} of batch B_k can be determined as

$$S_{B_k} = \begin{cases} \max\{C_j^p \mid j \in B_k\}, & 1 \leq k \leq m \\ \max\{\max\{C_j^p \mid j \in B_k\}, S_{B_{k-m}} + \tau\}, & m < k \leq \alpha. \end{cases} \quad (1)$$

Furthermore, the completion time C_j of each job $j \in B_k$ is given by $C_j = S_{B_k} + \tau$.

Example 1: Consider an instance with $n = 5$ jobs, $m = 2$ vehicles with capacity $b = 2$, and delivery time $\tau = 4$. The job characteristics are defined as follows:

j	1	2	3	4	5
r_j	0	0	5	3	5
p_j	1	2	2	1	3
d_j	5	10	24	12	16

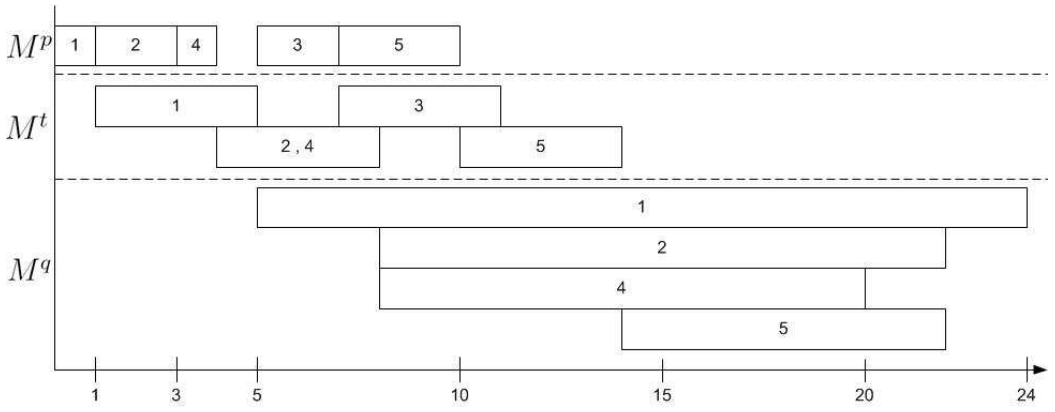


Figure 1: A feasible schedule for Example 1

With $D = d_3 = 24$ we get the tails $q_1 = 19, q_2 = 14, q_3 = 0, q_4 = 12, q_5 = 8$. Let us assume that on the production machine the jobs are processed in the sequence $(1, 2, 4, 3, 5)$; in the transportation stage the batch sequence is $(\{1\}, \{2, 4\}, \{3\}, \{5\})$. A corresponding feasible schedule with $C_{\max}^q = 24$ and $L_{\max} = 0$ (i.e. all due dates are respected) is presented in Figure 1. Here, the production machine is denoted by M^p, the transportation stage is denoted by M^t, and artificial tail-machines M^q are introduced to show the tail operations q_j for the jobs $j \in N$. □

The production-transportation model can be seen as a two-machine flow-shop system with a discrete machine at the first stage and m parallel batching machines at the second stage (cf. Ahmadi et al. [1]). It can be naturally decomposed into two subproblems. Relaxing the transportation stage by setting the transportation time to $\tau = 0$, the production subproblem corresponds to the classical single-machine problem $1|r_j|L_{\max}$, which is known to be strongly NP-hard, see [19]. Relaxing the production stage by setting $p_j = 0$ for all jobs j, the transportation subproblem for the vehicles corresponds to the parallel batching problem $P|\text{p-batch}, b < n, r_j, p_j = p|L_{\max}$. In this problem m parallel identical machines with bounded batch capacity $b < n$ are given which can process up to b jobs simultaneously, all having the same processing time p (corresponding to the transportation time τ).

Due to Condotta et al. [7], problem $P|\text{p-batch}, b < n, r_j, p_j = p|L_{\max}$ can be solved by the **barrier algorithm**. The time complexity of that algorithm is $O(n^3 \log^2 n)$ for multiple vehicles and $O(n^2 \log n)$ time for a single vehicle.

The special case where all due dates d_j are equal is equivalent to the makespan minimization problem $P|\text{p-batch}, b < n, r_j, p_j = p|C_{\max}$. We claim that it can be solved in $O(n \log n)$ time by the so-called *first-only-empty* **FOE-strategy** initially developed by Ikura and Gimple [15] for the single-machine version of the problem, $1|\text{p-batch}, b < n, r_j, p_j = p|C_{\max}$. It considers the jobs in non-descreasing order of their release dates grouping them in full batches of b jobs, except for the first batch which may contain less jobs, namely $n \bmod b$ jobs, if n is not divisible by b. A simple generalization of the FOE-strategy for the case of multiple batching machines $P|\text{p-batch}, b < n, r_j, p_j = p|C_{\max}$ groups the jobs in the batches as described above; their allocation to m vehicles (m parallel machines, equivalently) is a straightforward task since all batches have the same length τ: whenever a vehicle becomes available, it starts the transportation of the earliest available batch. The correctness of the generalized FOE-rule can be justified by the following two properties:

- there exists an optimal schedule in which the jobs are sequenced according to non-decreasing release dates;
- a schedule which does not satisfy the FOE-grouping (i.e., with some non-full batches B_k, $k \geq 2$) can be modified into a FOE schedule without increasing batch completion times by moving the jobs from earlier batches to non-full batches.

Extending the standard three-field scheduling notation we denote our two-stage problem as $(1|r_j) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau|L_{\max})$ or $(1|r_j) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau, q_j|C^q_{\max})$. In this notation, the first entry $(1|r_j)$ specifies that the production stage consists of a single machine and that release dates are given for the jobs. The second entry describes the transportation stage which consists of parallel batching machines with batch capacity $b < n$. The delivery times t_j are all equal to a constant τ. Finally, the last field denotes the objective function $L_{\max}$ for the minimization of the maximum lateness or $C^q_{\max}$ for the minimization of the extended makespan with tails.

We now review results on the two-stage model, which involves both production and transportation under the assumption that the processing times p_j at the first stage are arbitrary, while the transportation times $t_j = \tau$ in the second stage are the same for all jobs. Special cases of the corresponding two-machine flow-shop problem with one batching machine (i.e., $m = 1$) were studied by Ahmadi et al. [1] and Sung and Kim [27]. If all jobs are released simultaneously and have equal due dates (i.e., $r_j = 0$, $d_j = d$ for all $j \in N$), then the two-stage problem $(1|r_j = 0) \rightarrow (1|\text{p-batch}, b < n, t_j = \tau, d_j = d|L_{\max})$ is equivalent to the

makespan minimization problem $1 \rightarrow (1|\text{p-batch}, b < n, t_j = \tau|C_{\max})$, which can be solved by the **SPT-FOE**-rule in $O(n \log n)$ time due to Ahmadi et al. [1]. At the production stage the jobs are sequenced in SPT-order (*shortest processing time* first); at the transportation stage the jobs are sequenced in the same order and combined into batches in accordance with the FOE-rule.

On the other hand, the general case $(1|r_j) \rightarrow (1|\text{p-batch}, b < n, t_j = \tau|C_{\max})$ with arbitrary release dates is NP-hard in the strong sense as shown by Sung and Kim [27] and also by Lu et al. [22]. This implies that our problem with $m \geq 1$ batching machines and objective function $L_{\max}$ is also strongly NP-hard. If preemption for the production machine is allowed, problem $(1|r_j, pmtn) \rightarrow (1|\text{p-batch}, b < n, t_j = \tau|C_{\max})$ can be solved by the **SRPT-FOE-rule** [22, 27]:

SRPT: At the production stage the jobs are scheduled according to the *shortest remaining processing time* (SRPT) rule: at any decision point (given by a release date or the completion time of a job) schedule an available job with shortest remaining processing time.

FOE: At the transportation stage the jobs are sequenced according to the FOE-rule.

The described SRPT-FOE-rule can be generalized for multiple vehicles (multiple batching machines) employing the same SRPT-strategy at the production state and the same FOE-strategy to combine jobs for transportation. The only difference is in allocating the jobs to multiple batching machines rather than to one machine: whenever a vehicle becomes available, it starts the transportation of the earliest available batch.

The optimality of the SRPT-FOE-rule for multiple batching machines follows from the three properties stated by Sung and Kim [27] for a single machine:

Property 1: there exists an optimal solution with the same order of the jobs at the production and the transportation stage;

Property 2: the SRPT-strategy ensures the smallest possible completion time for each position $k = 1, 2, \ldots, n$, of the schedule;

Property 3: the FOE-strategy ensures the smallest possible makespan for scheduling batches of equal length τ on multiple machines [15].

Notice that the SRPT production schedule can be constructed in $O(n \log n)$ time if, e.g., a priority queue is used and the FOE transportation schedule can be constructed in $O(n)$ time.

Finally we observe that in prior research the case with m batching machines was studied only under an assumption of equal release dates and equal due dates: $(1|r_j = 0) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau|C_{\max})$. The $O(n \log n)$-time algorithm presented by Lee and Chen [16] can be considered as a generalization of a similar algorithm of Ahmadi et al. [1] with the same SPT-strategy at the production stage and the same FOE-strategy to create transportation batches.

The discussed results are summarized in Table 1.

Problem		Complexity		Reference
Transportation stage relaxed				
$1\|r_j\|L_{\max}$	–	NP-hard		[19]
Production stage relaxed				
–	$1\|\text{p-batch}, b < n, r_j, t_j = \tau\|C_{\max}$	$O(n \log n)$,	FOE	[15]
–	$P\|\text{p-batch}, b < n, r_j, t_j = \tau\|C_{\max}$	$O(n \log n)$,	FOE	this paper
–	$1\|\text{p-batch}, b < n, r_j, t_j = \tau\|L_{\max}$	$O(n^2 \log n)$,		[7]
–	$P\|\text{p-batch}, b < n, r_j, t_j = \tau\|L_{\max}$	$O(n^3 \log^2 n)$,	barrier alg.	[7]
Two-stage problem				
$(1\|r_j = 0)$	$\to (1\|\text{p-batch}, b < n, t_j = \tau\|C_{\max})$	$O(n \log n)$,	SPT-FOE	[1]
$(1\|r_j)$	$\to (1\|\text{p-batch}, b < n, t_j = \tau\|C_{\max})$	NP-hard		[22, 27]
$(1\|r_j, pmtn)$	$\to (1\|\text{p-batch}, b < n, t_j = \tau\|C_{\max})$	$O(n \log n)$,	SRPT-FOE	[22, 27]
$(1\|r_j = 0)$	$\to (P\|\text{p-batch}, b < n, t_j = \tau\|C_{\max})$	$O(n \log n)$,	SPT-FOE	[16]
$(1\|r_j, pmtn)$	$\to (P\|\text{p-batch}, b < n, t_j = \tau\|C_{\max})$	$O(n \log n)$,	SRPT-FOE	this paper

Table 1: Summary of complexity results

3 A mixed integer linear programming formulation

In this section we describe a mixed integer linear programming formulation (MIP) for problem $(1|r_j) \to (P|\text{p-batch}, b < n, t_j = \tau, q_j|C^q_{\max})$.

