KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Business and Economics

Order acceptance and scheduling in a single-machine environment: exact and heuristic algorithms

F. Talla Nobibon, J. Herbots and R. Leus

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0904

Order acceptance and scheduling in a single-machine environment: exact and heuristic algorithms

Fabrice Talla Nobibon*, Jade Herbots and Roel Leus

Department of Decision Sciences and Information Management
Faculty of Business and Economics
Katholieke Universiteit Leuven, Belgium

In this paper, we develop exact and heuristic algorithms for the order acceptance and scheduling problem in a single-machine environment. We consider the case where a pool consisting of firm planned orders as well as potential orders is available from which an over-demanded company can select. The capacity available for processing the accepted orders is limited and orders are characterized by known processing times, delivery dates, revenues and the weight representing a penalty per unit-time delay beyond the delivery date promised to the customer. We prove the non-approximability of the problem and give two linear formulations that we solve with CPLEX. We devise two exact branch-and-bound procedures able to solve problem instances of practical dimensions. For the solution of large instances, we propose six heuristics. We provide a comparison and comments on the efficiency and quality of the results obtained using both the exact and heuristic algorithms, including the solution of the linear formulations using CPLEX.

Keywords: order acceptance, scheduling, single machine, branch-and-bound, heuristics, firm planned orders.

---

1 Introduction

Many organizations give no formal consideration to either order acceptance or rejection. Instead, an order-entry process is operated that tacitly accepts *all* orders. In today's competitive manufacturing environment, an organization must respect order deadlines agreed to with customers, but order acceptance often takes place without consideration of the effect on the planning of the other jobs in the order portfolio. This is often the consequence of the functional separation between the order acceptance decision, which is made by the sales department, and capacity planning, which usually lies in the hands of the production department. These two departments generally have conflicting objectives: in order to boost sales, sales departments try to accept as many projects as possible, while production attempts to live up to promised delivery dates. This divergence of interests can result in considerable delays, violated due dates and/or excessive use of highly expensive non-regular capacity such as overtime and temporary labor. It is therefore essential that project selection and planning be integrated [17, 19, 42].

Order acceptance has gained increasing attention over the past decade. As clearly described by Rom and Slotnick [33], this decision is intricate because it should strike a balance between the revenue obtained from an accepted order on the one hand, and

*Corresponding author. E-mail: Fabrice.TallaNobibon@econ.kuleuven.be. Tel. +32 16 32 69 60. Fax +32 16 32 66 24.

the (opportunity) costs of capacity as well as potential tardiness penalties on the other hand. This paper examines the simultaneous order acceptance and sequencing decision. Order acceptance refers to the selection decision an over-demanded company has to make; sequencing determines the order in which jobs are executed. More specifically, the focus of this paper is on the order acceptance and scheduling decisions of an organization that has a pool consisting of firm planned orders as well as potential orders to choose from, while orders have known processing times, delivery dates and revenues. The capacity available for processing the accepted orders is limited. In addition, the urgency of individual orders may be emphasized by the importance of the client: even though some orders by themselves may not be very lucrative, they may still have strategic value for future business with the client. An order delivered past the agreed-upon delivery date incurs a penalty that is proportional to the overrun; there is no reward nor penalty, however, for order delivery before the promised date.

The selection of orders from a pool containing both firm planned orders as well as potential orders, which is the problem studied in this article, is a generalization of two boundary problems. The first of these two is the order acceptance and sequencing problem with weighted tardiness penalties studied by Slotnick and Morton [37]. Here, the pool of firm planned orders is empty and all tasks are eligible for rejection. The second boundary problem is the pure sequencing problem with weighted tardiness as solved by e.g. Potts and Van Wassenhove [31], where the pool of orders coincides with the pool of firm planned orders.

In practice, the sales department needs to have a clear insight into the relationship between the available resource capacities, possible workloads and the resulting manufacturing lead times in order to be able to quote realistic customer-order due dates. The lead time of an order can be defined as the difference between the promised due date of an order and its arrival time. Hence, quoting a lead time is equivalent to quoting a due date. Recent lead-time-related research has developed in multiple directions, including lead-time reduction [20, 38], predicting manufacturing lead times, the relationship between lead times and other elements of manufacturing such as lot sizes and inventory [15, 22, 29], and due-date management. The goal of due-date management is to develop a combined due-date setting and sequencing policy. This means that, in contrast with most of the scheduling literature, due dates are set endogenously; surveys in this area include [3, 8, 23].

In this article, we adopt an operational scheduling viewpoint and consider the due dates to be exogenous: they are the outcome of negotiations with the client and are fixed before a detailed workplan is to be developed. We examine order acceptance and planning on a specialized scarce resource, which is represented as a single machine and which constitutes the bottleneck of the manufacturing environment. We devise various exact and heuristic algorithms to solve the problem of deciding which non-planned orders to retain and which to reject for profit maximization, and we simultaneously determine the processing order of the accepted jobs. By means of computational experiments on a number of benchmark datasets, we show that we improve upon the performance of previously published exact algorithms and show the good behavior of our heuristics.

The contributions of this article are the following: we propose a new model for job selection and scheduling that builds a bridge between two well-known problems, namely single-machine scheduling to minimize total weighted tardiness and job selection and

sequencing with total weighted tardiness penalties. We prove the non-approximability of the problem and present two mixed-integer linear formulations. We develop two exact branch-and-bound algorithms able to solve medium-size instances and six heuristics for solving large instances.

The remainder of this article is structured as follows. First, we survey the existing literature in Section 2. In Section 3, we provide a formal description of the problem that we wish to solve. Section 4 contains the proof of the non-approximability result and the two linear formulations. Section 5 is devoted to the development of exact algorithms. In Section 6, we propose a number of heuristics. We comment the results of the computational experiments in Section 7 and we conclude in Section 8.

2 Literature review

Excellent literature surveys on the topic of order acceptance and scheduling are provided by Rom and Slotnick [33], Guerrero and Kern [17], Keskinocak and Tayur [23] and Roundy et al. [34]. The objective of this section is therefore not to provide an exhaustive listing of the existing literature, but rather to survey the different perspectives that have been developed on the topic by different researchers, with a particular focus on the work in single-machine environments. As mentioned before, the pure sequencing problem with weighted tardiness has been solved by Potts and Van Wassenhove [31]. This review will therefore only discuss the most closely related articles on the subject of order acceptance and scheduling. First, we discuss the objective of minimizing the total weighted tardiness; subsequently, we briefly consider alternative objective functions and on-line decision making.

Slotnick and Morton study the single-machine job selection and sequencing problem with deterministic job processing times and job rewards; their objective is to maximize the rewards in case of lateness [36] and tardiness [37] penalties. A pseudo-polynomial-time algorithm to solve the former problem was developed by Ghosh [14]. The problem was extended in [27] to multiple periods for the case where rejecting a job will result in the loss of all future jobs from that customer. An exact approach for solving the single-period weighted tardiness problem was developed in [37]. Since the proposed algorithm can only deal with very moderately sized problem instances (at most ten jobs), suboptimal algorithms were also presented. Other heuristics include genetic algorithms [33] and greedy algorithms for selection and ordering problems [2]. Yang and Geunes [40] consider an extension of the problem where job processing times are reducible at a cost and every job has a release time. In their paper, Yang and Geunes develop an optimal algorithm for maximizing schedule profit for a given sequence of jobs, along with heuristics to solve the entire problem. Sengupta [35] studies a special case of the problem where the weight of each job is one. Both an MIP formulation as well as a branch-and-bound algorithm are described in the very brief article of Yugma [41]; he reports results in reasonable time via MIP for up to 15 jobs, and up to 30 using branch-and-bound. Finally, Bilginturk et al. [6] look into a generalization of this problem with release times, deadlines and sequence-dependent setup times; they conclude that an MIP solution is not attainable for problem sizes exceeding ten jobs and resort to simulated annealing.

Other objective functions for the selection and sequencing problem have been consid-

ered in literature. Engels et al. [11] seek to minimize the sum of the weighted completion times of the scheduled jobs and the total rejection penalty of the rejected jobs. A related objective function is used in [28] for the unbounded parallel batch machine scheduling problem with release dates. A parallel batch machine can process a number of jobs simultaneously, so that the makespan is the same for all jobs in a batch. A different but related problem is the job-interval selection problem (JISP) where a job is determined by a set of intervals. In [9], some special cases of the JISP are considered. The paper develops algorithms that aim to maximize the number of jobs scheduled between their release dates and deadlines. Another objective function was examined by Gupta et al. [18], who develop an efficient polynomial-time dynamic-programming method that solves the project selection and sequencing problem (a fixed number of projects is selected from a set) while maximizing the net present value of the total return. De et al. [10] study the sequencing problem and minimize the weighted number of tardy jobs. In [10], a project-dependent cost is charged when starting the execution of a project so that it becomes a selection problem. It is assumed that a revenue is reaped at the completion of a project. The goal is to maximize the expected rewards of a selection of jobs with random processing times and random deadlines. Baptiste et al. [4] study a related problem with unit durations inspired by a practical case of a satellite launcher, and describe polynomial-time dynamic-programming recursions for some special cases.

