International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Single CNC machine scheduling with controllable processing times and multiple due dates

Mehmet Oguz Atan & M. Selim Akturk

To cite this article: Mehmet Oguz Atan & M. Selim Akturk (2008) Single CNC machine scheduling with controllable processing times and multiple due dates, International Journal of Production Research, 46:21, 6087-6111, DOI: 10.1080/00207540701262913

To link to this article: http://dx.doi.org/10.1080/00207540701262913

Published online: 09 Oct 2008.

International Journal of Production Research,
Vol. 46, No. 21, 1 November 2008, 6087–6111

Single CNC machine scheduling with controllable processing times and multiple due dates

MEHMET OGUZ ATAN† and M. SELIM AKTURK*‡

†Department of Industrial and Systems Engineering,
Lehigh University, Bethlehem, PA 18015, USA
‡Department of Industrial Engineering, Bilkent University,
Bilkent 06800, Ankara, Turkey

(*Revision received January 2007*)

In this study, we solve the single CNC machine scheduling problem with controllable processing times. Our objective is to maximize the total profit that is composed of the revenue generated by the set of scheduled jobs minus the sum of total weighted earliness and weighted tardiness, tooling and machining costs. Customers offer multiple due dates to the manufacturer, each coming with a distinct price for the order that is decreasing as the date gets later, and the manufacturer has the flexibility to accept or reject the orders. We propose a number of ranking rules and scheduling algorithms that we employ in a four-stage heuristic algorithm that determines the processing times for each job and a final schedule for the accepted jobs simultaneously, to maximize the overall profit.

Keywords: Scheduling; Total weighted tardiness and earliness; Multiple due dates; Controllable processing times; Heuristics; Order rejection

1. Introduction

In this study, we solve a scheduling problem in a single CNC machine environment. Although there is extensive research that deals separately with due date, pricing and order rejection considerations, controllable processing times and weighted earliness and tardiness penalties, there is no study that considers these issues collectively. We allow the flexibility to reject the jobs that do not provide profit, and hence determine the processing time of each job and schedule of the set of accepted jobs simultaneously. Common practice in the literature is rejecting the late jobs without considering whether they are still profitable. Customer determined due dates are rejected if they cause late deliveries in the study of Wester *et al.* (1992). Keskinocak *et al.* (2001) reject an order if it is not possible to start processing it without causing it to be completed late. Setting appropriate due dates that conform both to customer demand and manufacturer availability is another issue of consideration. The sensitivity of customers to the pricing policy is handled by the

*Corresponding author. Email: akturk@bilkent.edu.tr

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540701262913

game theoretical model presented by So (2000). Another study that relates the customer demand with delivery time and price is by Ray and Jewkes (2004). Geunes *et al.* (2006) proposed a single-stage planning model that combined pricing and order selection decisions to maximize the overall profit when production capacities are unlimited. Slotnick and Morton (2007) developed a branch-and-bound procedure to solve the job sequencing and job acceptance decisions jointly to minimize the total weighted tardiness. Engels *et al.* (2003) also allowed the possibility of not scheduling certain jobs such that these rejected jobs incurred a certain penalty. The overall objective is to minimize the sum of the weighted completion times of the jobs scheduled plus the sum of the penalties of the rejected jobs.

In our study, we assign multiple due dates to each customer order instead of dictating a single due date to a manufacturer, so that we provide the manufacturer a flexibility to choose the one that will not cause a congestion in the production. On the other hand, the customer pays a fair price according to the delivery time since the price of a part decreases as the due date gets later. This will be advantageous for both parties because the manufacturer will have different alternatives to schedule the job and consequently less difficulty in scheduling orders from other customers. The buyer will minimize the possibility of losses due to unexpected order delivery delays. The objective of the manufacturer is to decide on which due date (among the ones offered by the customer) to choose for a specific job and to determine its processing time while maximizing the total profit in light of manufacturing costs and weighted earliness/tardiness costs. As already discussed by Kaminsky and Lee (2001), reserving capacity may be helpful in order to prevent tardiness penalties. It is also seen that assigning jobs to pre-defined due dates is a way of reflecting the production availabilities of a manufacturer (Chand and Chhajed 1992). In order to capture the preferences of a customer in terms of delivery time, time windows and availability intervals are introduced in Charnsirisakskul *et al.* (2004) and Keskinocak *et al.* (2001), respectively. Different from these studies, we partition the delivery interval by providing alternative due dates in addition to the preferred due date and the deadline, and allow controllable processing times.

Although the vast majority of the studies in the scheduling literature assume that processing times are fixed, in fact utilization of flexible manufacturing systems makes it possible to control the processing times of jobs. By allocating additional resources or by changing machining parameters such as cutting speed and feed rate of a CNC machine, we can control the processing times. In the study of Daniels and Sarin (1989), job processing times are treated as decision variables that may be controlled through the assignment of an additional resource. They suggest a constructive procedure for developing the tradeoff curve between the number of tardy jobs and the total amount of allocated resource. Panwalkar and Rajagopalan (1992) find the optimal processing times, an optimal due date and an optimal sequence, where all jobs have a common due date and processing times are controllable with linear costs. Computationally efficient heuristics that consider controllable processing times in a single CNC machine environment are presented by Cheng *et al.* (1998) and Ng *et al.* (2003). Recently, Yang and Geunes (2007) studied a job selection problem with job tardiness and job compression costs to maximize the overall profit on a single machine. A tardiness cost is incurred if a job is completed after a given due date, and the job processing times can be controlled with a linear compression cost. They first present a compress-and-relax algorithm to handle the tardiness and linear

compression costs for a given fixed job sequence, and then employ a local search algorithm to generate different job sequences. In our study, we have an additional earliness cost term and hence a non-regular scheduling measure, so that we have to consider the possibility of inserting idle time. Furthermore, we have a nonlinear manufacturing cost function to represent the controllable processing cost component in the overall cost function. As already discussed in Gurel and Akturk (2007), the manufacturing cost of a typical machining operation (such as turning or milling) is a nonlinear convex function of its processing time. Handling linear compression costs is relatively easier since the overall problem can be formulated as an assignment problem as shown by Vickson (1980) and Cheng *et al*. (1996) for different scheduling measures, such as minimizing makespan or total completion time, for single or parallel machine environments.

On the other hand, there are also a number of studies on the weighted earliness and weighted tardiness problem. Ow and Morton (1989) proposed two dispatch priority rules and a filtered beam search method for the weighted earliness and tardiness problem with distinct due dates. Fry *et al*. (1984) suggest a heuristic solution that requires an enumeration procedure to solve the problem. In order to find lower and upper bounds for the problem, Azizoglu *et al*. (1991) assumed that there is no inserted idle time in the schedule and the earliness and tardiness penalties are the same. Hassin and Shani (2005) studied the scheduling problems with earliness/tardiness penalties where they also allowed that some jobs can be non-executed.