Let $\overline{\alpha} \leq n$ be an upper bound for the number of batches in the transportation stage and let $K := \{1, \ldots, \overline{\alpha}\}$. For each job $j \in N$ we introduce a variable S^p_j for the starting time on the production machine, for each $k \in K$ we have a variable S_{B_k} for the starting time of batch B_k. Additionally, we have binary variables $y_{ij} \in \{0, 1\}$ for $i, j \in N, i \neq j$ and $z_{jk} \in \{0, 1\}$ for $j \in N, k \in K$ where

$$y_{ij} = \begin{cases} 1, & \text{if job } i \text{ is processed before } j \text{ on the production machine,} \\ 0, & \text{otherwise;} \end{cases}$$

$$z_{jk} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } B_k, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, we have a variable $c \geq 0$ corresponding to the objective value $C^q_{\max}$. Using all these variables and three big values M_1, M_2, M_3 the problem can be formulated as follows.

$$\min c \tag{2}$$

$$\text{s.t.} \quad c - S_{B_k} + M_1(1 - z_{jk}) \geq \tau + q_j \qquad \forall j \in N, k \in K \tag{3}$$

$$S^p_j \geq r_j \qquad \forall j \in N \tag{4}$$

$$y_{ij} + y_{ji} = 1 \qquad \forall i, j \in N; i \neq j \tag{5}$$

$$S^p_j - S^p_i + M_2(1 - y_{ij}) \geq p_i \qquad \forall i, j \in N; i \neq j \tag{6}$$

$$S_{B_k} - S^p_j + M_3(1 - z_{jk}) \geq p_j \qquad \forall j \in N, k \in K \tag{7}$$

$$\sum_{k \in K} z_{jk} = 1 \qquad \forall j \in N \tag{8}$$

$$\sum_{j \in N} z_{jk} \leq b \qquad \forall k \in K \tag{9}$$

$$S_{B_k} - S_{B_{k-m}} \geq \tau \qquad m < k \leq \overline{\alpha} \tag{10}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in N; i \neq j \tag{11}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in N, k \in K \tag{12}$$
$$c \geq 0 \tag{13}$$
$$S_j^P \geq 0 \qquad \forall j \in N \tag{14}$$
$$S_{B_k} \geq 0 \qquad \forall k \in K \tag{15}$$

The objective function in (2) is to minimize the value $c = \max\{C_j + q_j\}$ which is correctly set due to (3): if job j is assigned to batch B_k, we have $c \geq S_{B_k} + \tau + q_j = C_j + q_j$. Constraints (4) ensure that the production of a job cannot start before its release date. Due to (5) two jobs cannot be processed simultaneously on the production machine. If i is processed before j on the production machine, we must have $S_j^p \geq S_i^p + p_i$ according to (6). Constraints (7) guarantee that a job cannot start its delivery in a batch before it is finished on the production machine. Due to (8) each job is assigned to exactly one batch, (9) guarantees that each batch contains at most b jobs. Finally, (10) implies that consecutive batches on the same vehicle do not overlap. Note that in the MIP no explicit assignment of batches to vehicles is modeled. As described in the previous section, such an assignment can be done in a cyclic way such that vehicle v processes batches B_v, B_{v+m}, B_{v+2m}, etc.

Additionally, we tried a second MIP based on time-indexed binary variables x_{it} where $x_{it} = 1$ if operation i completes at time t. It was much more time-consuming to solve.

4 Calculation of lower bounds

In this section we describe some methods to calculate lower bounds for problem $(1|r_j) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau, q_j|C^q_{\max})$.

At first we propose three constructive lower bounds which are based on different relaxations of the problem.

- LB_1: We relax the capacity of the production stage assuming that several jobs may be processed simultaneously on the production machine. We set the release dates for the batching machine to $r'_j := r_j + p_j$ and solve the resulting problem $P|\text{p-batch}, b < n, r'_j, t_j = \tau, q_j|C^q_{\max}$ for the transportation stage by the barrier algorithm from [7] with time complexity $O(n^3 \log^2 n)$.

- LB_2: We relax the capacity of the transportation stage assuming an unlimited number of vehicles. Additionally, we allow preemption for the production machine. Setting the tails for the production machine to $q'_j := q_j + \tau$, we obtain problem $1|r_j, pmtn, q'_j|C^q_{\max}$ or equivalently $1|r_j, pmtn|L_{\max}$ solvable in $O(n \log n)$ time by the *preemptive earliest due date* rule (cf. Horn [14]). For the classical $L_{\max}$-version of the problem, it schedules at each decision point (given by a release date or completion time of a job) an available job with the smallest due date; for the extended makespan version with $C^q_{\max}$ objective, the rule schedules at each decision point an available job with the largest tail q'_j.

- LB_3: We relax the tails q_j and allow preemption for the production machine. The resulting two-stage problem $(1|r_j, pmtn) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau|C_{\max})$ can be solved in $O(n \log n)$ time by the generalized version of the SRPT-FOE-rule described in Section 2.

Example 2: Consider an instance with $n = 6$ jobs, $m = 2$ vehicles with capacity $b = 2$, transportation time $\tau = 20$, and the following data:

j	1	2	3	4	5	6
r_j	3	1	2	2	1	4
p_j	12	7	2	5	6	2
q_j	9	11	0	13	7	10
r'_j	15	8	4	7	7	6
q'_j	29	31	20	33	27	30

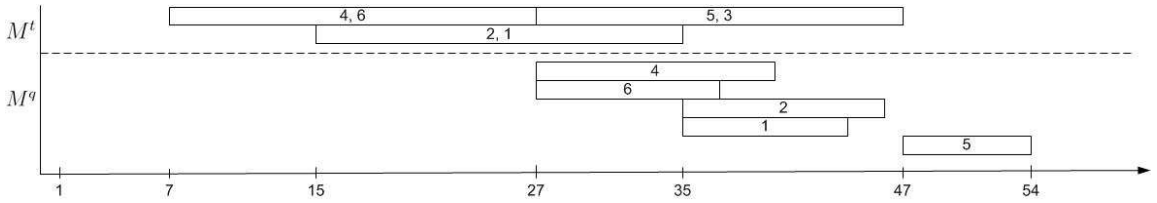


Figure 2: Schedule determining $LB_1 = 54$

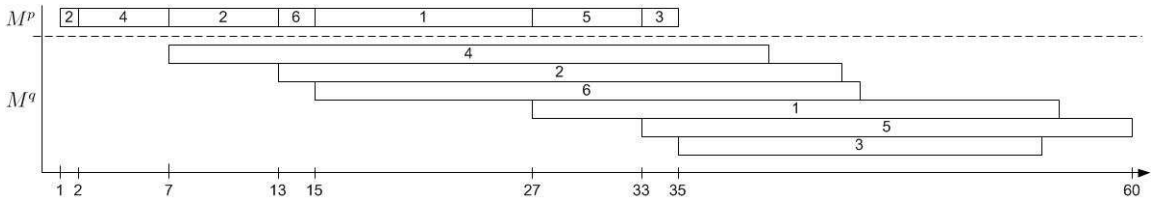


Figure 3: Schedule determining $LB_2 = 60$

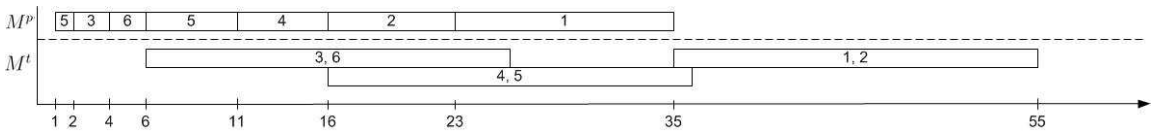


Figure 4: Schedule determining $LB_3 = 55$

For this data we get $LB_1 = 54$, $LB_2 = 60$, and $LB_3 = 55$. The corresponding schedules are illustrated in Figures 2, 3, and 4. □

For the above example we have $LB_2 = 60 > LB_3 = 55 > LB_1 = 54$, i.e. LB_2 provides the best lower bound. If we only consider a single vehicle and replace the batch capacity by $b = 1$, we obtain $LB_1 = 126 > LB_3 = 124 > LB_2 = 60$, i.e. LB_1 provides the best bound. There are other examples where LB_3 performs best.

In the following we consider some destructive lower bounds where a threshold value T for the makespan value $C^q_{\max}$ is given. A destructive test may result in one of the following three outcomes. It can provide a formal proof that for a relaxed problem no feasible schedule with $C^q_{\max} \leq T$ exists; then $T + 1$ is a valid lower bound value for the original problem. If the test fails to prove infeasibility, then in the best case it can provide a proof that T is a valid lower bound (by demonstrating, e.g., that for some job j its earliest starting time is equal to its latest starting time defined for the threshold value T); in the worst case the test may stop without making a conclusion about a lower bound.

Test 1: From a given threshold value T we derive deadlines $d'_j := T - q_j - \tau$ for the jobs on the production machine and study the feasibility problem $1|r_j, C_j \leq d'_j|-$ on the production machine. We apply the input and output tests originally developed for the job-shop problem by Carlier and Pinson [3].

In the *output test* we identify a job j and a set of jobs $\Omega \subseteq N$ which does not contain j such that condition

$$\max_{\mu\in\Omega} d'_\mu - \min_{\nu\in\Omega\cup\{j\}} r_\nu < \sum_{\nu\in\Omega\cup\{j\}} p_\nu$$

holds. Then activity j must end last in $\Omega\cup\{j\}$, i.e. we have precedence relations $\nu \to j$ for all $\nu \in \Omega$. This implies that j cannot be started before time $\tilde{r}_j := \max_{\nu\in\Omega} \{r_\nu + \sum_{\{h\in\Omega|r_h\geq r_\nu\}} p_h\}$ and any job $\nu \in \Omega$ should be completed before $d'_j - p_j$. As a consequence, we can update the release date of job j to $\max\{r_j, \tilde{r}_j\}$ and the deadline d'_ν for each job $\nu \in \Omega$ to $\min\{d'_\nu, d'_j - p_j\}$.

Symmetrically, in the *input test* a job j is identified which has to be processed before a subset Ω, which may lead to a decreased deadline d'_j and increased release dates r_ν for all $\nu \in \Omega$.

If a time window of a job becomes too small (i.e. $d'_j - r_j < p_j$), infeasibility is detected. Otherwise, the test may have produced increased release dates and decreased deadlines. The new deadlines may be transformed into increased tails $q_j := T - d'_j - \tau$. The input and output tests can be implemented in such a way that all pairs (j, Ω) satisfying the conditions for the current time windows can be found in $O(n \log n)$ time (cf. [4]). However, in our implementation we used a more simple version which runs in $O(n^2)$ time (cf. [2]). After time windows have been changed, the test may detect further pairs. Usually, the test is repeated until no more updates for the time windows are possible.