The on-line problem, in which projects arrive dynamically over time and need to be selected or rejected upon arrival, has been studied by several authors for a broad range of objective functions. For dynamic arrivals with a single resource constraint, Kleywegt and Papastavrou [24] studied a dynamic and stochastic knapsack problem, where the size of the knapsack represents the deterministic available resource quantity. Each arrival demands some amount of the resource, and a reward (unknown prior to the job arrival) is received upon acceptance. They provide an optimal acceptance policy that maximizes expected profits. For the batch process industry, Ivănescu et al. [21] develop policies that focus on delivery reliability, while keeping utilization rates up. Epstein et al. [12] consider a single-machine on-line job selection and scheduling problem with job-dependent rejection penalties. Their algorithm aims to minimize the total completion time of accepted jobs plus job rejection penalties.

3 Problem statement

A set of jobs $N = \{1, 2, \ldots, n\}$ with durations p_i ($i \in N$) is to be scheduled on a single machine; all jobs are available for processing at the beginning of the planning period. Each job i has a due date d_i and a revenue Q_i; the weight w_i represents a penalty per unit-time delay beyond d_i in the delivery to the customer. The pool N of jobs consists of two disjoint subsets F and $\bar{F}$ ($N = F \cup \bar{F}$ and $F \cap \bar{F} = \emptyset$), for which F comprises the firm planned orders and its complement $\bar{F}$ contains the 'optional' jobs (the ones that can still be rejected). Our objective is to maximize total net profit, that is, the sum of the revenues of the selected jobs minus applicable total weighted tardiness penalties incurred for those jobs.

There are two decisions to be made: which jobs in $\bar{F}$ to accept, and in which order to process the selected subset. We call M the set of jobs selected for processing, so

$F \subseteq M \subseteq N$. If we let $m = |M|$, the sequencing decisions can be represented by a bijection $\pi : \{1, 2, \ldots, m\} \mapsto M$, where $\pi(t)$ is the index of the job in position t in the sequence. Each such bijection is in one-to-one correspondence with a total order on set M. The objective is thus

$$\max_{M,\pi} \quad \sum_{t=1}^{m} Q_{\pi(t)} - w_{\pi(t)}(C_{\pi(t)} - d_{\pi(t)})^+, \tag{1}$$

where C_i is the completion time of job i, i.e. $C_i = \sum_{t=1}^{\pi^{-1}(i)} p_{\pi(t)}$, and $s^+ = \max\{s, 0\}$. Formulation (1) also models a special case of scheduling with subcontracting options [7] where the subcontractor has an unlimited capacity. In the sequel, we refer to (1) as either the objective function or the problem formulation when no confusion can be made.

When $N = F$, the pool of firm planned orders is equal to the set of mandatory jobs and the problem reduces to the single-machine total weighted tardiness problem $1||\sum w_i T_i$. Although this problem is strongly NP-hard [25, 26], Potts and Van Wassenhove [31] have developed a branch-and-bound algorithm that efficiently solves problems with up to 50 jobs.

On the other hand, when $N = \bar{F}$, the problem is equivalent to the selection and sequencing problem discussed in [33, 37], which is akin to a number of other recently examined optimization problems, as discussed in the literature review. This problem contains the scheduling problem; it is therefore strongly NP-hard. The branch-and-bound procedure in [37] only solves relatively small problem instances (at most ten jobs) and is primarily used as a benchmark for evaluating the performance of heuristics.

4 Non-approximability and linear formulations

In this section, we show that it is unlikely that a constant-factor approximation algorithm can be developed for problem (1). We present two mixed-integer linear formulations of (1) that can be solved using the IP solver of CPLEX.

The next result shows that (1) is difficult to solve even approximately. We prove this non-approximability result by showing that any polynomial-time constant-factor approximation algorithm for solving (1) can be used to solve the following variant of Partition (see [13]; by adding $|A|$ dummy elements of size 0 to an instance of the usual Partition problem, one easily sees that this variant of Partition is as hard as the original problem):
INSTANCE: A finite set $A = \{1, 2, \ldots, 2q\}$ (where q is an integer greater than 0) with size $s(i) \in \mathbb{Z}^+$ for each $i \in A$, and $K = \frac{1}{2}\sum_{i \in A} s(i)$.
QUESTION: Does there exist a subset $A' \subset A$ with $|A| = q$ and $\sum_{i \in A'} s(i) = K$?

Theorem 1. *Unless $P = NP$, there is no polynomial-time algorithm that guarantees a constant-factor approximation for solving the problem (1).*

Proof: For a given arbitrary instance of Partition, consider the following polynomial time reduction to an instance of (1) with $n = 2q + 2$ jobs. The set of firm planned orders $F = \{2q + 1, 2q + 2\}$ and we let $\delta = 2K + 1$. The properties of each job are the following: for job $i = 1, \ldots, 2q$ we have a revenue $Q_i = \delta s(i)$, the processing time $p_i = \delta + s(i)$, the due date $d_i = q\delta + K + 2$ and the weight $w_i = s(i)(\delta + 1)$. For the

last two jobs, we have $Q_{2q+1} = Q_{2q+2} = \delta$, $p_{2q+1} = p_{2q+2} = 1$, $d_{2q+1} = d_{2q+2} = 1$ and $w_{2q+1} = w_{2q+2} = \delta(K+2)$. We prove that by solving this instance of (1) with a constant-factor approximation algorithm, we can infer the answer to the Partition instance.

If the instance of Partition is a YES instance then an appropriate set A' exists. By selecting the jobs in A' (jobs built from the elements in A') plus those in F and by scheduling them optimally (for example, schedule job $2q+1$ first, job $2q+2$ second and the jobs in A' in any order thereafter) we achieve an optimal profit of 0. Therefore, a constant-factor approximation algorithm will always provide an optimal solution, from which we can easily infer A'.

Conversely, if a constant-factor approximation algorithm leads to an objective value different from 0, we conclude that the instance of Partition is a NO instance. □

We next present two mixed-integer linear formulations for our job selection and sequencing problem. For the first formulation, we use the binary decision variable $y_i \in \{0,1\}$ $(i \in N)$, which takes the value 1 if job i is accepted and 0 otherwise. Notice that each job in F must be accepted. This constraint translates into:

$$y_i = 1, \quad \forall i \in F. \tag{2}$$

The second set of binary variables $x_{it} \in \{0,1\}$ $(i \in N,\, t \in \{1,\ldots,n\})$ is used to identify the position of the accepted jobs. The variable x_{it} is equal to 1 if job i is accepted and is the t^{th} job processed, and to 0 otherwise. Clearly, the following set of equality constraints holds:

$$y_i = \sum_{t=1}^{n} x_{it}, \quad i = 1,\ldots,n. \tag{3}$$

The set of constraints (3) allows to relax the integrality constraints on the variable y_i and to use $0 \le y_i \le 1$, $i = 1,\ldots,n$.

The capacity constraints entail that a given position can be attributed to at most one job; this is enforced by adding the following set of constraints.

$$\sum_{i=1}^{n} x_{it} \le 1, \quad t = 1,\ldots,n. \tag{4}$$

To linearize the objective function of (1), we introduce the binary variable $z_{ji} \in \{0,1\}$ $(i \neq j)$, which is equal to 1 if both jobs i and j are accepted and job j is executed before job i, otherwise z_{ji} takes the value 0. Since both jobs i and j must be accepted when $z_{ji} = 1$, we have

$$z_{ji} \le y_i, \quad z_{ji} \le y_j, \quad i,j = 1,\ldots,n,\ i \neq j. \tag{5}$$

We add the following set of constraints to enforce the fact that if job i is accepted and job j is processed before job i (meaning that job j is also accepted) then $z_{ji} = 1$:

$$\sum_{q<t} x_{jq} + \sum_{q \ge t} x_{iq} \le 1 + z_{ji}, \quad i,j,t = 1,\ldots,n,\ i \neq j,\ t \neq 1. \tag{6}$$

To complete our formulation, we need a real variable $T_i \geq 0$ representing the tardiness of job $i = 1, \ldots, n$, satisfying

$$T_i \geq \sum_{j=1}^{n} p_j z_{ji} + p_i y_i - d_i, \quad i = 1, \ldots, n. \tag{7}$$

The objective function is

$$\text{maximize} \qquad \sum_{i=1}^{n} \left(Q_i y_i - w_i T_i \right). \tag{8}$$

We refer to the mixed-integer linear formulation given by the objective function (8) and the constraints (2–7) as MIP. The following result shows that MIP is equivalent to (1).

Theorem 2. *From any solution to MIP, we can infer a solution to the problem (1) with the same objective value and vice versa.*

Proof: $\Rightarrow$) Suppose that there is a solution y_i, x_{it}, z_{ji} and T_i to MIP. Consider $M = \{i \in N : y_i = 1\}$; it holds that $F \subseteq M$. Each job $i \in M$ has a unique value t that represents its position, namely t for which $x_{it} = 1$. The function π that orders the jobs in M according to increasing position is a bijection. Choices M and π form a feasible solution to (1) and it is easy to see that this solution has the same objective value as y_i, x_{it}, z_{ji}, T_i.
$\Leftarrow$) On the other hand, suppose that we have a solution (M, π) to (1), then take $x_{it} = 1$ if $i \in M$ and $\pi(t) = i$, otherwise set $x_{it} = 0$. We can easily infer y_i, z_{ji} and T_i. Moreover, this solution is feasible and has the same objective value as (M, π). □

The formulation MIP can be strengthened by adding the following inequalities:

$$\text{(Cuts)} \qquad \sum_{i=1}^{n} x_{it+1} \leq \sum_{i=1}^{n} x_{it}, \quad t = 1, \ldots, n-1.$$

These inequalities enforce that there should be no empty position between the execution of two consecutive jobs. We refer to these inequalities as 'cuts', although a number of integer solutions to MIP are also eliminated.