In this study, we aim to show how the flexibility of controlling the processing times can be utilized to solve the single CNC machine scheduling problem, while the interactions between the realistic features such as earliness and tardiness penalties, due dates and deadlines, and order acceptance and rejection decisions are carefully investigated. The remainder of this paper is organized as follows. In the following section, the problem is defined with its underlying assumptions. The solution properties that reflect the characteristics of the problem are explained in section 3. The proposed algorithms and ranking rules are presented in section 4. A numerical example is provided in order to clarify the basic steps of the proposed algorithm in section 5. In section 6, a computational study is performed to test the performance of the proposed algorithm. In the last section, some concluding remarks are provided.

2. Problem definition

In this study, we make the following assumptions. There are N jobs to be scheduled, which are all ready at time zero, without any precedence relations. There is a single CNC machine that is continuously available. The machine can produce one job at a time (i.e. non-interference constraints). Job specifications such as maximum allowable surface roughness, length and diameter of the surface, and depth of cut values are fixed and known. The required tool to produce each job is known in advance with associated tool parameters. Tools can be changed offline. Therefore, tool change time is negligible. Job loading and unloading times are neglected. Preemption is not allowed. The due dates, deadlines, tardiness and earliness penalties, prices on each due date and deadline are distinct and known in advance for each job.

Our objective is to maximize the total profit. The cost components are weighted earliness and tardiness, tooling, and manufacturing costs. In this study, the manufacturing cost is the summation of machining and tooling costs. We can decrease the processing time, and hence machining cost, by increasing the speed and feed rate. However, this will increase the tooling costs due to additional tool wear. We use a nonlinear tooling cost function to represent the controllable processing costs in a CNC manufacturing environment. The total weighted tardiness and earliness vary according to the change in the sum of processing times. Furthermore, when processing times are small, it is possible to complete jobs on earlier due dates to obtain higher prices. The processing time of a job is controllable, and can take any value between the lower and upper bounds that are calculated by solving the single machine operation problem (SMOP). For the associated mathematical model and the steps for the calculation of the lower and upper bounds, we refer to Kayan and Akturk (2005). Consequently, the manufacturer should decide which due date to choose, which processing time to use and when to schedule the part altogether, while trying to maximize the total profit.

The notation used throughout the paper is as follows.

Parameters:

T	planning horizon
p	index of a job, $p = 1, \ldots, N$
D_p^i	due date i of job p
$\bar{D}_p$	deadline for job p
χ_p	total number of due date settings (including the deadline) for job p
Pr_p^i	price of job p when it is delivered at due date i
$\bar{Pr}_p$	price of job p when it is delivered at its deadline
τ_p	unit tardiness penalty for job p
ϵ_p	unit earliness penalty for job p
$mt_p^{\mathrm{l}}, mt_p^{\mathrm{u}}$	lower and upper bounds for the processing time of job p
Ω	operating cost of the CNC machine (\$/min)
A_p, B_p	tooling cost multiplier and exponent for job p

Decision variables:

Z_p	equal to 1 if job p is accepted for processing, and 0 otherwise
prof_p	profit obtained by processing job p
rev_p	revenue obtained by processing job p
mt_p	processing time of job p
s_p	starting time of job p
c_p	completion time of job p
$w_{pp'}$	equal to 1 if job p precedes job p' (not necessarily immediately), and 0 otherwise

Given the completion time of a job, the due date that gives the maximum difference between price and weighted earliness/tardiness cost is agreed on for delivery. Note that if a job is completed before the first due date, revenue is Pr_p^1 minus the earliness penalty, while a completion after the deadline provides no revenue. For the completion times that lie in the interval $[D_p^i, D_p^{i+1}]$, revenue is given by $\max\{Pr_p^{i+1} - \epsilon_p \cdot (D_p^{i+1} - c_p), Pr_p^i - \tau_p \cdot (c_p - D_p^i)\}$. The form of the revenue function can be observed in figure 1.

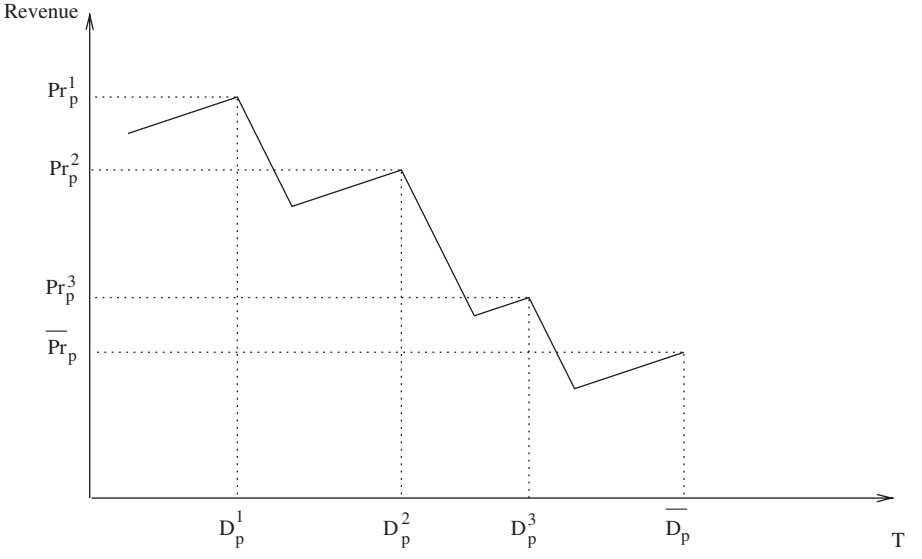

Figure 1. Revenue function.

The manufacturing cost of job p as a function of processing time mt_p can be calculated as $[\Omega \cdot mt_p + A_p \cdot (mt_p)^{B_p}]$. The first term is the operating cost which is the cost of running the CNC machine for job p and it is an increasing linear function of mt_p. The second term is the tooling cost which is the cost of tool usage for job p and it is a nonlinear decreasing function of mt_p. The parameters A_p and B_p are determined using tool specific (speed, feed, depth of cut exponents) and operation related (depth, diameter, length of cut, and surface roughness requirement) parameters. Since $A_p > 0$ and $B_p < 0$ always hold, the manufacturing cost is a nonlinear convex function. Furthermore, there exist processing time lower and upper bounds that are determined by the manufacturing properties of job p and the maximum applicable cutting power of the CNC machine. Therefore, mt_p^{l} and mt_p^{u} are also different for each job. The calculation of the cost components as well as lower and upper bounds on processing times can be found in Kayan and Akturk (2005).