Test 2: For each job $j \in N$ denote by $\bar{S}_j = T - q_j - \tau$ its latest possible starting time at the transportation stage in a feasible schedule with $C^q_{\max} \leq T$. Furthermore, let

$$u := \begin{cases} \tau, & \text{if } m = 1, \\ 0, & \text{if } m > 1. \end{cases}$$

For a fixed test job j we calculate its earliest possible starting time $r'_j := r_j + p_j$ at the transportation stage and partition the set N of all jobs into three subsets:

- $N_1 := \{i \in N \mid \bar{S}_i < r'_j\}$. All these jobs must be processed in a batch (or in several batches) starting before the batch of job j.
- $N_2 := \{i \in N \setminus N_1 \mid \bar{S}_i < r'_j + u\} \cup \{j\}$. All these jobs cannot be processed in a batch starting later than the batch of job j.
- $N_3 := N \setminus (N_1 \cup N_2)$.

In the following we relax the batch capacity in the transportation stage and proceed as follows.

1. Calculate the earliest possible completion time $C(N_1)$ for all jobs in N_1 on the production machine (by solving problem $1|r_j|C_{\max}$ defined on the set of jobs N_1). Since all these jobs have to be processed in a batch (or batches) before j, job j cannot be started at the transportation stage before time $C(N_1) + u$, i.e. we may update r'_j to $r'_j := \max\{r'_j, C(N_1) + u\}$.

2. Calculate the earliest possible completion time $C(N_{12})$ for all jobs in $N_1 \cup N_2$ on the production machine (by solving problem $1|r_j|C_{\max}$ defined on the set of jobs $N_1 \cup N_2$). Since all these jobs cannot be processed in a batch starting later than j, job j cannot be started before time $C(N_{12})$ on the transportation stage, i.e. we may update r'_j to $r'_j := \max\{r'_j, C(N_{12})\}$.

3. Check whether $\bar{S}_j \leq r'_j$ holds. If this inequality holds strictly, obviously, no feasible schedule exists and the whole procedure can be stopped.

   If $\bar{S}_j = r'_j$, then job j should start exactly at time $\bar{S}_j$, which implies that its completion time in a feasible schedule is no less than $\bar{S}_j + \tau + q_j = T$ and therefore T is a valid lower bound.

If $\bar{S}_j > r'_j$ and the value r'_j has been updated, repeat the procedure with new sets N_1, N_2 based on the new value r'_j.

For an efficient implementation of the described procedure we use two lists of jobs, one in non-decreasing order of r_i, $i \in N$, and another one in non-decreasing order of $\bar{S}_i$, $i \in N$. Given a fixed test job j, there can be up to n iterations of the described procedure. Each time the value r'_j increases and at least one of the sets N_1 or N_2 gets extra jobs, with exception of (possibly) the last two iterations when the sets N_1 and N_2 may remain unchanged resulting in unchanged r'_j.

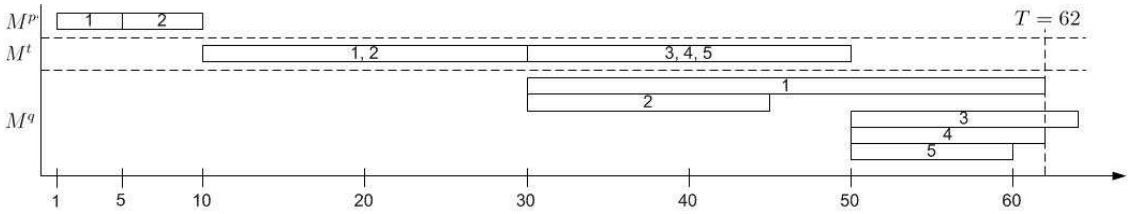
In each iteration, the splitting $N_1 \cup N_2 \cup N_3$ is obtained in $O(n)$ time, using the ordered list of $\bar{S}_i$-values, and each of the two problems $1|r_j|C_{\max}$ which emerge in Steps 1 and 2 can be solved in $O(n)$ time, using the ordered list of r_i-values. Thus, the overall time complexity is $O(n^2)$.

Example 3: Consider an instance with $n = 5$ jobs, a single vehicle with capacity $b = 4$, transportation time $\tau = 20$, threshold $T = 62$ and the following data:

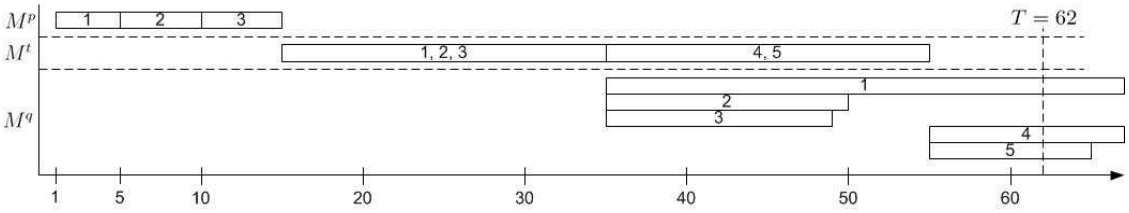
j	1	2	3	4	5
r_j	1	3	7	13	16
p_j	4	5	5	6	6
q_j	32	15	14	12	10

Due to $m = 1$ we have $u = \tau = 20$. For the latest starting times at the transportation stage we get $\bar{S}_1 = 10$, $\bar{S}_2 = 27$, $\bar{S}_3 = 28$, $\bar{S}_4 = 30$, and $\bar{S}_5 = 32$. We consider $j = 2$ as a test job and start with $r'_2 = r_2 + p_2 = 8$.

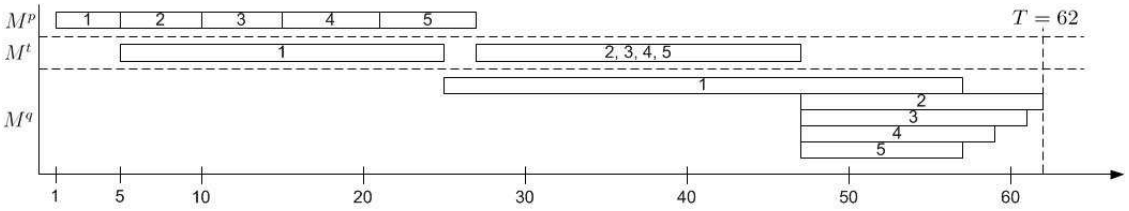
Iteration 1: We get $N_1 = \emptyset$, $N_2 = \{1, 2\}$ since $\bar{S}_1 = 10 < r'_2 + u = 28$, and $N_3 = \{3, 4, 5\}$. We calculate $C(N_{12}) = r_1 + p_1 + p_2 = 10$ and update $r'_2 = 10$.



Iteration 2: We get $N_1 = \emptyset$, $N_2 = \{1, 2, 3\}$ since $\bar{S}_3 = 28 < r'_2 + u = 30$, and $N_3 = \{4, 5\}$. We calculate $C(N_{12}) = r_1 + p_1 + p_2 + p_3 = 15$ and update $r'_2 = 15$.



Iteration 3: We get $N_1 = \{1\}$ due to $\bar{S}_1 = 10 < r'_2 = 15$, $N_2 = \{2, 3, 4, 5\}$, and $N_3 = \emptyset$. We calculate $C(N_1) = r_1 + p_1 = 5$ and update $r'_2 = 5 + u = 25$. We calculate $C(N_{12}) = r_1 + p_1 + p_2 + p_3 + p_4 + p_5 = 27$ and update $r'_2 = 27$.



Due to $\bar{S}_2 = r'_2$ we can conclude that $T = 62$ is a valid lower bound.

For this example the constructive lower bounds are $LB_1 = 60$, $LB_2 = 57$, and $LB_3 = 47$. □

Test 3: In this test we calculate for each job j the number of the latest batch in which j must be delivered in order to satisfy the given threshold T. For this purpose we determine an interval $[\underline{S}_j, \overline{S}_j]$ in which j has to start its transportation. The upper value $\overline{S}_j$ can be calculated as before by $\overline{S}_j = T - q_j - \tau$. The lower value $\underline{S}_j$ is set to the earliest starting time of the first batch. Initially, it is given by $\min_{j=1}^{n}\{r_j + p_j\}$, afterwards during the test this value may be increased. The main part of the test is to determine how many jobs (and if possible, also which jobs) must be processed in the first batch (which influences the starting time of the first batch). If the number of jobs which have to be started until a certain time exceeds the total capacity of the batches which can be started until that time, we may state infeasibility. Furthermore, infeasibility is detected if $\underline{S}_j > \overline{S}_j$ holds.

Let r'_{B_1} be the earliest possible starting time of the first batch B_1, initially we have $r'_{B_1} = \min_{j=1}^{n}\{r_j + p_j\}$. For a given time value t, the maximum number of batches which can be started in the interval $[r'_{B_1}, t]$ is $\left\lceil \frac{t - r'_{B_1} + 1}{\tau} \right\rceil \cdot m$. Hence, each job j must be started at the latest in the batch with number $\overline{B}(j) := \left\lceil \frac{\overline{S}_j - r'_{B_1} + 1}{\tau} \right\rceil \cdot m$. Let us renumber the jobs according to non-increasing tails, i.e. $q_1 \geq q_2 \geq \ldots \geq q_n$. Then, for $i < j$ we have $\overline{B}(i) \leq \overline{B}(j)$, which implies that in any feasible schedule the jobs $1, \ldots, j$ should be contained in the batches $B_1, \ldots, B_{\overline{B}(j)}$, together with possibly some other jobs. In the following we calculate a lower bound z_1 for the number of jobs which have to be processed in the first batch. Furthermore, in the case $m = 1$ we also find a subset of jobs F_1 which should belong to the first batch in any feasible schedule, $|F_1| \leq z_1$.

For each job j we distinguish the following five cases.

1. If $\overline{B}(j) \leq 0$ holds, then $\overline{S}_j < r'_{B_1} \leq \underline{S}_j$, i.e. we may state infeasibility.

2. If $\overline{B}(j) \cdot b < j$ holds, then the capacity of the batches $B_1, \ldots, B_{\overline{B}(j)}$ is too small for processing the jobs $1, \ldots, j$, i.e. also infeasibility is detected.

3. If $\overline{B}(j) \cdot b = j$ holds, then in each of the batches $B_1, \ldots, B_{\overline{B}(j)}$ exactly b jobs have to be started. In particular, the first batch must contain b jobs, i.e. we may set $z_1 := b$.

4. If $\overline{B}(j) = m$ holds, then in the batches $B_1, \ldots, B_m$ at least j jobs must be started. Since in the batches $B_2, \ldots, B_m$ at most $b(m-1)$ jobs can be started, in the first batch at least $z_1 := j - b(m-1)$ jobs must be contained. In the special case $m = 1$ we know that j must be contained in B_1, i.e. we may add j to the set F_1.