The following result states that we can relax the integrality constraints on the variables z_{ji} without harm. Let MIP′ be the formulation MIP in which the domain $\{0, 1\}$ of the variables z_{ji} is replaced by $[0, 1]$.

Proposition 1. *In any feasible solution to MIP′, each* $z_{ji} \in \{0, 1\}$.

Proof: Consider a given optimal solution y_i, x_{it}, z_{ji} and T_i to MIP′ and suppose that there exists a variable z_{ji} with $0 < z_{ji} < 1$. Due to constraints (5), $y_i = y_j = 1$, and from (3), there are two distinct positions t_i, t_j with $t_i \neq t_j$ to which i, respectively j, are assigned. The constraints (6) then imply that either z_{ji} or z_{ij} equals 1, and the other variable will then be 0 due to constraint (7). □

The second linear formulation is a time-indexed formulation [5, 39]. It is based on a discretization of the time horizon $[0, T]$ into T one-unit time buckets, where $T = \sum_{i \in N} p_i$ and the bucket t is the time interval $[t-1, t]$. The decision variables are the binary quantities $x_{jt} \in \{0, 1\}$ with $j = 1, \ldots, n$ and $t = 1, \ldots, T - p_j + 1$, which equal 1 if job j is selected and its execution starts in bucket t, and 0 otherwise. We let $c_{jt} = \max\{t + p_j - d_j - 1; 0\}$, the tardiness of job j associated with the decision x_{jt}. The time-indexed formulation is given by:

$$\text{(TIF)} \quad \max \quad \sum_{j=1}^{n} \sum_{t=1}^{T-p_j+1} x_{jt} \left(Q_j - w_j c_{jt}\right) \tag{9}$$

$$\text{subject to} \quad \sum_{t=1}^{T-p_j+1} x_{jt} = 1 \qquad j \in F, \tag{10}$$

$$\sum_{t=1}^{T-p_j+1} x_{jt} \leq 1 \qquad j \in \bar{F}, \tag{11}$$

$$\sum_{j=1}^{n} \sum_{s=t-p_j+1}^{t} x_{jt} \leq 1 \qquad t = 1, \ldots, T, \tag{12}$$

$$x_{jt} \in \{0, 1\} \qquad j = 1, \ldots, n,\ t = 1, \ldots, T - p_j + 1. \tag{13}$$

The set of constraints (10) specifies that each job $j \in F$ is processed only once, the set of constraints (11) states that each job $j \in \bar{F}$ can be selected or not and if selected it is executed only once. The set of constraints (12) avoids the processing of more than one job at the same time.

5 Implicit-enumeration algorithms

In this section, we present two branch-and-bound (B&B) algorithms for solving our problem. These algorithms are inspired by the work of Slotnick and Morton [37]. The first B&B algorithm is hierarchical, in that it performs selection and scheduling separately. At each node of the branching tree, the B&B algorithm developed by Potts and Van Wassenhove [31] is used to schedule the set of selected jobs. The second B&B algorithm performs both selection and scheduling simultaneously. In what follows, these two B&B algorithms are called *two-phase* and *direct*, respectively.

We will use the same example instance to illustrate the working of the two algorithms. This instance has four jobs and the firm has planned to certainly accept the last job, so $F = \{4\}$. The properties of each job are given in Table 1.

5.1 Two-phase B&B algorithm

In what follows, we describe in detail each step of the two-phase B&B algorithm.

5.1.1 Removable set

We wish to distinguish a set $\bar{R}$ of jobs for which we are certain that they are part of each optimal solution, we call these jobs *non-removable*. By symbol R, we denote the set of

Job	1	2	3	4
Q_i	5	4	5	6
w_i	3	2	1	4
p_i	2	3	2	2
d_i	4	4	3	2

Table 1: Job properties for the example.

removable jobs: $R = N \setminus \bar{R}$. Slotnick and Morton [37] propose a procedure to identify R when $F = \emptyset$: given an optimal sequence of all the jobs and given a job j, if removing job j from the sequence decreases the net profit then $j \in \bar{R}$, otherwise $j \in R$. The following result is a generalization for arbitrary F.

Theorem 3. *For a given instance of problem (1) and a removable set A for the corresponding instance with $F = \emptyset$, the set R of removable jobs of our problem is given by $R = A \setminus F$.*

Proof: This follows from the reasoning given by Slotnick and Morton [37] for the total weighted lateness penalties and the fact that jobs in F must be selected. □

In the sequel, we denote the jobs in R by$J_1, \ldots, J_{|R|}$.

5.1.2 Branching strategy

Figure 1 depicts the branching tree explored by the two-phase B&B algorithm for the example instance. The top node (node 0) in Figure 1 represents the scheduling instance

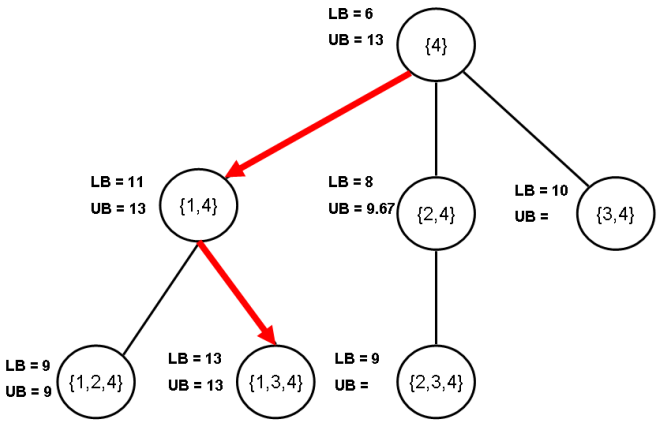

Figure 1: Illustration of the two-phase B&B algorithm. The highlighted path leads from the root node to the optimal solution.

with total weighted tardiness penalties containing all the jobs in $M_0 = \bar{R}$, in the example

$\bar{R} = F = \{4\}$. The solution of this scheduling instance provides us with a lower bound (LB) of 6 on the optimal objective function value; at the same time, the solution of an assignment problem leads to an upper bound (UB) of 13 (more details on UB are provided in Subsection 5.1.6). Each node u at the next level of the search tree represents a new scheduling instance in which one extra job $J_j \in R$ is added, leading to $M_u = M_0 \cup \{J_j\}$. At each subsequent level, one job is added to M_u, so at level k of the branching tree, each node corresponds with the selection of $|\bar{R}| + k$ jobs. To avoid repetition, a removable job J_j is added to a set of selected jobs if and only if its index j is greater than the highest index of a removable job already selected. Observe that the branching tree is strongly unbalanced and that for some nodes, we do not need to compute a UB.

5.1.3 Scheduling algorithm

In each node u, we use the B&B algorithm developed by Potts and Van Wassenhove [31] to schedule the set of jobs M_u to minimize the total weighted tardiness.

5.1.4 Node selection

We visit the branching tree in a *best-first search* (BFS) manner. At any point in time, a list of unfathomed nodes is kept in non-increasing order of their UB. The first node in the list is the next node to investigate. When a new node is created, the list is updated by inserting that node at the appropriate position such that the non-increasing-UB order is preserved.

5.1.5 Lower bound

At a given node u of the branching tree with the set M_u of selected jobs, a LB is given by the objective value of an optimal schedule.

5.1.6 Upper bound

At a given node u, a UB is computed if $J_{|R|} \notin M_u$. We have implemented two UBs. The first one is the output of an assignment problem; this bound is inspired by Slotnick and Morton [37]. The second bound is based on the solution to a job selection and sequencing problem with total weighted lateness penalties.

Assignment bound This approach divides each job into joblets with a duration of one time unit. A UB is found by solving an assignment problem that assigns joblets to unit-duration time buckets, so the corresponding schedule may be preemptive. At node u, we include dummy joblets and dummy positions to allow for the rejection of joblets stemming from jobs not in M_u.

Consider a node u at level l of the search tree and with J_k the last added job; it holds that $k \geq l$. Set $S_u = N \setminus (\{J_1, \ldots, J_{k-1}\} \setminus M_u)$ contains the jobs that can still be selected in the children of u; let $K = \sum_{i \in S_u} p_i$. The assignment of any joblet to a time instant larger than K and of any dummy joblet receives a return of zero. A non-dummy joblet assigned to a time bucket with index lower than or equal to K corresponds with acceptance of that joblet; the contribution of such a combination

to the objective function is determined based on the per-joblet reward, weight and due date. The per-joblet reward and weight are obtained by dividing the original job's reward and weight by the job's processing time; the due date of a joblet is a function of the position of the joblet (within the job) and the job's due date [33].

The assignment problem is solved using a cost-scaling algorithm [1, 16] and the opposite of its optimal value is used as UB.

Lateness bound At the node u, the total net profit is

$$\max_{M_u \subseteq M, \pi} \quad \sum_{t \in M} Q_{\pi(t)} - w_{\pi(t)}(C_{\pi(t)} - d_{\pi(t)})^+ \leq \max_{M_u \subseteq M, \pi} \quad \sum_{t \in M} Q_{\pi(t)} - w_{\pi(t)}(C_{\pi(t)} - d_{\pi(t)}).$$

The latter problem is a variant of the job selection and sequencing problem with total weighted lateness penalties [14, 36]. This variant imposes the selection of some jobs (namely those in M_u). The pseudo-polynomial-time dynamic-programming algorithm proposed by Ghosh [14] can be modified to solve this problem. The optimal objective value is then a UB for our problem.