The profit obtained from a job is found by subtracting the nonlinear manufacturing costs from the revenue as follows:

$$\mathrm{prof}_p = \mathrm{rev}_p - [\Omega \cdot mt_p + A_p \cdot (mt_p)^{B_p}]. \tag{1}$$

A mixed-integer nonlinear programming (MINLP) model for the problem is

$$\text{Maximize} \sum_p Z_p \cdot \mathrm{prof}_p, \tag{2}$$

subject to

$$mt_p^{\mathrm{l}} \le mt_p \le mt_p^{\mathrm{u}}, \quad \forall p, \tag{3}$$

$$Z_p \cdot (s_p + mt_p) \le s_{p'} + M \cdot (1 - w_{pp'}), \quad p, p' = 1, \ldots, N,\ p \ne p', \tag{4}$$

$$Z_p \cdot (s_{p'} + mt_{p'}) \le s_p + M \cdot w_{pp'}, \quad p, p' = 1, \ldots, N,\ p \ne p', \tag{5}$$

$$s_p, mt_p \ge 0 \quad \text{and } Z_p, w_{pp'} \in \{0, 1\}, \quad p, p' = 1, \ldots, N,\ p \ne p'. \tag{6}$$

Our objective is to maximize the total profit. Note that the profit obtained from each job is described by equation (1) and in figure 1. The binary variables in (2) provide the flexibility to reject jobs. Constraints (3) force the processing times to satisfy their corresponding upper and lower bounds. Furthermore, constraint sets (4) and (5) are non-interference constraints such that only one job can be processed at a time on the CNC machine, where M is a very large positive number. It is also straightforward to modify the above formulation to portray high competition in industry by assuming mandatory compliance or goodwill costs.

MINLP problems are very difficult to solve in general since they possess the difficulties of both mixed-integer programs and nonlinear programs. Our model maximizes the overall profit by choosing the optimal processing and completion times (among infinitely many possibilities) for each job. Moreover, not only optimization but also the calculation of the profit obtained from a job is a difficult problem on its own since we first need to identify the interval that the completion time lies and then figure out whether the job should be delivered on the earlier or the later due date after comparing the revenues for both dates. Therefore, in addition to the global maximization of overall profit, we have a second maximization problem embedded in our model to choose the revenue maximizing interval, and a third maximization problem to decide on the committed due date in that interval. Consequently, commercial MINLP solvers such as BARON and SBB fail to provide a global optimal solution for our model even of trivial problem sizes. In the following sections, we will present new solution properties and procedures that will help us accept and schedule jobs with higher profit opportunities.

3. Solution properties

In order to construct an efficient algorithm, we first determine some solution properties that will help us to reduce the computational burden and to capture the characteristics of the problem as well. The detailed explanation for these properties are presented below.

3.1 *Critical points*

For each interval $[D_p^i, D_p^{i+1}]$, the critical point $\kappa_{i,i+1}^p$ is defined to be the point that when a job is completed on it, the manufacturer is indifferent between choosing D_p^i or D_p^{i+1}. The manufacturer obtains the lowest possible revenue in a given interval when the job is completed on a critical point. Figure 2 displays the critical points on the revenue function. If $c_p < \kappa_{i,i+1}^p$, D_p^i is chosen for delivery, while D_p^{i+1} provides a greater revenue when $c_p > \kappa_{i,i+1}^p$. The critical point for each job p can be found as follows:

$$\kappa_{i,i+1}^p = \frac{(Pr_p^i - Pr_p^{i+1}) + (D_p^i \cdot \tau_p + D_p^{i+1} \cdot \epsilon_p)}{\epsilon_p + \tau_p}.$$

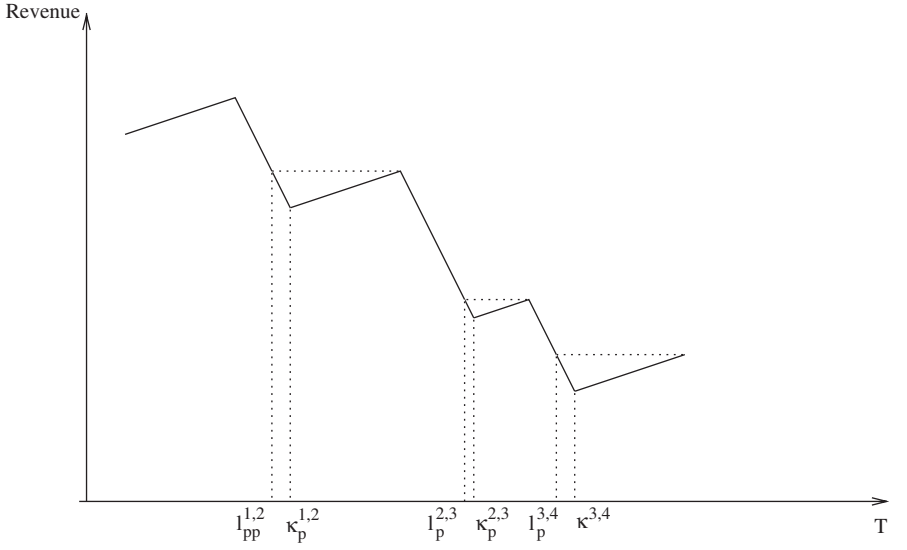

Figure 2. Critical points and same revenue points.

3.2 *Undesirable periods*

We define the completion time point that would provide as much revenue as completion on the next due date as a same revenue point. Undesirable periods start with a same revenue point $l^p_{i,i+1}$ and end with a critical point $\kappa^p_{i,i+1}$. Completing a job in this period is not desired because it is possible to obtain higher revenues if the job is finished before or after this period. Figure 2 illustrates these same revenue points, which are found by the formula

$$l^p_{i,i+1} = D^i_p + \frac{(Pr^i_p - Pr^{i+1}_p)}{\tau_p}.$$

3.3 *Critical processing time*

The critical processing time provides the completion time of the job at the point where the slope of the manufacturing cost is equal to the slope of the weighted tardiness cost line. Up to this point, for every unit increase in the processing time, the amount of savings obtained from the decrease of manufacturing cost is greater than the increase in the weighted tardiness cost. Beyond this point, savings from the manufacturing cost is less than the tardiness cost increase. The formula for critical processing time is given below, while the illustration is provided by figure 3.

$$mt^*_p = \left(\frac{-\tau_p - \Omega}{A_p \cdot B_p}\right)^{1/(B_p-1)}.$$

On the other hand, although the possibility is very low, for the instances that the critical processing time we find from the derivation is smaller than the lower bound of the processing time, $mt^*_p < mt^l_p$, we set $mt^*_p = mt^l_p$.