5. If $(\overline{B}(j) - 1)b < j$ holds, then the first $\overline{B}(j) - 1$ batches do not have sufficient capacity for the jobs $1, \ldots, j$. Then the smallest number of jobs in batch B_1 is $j \bmod b$, which may happen only if each of the subsequent batches $B_2, \ldots, B_{\overline{B}(j)}$ is full (i.e. contains b jobs). Thus, we conclude that $z_1 \geq j \bmod b$.

The last three cases are not mutually exclusive. We set z_1 to the largest value which may be derived from these considerations.

It remains to describe how the value z_1 (and additionally the set F_1 in the case $m = 1$) can be used to update the value r'_{B_1}. In the case $m > 1$ we determine the earliest time on the production machine when z_1 (arbitrary) jobs can be completed. In the case $m = 1$ we determine the earliest time on the production machine when z_1 jobs, among them the jobs from the set F_1, can be completed. Both cases can be solved using the (preemptive) SRPT-rule.

- If $m > 1$, we schedule at each decision point (given by a release date or the completion time of a job) an available job with the shortest remaining processing time. The scheduling process is stopped as soon as z_1 jobs are completed. The validity of this procedure follows from Property 2 formulated in Section 2: the completion time of the last scheduled job is a lower bound for the completion time of any non-preemptive schedule consisting of z_1 jobs.

- If $m = 1$, we allocate first all jobs from F_1 starting them at their earliest starting times; after that we proceed with the SRPT-rule adding new jobs to the partial schedule until exactly $z_1 - |F_1|$ jobs are fully allocated. Time intervals used by the jobs F_1 are treated as infeasible time intervals for $z_1 - |F_1|$ jobs. Clearly, allocating compulsory jobs F_1 as early as possible leaves more flexibility for allocating the remaining jobs in comparison with later allocation of F_1; allocating $z_1 - |F_1|$ jobs by the SRPT-rule ensures that the last scheduled job is completed as early as possible, subject to forbidden intervals used by jobs F_1.

The time complexity of test 3 applied for all jobs $j = 1, 2, \ldots, n$ is $O(n \log n)$: it takes $O(n \log n)$ time to perform pre-processing, $O(1)$ time to check the conditions of the five cases for each fixed j, and $O(n \log n)$ time to apply the SRPT-rule with $z_1 \leq n$ jobs. In our implementation, whenever an update is done, the test is restarted from the beginning with $j = 1, 2, \ldots, n$.

Example 4: Consider an instance with $n = 7$ jobs, a single vehicle with capacity $b = 2$, transportation time $\tau = 20$, threshold $T = 95$ and the following data:

j	1	2	3	4	5	6	7
r_j	8	19	22	10	12	3	20
p_j	5	5	1	1	6	7	5
q_j	47	28	24	22	15	7	2

Initially, we have $r'_{B_1} = r_6 + p_6 = 10$, which implies $\overline{B}(1) = \lceil \frac{95-47-20-10+1}{20} \rceil = 1$. Hence, job 1 must be started in the first batch and we update r'_{B_1} to $r_1 + p_1 = 13$. After a restart we get $\overline{B}(1) = \lceil \frac{16}{20} \rceil = 1$, $\overline{B}(2) = \lceil \frac{35}{20} \rceil = 2$, $\overline{B}(3) = \lceil \frac{39}{20} \rceil = 2$. Thus, at least 3 jobs must be contained in the first two batches. Due to the capacity $b = 2$ at least one job must be started in the first batch (we already know that it is job 1). We continue with $\overline{B}(4) = \lceil \frac{41}{20} \rceil = 3$, $\overline{B}(5) = \lceil \frac{48}{20} \rceil = 3$, $\overline{B}(6) = \lceil \frac{56}{20} \rceil = 3$. Thus, at least 6 jobs must be contained in the first three batches, i.e. according to the third case we must have exactly two jobs in each batch. If we apply the SRPT-rule with the set $F_1 = \{1\}$ and $z_1 = 2$, we get the sequence $(1, 4)$ on the production machine. Since this sequence cannot complete before time $r_1 + p_1 + p_4 = 14$, we may update $r'_{B_1} = 14$.

After starting again with job 1, we get $\overline{B}(1) = \lceil \frac{15}{20} \rceil = 1$, $\overline{B}(2) = \lceil \frac{34}{20} \rceil = 2$, $\overline{B}(3) = \lceil \frac{38}{20} \rceil = 2$, $\overline{B}(4) = \lceil \frac{40}{20} \rceil = 2$, $\overline{B}(5) = \lceil \frac{47}{20} \rceil = 3$, $\overline{B}(6) = \lceil \frac{55}{20} \rceil = 3$, $\overline{B}(7) = \lceil \frac{60}{20} \rceil = 3$. We have the second case: 7 jobs must be started in 3 batches, which is impossible due to $b = 2$. Thus, $T = 95$ is infeasible and $T + 1 = 96$ is a valid lower bound value. For comparison, the constructive lower bounds are $LB_1 = 94$, $LB_2 = 80$, and $LB_3 = 91$. □

All three tests can be used together in a destructive lower bound calculation. We are given a threshold value T for the objective function $C^q_{\max}$ and try to prove that no feasible schedule with $C^q_{\max} \leq T$ exists. At first test 1 is applied. If it does not detect infeasibility, it may have produced increased release dates and tails for the jobs. Afterwards, test 2 is applied with each job $j = 1, 2, \ldots, n$ selected as a test job. Finally, test 3 is performed for all jobs $j = 1, 2, \ldots, n$. By trying different T-values in a binary search procedure, the largest infeasible T is calculated. Then $T + 1$ is the destructive lower bound value.

5 A tabu search algorithm

In this section we describe a tabu search algorithm to find heuristic solutions for problem $(1|r_j) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau, q_j|C^q_{\max})$. We decompose the problem into the two subproblems of production and transportation. First a solution to the production subproblem is defined as a sequence of all jobs on the production machine. Then the resulting subproblem for the transportation stage becomes the parallel batching problem $P|\text{p-batch}, b < n, r'_j, p_j = p|C^q_{\max}$ where the release dates r'_j correspond to the completion times C^p_j at the production stage and the processing times are equal to the delivery time τ. An optimal solution to the latter problem can be found in $O(n^3 \log^2 n)$ time by the barrier algorithm from Condotta et al. [7].

Thus, it is sufficient to represent a solution by a production sequence $(j_1, \ldots, j_n)$. In order to reduce the search space we take precedence relations into account. If we know that a job i has to precede another job j, we only consider sequences in which i is placed before j. As mentioned in Section 4, test 1 may produce precedences $i \rightarrow j$ for certain pairs of jobs i, j. In every iteration where a new best solution is found, we start the input and output test based on the new upper bound T and store all precedences which can be derived by test 1.

In order to find a good production sequence, we first construct a starting solution by a priority-based heuristic, afterwards we try to improve this solution by tabu search. To calculate an initial production sequence we use one of the following four rules:

Rule 1: Sort the jobs according to non-decreasing release dates r_j.

Rule 2: Sort the jobs according to non-increasing tails q_j.

Rule 3: At any decision point (given by a release date or the completion time of a job) schedule an available job with the largest tail.

Rule 4: Calculate a preemptive schedule by using the SRPT-rule. Remove the preemptions by scheduling the jobs without preemption according to increasing starting times in the preemptive schedule.

The production schedules generated by the four rules are then complemented by optimal transportation schedules obtained by the barrier algorithm [7]. The resulting four schedules are used as initial solutions for the tabu search algorithm described below.

We consider the following two neighborhoods for a production sequence: in the swap-neighborhood $\mathcal{N}_{swap}$ we interchange two jobs in the sequence, in the shift-neighborhood $\mathcal{N}_{shift}$ we move one job to another position. We tested different attributes for characterizing solutions to be tabu. It turned out that the following two criteria performed best:

- If in the swap-neighborhood $\mathcal{N}_{swap}$ we interchange two jobs j_λ, j_μ at the positions $\lambda < \mu$ in the sequence $(j_1, \ldots, j_\lambda, \ldots, j_\mu, \ldots, j_n)$, we store the pair (j_λ, λ). The corresponding swap-operation is denoted by $swap_{\lambda,\mu}$. As long as the pair (j_λ, λ) is contained in the tabu list, we forbid all solutions in which job j_λ is swapped to a position $\nu \leq \lambda$.
- If in the shift-neighborhood $\mathcal{N}_{shift}$ we move the job j_λ from position λ to another position μ, we simply store the job j_λ. The corresponding shift-operation is denoted by $shift_{\lambda,\mu}$. As long job j is contained in the tabu list, it is forbidden to shift job j_λ again.

We use a first-fit strategy where at first the swap-neighborhood and then the shift-neighborhood is searched for an improving solution. In our implementation, in order to reduce the size of the neighborhood, we restrict both operators $swap_{\lambda,\mu}$ and $shift_{\lambda,\mu}$ to positions λ, μ with $|\mu - \lambda| \leq d$ for a fixed integer $d > 0$. Additionally, to speed up neighbor generation, for both operators we only consider jobs j_λ which are processed before a so-called "critical" job at the production stage. Notice that the idea of re-structuring a "critical" part of a schedule is often employed in neighbor generation for many scheduling models (e.g. the job-shop problem).

Here, we call a job j critical if it defines the overall makespan of the schedule, i.e. if

$$C^q_{\max} = C_j + q_j = S^t_j + \tau + q_j,$$

where S^t_j is the starting time of j at the transportation stage. If there are several jobs with this property, we select the earliest one. It is easy to see that if B is the batch in which job j is delivered, in a solution with a smaller $C^q_{\max}$-value batch B must be started earlier. We try to achieve this by restricting the operators $swap_{\lambda,\mu}$ and $shift_{\lambda,\mu}$ to move only jobs j_λ satisfying $C^p_{j_\lambda} \leq S^t_j$. However, as discussed in appendix A, a decrease in the objective value may also be achieved by other moves.