Combinations Two combinations of the above UBs are tested in Section 7. The first combination (comb. 1) with a parameter α ($0 < \alpha \leq 1$) proceeds as follows: at the root node, an assignment bound is computed. At any subsequent node, a lateness bound is computed first; if its value is less than or equal to α times the UB of its parent node then an assignment bound is computed, in an attempt to prune the node. The second combination (comb. 2) is based on similar principles but the value of α now varies during the search: α now equals the number of jobs accepted at the parent node divided by the number of jobs accepted at the child node. Notice that when the number of jobs accepted at the parent node increases, α tends to one.

5.2 Direct B&B algorithm

The main steps of the direct B&B algorithm are described in this section.

5.2.1 Branching strategy

Figure 2 depicts a part of the search tree explored by the direct B&B algorithm when applied to the example instance presented earlier. The top node in the figure has a LB of 6, a UB of 13 and represents the situation where no job has yet been accepted. The LB corresponds with the heuristic solution that accepts all the jobs in F, sequenced in increasing job index; Subsection 5.2.3 provides more details on the UB computation. At level 1, n new nodes are created (four, in the example) representing n (possibly partial) solutions, each generated by selecting a single job and scheduling it at the first position. Given a parent node at level $k-1$ ($k \geq 1$), we create $n-k+1$ child nodes each with k jobs selected with known position: the job added at level k takes position k. At a given node u of our branching tree with $M_u \subseteq N$ the set of selected jobs, we denote by $\bar{M}_u$ its complement $N \setminus M_u$. Similarly to the two-phase algorithm, the branching tree is explored in a BFS manner.

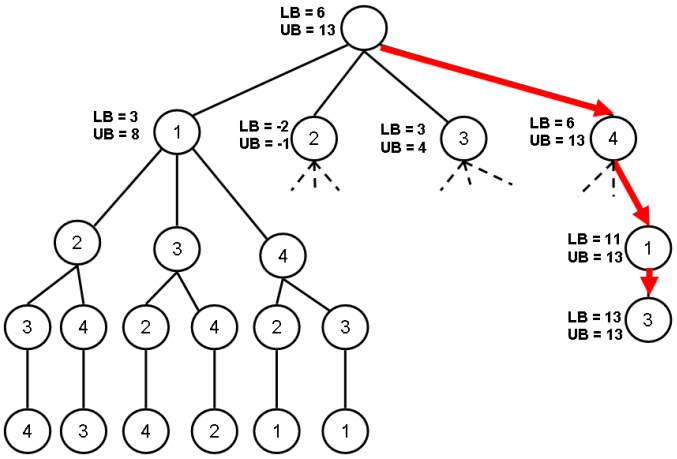

Figure 2: Illustration of the direct B&B algorithm. The highlighted path leads from the root node to the optimal solution.

5.2.2 Lower bound

At a given node u of the branching tree, let V_u be the return (net profit) obtained for the scheduled set M_u of jobs. Two cases can occur: (1) $F \subseteq M_u$, in which case V_u is a LB; (2) $F \nsubseteq M_u$, and then a LB is obtained by appending the jobs in $F \setminus M_u$ to the rear of the sequence in increasing order of their indices.

5.2.3 Upper bound

We have implemented the two UBs that were discussed in Section 5.1.6 also for the direct B&B. Notice that the contribution to the objective function of the jobs in M_u is known exactly; therefore, only the contribution of jobs in $\bar{M}_u$ needs to be upper-bounded. As the $|M_u|$ selected jobs are scheduled at the first $|M_u|$ positions, the due date of each job in $\bar{M}_u$ is updated accordingly by subtracting $\sum_{i \in M_u} p_i$, we refer to this new value as $\bar{d}$.

Assignment bound This UB is similar to the one in [37], apart from the updated due dates. Our UB is the sum of V_u and the objective function value of the assignment problem.

Lateness bound At node u, the unscheduled jobs can achieve a net profit of

$$\max_{M \subseteq \bar{M}_u, \pi} \sum_{t \in M} Q_{\pi(t)} - w_{\pi(t)}(C_{\pi(t)} - \bar{d}_{\pi(t)})^+ \leq \max_{M \subseteq \bar{M}_u, \pi} \sum_{t \in M} Q_{\pi(t)} - w_{\pi(t)}(C_{\pi(t)} - \bar{d}_{\pi(t)}) = \Delta,$$

where $F \setminus M_u \subseteq M$. The UB is then $\Delta + V_u$.

Combinations In Section 7, experimental results are reported for four combinations of these UBs. The first (comb. 1) and the second combination (comb. 2) correspond with those described for the two-phase B&B algorithm. The third combination

(comb. 3) is a variant of comb. 1 where the assignment bound is computed when the value of the LB (instead of UB) is less than or equal to α times the LB (rather than UB) of its parent node. The fourth combination (comb. 4) is a similar variation of comb. 2.

5.2.4 Dominance rules

Below, we outline some global and local dominance rules that can be used as pruning devices for the direct B&B algorithm.

Global dominance rules The global dominance rules presented here are extensions of Emmons' rules [5, 32]. The goal of these rules is to identify for each job j a set B_j, containing jobs that can be processed before job j, and A_j, the set of jobs that can be executed after job j. Let B_j^F be the set of jobs in F that must be processed before job j, and A_j^F the set of jobs in F that must be processed after job j. Both B_j^F and A_j^F are initially empty, and the rules below are applied iteratively.
Global rules: If job i and job j are selected, then there is an optimal sequence in which job i is processed before job j, if one of the following conditions holds:
Rule 1: $p_i \leq p_j$, $w_i \geq w_j$ and $d_i \leq \max\{d_j, p_j + \sum_{h \in B_j^F} p_h\}$.
Rule 2: $w_i \geq w_j$, $d_i \leq d_j$ and $d_j + p_j \geq \sum_{h \in N \setminus A_i^F} p_h$.
Rule 3: $d_j \geq \sum_{h \in N \setminus A_i^F} p_h$.
Remark that when $F = N$ these three rules are exactly Emmons' rules [32].

The proof of these three rules follows from a slight modification of the proof proposed by Rinnooy Kan et al. [32]. These three rules are used to construct a predecessor graph. However, the implementation of transitivity is slightly different because we also need to select jobs as well as schedule them. In our case, we use the following implementation.
Transitivity: whenever we identify a new relation "job j precedes job k" we distinguish two cases.
Case 1: If job $j \in F$ then we make sure that job k and each job coming after k come after job j and any job coming before j.
Case 2: If job $k \in F$ then we enforce that job j and each job coming before j come before job k and any job coming after k.

Local dominance rules At each node of the branching tree, we use two local dominance rules as pruning devices. The adjacent-job interchange rule is used as a scheduling rule. The following lemma presents the selection criterion. At node u, let J_u be the last job selected and scheduled at position $|M_u|$ and t_u the start time of the execution of job J_u.

Lemma 1. *At a given node u of the branching tree, let $j_i \in \bar{M}_u \setminus F$.*
(1) If $Q_{j_i} - w_{j_i}(t_u + p_{j_u} + p_{j_i} - d_{j_i}) \leq 0$ then the child node of u obtained by selecting and scheduling the job j_i at the position $|M_u| + 1$ can be pruned without losing all optimal solutions.
(2) If $F \cap \bar{M}_u \neq \emptyset$, let $B = Q_{j_i} - w_{j_i}(t_u + p_{j_u} + p_{j_i} - d_{j_i}) > 0$, $t_1 = t_u + p_{j_u} + p_{j_i}$ and $t_2 = t_u + p_{j_u}$. Consider two optimal schedules of all jobs in $F \cap \bar{M}_u$ after time

t_1 and time t_2 respectively with optimal value T_1 and T_2. If $B + T_1 \leq T_2$ then the child node of u obtained by selecting and scheduling job j_i at the position $|M_u| + 1$ can be pruned without losing all optimal solutions.

Proof: The proof follows from the fact that selecting job j_i will lead in the best case to a net profit equal to the optimal net profit obtained when job j_i is not selected. □

6 Heuristics

In this section, we present six heuristics for solving problem (1). These heuristics are developed based on the structure of the problem and the exact algorithms presented in Section 5. The first is a slightly modified version of the myopic heuristic proposed by Rom and Slotnick [33]; the modification allows to take into account the jobs in F. The second heuristic is an improvement of the first. The third heuristic is based on the LP relaxation of formulation MIP′ while the fourth heuristic is based on the LP relaxation of TIF. Heuristic 5 is a depth-first B&B heuristic based on the two-phase B&B algorithm and the last one is a truncated direct B&B heuristic.

6.1 Heuristic 1

This is a slightly modified version of the myopic heuristic presented by Rom and Slotnick [33]. This heuristic is described by Pseudocode 1.

Pseudocode 1

1: calculate the profit when all jobs are accepted and ordered in increasing job index
2: decompose the jobs into joblets with unit processing time, apportion the weights and revenues accordingly, and use the assignment algorithm of Section 5.1.6 to find the optimal sequence of this relaxation by maximizing the return of accepting or rejecting each joblet while making sure that the firm planned orders are accepted
3: accept all jobs in $\bar{F}$ that have at least 75% of their joblets accepted in the relaxed solution
4: sequence these (reassembled) jobs in ascending order of completion time (the completion time of the latest component joblet scheduled in the assignment solution) minus processing time, and calculate the profit of this set
5: order the remaining jobs using the Rachamadugu-and-Morton heuristic for weighted tardiness [30], place them after the previously accepted jobs in 4 above, and calculate the profit of this set
6: return the best solution of 1, 4 and 5 above

6.2 Heuristic 2

Heuristic 2 is an improvement of Heuristic 1. Here, an exact algorithm is used at line 4: to schedule the accepted set of jobs, namely the B&B developed by Potts and Van Wassenhove [31].