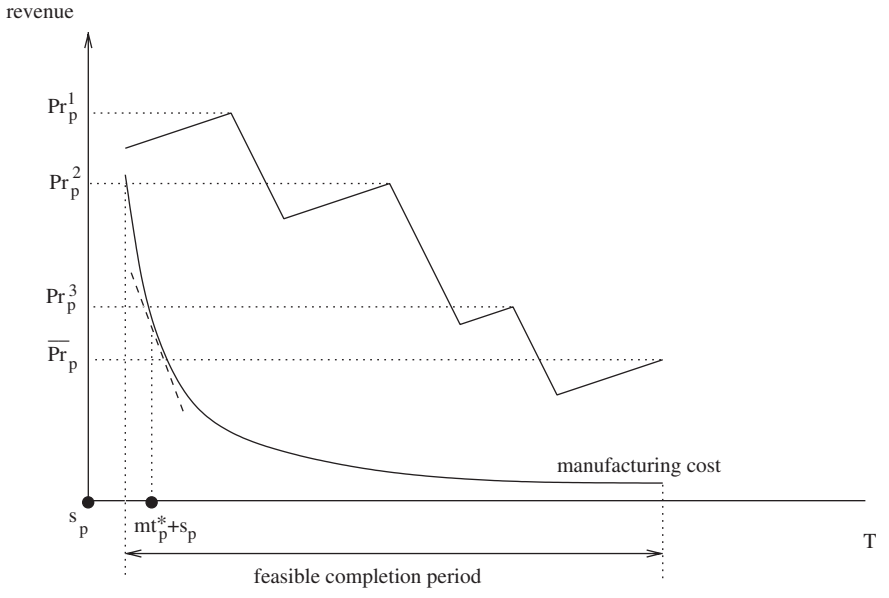

Figure 3. Critical processing time.

The maximum possible profit is obtained if the completion time of a job is exactly at the point where the difference between the revenue function and the manufacturing cost function is the largest. However, since our revenue function behaves differently in different regions, we need to consider each region separately while finding the maximum profit, and then choose the profit maximizing completion time point among them. In this problem, we have to make the following decisions at the same time: the processing time of each job (since we have controllable processing times), the starting time of each job, and the completion time of each job as a function of the selected processing time to find the final schedule that maximizes the total profit. Furthermore, we have a non-regular scheduling measure and, hence, inserting idle time could improve the overall total profit measure. In summary, at each decision point, we have to select the next job to be scheduled, whether we should insert idle time or not before the start time, and determine the job processing time, which will collectively specify the individual job completion times. Inserted idle time can take any real value. Similarly, individual job processing times can take any real value between the given lower and upper bounds. Therefore, there are infinitely many values for a possible completion time of each job. Using the proposed solution properties, we present a new lemma that will be helpful in any exact or approximate scheduling algorithm. In this lemma, at any time t for a job that will be scheduled at position $[r]$, the number of completion time points, among which we should consider before choosing the best one to be the completion time, is restricted to a limited number of completion time points as shown in Lemma 3.1. This property is defined as a time-based (or position based) local dominance rule, since it only specifies local optimality conditions based on the available information at time t (or position $[r]$) and the selection based on the local dominance rules may not necessarily lead to a global optimum solution. Local dominance rules are extremely

useful in branch-and-bound type exact algorithms, in local search algorithms, or dispatching rule based approximation algorithms where we have to make a decision at each node (or at position [r]) or each time point t as already shown by Akturk and Ozdemir (2000) for a branch-and-bound algorithm and by Avci *et al.* (2003) for a dispatching rule based local search algorithm. For a single-machine weighted tardiness problem, Avci *et al.* (2003) have shown that searching over a set of locally optimum solutions will guide the search process to the areas most likely to contain good solutions.

Lemma 3.1: *A job at position* [r] *should be completed on one of the following points*:

(i) $c_{p_{[r-1]}} + mt^{*}_{p_{[r]}}$, *i.e. the critical processing time is chosen as the processing time*;
(ii) $D^{k}_{p_{[r]}}$ *such that* $D^{k}_{p_{[r]}} \leq c_{p_{[r-1]}} + mt^{u}_{p_{[r]}}$, *i.e. completion time is at one of the due dates that are no later than the maximum completion time, when there is no inserted idle time. The maximum completion time for job* $p_{[r]}$ *is defined as* $c_{p_{[r-1]}} + mt^{u}_{p_{[r]}}$;
(iii) $D^{k}_{p_{[r]}}$ *such that* $D^{k}_{p_{[r]}} = min\{D^{i}_{p_{[r]}}\}$ *where* $D^{i}_{p_{[r]}} > c_{p_{[r-1]}} + mt^{u}_{p_{[r]}}$, *i.e. completion time is at the earliest due date after the maximum completion time, when there is no inserted idle time.*

Proof: While scheduling a job, we have two opportunities; we can start processing either immediately when the preceding job is completed or after inserting some idle time. Therefore, we should investigate the problem in two different cases.

Case 1: $c_{p_{[r]}} = c_{p_{[r-1]}} + mt_{p_{[r]}}$ where $mt_{p_{[r]}} \in \{mt^{l}_{p_{[r]}}, mt^{u}_{p_{[r]}}\}$. Then, we can say that the completion time is in one of the following four region types.

(i) Region before the first due date. In this region the job is early. Therefore, we have an increasing revenue function. For the maximum profit, the job should be completed on the first due date.
(ii) Early regions (the regions between $\kappa^{p}_{i,i+1}$ and D^{i+1}_{p}, $i = \{1, \ldots, \chi_p - 1\}$). Since the revenue function is increasing, the maximum profit is obtained at the end of the region, which is the due date earlier than the maximum completion time.
(iii) Tardy regions (the regions between D^{i}_{p} and $\kappa^{p}_{i,i+1}$, $i = \{1, \ldots, \chi_p - 1\}$). The revenue function and the manufacturing cost function are both decreasing in tardy regions. Therefore, if $s_p + mt^{*}_{p}$ is in the tardy region, the profit maximizing processing time is mt^{*}_{p}. If the region is later than $s_p + mt^{*}_{p}$, D^{i}_{p} is the maximizing completion time while $\kappa^{p}_{i,i+1}$ is the one for earlier regions. However, we have already shown that critical points are profit minimizers for early regions. Thus, we do not need to consider them.
(iv) Region after the deadline. The deadline is the only point that provides a positive revenue.

Consequently, without an inserted idle time, the job is finished before the maximum completion time, either using the critical processing time or on a due date (including the deadline).

Case 2: Idle time is inserted before the processing of the next job in the ranking order. We always prefer increasing the processing time instead of inserting idle time, because manufacturing costs decrease in that case. Therefore, the only case we would

choose inserting idle time is when we are already at the upper bound on the processing time.

(i) In early regions, time is inserted to delay the completion until the due date.
(ii) If the maximum completion time point is in a tardy region:
  (a) Before the same revenue point, we cannot increase our profit by inserting idle time.
  (b) After the same revenue point, the maximum revenue is obtained by delaying the completion time until the next due date by inserting an idle time.

Therefore, if we are inserting idle time, the only point that we would like to complete the job is the first due date after the maximum completion time point. □

4. The solution procedure

In this study, we propose a four-stage algorithm to solve the problem. In the first stage, we find the ranking order to determine in which position the jobs are going to be handled by the scheduling algorithms. In the second stage, referred to as the initial scheduling stage, we construct an initial feasible schedule in reasonably small computation times. In the third stage, we try to improve the objective function value by increasing the number of accepted jobs. Finally, in the last stage, we introduce the controllable processing times and calculate the optimum processing times for a given sequence found in the previous stage using the commercial GAMS/MINOS solver. The detailed information about the stages can be found in the following sections.