Example 5: Consider an instance with $n = 7$ jobs, $m = 1$ vehicle with capacity $b = 2$ and delivery time $\tau = 6$. The job characteristics are defined as follows:

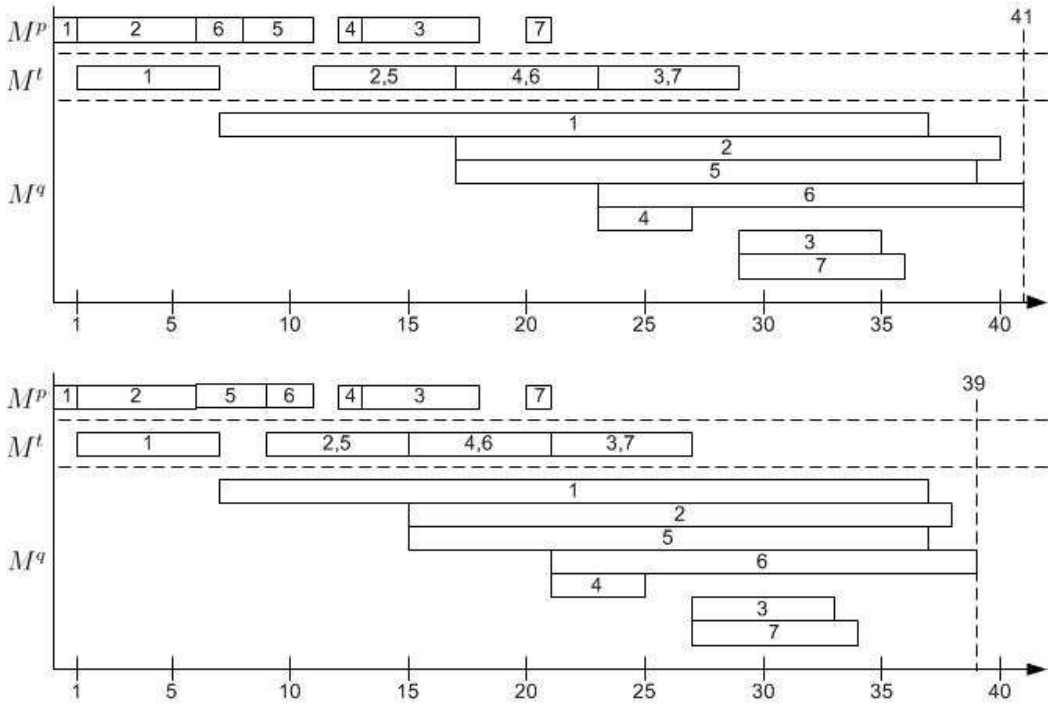


Figure 5: Neighbor generation of a schedule

j	1	2	3	4	5	6	7
r_j	0	1	9	12	3	6	20
p_j	1	5	5	1	3	2	1
q_j	30	23	6	4	22	18	7

A schedule corresponding to the production sequence $(1, 2, 6, 5, 4, 3, 7)$ is shown in the upper part of Figure 5. The only critical job is job 6 since there is no other job defining the makespan. Job 6 is delivered in batch B_3 starting at time $S_6^t = 17$. The jobs j_λ satisfying $C_{j_\lambda}^p \leq S_j^t = 17$ (i.e., completing on the production machine before the start of B_3) are the jobs $1, 2, 6, 5, 4$. In accordance with the formulated rules for neighbor generation, one of this jobs is selected as j_λ and moved elsewhere. If we swap jobs 6 and 5 in the production sequence, we get the schedule shown in the lower part of the figure. It has the improved objective value $C_{\max}^q = 39$. □

In preliminary computational experiments we tested different variants and parameters for the tabu search algorithm. It turned out that the following settings performed quite well on average. We run the tabu search procedure four times with different initial solutions generated by Rules 1-4 and stop each run after 3 non-improving iterations, when a time limit is reached or when a solution is found which is verified to be optimal by our common lower bound. We use a tabu list of fixed length 5. The parameter d restricting the range of the operators is set to $d \in \{\lceil n/2 \rceil, \lceil n/3 \rceil, \lceil n/4 \rceil\}$ depending on $n \in \{50, 100, 200\}$.

6 Computational results

In this section we describe some computational results for the proposed methods dealing with the production-transportation problem. We implemented all algorithms in Java and tested them on a computer with a 2.8 GHz-processor and 2 GB main storage.

As test data, we generated random instances similarly to the experiments of Sung and Kim [27]. We consider various combinations of the following parameters: the number of jobs $n \in \{50, 100, 200\}$, the number of vehicles $m \in \{1, 2, 5\}$, the delivery time $\tau \in \{40, 100, 500\}$, and the vehicle capacity $b \in \{5, 10, 15\}$. Furthermore, the release dates r_j are uniformly distributed from an interval $[1, 10 \cdot \theta \cdot n]$ with $\theta \in \{0.1, 0.5, 1\}$, the processing times p_j are uniformly distributed from $[1, 10]$ or $[1, 100]$, and the due dates d_j are uniformly distributed from $[r_j+p_j+\tau, (r_j+p_j+\tau)\delta]$ with $\delta \in \{1.2, 1.5, 1.8\}$. For each combination of $n, \tau, b, \theta, \delta$ and a processing time interval we generated 10 instances, i.e., in total we have $3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 2 \cdot 10 = 4860$ instances. Each of these instances can be tested with $m = 1, 2, 5$ vehicles.

At first we calculated the six individual lower bounds LB_1, LB_2, LB_3, LB_{T1}, LB_{T2}, LB_{T3} and the common bound $LB = \max\{LB_1, LB_2, LB_3, LB_{T1}, LB_{T2}, LB_{T3}\}$. In order to estimate their quality, we determine for each lower bound LB_v the relative deviation $\Delta_v = \frac{LB-LB_v}{LB}$ from the best lower bound and count how often $LB_v = LB$ holds. In Tables 5 to 10 in appendix B individual results for all lower bounds and all instances are reported.

In Table 2 aggregated results are shown where each row averages over 4860 instances and the best results are written in bold. The first section corresponds to instances with small processing times $p_j \in [1, 10]$ with a summary in the fourth row; the second section corresponds to large processing times $p_j \in [1, 100]$, also with a summary in the fourth row. The last row summarizes the results for all instances. For each lower bound LB_v the relative deviation Δ_v (in percent) and the number of occurrences $LB_v = LB$ are reported. In the last column the average and maximum computation times (in seconds) to calculate the common lower bound can be found.

m	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}		time	
1	3.0	902	35.1	184	14.8	86	35.1	184	38.1	466	**2.9**	**1163**	1.7	4.5
2	**7.1**	**1147**	18.9	859	19.5	205	18.9	884	30.2	181	7.8	624	2.0	15.8
5	12.5	1062	**5.0**	1586	24.4	273	**5.0**	**1675**	17.7	636	13.6	328	2.4	26.7
	7.5	**3111**	19.7	2629	19.6	564	19.7	2743	28.7	1283	8.1	2115	2.0	
1	53.6	7	5.1	**2105**	**0.5**	1537	5.1	**2105**	67.8	0	53.1	325	8.0	116.8
2	65.1	2	**0.1**	2372	0.4	1719	**0.1**	**2373**	70.0	0	65.1	56	8.5	108.9
5	75.3	0	**0.0**	2429	0.3	1747	**0.0**	**2430**	69.9	0	75.6	0	10.3	106.8
	64.7	9	1.7	6906	**0.4**	5003	1.7	**6908**	69.2	0	64.4	381	8.9	
	36.1	3120	10.7	9535	**10.0**	5567	10.7	**9651**	49.0	1283	36.4	2496	5.5	

Table 2: Comparison of lower bounds

As it can be seen there is no clear dominance of one lower bound on all instances. Depending on the characteristics of the instances the following conclusions can be drawn:

- LB_1 is mainly based on the transportation stage and provides better results when this stage is dominant (i.e. for smaller processing times on the production machine and larger τ). For instances with $m = 2$ and $p_j \in [1, 10]$ this bound is on average the best bound. On the other hand, for instances with $p_j \in [1, 100]$ and $m = 2, 5$ the bound LB_1 has very large deviations.

- LB_2 and LB_{T1} are mainly based on the production stage and provide very similar (often even identical) results. While for LB_2 the preemptive problem $1|pmtn, r_j, q_j|C^q_{\max}$ is solved, for LB_{T1} the non-preemptive feasibility problem $1|r_j, C_j \leq d_j|-$ is considered, sometimes leading to slightly better results. Both bounds are very suitable for instances with a dominating production stage (i.e. large processing times p_j). For instances with $p_j \in [1, 100]$ they are the best bounds, also for $p_j \in [1, 10]$ and $m = 5$ they perform best.

- LB_3 is based on the two-stage problem $(1|r_j, pmtn) \rightarrow (P|\text{p-batch}, b < n, t_j = \tau|C_{\max})$ and provides good results for instances where both stages have a similar impact. For $p_j \in [1, 100]$ it provides better results than for $p_j \in [1, 10]$. For $m = 1$ and $p_j \in [1, 100]$ it achieves the smallest average deviation from the best known lower bound. Although almost for every instance LB_3 is dominated by another bound, on average it provides quite good results.

- LB_{T2} and LB_{T3} are mainly based on the transportation stage which is also confirmed by the results. These two bounds provide better results for smaller processing times and $m = 1$.

Due to the fact that the bounds behave very differently for different instances and the computation times are small for most instances, it can be concluded that all bounds should be calculated to get the best result.

In order to estimate the quality of the tabu search algorithm we compared it with the MIP formulation (solved by CPLEX 12) described in Section 3. Both algorithms were run on 1800 instances with the same time limit (10 or 15 minutes). More specifically, we used the following three classes of test instances:

(I) $m = 1, n = 50, p_j \in [1, 10]$, 810 instances, 467 of them could be solved to optimality by the MIP formulation within a time limit of 10 minutes

(II) $m = 1, n = 50, p_j \in [1, 100]$, 810 instances, 49 could be solved to optimality by the MIP formulation within a time limit of 10 minutes

(III) $m = 1, n = 100, p_j \in [1, 100]$, 90 instances, none of them could be solved to optimality by the MIP formulation within a time limit of 15 minutes

In Table 3 aggregated results are shown for these instances. In the first three columns we show the number of instances for which the tabu search produces better, equal or worse results compared with the MIP. In the next four columns the heuristic results of tabu search and the MIP are compared with respect to the best known lower bounds. We report the number of instances for which the heuristic upper bounds could be verified to be optimal by the lower bound values and show the average relative deviations of tabu search solutions and MIP solutions from the lower bound values: $\Delta^{TS}_{LB} = \frac{UB^{TS}-LB}{LB}$ and $\Delta^{MIP}_{LB} = \frac{UB^{MIP}-LB}{LB}$ (in percent). Finally, in the last two columns the average computation times (in seconds) can be found.

It can clearly be seen that the MIP is outperformed by the tabu search (especially for larger instances). For all 90 instances in (III) the tabu search obtained better results than the MIP formulation and the average relative deviation Δ^{TS}_{LB} is much smaller than Δ^{MIP}_{LB}. For the instances in (I) and (II) on average tabu search produces better solutions in a smaller

	$TS <$	$TS =$	$TS >$	$UB^{TS} = LB$	$UB^{MIP} = LB$	Δ_{LB}^{TS}	Δ_{LB}^{MIP}	time TS	time MIP
(I)	268	519	23	456	334	0.4	1.2	269	300
(II)	751	53	6	374	49	1.1	5.7	387	575
(III)	90	0	0	0	0	2.2	53.2	900	900

Table 3: Comparison of tabu search and MIP

amount of time. Notice that for half of these instances optimal solutions with $UB^{TS} = LB$ were found and the computation time was smaller than the time limit.