6.3 Heuristic 3

This heuristic is based on the LP relaxation of MIP′. Given a solution to the LP relaxation, if that solution is integer then it is the output of our heuristic; otherwise, we round

the variables y_i, $i = 1, \ldots, n$, to integer values as follows. If $i \in F$ then $y_i = 1$. On the other hand, if $i \in \bar{F}$ and there exists $t \in \{1, 2, \ldots, n\}$ such that $x_{it} \geq \frac{1}{4}$ we set $y_i = 1$, otherwise $y_i = 0$. The jobs $i \in N$ with $y_i = 1$ form the set of selected jobs. To limit the running time, the LP relaxation is solved as follows: for an instance with n jobs, we impose a time limit of $5n$ seconds. Subsequently, we apply the B&B of Potts and Van Wassenhove [31] to produce an optimal schedule for the selected jobs. We mention that the parameter $\frac{1}{4}$ used in this heuristic was chosen after many trials.

6.4 Heuristic 4

This heuristic is based on the LP relaxation of TIF. Given a solution x_{jt}, $j = 1, \ldots, n$, $t = 1, \ldots, T - p_j + 1$, to the LP relaxation, there are two possibilities: either the solution is integer, in which case it is the output of the heuristic, or it is fractional and we select the jobs to execute in the following way. Job j is selected if $j \in F$ or if there exists $t \in \{1, \ldots, T\}$ such that $\sum_{s=t-p_j+1}^{t} x_{jt} \geq \frac{1}{4}$. As for Heuristic 3, a time limit of $5n$ seconds is set for the LP relaxation and selected jobs are again scheduled with an exact algorithm.

6.5 Heuristic 5

This heuristic is a depth-first B&B heuristic; it is a variant of the two-phase B&B algorithm. Pseudocode 2 depicts the steps followed by Heuristic 5.

Pseudocode 2

1: schedule all jobs to minimize total weighted tardiness
2: identify the set R of removable jobs
3: order the jobs in R in decreasing order of their effect on the net profit
4: compute a LB at the root node
5: add jobs in a greedy fashion; keep a job if and only if it increases the net profit

6.6 Heuristic 6

Heuristic 6 is a truncated direct B&B heuristic: the algorithm is halted when a time limit is reached. We impose a time limit of $10n$ seconds when solving instances with n jobs.

7 Computational experiments

All algorithms have been coded in C using Visual Studio C++ 2005; all the experiments were run on a Dell Optiplex $GX620$ personal computer with Pentium R processor with 2.8 GHz clock speed and 1.49 GB RAM, equipped with Windows XP. CPLEX 10.2 was used for solving the linear formulations. Below, we first provide some details on the generation of the datasets and subsequently, we discuss the computational results.

7.1 Data generation

The B&B algorithms and the heuristics are tested on randomly generated instances with n jobs, for $n = 10$, 20, 30, 40 and 50. For each job i, an integer processing time p_i and

integer weight w_i are drawn from the discrete uniform distribution on $[1, 10]$.

To diversify the test instances, we follow Potts and Van Wassenhove [31] and use different relative ranges of due dates r and different average tardiness factor T. The values chosen for r are 0.3, 0.6 and 0.9; the same values apply for T. For given values of r and T and with $P = \sum_{i=1}^{n} p_i$, we select for each job i an integer due date d_i from the uniform distribution with support the set of integers in the interval $[\max\{P(1 - T - \frac{r}{2}), p_i\}, \max\{P(1 - T + \frac{r}{2}) - 1, p_i\}]$. In this way, we avoid the creation of instances in which some jobs can be processed last without being late, if all jobs are executed. Each job revenue Q_i is an observation of a lognormal distribution with an underlying normal distribution with mean 0 and standard deviation 1, rounded to the nearest integer. This reflects a situation in which job characteristics are fairly similar but where the potential revenue of a job may vary widely (see Slotnick et al. [36, 37]).

For every value of n and for each of the nine pairs of values of r and T, one instance is generated. These instances are subsequently modified by randomly selecting the firm planned orders out of the n jobs as follows. Two extreme cases considered are $|F| = 0$ and $|F| = n$. We also consider the intermediary choices with $|F|$ equal to $0.2n$, $0.4n$, $0.6n$ and $0.8n$. For each of the latter values for $|F|$, we make two choices for F, and we make sure that each smaller set F is embedded in the larger F of another instance. This means, for example, that for a given instance with a set F_1 of firm planned orders with $|F_1| = 0.2n$, there exists an instance with a set of firm planned orders F_2 with $|F_2| = 0.4n$ such that $F_1 \subset F_2$. For a given value of n, we have $9 \times (2 + 2 \times 4) = 90$ test instances, yielding $5 \times 90 = 450$ instances in total.

Moreover, to study the effect of processing-time variability on the developed algorithms, we have also generated a set of 90 instances with $n = 10$ jobs each, following the methodology described above but with larger processing times: the p_i are generated from the discrete uniform distribution on $[10, 100]$.

7.2 Computational results

In this section, computation time is referred to as *Time* and is expressed in seconds; *Nodes* is the number of nodes explored in the search tree of the considered algorithm. Furthermore, each cell in the tables appearing in this section is (unless mentioned otherwise) the average of either nine values (corresponding to the cells with $|F|$ equal to $100\%n$ and $0\%n$) or 18 values (corresponding to the cells with 80%, 60%, 40% and 20%). Some tables contain rows entitled *Unsolved*, which indicate the number of instances of each group that remained unsolved when the time limit was reached.

7.2.1 Size $n = 10$

Table 2 displays the output of the mixed-integer linear formulations for the dataset with $n = 10$ and processing times between 1 and 10. The table shows the domination of the time-indexed formulation (TIF) over the other linear formulations (when solved using CPLEX). As for the other formulations, we observe that MIP′ dominates MIP+Cuts, which is the formulation MIP with the addition of cuts. Moreover, CPLEX solves MIP′ faster than MIP′+Cuts for instances with at least 60% of firm planned orders. On the other hand, for instances with at most 40% of firm planned orders, the running time of

Firm planned orders		100%	80%	60%	40%	20%	0%
MIP+Cuts	Nodes	96680 (2)	120075	97140	48417	59467	51417
	Time	505.58	657.61	548.21	342.52	401.73	378.15
MIP′	Nodes	17870	17920	21137	14935	15256	19118
	Time	123.46	91.28	97.30	70.97	78.40	118.30
MIP′+Cuts	Nodes	30556	10058	16110	7339	6315	6989
	Time	194.94	99.89	107.67	61.06	52.42	54.39
TIF	Nodes	0	0	7	0	1	0
	Time	0.05	0.07	0.07	0.08	0.07	0.07

Table 2: Linear formulations for $n = 10$ with small processing time.

MIP′ is greater than that of MIP′+Cuts. With the addition of cuts to MIP, there are two instances with 100% of firm planned orders that CPLEX was not able to solve within the time limit of one hour (indicated by the number 2 between brackets).

Table 3 reports the output of the two-phase B&B algorithm for the dataset with $n = 10$ and small processing times, including the two UB procedures and the two combinations described in Section 5. The two-phase B&B algorithm with assignment bound is identified by *assign.* while *lateness* refers to the algorithm with the lateness bound. The entry *min.* is used to indicate the best (minimum computation time) of the two foregoing implementations. The first combination of these two bounds with constant factor α identified by *comb.* 1 is implemented for four different values of α, namely 1, 0.9, 0.8 and 0.7. From Table 3, we see that the assignment bound leads to a branching tree with quite fewer nodes than the lateness bound. However, it turns out that the computation time required for the assignment bound is considerable; this can be observed in the row corresponding to min., which contains exactly the result obtained with the lateness bound. With respect to the combinations of the UB procedures, comb. 2, which is the combination with a variable factor α, seems to behave rather well. The number of nodes explored by comb. 1 seems to increase when α decreases.

Table 4 reports the results of the direct B&B algorithm for $n = 10$ with processing

Firm planned orders				100%	80%	60%	40%	20%	0%
two-phase	assign.		Nodes	0	1	4	12	20	39
			Time	0.00	0.02	0.05	0.13	0.21	0.41
	lateness		Nodes	0	2	10	34	125	442
			Time	0.00	0.00	0.00	0.00	0.01	0.04
	min.		Nodes	0	2	10	34	125	442
			Time	0.00	0.00	0.00	0.00	0.01	0.04
	comb. 1	1	Nodes	0	2	7	19	38	79
			Time	0.00	0.01	0.04	0.10	0.21	0.41
		0.90	Nodes	0	2	8	22	66	206
			Time	0.00	0.01	0.03	0.07	0.13	0.26
		0.80	Nodes	0	2	8	27	99	301
			Time	0.00	0.01	0.03	0.06	0.09	0.17
		0.70	Nodes	0	2	9	29	108	342
			Time	0.00	0.01	0.02	0.05	0.07	0.13
	comb. 2		Nodes	0	2	8	26	96	367
			Time	0.00	0.01	0.03	0.06	0.11	0.16

Table 3: Two-phase B&B algorithm for $n = 10$ with small processing time.