4.1 *Stage 1—ranking the jobs*

In this first stage, we choose the dispatching rule that will rank the jobs to determine the sequence they are going to be scheduled. In order to maintain efficient sequences, we implemented 12 different dispatching rules among which we will choose the one that captures the specifications of our problem the best. The ranking rules and their priority indices are summarized in table 1. Among these rules, SPT, LPT, WPD, WSPT, and EDD are static dispatching rules, while the remaining rules are dynamic. Note that the generalizations of the ATC rule are developed to take earliness penalties into account. In all these ATC based rules, we choose the look-ahead parameter k to be equal to 2, while $\bar{\epsilon}$ is equal to the sum of earliness penalties and $\bar{p}$ is equal to the sum of processing times of the unscheduled jobs at their lower bound.

4.2 *Stage 2—initial scheduling*

In the second stage, the processing times and start times of jobs are determined simultaneously to construct an initial schedule. Furthermore, the decision of rejecting an order is also under consideration. Our aim is to build a schedule in a reasonable computation time that is open to further improvement. In order to construct an initial schedule, we propose three algorithms: the Schedule-ahead

Table 1. Ranking rules.

Rule	Rank and priority index
SPT	$\min(mt_p)$
LPT	$\max(mt_p)$
WPD	$\max\left(\frac{\tau_p}{mt_p \cdot D_p^1}\right)$
WSPT	$\max\left(\frac{\tau_p}{mt_p}\right)$
EDD	$\min(D_p^1)$
COVERT	$\max\left(\frac{\tau_p}{mt_p} \cdot \max\left[0, 1 - \frac{\max(0, D_p^1 - t - mt_p)}{k \cdot mt_p}\right]\right)$
ATC	$\max\left(\frac{\tau_p}{mt_p} \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}}\right]\right)$
ATC-2	$\max\left(\frac{\tau_p}{mt_p} \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}}\right] \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}} \cdot \frac{\epsilon_p}{mt_p}\right]\right)$
ATC-3	$\max\left(\frac{\tau_p}{mt_p} \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}}\right] \cdot \exp\left[\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}} \cdot \frac{\epsilon_p}{mt_p}\right]\right)$
ATC-4	$\max\left(\frac{\tau_p}{mt_p} \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}}\right] \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}} \cdot \frac{\epsilon_p + \tau_p}{\epsilon_p}\right]\right)$
ATC-5	$\max\left(\frac{\tau_p}{mt_p} \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}}\right] \cdot \exp\left[-\frac{\epsilon_p}{k \cdot \bar{\epsilon}}\right]\right)$
ATC-6	$\max\left(\frac{\tau_p}{mt_p} \cdot \exp\left[-\frac{\max(0, D_p^1 - t - mt_p)}{k \cdot \bar{p}}\right] \cdot \exp\left[-\frac{\epsilon_p + \tau_p}{\epsilon_p}\right]\right)$

algorithm (SA), the Back-forth algorithm (BF), and the Crush-back-forth algorithm (CBF), as described below. In all algorithms, we decide on the processing time, mt_p, and the completion time, c_p, for each job p simultaneously.

4.2.1 Schedule-ahead algorithm. We schedule one job at a time to obtain the greatest possible profit. In the initial schedule, the sequence will be the same as the job order given by the selected dispatching rule, although there might be some rejected jobs in the initial schedule due to the deadline constraint. The algorithm is terminated when the last job of the sequence is scheduled or rejected. Step 1 rejects a job if it is impossible to complete before the deadline. Step 2 considers completion on the preferred due date using mt^{u}, which is the most profitable way to process a job. Steps 3 and 4 use Lemma 3.1 to schedule the job providing the greatest possible profit, as outlined below.

Step 0: $T \leftarrow 0, \quad r \leftarrow 1$. While $r \leq N$, do:

Step 1: If deadline cannot be met even if $mt_{[r]}^{\mathrm{l}}$ is used, reject the job. $r = r + 1$. Repeat Step 1.

Step 2: If $T + mt^{u}_{[r]} < D^{l}_{[r]}$, set $c_{[r]} = D^{l}_{[r]}$, $T = c_{[r]}$, $mt_{[r]} = mt^{u}_{[r]}$. $r = r + 1$. Go to Step 1.

Step 3: Calculate the profits that would be obtained when:

(i) $mt^{*}_{[r]}$ is used without inserted idle time;
(ii) job is completed on a due date without using inserted idle time;
(iii) job is completed on the earliest due date after $T + mt^{u}_{[r]}$, using inserted idle time.

Step 4: If the maximum of the profits calculated in Step 3 is positive, schedule the job using the corresponding policy and update $T = c_{[r]}$. Otherwise, reject the job. $r = r + 1$. Go to Step 1.

4.2.2 Back-forth algorithm. This algorithm is developed to provide the possibility of scheduling jobs not only after previously scheduled jobs but also at some point in the schedule without violating the non-interference constraints. Thus, the sequence after the initial schedule is completed will not be the same as the rank order obtained from the dispatching rule. Every time a new job is being handled, the algorithm determines all idle time blocks in the schedule and chooses the one that would provide the maximum profit, using the settings offered by Lemma 3.1.

After scheduling the first job using the SA algorithm in Step 1, the BF algorithm finds all idle time blocks in Step 2. The number of idle time blocks may change at each iteration, and is denoted by η. Furthermore, $\underline{id}_{[y]}$, $\overline{id}_{[y]}$, and $|id_{[y]}|$ refer to start time, end, and length of the yth idle time block from the beginning of the schedule. For each idle time block, we check if the job is acceptable and if it can be scheduled before the first due date, in Steps 2.1 and 2.2, respectively. Then, in Steps 2.3 and 2.4, when the idle time is greater than mt^{u} or mt^{l}, we look for the best way of scheduling the job while trying to maximize the profit using Lemma 3.1, respectively. The remaining steps choose the most profitable idle time block and schedule the job, or reject it, as outlined below.

Step 0: $T \leftarrow 0$, $r \leftarrow 1$.

Step 1: Schedule the job using the SA algorithm. Set $T = c_{[r]}$, $r = r + 1$. While $r \leq N$, do:

Step 2: Find η. Set the values for $\underline{id}_{[y]}$ and $\overline{id}_{[y]}$, $y = \{1, \ldots, \eta\}$. Set $y = 1$.

Step 2.1 If $\underline{id}_{[y]} + mt^{l}_{[r]} > \bar{D}_{[r]}$, reject the job, $r = r + 1$, repeat Step 2.1. Else, go to Step 2.2.

Step 2.2: If $|id_{[y]}| > mt^{u}_{[r]}$ and if $D^{l}_{[r]}$ is met using $mt^{u}_{[r]}$, go to Step 2.2.1. Else, go to Step 2.3.

Step 2.2.1: If $D^{l}_{[r]} < \overline{id}_{[y]}$, complete on the due date. Otherwise, complete on $\overline{id}_{[y]}$. Use $mt^{u}_{[r]}$ in both cases. Calculate the profit. Go to Step 3.

Step 2.3: If $|id_{[y]}| > mt^{u}_{[r]}$, go to Step 2.3.1. Else, go to Step 2.4.