Afterwards we tried to solve the MIP formulation for instances with $m > 1$. For all instances tested (90 instances with $m = 2, n = 200, p_j \in [1, 100]$) even a feasible solution could not be found within a time limit of 20 minutes. Thus, MIP cannot be used for these instances, and therefore we compare the tabu search results only with lower bound values.

In Table 4 aggregated results are shown for all instances with $m = 2$ and $m = 5$. For $m = 2$ each row summarizes the results of 810 instances, for $m = 5$ each row corresponds to $3 \cdot 810 = 2430$ instances. The tabu search procedure was executed with a time limit of 15 minutes. We report the number of instances for which the heuristic upper bounds (best initial upper bound value UB^0 and tabu search value UB^{TS}) could be verified to be optimal by the lower bound values and present the average relative deviations from the lower bounds: $\Delta_{LB}^{UB^0} = \frac{UB^0 - LB}{LB}$ and $\Delta_{LB}^{TS} = \frac{UB^{TS} - LB}{LB}$ (in percent). In the next column we show how often the tabu search improves UB^0. Finally, in the last column the average computation times (in seconds) can be found. Notice that the latter values are less than the time limit of 15 minutes since in 25%-97% of cases provable optimal solutions were obtained before that time.

	$UB^0 = LB$	$UB^{TS} = LB$	$\Delta_{LB}^{UB^0}$	Δ_{LB}^{TS}	$UB^{TS} < UB^0$	time
$m = 2, [1, 10], n = 50$	198	294	1.29	0.91	293	564
$m = 2, [1, 10], n = 100$	177	248	0.79	0.59	289	618
$m = 2, [1, 10], n = 200$	156	205	0.47	0.40	261	675
$m = 2, [1, 100], n = 50$	565	573	0.59	0.46	64	264
$m = 2, [1, 100], n = 100$	570	580	0.42	0.29	68	261
$m = 2, [1, 100], n = 200$	547	550	0.25	0.20	54	313
$m = 5, [1, 10]$	1214	1430	0.66	0.55	639	374
$m = 5, [1, 100]$	2357	2359	0.01	0.01	3	41

Table 4: Comparison of tabu search results with lower bounds

The computational results show that the average relative deviations between lower and upper bounds are quite small, i.e. both, the lower bound algorithm and the tabu search procedure provide good results in a relatively small amount of time. It turns out that the instances with $m = 5$ become easier to solve. This observation is in agreement with a statement by Gupta and Tunc [10] who studied a related flow-shop problem with parallel machines at the second stage and noticed that "the availability of a large number of machines at second stage increases the chances of obtaining a minimum makespan schedule".

7 Concluding remarks

In this paper we studied a combined production-transportation problem where the production stage consists of a single machine and the transportation stage consists of vehicles with limited capacity. This NP-hard problem appears to be computationally challenging, as shown in our experiments with the MIP formulation presented in Section 3. Therefore, in

our study we focus mainly on heuristic algorithms and on special techniques for improving their performance.

In our algorithms we essentially make use of our earlier result [7] which allows us to construct efficiently an optimal solution to the transportation subproblem based on a given solution to the production subproblem. The implications of this result are twofold:

- our approach ensures that production and transportation schedules are coordinated in the best possible way;
- based on [7], we introduce a compact representation of solutions enumerated by tabu search limiting the search space to production sequences only.

The neighbor generation strategy for the production-transportation is not a trivial task. As we show in the appendix, a decrease in the objective function may be achieved not only by removing the jobs from the part of the production sequence preceding the critical job, but also by re-allocated jobs from the later part of the production sequence moving them in front of the critical job. To the best of our knowledge, this phenomenon has not been observed in the past for traditional scheduling problems.

In addition to tabu search, we develop several algorithms for lower bound calculation. While the primary goal is to perform a more accurate evaluation of tabu search, a systematic study of lower bounds leads to further advancements in the tabu search design. Some of our procedures for lower bound calculation provide additional information for tabu search making it possible to fix precedence relations in an optimal way and to speed up the search. We strongly believe that the computational effort spent on lower bound calculation should bring a pay off for other solution methods as well. In particular, our study of lower bounds will be most useful in the design of branch-and-bound algorithms.

Our computational results show that the average relative deviations between lower and upper bounds are quite small, i.e. both, the lower bound algorithm and the tabu search procedure provide good results in a relatively small amount of time. For the tested instances with $m = 1$ the gap is 1.2% on average, for all instances with $m = 2$ it is 0.48 % and for instances with $m = 5$ (which appear to be easier to solve) it is only 0.28%.

The outcomes of our study may be useful for addressing more general models. For example, the production stage may have parallel machines rather than a single machine, as assumed in our study. A more complicated supply chain model may include an additional transportation stage before production (transportation from a supplier) followed by another transportation stage (transportation to customers).

Appendix A

Example 6 : Consider an instance with $n = 6$ jobs and one vehicle with capacity $b = 2$ and delivery time $\tau = 6$. The job characteristics are defined as follows:

j	1	2	3	4	5	6
r_j	0	1	2	8	10	1
p_j	1	4	2	2	1	1
q_j	13	7	2	8	1	1

For the production sequence $(1, 2, 3, 4, 5, 6)$ the optimal transportation sequence found by the barrier algorithm is $(\{1, 2\}, \{3, 4\}, \{5, 6\})$ resulting in $C^q_{\max} = 25$. In this solution (cf. the

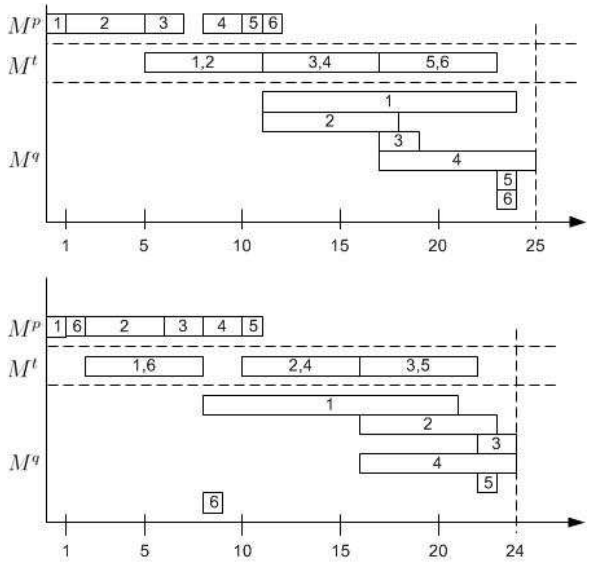


Figure 6: Moving a job processed in the production sequence after a critical job

upper schedule in Figure 6), job 4 determines the objective value and is the only critical job according to our definition from Section 5. A critical path (in the common sense) consists of the sequence $(1, 2)$ on M^p, the batches $\{1, 2\}, \{3, 4\}$ on M^t and the tail of job 4.

Typically, restructuring the schedule after a critical job or moving new jobs in front of a critical job cannot reduce the completion time of the critical job. Contrary to other scheduling problems, this property does not hold for our problem. If we move job 6 scheduled after the critical job 4 to the second position, we get the production sequence $(1, 6, 2, 3, 4, 5)$. The corresponding optimal transportation sequence found by the barrier algorithm is $(\{1, 6\}, \{2, 4\}, \{3, 5\})$ resulting in a smaller objective value of $C^q_{\max} = 24$ (cf. the lower schedule in Figure 6).

This example shows that unlike the classical job-shop problem, moving new jobs in front of a critical job can reduce the completion time of the critical job. There are other examples which show that changing the order of jobs not belonging to a critical path may also decrease the makespan value. □

Appendix B

In Tables 5 to 10 the individual results for all lower bounds are summarized. In each row we have fixed values n, τ, and b, we varied θ and δ, i.e. always the average over $90 = 10 \cdot 3 \cdot 3$ instances is taken. For each lower bound LB_v the relative deviation $\Delta_v = \frac{LB - LB_v}{LB}$ (in percent) and the number of occurences $LB_v = LB$ (from 90) are reported. Additionally, we state the deviation $\Delta^{UB^0}_{LB} = \frac{UB^0 - LB}{LB}$ (in percent) from the best upper bound UB^0 of the four initial heuristic solutions.

n	τ	b	$\Delta_{LB}^{UB^0}$	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}	
50	40	5	0.81	0.6	40	15.2	0	19.4	0	15.2	0	22.1	17	0.7	50
50	40	10	1.90	8.9	18	2.4	30	27.2	17	2.4	30	11.7	54	7.5	13
50	40	15	1.61	14.5	14	2.3	30	28.3	19	2.3	30	12.2	53	13.6	4
50	100	5	0.42	0.3	38	49.4	0	6.8	0	49.3	0	47.1	0	0.6	54
50	100	10	1.66	1.1	36	22.2	0	19.1	0	22.1	0	18.7	11	0.7	54
50	100	15	1.22	1.4	29	12.6	0	27.7	0	12.6	0	9.1	33	0.8	54
50	500	5	0.11	0.1	35	78.6	0	1.9	0	78.6	0	73.4	0	0.1	58
50	500	10	0.39	0.2	43	60.0	0	6.3	0	60.0	0	50.3	0	0.5	49
50	500	15	0.56	0.4	41	49.7	0	6.7	0	49.7	0	37.7	0	0.8	53
100	40	5	0.40	0.3	39	15.3	0	18.4	0	15.3	0	25.7	9	0.3	50
100	40	10	1.04	9.2	23	1.3	30	28.9	14	1.3	30	15.8	52	8.9	2
100	40	15	1.03	15.9	19	1.2	31	27.7	16	1.2	31	15.7	53	15.2	11
100	100	5	0.19	0.1	36	52.9	0	4.8	0	52.9	0	54.4	0	0.3	55
100	100	10	0.89	0.6	39	23.9	0	16.1	0	23.9	0	27.3	10	0.5	55
100	100	15	1.23	0.9	42	10.2	0	26.0	0	10.2	0	15.4	18	0.5	48
100	500	5	0.05	0.0	35	84.8	0	1.2	0	84.8	0	81.4	0	0.1	55
100	500	10	0.21	0.1	36	70.8	0	3.8	0	70.8	0	64.0	0	0.3	56
100	500	15	0.34	0.2	38	59.5	0	6.0	0	59.5	0	50.2	0	0.5	54
200	40	5	0.18	0.2	32	16.1	2	18.6	0	16.1	2	28.1	15	0.1	55
200	40	10	0.70	8.9	25	0.6	31	28.3	10	0.6	31	17.6	55	8.6	7
200	40	15	0.66	16.2	18	0.6	30	28.5	10	0.6	30	17.7	58	16.0	7
200	100	5	0.10	0.1	35	54.3	0	3.9	0	54.3	0	57.9	0	0.2	57
200	100	10	0.46	0.3	36	24.8	0	14.3	0	24.8	0	31.8	9	0.2	53
200	100	15	0.57	0.5	36	9.4	0	24.5	0	9.4	0	19.7	19	0.3	54
200	500	5	0.03	0.0	36	87.8	0	0.8	0	87.8	0	86.5	0	0.0	56
200	500	10	0.11	0.1	41	76.2	0	2.4	0	76.2	0	73.8	0	0.1	51
200	500	15	0.19	0.1	42	66.2	0	2.9	0	66.2	0	62.8	0	0.3	48
			0.63	3.0	902	35.1	184	14.8	86	35.1	184	38.1	466	2.9	1163