Firm planned orders				100%	80%	60%	40%	20%	0%
direct	assign.		Nodes	61	33	31	38	45	84
			Time	0.23	0.16	0.21	0.32	0.39	0.59
	lateness		Nodes	58	56	82	138	267	579
			Time	0.00	0.00	0.01	0.02	0.05	0.13
	min.		Nodes	58	56	82	138	267	579
			Time	0.00	0.00	0.01	0.02	0.05	0.13
	comb. 1	1	Nodes	64	36	40	51	72	148
			Time	0.14	0.11	0.15	0.25	0.37	0.64
		0.90	Nodes	63	36	39	51	72	147
			Time	0.15	0.11	0.15	0.25	0.39	0.64
		0.80	Nodes	63	38	48	55	80	165
			Time	0.16	0.11	0.16	0.26	0.39	0.68
		0.70	Nodes	63	44	50	73	127	251
			Time	0.16	0.11	0.16	0.28	0.44	0.76
	comb. 2		Nodes	63	44	54	76	133	288
			Time	0.15	0.11	0.16	0.28	0.45	0.80
	comb. 3	1	Nodes	61	34	37	53	109	579
			Time	0.20	0.14	0.17	0.21	0.26	0.14
		0.90	Nodes	63	39	55	113	250	579
			Time	0.17	0.08	0.07	0.11	0.10	0.14
		0.80	Nodes	63	40	56	114	250	579
			Time	0.17	0.08	0.07	0.10	0.09	0.14
		0.70	Nodes	63	40	58	117	251	579
			Time	0.17	0.08	0.07	0.10	0.08	0.14
	comb. 4		Nodes	63	40	55	95	227	579
			Time	0.17	0.09	0.09	0.11	0.10	0.14

Table 4: Direct B&B algorithm for $n = 10$ with small processing time.

time between 1 and 10. Just as for the two-phase B&B algorithm, the best computation time here is obtained using the lateness bound. A number of combinations of the two UB procedures are described in Section 5; the combinations comb. 1 and comb. 2 seem to have no positive effect on the direct B&B algorithm as the running times generally increase. Options comb. 3 and comb. 4, although usually better than comb. 1 and comb. 2, are still dominated by the lateness bound.

Overall, two observations can be made. When using either the two-phase B&B or the direct B&B for small instances ($n = 10$), the lateness bound seems to display the best ratio quality/time. Moreover, the two-phase B&B algorithm with lateness bound

Firm planned orders		100%	80%	60%	40%	20%	0%
MIP′	Unsolved	–	–	–	–	–	–
	Nodes	23875	20054	31021	36362	58677	48539
	Time	214.50	88.88	105.12	141.57	202.97	179.38
MIP′+Cuts	Unsolved	all	–	–	–	–	–
	Nodes	39134	9699	9192	11663	7064	13548
	Time	284.14	78.53	78.13	75.12	66.41	124.38
TIF	Unsolved	–	1	2	3	3	1
	Nodes	0	2585	1033	3202	1348	1395
	Time	7.08	228.04	54.03	38.06	15.74	27.38

Table 5: Linear formulations for $n = 10$ with large processing time.

Firm planned orders				100%	80%	60%	40%	20%	0%
two-phase	assign.		Nodes	0	1	3	9	25	53
			Time	0.00	1.54	3.93	8.13	18.79	40.59
	lateness		Nodes	0	2	10	44	178	722
			Time	0.00	0.01	0.01	0.02	0.03	0.08
	min.		Nodes	0	2	10	44	178	722
			Time	0.00	0.01	0.01	0.02	0.03	0.08
	comb. 1	1	Nodes	0	2	7	17	47	110
			Time	0.00	0.92	3.21	7.95	17.85	41.51
		0.90	Nodes	0	2	8	29	99	318
			Time	0.00	0.93	2.61	6.16	11.70	28.76
		0.80	Nodes	0	2	9	33	130	446
			Time	0.00	0.93	2.38	5.32	9.36	21.76
		0.70	Nodes	0	2	9	36	144	553
			Time	0.00	0.92	2.09	4.17	7.76	15.91
	comb. 2		Nodes	0	2	9	32	133	598
			Time	0.00	0.92	2.47	5.56	9.97	18.03

Table 6: Two-phase B&B algorithm for $n = 10$ with large processing time.

appears to have the best computation times. Remark that the average computation time of the worst exact algorithm presented so far (linear formulations included) is far better than the average computation time reported by Slotnick and Morton [37] for $n = 10$.

Firm planned orders				100%	80%	60%	40%	20%	0%
direct	assign.		Nodes	67	31	25	34	52	93
			Time	13.87	8.88	10.45	15.46	24.57	41.17
	lateness		Nodes	53	55	87	155	259	536
			Time	0.00	0.01	0.01	0.03	0.07	0.20
	min.		Nodes	53	55	87	155	259	536
			Time	0.00	0.01	0.01	0.03	0.07	0.20
	comb. 1	1	Nodes	74	35	32	48	78	144
			Time	9.74	6.40	7.52	11.76	19.18	31.97
		0.90	Nodes	70	34	32	48	78	145
			Time	10.44	6.58	7.83	12.10	19.22	33.03
		0.80	Nodes	69	34	33	50	84	156
			Time	11.47	6.75	7.97	10.86	17.37	29.16
		0.70	Nodes	69	35	34	57	94	178
			Time	10.93	5.95	7.08	10.96	16.15	27.55
	comb. 2		Nodes	69	36	38	63	110	217
			Time	9.92	5.74	6.84	10.53	15.18	25.32
	comb. 3	1	Nodes	67	31	28	43	113	536
			Time	10.86	7.12	8.59	12.12	15.43	2.83
		0.90	Nodes	67	35	43	83	231	536
			Time	11.69	6.85	7.15	8.49	5.98	2.83
		0.80	Nodes	67	36	44	87	214	536
			Time	11.71	6.98	7.12	8.32	5.90	2.84
		0.70	Nodes	67	36	45	88	216	536
			Time	11.86	7.02	6.94	8.03	5.52	2.84
	comb. 4		Nodes	67	35	44	81	205	536
			Time	11.49	6.91	7.20	8.75	6.99	2.84

Table 7: Direct B&B algorithm for $n = 10$ with large processing time.

Firm planned orders				100%	80%	60%	40%	20%	0%
TIF			Unsolved	–	2	2	–	–	–
			Nodes	14	9255	19288	192019	4129	1677
			Time	0.54	19.45	27.11	703.54	6.25	4.21
two-phase	assign.		Unsolved	–	–	–	–	–	–
			Nodes	0	5	20	70	140	315
			Time	0.00	0.23	0.90	3.11	6.39	13.32
	lateness		Unsolved	–	–	–	–	–	–
			Nodes	0	13	162	2461	35565	471566
			Time	0.00	0.02	0.16	1.39	12.87	208.35
	min.		Unsolved	–	–	–	–	–	–
			Nodes	0	13	162	2461	140	315
			Time	0.00	0.02	0.16	1.39	6.39	13.32
	comb. 1	1	Unsolved	–	–	–	–	–	–
			Nodes	0	9	41	168	401	977
			Time	0.00	0.20	1.08	4.42	11.21	26.94
		0.90	Unsolved	–	–	–	–	–	–
			Nodes	0	12	107	1029	8671	72900
			Time	0.00	0.13	0.77	4.70	26.21	110.44
		0.80	Unsolved	–	–	–	–	–	–
			Nodes	0	12	135	1871	23456	275274
			Time	0.00	0.12	0.57	4.77	34.15	212.77
		0.70	Unsolved	–	–	–	–	–	–
			Nodes	0	12	145	2238	31922	395154
			Time	0.00	0.12	0.52	3.85	25.54	244.56
	comb. 2		Unsolved	–	–	–	–	–	–
			Nodes	0	12	91	853	11725	239363
			Time	0.00	0.14	0.88	5.24	36.17	244.69

Table 8: Results for $n = 20$ with exact algorithms.

Next, we study the effect of the processing time on the exact algorithms. Table 5 contains a comparison of the two linear formulations when processing times range from 10 to 100. In this table, we do not include the formulation MIP+Cuts as this setting is dominated by the formulation MIP′ (see Table 2). The results of Table 5 indicate a strong dependency of the formulation TIF on the scale of the processing time. Both by the formulation MIP′ and MIP′+Cuts, all the instances are solved within a timespan of less than three times the running time needed for the instances with small processing time. For the formulation TIF, on the other hand, CPLEX was not able to solve some instances within the time limit of one hour and the computation time for instances solved within the time limit are more than 1000 times the running time reported in Table 2. Tables 2 & 5 suggest to use TIF when the processing time are small while either MIP′ or MIP′+Cuts is preferable for instances with large processing time.

Table 6 reports the results of the two-phase B&B algorithm for $n = 10$ with large processing time. We observe that the two-phase B&B algorithm with the lateness bound is almost insensible to the variation in processing times, with running times comparable to those for small p_i. Unlike the lateness bound, the assignment bound leads to a marked increase in running times. The behavior of the combinations (comb. 1 and comb. 2) observed in Table 3 remains unchanged.