Step 2.3.1: Using Lemma 3.1, calculate the profit that would be obtained when the job is:

(i) completed on the first due date after $\underline{id}_{[y]}$ using $mt^{u}_{[r]}$;

(ii) completed on the first due date after $\underline{id}_{[y]}$, before the maximum completion time;
(iii) started on $\underline{id}_{[y]}$ using the critical processing time, $mt^{*}_{[r]}$;
(iv) started on $\underline{id}_{[y]}$ using $mt^{u}_{[r]}$;
(v) completed on $\overline{id}_{[y]}$ using $mt^{u}_{[r]}$;
(vi) started on $\underline{id}_{[y]}$ using $mt^{l}_{[r]}$.

Step 2.3.2: Get the maximum profit obtained in Step 2.3.1. Go to Step 3.

Step 2.4: If $|id_{[y]}| > mt^{l}_{[r]}$, go to Step 2.4.1. Else, go to Step 3.

Step 2.4.1: Using Lemma 3.1, calculate the profit that would be obtained when the job is:

(i) started on $\underline{id}_{[y]}$ and completed on the next due date, which is after $\underline{id}_{[y]} + mt^{*}_{[r]}$;
(ii) started on $\underline{id}_{[y]}$ and completed on $\overline{id}_{[y]}$;
(iii) started on $\underline{id}_{[y]}$ using the critical processing time, $mt^{*}_{[r]}$.

Step 2.4.2: Get the maximum profit obtained in Step 2.4.1. Go to Step 3.

Step 3: If profit obtained for this block is greater than the profit obtained by any of the previous blocks, keep the profit, policy and the block information. $y = y + 1$. If $y \leq \eta$, go to Step 2.1.

Step 4: If the maximum profit after all blocks are considered is non-positive, reject the job. Else, schedule the job using the block and policy information corresponding to the maximum profit. Reset the profit information for the next job. $r = r + 1$, go to Step 2.

The main advantage of the Back-forth algorithm when compared to the Schedule-ahead algorithm is that it provides greater utilization in the production schedule. Since the Schedule-ahead algorithm schedules the jobs to obtain the maximum profit, the first job in the sequence is scheduled to be completed on the first due date. Therefore, especially when the first job in the sequence has a considerably late first due date, then a large portion of the time horizon, which is before the start time of the first job, becomes unavailable for scheduling the remaining jobs. In this respect, the possibility of creating large idle time blocks between other jobs in the schedule is also greater in the schedules that are constructed by the Schedule-ahead algorithm. However, since the Back-forth algorithm allows jobs to be scheduled at any place on the time horizon, we rarely encounter such instances with large portions of idle time blocks.

4.2.3 Crush-back-forth algorithm. The Crush-back-forth algorithm is able to decrease the processing times of jobs that were scheduled previously. In this algorithm we can also use the idle time blocks that are smaller than the lower bound on the processing time of the job we are trying to schedule. Other than these, it works exactly in the same way the Back-forth algorithm does. The steps of the algorithm are the same up to Step 3 of the Back-forth algorithm, before which it continues with checking the crushing possibility in substeps of 2.5, when mt^{l} is greater than the block length. In particular, in order to obtain the CBF algorithm, the following are inserted into the steps of BF algorithm just before Step 3. Moreover, in CBF,

Step 2.4 should send us to Step 2.5 instead of Step 3, if the condition is not satisfied. We use $[y^+]$ to denote the job that starts at the end of idle time block y.

Step 2.5: If $|id_{[y]}| < mt^1_{[r]}$, go to Step 2.5.1. Else, go to Step 3.

Step 2.5.1: If the processing time of the job $[y^+]$ can be decreased enough to fit job $[r]$ in block y, go to Step 2.5.2. Else, go to Step 3.

Step 2.5.2: If the profit obtained by scheduling job $[r]$ in block y is greater than the loss in profit obtained from the job $[y^+]$, keep this difference as the profit obtained from scheduling job $[r]$. Go to Step 3.

The Crush-back-forth algorithm is expected to work better especially in tight scheduling horizons that contain many small idle time blocks. If the earlier jobs in the ranking order are scheduled at noticeably larger processing times, then the scheduling horizon, particularly the ones with tight due date assignments, may not contain enough space for scheduling of the later jobs in the ranking order, except the idle time blocks that are even smaller than the minimum processing times of jobs. In such cases, a greater number of jobs can be scheduled than could be in the Back-forth algorithm. More efficient initial schedules that contain less idleness can be obtained using the Crush-back-forth algorithm.

4.3 *Stage 3—improving the initial schedule*

In the initial scheduling algorithm, we try to schedule one job at a time based on the ranking order. Consequently, it is possible to still have a place for scheduling a rejected job that will yield a positive profit. Therefore, the aim of this stage is to insert previously rejected jobs into the schedule. We present two improvement algorithms below that could be helpful to increase the number of accepted jobs in the schedule with a positive profit.

4.3.1 Ranked-insertion algorithm. The jobs that were rejected by the initial scheduling algorithm are considered in the same order obtained by the dispatching rule. The algorithm works in the same way the Crush-back-forth algorithm works, and it is terminated when all jobs in the ranking order are considered. The steps for the Ranked-insertion algorithm (RDI) are summarized below.

Step 1: From the ranking order, get the jobs that could not be scheduled by the initial scheduling algorithm. Let the number of unscheduled jobs be ζ. Sequence the jobs starting from 1 to ζ, without changing the order of the jobs with respect to each other. Set $r = 0$.

Step 2: Use the Crush-back-forth algorithm, starting from Step 2. Stop when the Crush-back-forth algorithm stops.

4.3.2 Rankless-insertion algorithm. In the Rankless-insertion algorithm (RLI), we evaluate all job–idle time block combinations for all rejected jobs and select the one that provides the largest positive profit. After a job is scheduled, the idle time block information is updated, and a new search begins. If there exists no combination to yield a positive profit, the algorithm is terminated.