Table 5: Lower bounds for $m = 1$, $p_j \in [1, 10]$

n	τ	b	$\Delta_{LB}^{UB^0}$	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}	
50	40	5	1.04	9.4	38	0.2	67	25.9	20	0.2	69	17.3	15	10.3	1
50	40	10	1.03	17.2	38	0.2	69	25.7	17	0.2	70	16.8	18	18.4	0
50	40	15	0.92	19.3	37	0.1	66	26.8	19	0.1	69	17.3	16	21.2	2
50	100	5	1.63	0.4	47	18.9	4	16.6	2	18.8	4	27.8	4	1.2	43
50	100	10	2.58	2.7	59	2.1	28	26.5	12	2.1	28	14.4	2	4.8	7
50	100	15	2.27	8.9	55	1.7	38	26.7	14	1.6	38	14.0	3	11.6	0
50	500	5	0.30	0.1	38	59.3	0	5.1	0	59.3	0	60.0	0	0.2	54
50	500	10	0.62	0.3	44	36.9	0	9.5	0	36.8	0	37.8	0	0.6	46
50	500	15	1.30	0.6	49	20.2	0	18.1	0	20.2	0	21.4	0	2.0	41
100	40	5	0.61	9.7	38	0.1	73	27.3	12	0.1	74	18.6	20	10.1	0
100	40	10	0.55	19.4	37	0.1	64	28.1	14	0.1	67	18.6	13	19.7	2
100	40	15	0.56	21.8	36	0.1	66	26.9	16	0.1	71	18.4	15	22.6	0
100	100	5	0.72	0.2	43	22.3	3	14.5	1	22.3	3	32.1	3	0.7	49
100	100	10	1.80	4.1	56	0.9	37	27.0	13	0.9	37	16.8	3	5.0	4
100	100	15	1.55	10.1	53	0.9	39	27.3	12	0.9	39	16.6	2	11.7	0
100	500	5	0.14	0.1	36	70.8	0	3.4	0	70.8	0	72.0	0	0.2	54
100	500	10	0.66	0.2	38	46.7	0	8.7	0	46.7	0	49.1	0	0.5	54
100	500	15	0.58	0.3	40	33.6	0	10.1	0	33.6	0	36.5	0	0.7	52
200	40	5	0.33	9.3	31	0.0	72	28.0	13	0.0	78	19.4	17	9.6	2
200	40	10	0.36	19.8	36	0.0	74	27.9	10	0.0	77	19.0	22	20.0	1
200	40	15	0.35	22.9	38	0.0	74	28.2	10	0.0	75	19.1	22	23.4	1
200	100	5	0.33	0.1	41	23.6	2	13.6	0	23.6	2	34.2	2	0.3	51
200	100	10	1.08	3.4	51	0.4	40	28.0	11	0.4	40	18.3	1	3.8	3
200	100	15	0.89	11.6	50	0.4	43	28.3	9	0.4	43	18.1	3	12.3	0
200	500	5	0.07	0.0	37	76.1	0	2.2	0	76.1	0	77.9	0	0.1	55
200	500	10	0.31	0.1	40	55.4	0	6.0	0	55.4	0	58.9	0	0.3	52
200	500	15	0.51	0.2	41	39.2	0	8.9	0	39.2	0	44.3	0	0.5	50
			0.85	7.1	1147	18.9	859	19.5	205	18.9	884	30.2	181	7.8	624

Table 6: Lower bounds for $m = 2$, $p_j \in [1, 10]$

n	τ	b	$\Delta_{LB}^{UB^0}$	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}	
50	40	5	0.09	19.8	26	0.0	84	25.8	20	0.0	90	17.1	28	21.2	3
50	40	10	0.07	20.2	29	0.0	85	25.6	17	0.0	90	16.7	32	21.8	0
50	40	15	0.06	20.8	25	0.0	83	26.8	19	0.0	90	17.2	27	22.5	2
50	100	5	0.46	10.2	42	0.0	82	25.6	15	0.0	86	12.8	38	11.5	2
50	100	10	0.37	14.0	38	0.0	79	25.4	13	0.0	86	12.5	34	15.5	3
50	100	15	0.39	14.2	41	0.0	84	25.7	14	0.0	87	12.3	37	16.9	4
50	500	5	1.28	0.3	51	19.0	0	15.3	6	19.0	0	20.5	0	1.3	36
50	500	10	1.59	2.1	53	1.4	30	25.7	5	1.4	32	3.3	15	3.8	22
50	500	15	1.02	1.8	69	1.0	44	25.8	6	1.0	45	2.7	25	5.4	1
100	40	5	0.04	21.8	30	0.0	88	27.3	12	0.0	90	18.6	33	22.3	2
100	40	10	0.05	22.7	29	0.0	83	28.1	14	0.0	90	18.5	29	23.6	2
100	40	15	0.06	22.9	29	0.0	81	26.9	16	0.0	90	18.4	29	23.9	1
100	100	5	0.29	9.2	44	0.0	80	26.8	10	0.0	83	15.8	37	10.0	1
100	100	10	0.24	17.5	39	0.0	84	26.6	13	0.0	85	15.9	36	18.5	2
100	100	15	0.30	18.3	41	0.0	81	26.8	12	0.0	85	15.8	36	19.9	0
100	500	5	0.81	0.1	40	36.5	0	10.3	1	36.5	0	39.5	0	0.4	50
100	500	10	3.23	0.5	51	7.6	1	21.3	10	7.6	1	12.9	1	1.7	30
100	500	15	1.96	0.4	54	4.0	4	25.4	1	4.0	4	9.3	3	2.0	35
200	40	5	0.03	22.4	26	0.0	83	28.0	13	0.0	90	19.3	27	22.7	1
200	40	10	0.03	23.6	27	0.0	84	27.9	10	0.0	90	19.0	29	24.1	2
200	40	15	0.04	23.8	31	0.0	87	28.2	10	0.0	90	19.1	33	24.3	3
200	100	5	0.19	9.4	34	0.0	84	27.5	9	0.0	89	17.9	33	9.8	2
200	100	10	0.15	19.0	33	0.0	81	27.8	11	0.0	87	17.8	31	19.3	0
200	100	15	0.17	21.2	41	0.0	84	28.1	9	0.0	85	17.7	36	22.1	1
200	500	5	0.38	0.1	38	45.9	0	7.6	0	45.9	0	50.3	0	0.1	52
200	500	10	1.85	0.3	45	15.7	3	17.9	4	15.7	3	23.7	3	0.7	43
200	500	15	2.74	0.3	56	3.2	7	25.0	3	3.2	7	14.0	4	1.4	28
			0.66	12.5	1062	5.0	1586	24.4	273	5.0	1675	17.7	636	13.6	328

Table 7: Lower bounds for $m = 5$, $p_j \in [1, 10]$

n	τ	b	$\Delta_{LB}^{UB^0}$	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}	
50	40	5	0.01	75.7	0	0.0	90	0.2	77	0.0	90	69.7	0	75.9	0
50	40	10	0.04	78.3	0	0.0	90	0.3	71	0.0	90	69.4	0	78.5	0
50	40	15	0.02	79.5	0	0.0	90	0.2	76	0.0	90	70.3	0	79.8	0
50	100	5	0.45	55.7	0	0.0	90	0.3	71	0.0	90	65.3	0	55.0	0
50	100	10	0.52	71.0	0	0.0	90	0.2	77	0.0	90	65.6	0	70.3	0
50	100	15	0.55	73.2	0	0.0	90	0.2	78	0.0	90	65.2	0	72.9	0
50	500	5	1.50	0.6	4	41.2	0	2.1	0	41.2	0	64.8	0	0.0	90
50	500	10	5.56	8.9	0	0.4	79	0.9	63	0.4	79	40.1	0	6.1	11
50	500	15	4.94	26.5	0	0.0	90	0.4	68	0.0	90	39.4	0	23.7	0
100	40	5	0.02	77.7	0	0.0	90	0.3	66	0.0	90	73.3	0	77.7	0
100	40	10	0.01	80.8	0	0.0	90	0.1	72	0.0	90	73.1	0	80.9	0
100	40	15	0.02	81.5	0	0.0	90	0.3	64	0.0	90	73.1	0	81.6	0
100	100	5	0.28	57.5	0	0.0	90	0.4	60	0.0	90	70.8	0	75.2	0
100	100	10	0.23	73.7	0	0.0	90	0.1	72	0.0	90	70.9	0	73.5	0
100	100	15	0.30	77.0	0	0.0	90	0.3	69	0.0	90	70.8	0	76.8	0
100	500	5	0.78	0.3	1	45.8	0	1.2	0	45.8	0	76.2	0	0.0	90
100	500	10	3.29	5.7	0	0.8	67	1.0	47	0.8	67	57.2	0	4.3	23
100	500	15	2.80	31.4	0	0.0	90	0.4	62	0.0	90	57.0	0	29.6	0
200	40	5	0.01	78.5	0	0.0	90	0.4	60	0.0	90	74.9	0	78.5	0
200	40	10	0.01	82.0	0	0.0	90	0.3	61	0.0	90	74.9	0	82.0	0
200	40	15	0.02	82.7	0	0.0	90	0.5	57	0.0	90	74.7	0	82.7	0
200	100	5	0.16	58.5	0	0.0	90	0.3	53	0.0	90	73.6	0	58.4	0
200	100	10	0.15	75.0	0	0.0	90	0.3	53	0.0	90	73.4	0	74.8	0
200	100	15	0.14	78.8	0	0.0	90	0.3	61	0.0	90	73.5	0	78.8	0
200	500	5	0.42	0.1	2	47.6	0	0.7	0	47.6	0	82.2	0	0.0	90
200	500	10	1.92	3.8	0	0.5	69	0.7	42	0.5	69	66.5	0	3.1	21
200	500	15	1.66	31.4	0	0.0	90	0.4	57	0.0	90	66.2	0	30.6	0
			0.96	53.6	7	5.1	2105	0.5	1537	5.1	2105	67.8	0	53.1	325