Table 7 presents the results of the direct B&B for $n = 10$ and large processing times. In line with the two-phase algorithm, the direct B&B with lateness bound is insensible to

Firm planned orders				100%	80%	60%	40%	20%	0%
direct	assign.		Unsolved	–	–	–	–	–	–
			Nodes	182	412	249	621	1446	2948
			Time	2.60	4.52	6.12	15.52	34.63	65.66
	lateness		Unsolved	–	–	–	–	2	5
			Nodes	1138	3412	11028	64553	416139	904924
			Time	0.01	0.43	2.98	40.42	1346.03	2332.84
	min.		Unsolved	–	–	–	–	–	–
			Nodes	1138	3412	11028	621	1446	2948
			Time	0.01	0.43	2.98	15.52	34.63	65.66
	comb. 1	1	Unsolved	–	–	–	–	–	–
			Nodes	220	491	335	961	2767	7050
			Time	1.53	3.62	5.38	16.75	49.58	161.96
		0.90	Unsolved	–	–	–	–	–	–
			Nodes	228	566	372	1091	3104	7815
			Time	1.83	3.59	4.82	16.40	52.18	169.53
		0.80	Unsolved	–	–	–	–	–	–
			Nodes	244	839	1797	8807	36239	150594
			Time	1.79	3.36	8.64	54.50	284.38	1374.04
		0.70	Unsolved	–	–	–	–	–	4
			Nodes	263	1189	5807	34494	213449	176995
			Time	1.91	4.32	21.19	135.69	902.26	1141.85
	comb. 2		Unsolved	–	–	–	–	–	–
			Nodes	222	651	664	2602	12450	61122
			Time	1.72	3.76	7.94	39.42	219.59	1048.19
	comb. 3	1	Unsolved	–	–	–	–	–	4
			Nodes	194	427	289	1387	29417	891555
			Time	2.03	3.84	5.29	14.56	148.86	2125.73
	comb. 4		Unsolved	–	–	–	–	–	4
			Nodes	212	644	3692	39971	323021	891555
			Time	1.92	2.98	5.60	36.68	837.02	2164.71

Table 9: Results for $n = 20$ with exact algorithms (continued).

the variation of the processing time, while the assignment bound increases the running times and the combinations of these two bounds display the same behavior as in Table 4.

Overall, both B&B algorithms (two-phase and direct) outperform CPLEX for instances with large processing time.

7.2.2 Size $n = 20$

Tables 8 & 9 display the results of the exact algorithms for the instances with $n = 20$. Table 8 contains the results of the time-indexed formulation and the two-phase B&B algorithm. CPLEX seems to solve extreme instances (100% and 0%) faster than the intermediary cases. Moreover, four instances (two with 80% and two with 60% of firm planned orders) are not solved within the time limit of two hours. The two-phase B&B solves all instances, regardless of the UB procedure or the combination used. The best computation time (represented by *min.*) is obtained by the lateness bound when the percentage of firm planned orders is greater than or equal to 40%, and by the assignment bound for the other cases. This observation is different from our analysis of the ten-job instances, where the lateness bound was unconditionally recommended.

The results for the direct B&B can be found in Table 9. Using the lateness bound, there are two (respectively five) instances in the set with 20% (respectively 0%) of firm planned orders that are not solved within the time limit of two hours, while the assignment bound solves all the instances. Exactly as for the two-phase B&B, the best computation time is obtained by the lateness bound, respectively by the assignment bound, dependent on whether or not the percentage of firm planned orders is at least 60%. The combinations of the two UB procedures do not seem to present any real advantage.

7.2.3 Size $n = 30, 40$ and 50

Firm planned orders			100%	80%	60%	40%	20%	0%
TIF		Unsolved	–	2	2	2	–	1
		Node	5	1334	4486	3223	4855	25063
		Time	2.44	14.99	26.43	15.90	25.21	115.33
two-phase	assign.	Unsolved	–	–	–	–	–	–
		Nodes	0	7	61	174	765	1385
		Time	0.30	1.06	5.40	16.66	72.83	128.60
	lateness	Unsolved	–	–	–	–	8	7
		Nodes	0	33	1742	98302	416965	0
		Time	0.30	2.43	21.60	585.66	560.51	0.00
	min.	Unsolved	–	–	–	–	–	–
		Nodes	0	7	61	174	765	1385
		Time	0.30	1.06	5.40	16.66	72.83	128.60
direct	assign.	Unsolved	–	–	–	3	3	2
		Nodes	14279	9807	9907	5441	13441	6146
		Time	408.47	192.93	166.30	149.44	336.47	300.42
	lateness	Unsolved	–	–	3	11	15	7
		Nodes	15088	89888	192703	516127	267993	257571
		Time	0.50	132.66	293.13	1207.88	619.58	1055.19
	min.	Unsolved	–	–	–	3	3	2
		Nodes	15088	89888	9907	5441	13441	6146
		Time	0.50	132.66	166.30	149.44	336.47	300.42

Table 10: Results for $n = 30$ with exact algorithms.

The results of the exact algorithms for $n = 30$ are gathered in Table 10. Although the time limit is now three hours, the number of unsolved instances for the TIF increases compared to Table 8. The two-phase B&B with assignment bound displays the best running time amongst the B&B algorithms and solves all the instances. The TIF formulation does better only for the set of instances with 20% of firm planned orders.

In Table 11, we find the output of the two-phase B&B algorithm and the TIF formulation for $n = 40$ and of the two-phase B&B algorithm for $n = 50$. A two-hour time limit is applied. For the 40-job instances, the two-phase B&B algorithm solves all the instances within the time limit. Using the linear formulation TIF, the number of instances unsolved increases up to six out of 18 for some settings. For the 50-job dataset, the two-phase B&B algorithm is able to solve all but one of the instances within two hours.

Table 12 exhibits the outcomes of the heuristics for $n = 30$. Here, *GAP* equals $\frac{|z_H - z_{OP}|}{z_{OP}} \times 100\%$, where z_{OP} is the optimal objective value and z_H is the objective value obtained by the heuristic. GAP is computed only for the instances for which z_H and z_{OP} have the same sign and $z_{OP} \neq 0$. For each cell in the table, *Comp.* counts the

Firm planned orders		100%	80%	60%	40%	20%	0%
40 jobs							
TIF	Unsolved	–	6	4	6	5	1
	Time	7.73	169.24	222.80	122.98	320.44	182.19
two-phase	Unsolved	–	–	–	–	–	–
	Time	8.96	105.01	107.87	178.76	290.79	533.76
50 jobs							
two-phase	Unsolved	–	–	–	–	1	–
	Time	24.51	345.38	379.71	878.53	1334.96	2091.59

Table 11: Results for $n = 40$ and $n = 50$ with exact algorithms.

Firm planned orders		100%	80%	60%	40%	20%	0%
Heuristic 1	Comp.	1	–	1	–	–	–
	GAP	52.11	80.62	16.64	15.51	7.68	4.96
	Time	0.08	0.12	0.15	0.19	0.23	0.31
Heuristic 2	Comp.	–	–	1	–	–	–
	GAP	0.00	3.77	3.16	8.28	2.61	1.31
	Time	0.37	0.13	0.16	0.20	0.23	0.29
Heuristic 3	Comp.	–	–	1	1	–	–
	GAP	0.00	10.18	23.25	50.16	72.03	99.63
	Time	30.18	27.26	28.76	27.09	32.66	30.60
Heuristic 4	Comp.	–	–	–	1	–	–
	GAP	0.00	3.55	3.28	8.69	3.40	4.36
	Time	0.59	0.29	0.17	0.13	0.11	0.10
Heuristic 5	Comp.	–	–	–	–	–	–
	GAP	0.00	0.32	1.72	12.59	20.98	30.24
	Time	0.28	0.38	0.31	0.29	0.28	0.29
Heuristic 6	Comp.	4	2	3	2	2	–
	GAP	4.96	29.62	41.65	17.70	22.18	39.69
	Time	0.00	4.40	76.59	120.12	144.00	186.06

Table 12: Comparison of heuristics for $n = 30$.

number of instances for which we have $z_H < 0$ and $z_{OP} \geq 0$. From Table 12, we see that Heuristic 1 is dominated by Heuristic 2, which shows the importance of using an exact algorithm for scheduling the selected set of jobs. Heuristic 4 dominates Heuristic 3, which might be explained by the better behavior of the time-indexed formulation compared to the formulation MIP$'$ for instances with small processing time. Finally, Heuristic 5 dominates Heuristic 6 both in computation time and based on GAP. Below, only Heuristic 2, Heuristic 4 and Heuristic 5 are used for the datasets with 40 and 50 jobs per instance.

Table 13 shows that the heuristics perform quite well for the 40-job and 50-job instances. Although GAP and the running time increase with the number of jobs, they are still reasonable. For the one instance with 50 jobs for which the two-phase B&B could not guarantee an optimal solution within the time limit, we use the UB provided by the solution to the LP relaxation to compute the GAP. For $n = 50$, there are two instances with 80% of firm planned orders for which the three heuristics are unable to produce an objective value with the same sign as the optimal value.