4.4 *Stage 4—improvement via MINOS*

The overall aim of the last stage is to calculate the optimum processing times for the accepted jobs using the commercial GAMS/MINOS 5.3 solver under the assumption that the sequence found in the previous step is fixed. If the processing time of a job is decreased, not only will its completion time be decreased, but also the completion times of the succeeding jobs as well. Therefore, the amount of reduction in the total completion time and its impact on the total profit is directly related to the position of the job in the given sequence. As a result, we maximize the following objective function:

$$\text{Maximize } \text{Reg}_1 + \text{Reg}_2,$$

subject to

$$mt_p^{\text{l}} \leq mt_p \leq mt_p^{\text{u}},$$

where

$$\begin{aligned}\text{Reg}_1 = {} & \min\left[\phi_p, \left(\sum_{i<p}\delta_i\right)\right]\tau_p - \max\left[0, \left(\sum_{i<p}\delta_i\right) - \phi_p\right]\epsilon_p \\ & - \min[\phi_p, \delta_p]\tau_p - \max[0, \delta_p - \phi_p]\epsilon_p \\ & - A_p[(mt_p - \delta_p)^{B_p} - (mt_p)^{B_p}] + \delta_p\Omega\end{aligned}$$

and

$$\begin{aligned}\text{Reg}_2 = {} & -\min\left[\phi_p, \left(\sum_{i<p}\delta_i\right)\right]\epsilon_p + \max\left[0, \left(\sum_{i<p}\delta_i\right) - \phi_p\right]\tau_p \\ & - \min[\phi_p, \delta_p]\epsilon_p + \max[0, \delta_p - \phi_p]\tau_p \\ & - A_p[(mt_p - \delta_p)^{B_p} - (mt_p)^{B_p}] + \delta_p\Omega.\end{aligned}$$

The Reg_1 term in the objective function is written for jobs that are tardy or on time, while Reg_2 is for all jobs that are early. The time period that a job is tardy or early is denoted by ϕ_p, whereas the amount of compression for a job is denoted by δ_p. The first lines of Reg_1 and Reg_2 refer to the increase in profit of a job by early completion due to the decrease in the previous jobs' processing times, while the second lines correspond to the decrease of that job's own processing time. Finally, the last lines refer to the increase in the tooling cost and the decrease in machining cost due to the decrease in the processing time of that job. Since the NLP solvers are not successful in solving problems that contain functions that have discontinuous derivatives, we used the standard reformulation approach for min and max functions. The smooth GAMS approximation we used for $\max(f(x), g(y))$ and $\min(f(x), g(y))$ are $(f(x) + g(y) + [(f(x) - g(y))^2 + \theta^2]^{1/2} - \theta)/2$ and $(f(x) + g(y) + [(f(x) - g(y))^2 + \theta^2]^{1/2} + \theta)/2$, respectively. The approximation error is $\theta/2$ when $f(x) = g(y)$, and decreases with the difference between the two terms.

Table 2. Job data used in the example.

Job	mt_p^l	mt_p^u	mt_p^*	τ_p	ϵ_p	D_p^1	D_p^2	D_p^3	$\bar{D}_p$	Pr_p^1	Pr_p^2	Pr_p^3	$\bar{Pr}_p$
1	0.09	1.31	0.67	4.9	1.4	6.71	8.14	11.35	14.25	81.65	62.21	42.77	23.33
2	0.99	6.28	3.42	4.8	1.5	5.51	9.25	13.47	17.84	56.66	43.67	30.69	17.70
3	0.32	4.65	2.26	4.8	1.5	6.12	7.23	8.35	9.54	28.73	22.14	15.56	8.978
4	0.55	4.94	2.69	3.7	1	12.33	14.57	17.98	20.01	60.57	45.84	31.10	16.37
5	2.11	13.84	6.42	5.4	1.3	8.56	9.34	10.03	10.99	56.94	42.53	28.11	13.70

5. Numerical example

We consider a problem instance with five jobs, the details of which are provided in table 2. We did not provide the calculations for the lower and upper bounds of the processing times, since we will focus on the scheduling aspect. The operating cost, Ω, and maximum available machine power, HP, combination is set at \$1.2/min and 30 hp for this example, respectively. We will only demonstrate how the Crush-back-forth algorithm works, since it also captures properties of the Schedule-ahead and the Back-forth algorithms. We used the COVERT rule to initially rank the jobs in Stage 1.

5.1 *Crush-back-forth algorithm*

- The first job is job 1. Since $mt_1^u < D_1^1$, we set $mt_1 = mt_1^u = 1.313$, $s_1 = 5.397$, and $c_1 = D_1^1 = 6.71$.
- The second job in the ranking order is job 3. We search for idle time blocks. The first one is (0, 5.39), and the other one starts at 6.71.
  (i) The first block is large enough to schedule the job at mt_3^u. Profit is 17.50, when $c_3 = 5.39$.
  (ii) For the second block, there is no profitable way:
    (a) when $c_3 = D_3^2$, the revenue is 22.14, while the manufacturing cost is 67.99;
    (b) when $c_3 = D_3^3$, the revenue is 15.56, the manufacturing cost is 18.38;
    (c) when $c_3 = \bar{D}_3$, the revenue is 8.97. However, the manufacturing cost is 11.79;
    (d) when $mt_3 = mt_3^*$, the revenue is 8.12, and the manufacturing cost is 13.76.
- The third job is job 2. Idle time blocks are (0, 0.73) and (6.71, −). mt_2^l is less than 0.73, but crushing is not profitable. Thus, the job is scheduled at (6.71, 9.25) according to Lemma 3.1.
- The next job is job 4. According to the lemma, the alternatives for the second idle time block are finishing on D_4^1, D_4^2 or starting at 9.25 and using mt_4^u. The job can also be fit into the first idle time block. Among these choices, the first one is the most profitable, with $mt_4 = 3.08$ and $c_4 = 12.33$.
- The last job is job 5. We have two idle time blocks, (0, 0.73) and (12.33, −).
  (i) The second block starts after the deadline, so we cannot use this block.

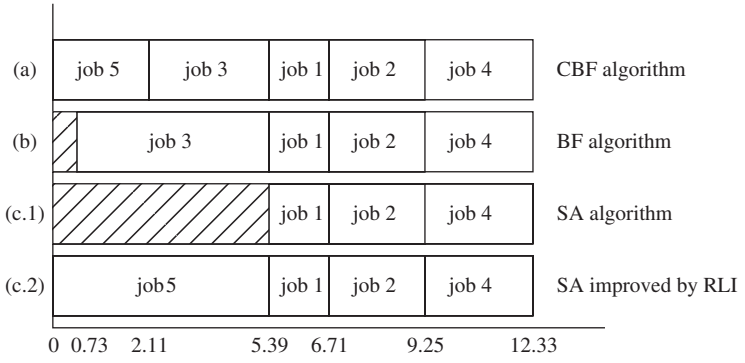

Figure 4. Gantt chart for schedules constructed by different algorithms.

(ii) The length of the first block is less than mt_5^1. Therefore, we need to crush job 3 in order to fit job 5 in this block. The required amount of crush is $2.11 - 0.73 = 1.38$, which is greater than $mt_3^1 = 0.32$. The additional manufacturing cost is 9.25, while the gain is 13.4 when we schedule job 5 to start at 0 and finish at 2.11.

We obtain \$148.62 as the total profit in this case as shown in figure 4(a). On the other hand, for the same set of jobs, the Schedule-ahead (figure 4(c1)) and the Back-forth algorithms (figure 4(b)) provide total profits of \$126.97 and \$144.47, respectively. The schedules obtained by all three algorithms are given in figure 4.

The proposed improvement algorithms in Stage 3 could be used to improve any given schedule. Next, we illustrate how the Rankless-insertion algorithm could be used after the initial schedule is constructed by the Schedule-ahead algorithm given in figure 4(c1).