Table 8: Lower bounds for $m = 1$, $p_j \in [1, 100]$

n	τ	b	$\Delta_{LB}^{UB^0}$	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}	
50	40	5	0.00	79.0	0	0.0	90	0.2	77	0.0	90	71.2	0	79.5	0
50	40	10	0.00	79.3	0	0.0	90	0.3	71	0.0	90	70.9	0	79.9	0
50	40	15	0.00	80.1	0	0.0	90	0.2	76	0.0	90	71.9	0	80.7	0
50	100	5	0.01	71.6	0	0.0	90	0.3	71	0.0	90	69.1	0	71.8	0
50	100	10	0.04	75.9	0	0.0	90	0.2	77	0.0	90	69.4	0	76.9	0
50	100	15	0.02	76.1	0	0.0	90	0.2	78	0.0	90	69.1	0	77.1	0
50	500	5	3.02	10.2	1	0.5	79	0.8	62	0.5	80	56.4	0	9.2	9
50	500	10	1.18	41.6	0	0.0	90	0.5	67	0.0	90	56.2	0	40.8	0
50	500	15	1.18	52.7	0	0.0	90	0.4	68	0.0	90	56.0	0	51.9	0
100	40	5	0.00	81.2	0	0.0	90	0.3	66	0.0	90	74.1	0	81.4	0
100	40	10	0.00	82.1	0	0.0	90	0.1	72	0.0	90	73.8	0	82.4	0
100	40	15	0.00	82.1	0	0.0	90	0.3	64	0.0	90	73.9	0	82.5	0
100	100	5	0.01	74.1	0	0.0	90	0.4	60	0.0	90	72.8	0	74.3	0
100	100	10	0.01	79.1	0	0.0	90	0.1	72	0.0	90	72.8	0	79.4	0
100	100	15	0.01	79.7	0	0.0	90	0.3	69	0.0	90	72.7	0	80.2	0
100	500	5	2.27	5.6	0	0.8	70	0.9	49	0.8	70	65.4	0	5.0	21
100	500	10	0.80	47.7	0	0.0	90	0.4	60	0.0	90	65.7	0	46.4	0
100	500	15	0.73	58.1	0	0.0	90	0.4	62	0.0	90	65.9	0	57.5	0
200	40	5	0.00	82.1	0	0.0	90	0.4	60	0.0	90	75.2	0	82.2	0
200	40	10	0.00	83.3	0	0.0	90	0.3	61	0.0	90	75.3	0	83.4	0
200	40	15	0.00	83.4	0	0.0	90	0.5	57	0.0	90	75.1	0	83.6	0
200	100	5	0.01	75.5	0	0.0	90	0.3	53	0.0	90	74.6	0	75.6	0
200	100	10	0.01	80.5	0	0.0	90	0.3	53	0.0	90	74.3	0	80.7	0
200	100	15	0.01	81.6	0	0.0	90	0.3	61	0.0	90	74.5	0	81.8	0
200	500	5	1.33	3.3	1	0.8	63	0.7	45	0.8	63	70.9	0	3.1	26
200	500	10	0.45	48.8	0	0.0	90	0.4	51	0.0	90	71.0	0	48.2	0
200	500	15	0.44	62.2	0	0.0	90	0.4	57	0.0	90	70.9	0	61.7	0
			0.43	65.1	2	0.1	2372	0.4	1719	0.1	2373	70.0	0	65.1	56

Table 9: Lower bounds for $m = 2$, $p_j \in [1, 100]$

n	τ	b	$\Delta_{LB}^{UB^0}$	LB_1		LB_2		LB_3		LB_{T1}		LB_{T2}		LB_{T3}	
50	40	5	0.00	79.7	0	0.0	90	0.2	77	0.0	90	71.2	0.0	80.2	0
50	40	10	0.00	79.4	0	0.0	90	0.3	71	0.0	90	70.9	0.0	79.9	0
50	40	15	0.00	80.3	0	0.0	90	0.2	76	0.0	90	71.9	0.0	80.7	0
50	100	5	0.00	76.7	0	0.0	90	0.3	71	0.0	90	69.1	0.0	77.4	0
50	100	10	0.00	77.3	0	0.0	90	0.2	77	0.0	90	69.4	0.0	78.2	0
50	100	15	0.00	76.9	0	0.0	90	0.2	78	0.0	90	69.1	0.0	77.6	0
50	500	5	0.02	53.6	0	0.0	89	0.4	68	0.0	90	56.1	0.0	53.2	0
50	500	10	0.02	60.8	0	0.0	90	0.5	67	0.0	90	56.2	0.0	60.9	0
50	500	15	0.02	60.6	0	0.0	90	0.4	68	0.0	90	56.0	0.0	62.5	0
100	40	5	0.00	82.4	0	0.0	90	0.3	66	0.0	90	74.1	0.0	82.7	0
100	40	10	0.00	82.3	0	0.0	90	0.1	72	0.0	90	73.8	0.0	82.5	0
100	40	15	0.00	82.2	0	0.0	90	0.3	64	0.0	90	73.9	0.0	82.5	0
100	100	5	0.00	79.9	0	0.0	90	0.4	60	0.0	90	72.8	0.0	80.2	0
100	100	10	0.00	80.9	0	0.0	90	0.1	72	0.0	90	72.8	0.0	81.3	0
100	100	15	0.00	80.7	0	0.0	90	0.3	69	0.0	90	72.7	0.0	81.1	0
100	500	5	0.03	56.0	0	0.0	90	0.4	57	0.0	90	65.0	0.0	55.7	0
100	500	10	0.02	69.2	0	0.0	90	0.4	60	0.0	90	65.7	0.0	69.3	0
100	500	15	0.02	70.3	0	0.0	90	0.4	62	0.0	90	65.9	0.0	71.2	0
200	40	5	0.00	83.4	0	0.0	90	0.4	60	0.0	90	75.2	0.0	83.6	0
200	40	10	0.00	83.6	0	0.0	90	0.3	61	0.0	90	75.3	0.0	83.7	0
200	40	15	0.00	83.5	0	0.0	90	0.5	57	0.0	90	75.1	0.0	83.6	0
200	100	5	0.00	81.5	0	0.0	90	0.3	53	0.0	90	74.6	0.0	81.7	0
200	100	10	0.00	82.4	0	0.0	90	0.3	53	0.0	90	74.3	0.0	82.6	0
200	100	15	0.00	82.7	0	0.0	90	0.3	61	0.0	90	74.5	0.0	82.9	0
200	500	5	0.01	57.4	0	0.0	90	0.4	59	0.0	90	70.6	0.0	57.3	0
200	500	10	0.02	72.9	0	0.0	90	0.4	51	0.0	90	71.0	0.0	72.9	0
200	500	15	0.01	75.7	0	0.0	90	0.4	57	0.0	90	70.9	0.0	76.1	0
			0.01	75.3	0	0.0	2429	0.3	1747	0.0	2430	69.9	0	75.6	0

Table 10: Lower bounds for $m = 5$, $p_j \in [1, 100]$

References

[1] J.H. Ahmadi, R.H. Ahmadi, S. Dasu, and C.S. Tang. Batching and scheduling jobs on batch and discrete processors. *Operations Research*, 40(4):750–763, 1992.

[2] P. Brucker and S. Knust. *Complex Scheduling*. Springer, 2012.

[3] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176, 1989.

[4] J. Carlier and E. Pinson. Adjustment of head and tails for the job-shop problem. *European Journal of Operational Research*, 78:146–161, 1994.

[5] Z.-L. Chen. Integrated production and distribution operations: Taxonomy, models, and review. *In D. Semchi-Levi and S.D. Wu and Z.-J. Shen (Eds.), Handbook of Quantitative Supply Chain Analysis: Modelling in the e-Business Era, Kluwer*, pages 711–747, 2004.

[6] Z.-L. Chen. Integrated production and outbound distribution scheduling: review and extensions. *Operations Research*, 58:130–148, 2010.

[7] A. Condotta, S. Knust, and N.V. Shakhlevich. Parallel batch scheduling of equal-length jobs with release and due dates. *Journal of Scheduling*, 13:463–477, 2010.

[8] Ş.S. Erengüc, N.C. Simpson, and A.J Vakharia. Integrated production/distributed planning in supply chains: An invited review. *European Journal of Operational Research*, 115:219–236, 1999.

[9] H.N. Geismar, G. Laporte, L. Lei, and C. Sriskandarajah. The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing*, 20:21–33, 2008.

[10] J.N.D. Gupta and E.A. Tunc. Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research*, 29:1489–1502, 1991.

[11] L Hall and D. Shmoys. Approximation schemes for constrained scheduling problems. *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 134–140, 1989.

[12] N.G. Hall and C.N. Potts. Supply chain scheduling: batching and delivery. *Operations Research*, 51:566–584, 2003.

[13] N.G. Hall and C.N. Potts. The coordination of scheduling with batch deliveries. *Annals of Operations Research*, 135:41–64, 2005.

[14] W.A. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21:177–185, 1974.

[15] Y. Ikura and M. Gimple. Efficient scheduling algorithms for a single batch processing machine. *Operations Research Letters*, 5:61–65, 1986.

[16] C.-Y. Lee and Z.-L. Chen. Machine scheduling with transportation considerations. *Journal of Scheduling*, 4:3–24, 2001.

[17] D. Lei and T. Wang. An effective neighbourhood search algorithm for scheduling a flow shop of batch processing machines. *Computers and Industrial Engineering (to appear).*

[18] D.M. Lei and X.P. Guo. Variable neighborhood search for minimizing tardiness objectives on flow shop with batch processing machines. *Journal of Production Research*, 49:519–529, 2011.

[19] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problem. *Annals of Discrete Mathematics (North-Holland, Amsterdam)*, 1:343–362, 1997.

[20] L.M. Liao and C.J. Huang. Tabu search heuristic for two-machine flow shop with batch-processing machines. *Computers and Industrial Engineering (to appear).*

[21] P. Liu and X. Lu. An improved approximation algorithm for single machine scheduling with job delivery. *Theoretical Computer Science*, 412:270–274, 2011.

[22] L. Lu, J. Yuan, and L. Zhang. Single machine scheduling with release dates and job delivery to minimize the makespan. *Theoretical Computer Science*, 393:102–108, 2008.

[23] P.K. Manjeshwar, P. Damodaran, and K. Srihari. Minimizing makespan in a flow shop with two batch-processing machines using simulated annealing. *Robotics and Computer-Aided Manufacturing*, 25:667–679, 2009.

[24] H.S. Mirsanei, B. Karimi, and F. Jolai. Flow shop scheduling with two batch processing machines and nonidentical job sizes. *International Journal of Advanced Manufacturing Technology*, 45:553–572, 2009.

[25] C.N. Potts. Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Operations Research*, 28:690–710, 1980.

[26] C.N. Potts and L.N. van Wassenhove. Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. *Journal of the Operational Research Society*, 43:395–406, 1992.

[27] C.S. Sung and Y.H. Kim. Minimizing makespan in a two-machine flowshop with dynamic arrivals allowed. *Computers and Operations Research*, 29(3):275–294, 2002.

[28] X. Wang and T.C.E. Cheng. Production scheduling with supply and delivery considerations to minimize the makespan. *European Journal of Operational Research*, 194:743–752, 2009.

[29] S.H. Zegordi, I.N. Kamal Abadi, and M.A. Beheshti Nia. A novel genetic algorithm for solving production and transportation scheduling in a two-stage supply chain. *Computers and Industrial Engineering*, 58:373–381, 2010.