Firm planned orders		100%	80%	60%	40%	20%	0%
40 jobs							
Heuristic 2	Comp.	–	–	–	–	–	–
	GAP	0.00	3.73	3.91	4.25	4.12	4.33
	Time	10.10	0.28	0.24	0.31	0.35	0.42
Heuristic 4	Comp.	–	–	–	–	–	–
	GAP	0.00	4.60	4.58	5.13	4.77	5.55
	Time	8.97	0.56	0.40	0.31	0.34	0.33
Heuristic 5	Comp.	–	–	–	–	–	–
	GAP	0.00	3.61	4.54	8.61	14.37	23.12
	Time	10.46	83.56	100.72	101.84	102.53	99.38
50 jobs							
Heuristic 2	Comp.	–	2	–	–	–	–
	GAP	0.00	10.14	7.18	12.47	10.98	12.28
	Time	25.97	1.25	0.42	0.50	0.60	0.71
Heuristic 4	Comp.	–	2	–	–	–	–
	GAP	0.00	10.42	16.96	10.00	10.21	10.05
	Time	26.78	1.21	0.70	0.49	0.46	0.45
Heuristic 5	Comp.	–	2	–	–	–	–
	GAP	0.00	0.76	5.16	7.48	14.76	28.09
	Time	25.96	340.56	287.77	286.36	286.18	286.13

Table 13: Comparison of heuristics for $n = 40$ and $n = 50$.

8 Summary and conclusions

In this paper, we have modeled the order acceptance and scheduling problem taking into account both firm planned orders as well as potential orders, where the latter are orders that can still be rejected. Our results show that it is unlikely that a constant-factor approximation algorithm can be developed for this problem. We have presented two mixed-integer linear formulations that are solved using the IP solver of CPLEX. The first of these formulations is rather intuitive, the second is a time-indexed formulation. Our results indicate that the IP solver of CPLEX solves the latter formulation faster than the former one even with the addition of cuts when the processing time of each job is relatively small. In case of larger processing times, however, we recommend the use of the former formulation.

Our linear formulations turn out to perform better (in term of computation time) than the exact B&B algorithm presented by Slotnick and Morton [37]. In this paper, we have also developed two new B&B algorithms that produce optimal solutions to the order acceptance and scheduling problem. The first B&B algorithm is hierarchical and performs selection and scheduling separately, while the second integrates these two decisions. Two upper-bound procedures have been implemented, one based on an assignment problem and one that uses a job selection problem with lateness penalties. Our experimental results demonstrate the good behavior of these two B&B algorithms. For small instances (with a low number of jobs), the implementation with the lateness bound is the most efficient; the full benefit of the assignment bound is achieved for instances with more than 20 jobs. Overall, the two-phase B&B algorithm with assignment bound dominates the other exact algorithms.

Both the linear formulations and the exact B&B algorithms are suited especially

for solving small and medium-sized instances. To extend the application to large-size instances with considerably more jobs, we have presented six heuristics. The first one is a slightly modified version of the myopic heuristic described by Rom and Slotnick [33], the second is an improvement of the first and the others are derived from the linear formulations and the exact algorithms. Based on our experimental results, we recommend the use of either the improved myopic heuristic (Heuristic 2), the heuristic based on the LP relaxation of the time-indexed formulation (Heuristic 4) or the depth-first B&B heuristic based on the two-phase B&B algorithm (Heuristic 5).

An important research direction that might be pursued in the future is an extension of this work to on-line scheduling, where not all jobs are available at the beginning of the planning horizon but arrive dynamically throughout time. A second obvious extension that deserves attention is the case where the manufacturing capacity consists of multiple machines in parallel.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc. NJ, 1993.

[2] B. Alidaee, G. Kochenberger, and M. Amini. Greedy solutions of selection and ordering problems. *European Journal of Operational Research*, 134:203–215, 2001.

[3] K.R. Baker. Sequencing rules and due-date assignments in a job shop. *Management Science*, 30:1093–1104, 1984.

[4] P. Baptiste, P. Chrétienne, J. Meng-Gérard, and F. Sourd. On maximizing the profit of a satellite launcher: selecting and scheduling tasks with time windows and setups. *Discrete Applied Mathematics*, to appear.

[5] L.-P. Bigras, M. Gamache, and G. Savard. Time-indexed formulations and the total weighted tardiness problem. *INFORMS Journal on Computing*, 20:133–142, 2008.

[6] Z. Bilginturk, C. Oguz, and S. Salman. Order acceptance and scheduling decisions in make-to-order systems. In P. Baptiste, G. Kendall, A. Munier-Kordon, and F. Sourd, editors, *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, pages 80–87, Paris, France, 28-31 August 2007.

[7] Z-H. Chen and C-L. Li. Scheduling with subcontracting options. *IIE Transactions*, 40:1171–1184, 2008.

[8] T.C.E. Cheng and M.C. Gupta. Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research*, 38(2):156–166, 1989.

[9] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. In *IEEE Symposium on Foundations of Computer Science*, pages 348–356, 2001. citeseer.ist.psu.edu/chuzhoy01approximation.html.

[10] P. De, J.B. Ghosh, and C.E. Wells. On the minimization of the weighted number of tardy jobs with random processing times and deadline. *Computers & Operations Research*, 18(5):457–463, 1991.

[11] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, and J. Wein. Techniques for scheduling with rejection. *Journal of Algorithms*, 49:175–191, 2003.

[12] L. Epstein, J. Nogab, and G.J. Woeginger. On-line scheduling of unit time jobs with rejection: minimizing the total completion time. *Operations Research Letters*, 30:415–420, 2002.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.

[14] J.B. Ghosh. Job selection in a heavily loaded shop. *Computers & Operations Research*, 24(2):141, 1997.

[15] P. Glasserman and Y. Wang. Leadtime-inventory trade-offs in assemble-to-order systems. *Operations Research*, 46:858–871, 1998.

[16] A.V. Goldberg and R. Kennedy. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71:153–177, 1995.

[17] H.H. Guerrero and G.M. Kern. How to more effectively accept and refuse orders. *Production and Inventory Management Journal*, 4:59–62, 1998.

[18] S.K. Gupta, J. Kyparisis, and C.-M. Ip. Project selection and sequencing to maximize net present value of the total return. *Management Science*, 38:751–752, 1992.

[19] J. Herbots, W. Herroelen, and R. Leus. Dynamic order acceptance and capacity planning on a single bottleneck resource. *Naval Research Logistics*, 54(8):874–889, 2007.

[20] W.J. Hopp and M.L. Spearman. *Factory Physics. Foundations of Manufacturing Management*. McGraw-Hill, 2001.

[21] V.C. Ivănescu, J.C. Fransoo, and J.W. Bertrand. A hybrid policy for order acceptance in batch process industries. *OR Spectrum*, 28:199–222, 2006.

[22] U. Karmarkar. Lot sizes, manufacturing lead times and throughput. *Management Science*, 33:409–418, 1987.

[23] P. Keskinocak and S. Tayur. Due date management policies. In D. Simchi-Levi, S.D. Wu, and Z.J. Shen, editors, *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, chapter 12, pages 485–547. Kluwer, 2004.

[24] A.J. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41, 2001.

[25] E.L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathemathics*, 1:331–342, 1977.

[26] J.K. Lenstra, A.H. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathemathics*, 1:343–362, 1977.

[27] H.F. Lewis and S.A. Slotnick. Multi-period job selection: planning work loads to maximize profit. *Computers & Operations Research*, 29:1081–1098, 2002.

[28] L. Lu, L. Zhang, and J. Yuan. The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan. *Theoretical Computer Science*, 396:283–289, 2008.

[29] Y.D. Lu, J.S. Song, and D.D. Yao. Order fill rate, leadtime variability, and advance demand information in an assemble-to-order system. *Operations Research*, 51(2):292–308, 2003.

[30] T.E Morton and R.M. Rachamadugu. Myopic heuristics for the single machine weighted tardiness problem. Working Paper 30-82-83, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, 1982.

[31] C.N. Potts and L.N. Van Wassenhove. A branch and bound algorithm for the total weighted tardiness problem. *Operations research*, 33(2):363–377, 1985.

[32] A.H.G. Rinnooy Kan, B.J. Lageweg, and J.K. Lenstra. Minimizing total cost in one-machine scheduling. *Operations Research*, 23:908–927, 1975.

[33] W.O. Rom and S.A. Slotnick. Order acceptance using genetic algorithms. *Computers & Operations Research*, 36(6):1758–1767, 2009.

[34] R. Roundy, D. Chen, P. Chen, M. Cakanyildirim, M.B. Freimer, and V. Melkonian. Capacity-driven acceptance of customer orders for a multi-stage batch manufacturing system: models and algorithms. *IIE Transactions*, 37:1093–1105, 2005.

[35] S. Sengupta. Algorithms and approximation schemes for minimum lateness/tardiness scheduling with rejection. *Lecture Notes in Computer Science*, 2748:79–90, 2003.

[36] S.A. Slotnick and T.E. Morton. Selecting jobs for a heavily loaded shop with lateness penalties. *Computers & Operations Research*, 23:131–140, 1996.

[37] S.A. Slotnick and T.E. Morton. Order acceptance with weighted tardiness. *Computers & Operations Research*, 34(10):3029–3042, 2007.

[38] R. Suri. *Quick Response Manufacturing. A Companywide Approach to Reducing Lead Times*. Productivity Press, 1998.

[39] J.M. Van Den Akker, C.A.J. Hurkens, and M.W.P. Savelsbergh. Time-indexed formulations for single-machine scheduling problems: column generation. *INFORMS Journal on Computing*, 12:111–124, 2000.

[40] B. Yang and J. Geunes. A single resource scheduling problem with job-selection flexibility, tardiness costs and controllable processing times. *Computers & Industrial Engineering*, 53:420–432, 2007.

[41] C. Yugma. Dynamic management of a portfolio of orders. *4OR: A Quarterly Journal of Operations Research*, 3:167–170, 2005.

[42] W.H.M. Zijm. Towards intelligent manufacturing planning and control systems. *OR Spektrum*, 22:313–345, 2000.