5.2 *Rankless-insertion algorithm*

- The idle time blocks are (0, 5.39) and (12.33, −).
  (i) Job 3 is under consideration. It can be scheduled into the first block with $mt_3^u = mt_3 = 0.73$. The additional profit is 17.51.
  (ii) Job 5 is under consideration. Since the end of the first block is earlier than D_5^1, the best way of scheduling it is to complete it at the end of the block, 5.39, with a profit of 31.18. The second idle time block is out of consideration, since $\bar{D}_5 < 12.33$.
- The most profitable is job 5. We set $s_5 = 0$, $mt_5 = 5.39$ and $c_5 = 5.39$.
- The idle time blocks are updated. The only remaining block is (12.33, −). The only unscheduled job is job 3. Since $\bar{D}_3 < 12.33$, we reject job 3.

The total profit we obtain in this case is 158.05, and the final schedule is given in figure 4(c2). When the Ranked-insertion algorithm is used, the resulting schedule is the same as in figure 4(a).

Table 3. Experimental design factors.

Factor	Definition	Level 1	Level 2	Level 3
N	Number of jobs	50	100	200
τ_p	Tardiness penalty weights	UN[3,7]	UN[7,12]	
RDD	Relative range of due dates	0.2	0.5	0.8
TF	Average tardiness factor	0.2	0.5	0.8
Ω, HP	Operating cost, machine power	0.8, 10	1.2, 30	

6. Computational results

We performed a computational study to test the performance of the proposed algorithms. All algorithms were coded in the C language and compiled with the Gnu C compiler. In the final improvement phase, the problem for a given schedule is formulated in GAMS 2.25 and solved by MINOS 5.3. All codes were run on a sparc station Sun Enterprise 4000 with 1024 MB memory and six CPU of 248 MHz, under SunOS 5.7. MINOS solver was run on a station with 512 MB memory and 2.4 GHz CPU, under Windows Xp. There are five primary experimental factors that affect the efficiency of our base heuristic which can be seen in table 3. The experimental design is a $2 * 2 * 3 * 3 * 3$ full-factorial design. We took five replications for each factor combination, resulting in 540 randomly generated runs.

The number of jobs to be processed, N, affects the load on the system. When the tardiness penalty weight, τ_p, is large, the manufacturer prefers scheduling the job at an earlier position, which will affect the revenues of other jobs. Two factors, TF and RDD, are employed to represent the difficulty of a specific earliness/tardiness problem. The first due date of each job was randomly generated from the following uniform distribution (UN):

$$D_p^1 = \mathrm{UN}[(1 - \mathrm{TF} - \mathrm{RDD}/2), \quad (1 - \mathrm{TF} + \mathrm{RDD}/2)] * \sum_{p=1}^{N} mt_p^1.$$

Deadlines were also randomly generated using a similar uniform distribution in which mt_p^u is used instead of mt_p^l. The remaining due dates between the first one and the deadline were assigned with equal time intervals between each of them. The last experimental factor is the combination of the operating cost, Ω, and the maximum available machine horsepower, HP. This combination reflects the technological attributes of a CNC machine such that at level 2 we consider a CNC machine which provides a greater machine power (or, equivalently, higher cutting speeds and feed rates), but incurs a higher operating cost in response.

The other variables are assumed to be fixed parameters. The earliness penalty weights, ϵ_p, were generated from the uniform distribution [1, 2]. For each job, we generated operation and tool related parameters as discussed in Kayan and Akturk (2005). Prices of jobs are generated using τ_p and the due dates. The price of a job at the first due date, Pr_p^1, is calculated by multiplying the unit tardiness penalty of that job, τ_p, by the length of the period between the first due date and the deadline, $\bar{D}_p - D_p^1$. For the remaining prices, the difference between the unit tardiness penalty

Table 4. Summary of total profit values after initial scheduling.

	Schedule-ahead			Back-forth			Crush-back-forth		
Ranking	Min	Max	Average	Min	Max	Average	Min	Max	Average
COV	2902	1 191 532	189 281	59	977 854	162 997	59	1 057 829	171 372
ATC	3343	1 122 948	162 018	66	974 967	164 127	66	988 545	169 114
ATC-2	2848	1 121 292	159 186	51	952 168	150 308	51	962 094	154 301
LPT	943	753 034	85 797	26	354 804	61 669	26	357 894	63 450
SPT	2668	1 095 670	159 495	398	976 333	183 903	398	1 020 420	192 761
WPD	3214	1 101 179	160 652	105	981 972	167 837	105	1 060 100	177 567
WSPT	2775	1 117 695	161 464	398	973 013	186 613	398	1 029 953	196 888
EDD	1419	989 834	136 463	107	603 783	102 575	107	604 805	104 907
ATC-3	2692	1 117 808	156 882	2808	973 620	183 434	2964	1 001 528	193 867
ATC-4	2848	1 124 979	162 515	51	961 835	152 230	51	1 059 561	163 571
ATC-5	3343	1 122 948	162 018	66	974 922	164 109	66	1 050 494	176 489
ATC-6	2335	959 561	122 033	66	750 373	132 739	66	827 331	142 673

Table 5. Deviation averages in percentages at initial schedule.

	SA		BF		CBF	
Ranking	Profit	CPU	Profit	CPU	Profit	CPU
COV	0.07	0.25	0.43	0.32	0.41	0.10
ATC	0.11	0.28	0.31	0.29	0.29	0.06
ATC-2	0.07	0.25	0.42	0.22	0.40	0.06
LPT	0.04	0.23	0.39	0.30	0.37	0.13
SPT	0.17	0.59	0.07	0.57	0.02	0.01
WPD	0.12	0.41	0.25	0.57	0.21	0.13
WSPT	0.16	0.57	0.07	0.50	0.02	0.01
EDD	0.04	0.25	0.36	0.35	0.34	0.14
ATC-3	0.18	0.21	0.06	0.29	0.02	0.07
ATC-4	0.09	0.27	0.39	0.24	0.35	0.07
ATC-5	0.10	0.22	0.27	0.32	0.23	0.10
ATC-6	0.15	0.18	0.27	0.24	0.23	0.07

and the unit earliness penalty, $\tau_p - \epsilon_p$, is multiplied by the length of the time interval between any two consecutive due dates, $D_p^{i+1} - D_p^i$, $i = \{1, \ldots, \chi_p - 1\}$, which is subtracted from the price at the preceding due date, Pr_p^i, to calculate Pr_p^{i+1}.

In order to construct an initial schedule, the Schedule-ahead, Back-forth, and Crush-back-forth algorithms were used with the rankings obtained from the 12 different dispatching rules that were previously described. A total of 6480 runs were taken for each algorithm. Table 4 shows that, on average, the Crush-back-forth algorithm performs the best in terms of total profit. However, we see that the Schedule-ahead algorithm obtains the maximum profit in all cases. If we normalize the total profit values, we can measure the performance of the algorithms in terms of percentage difference from the best result as summarized in table 5. The formula for the deviation, dev_h, of the result of a single run, r_h, is written by using the best and

Table 6. Averages for the number of scheduled jobs.

	SA			BF			CBF		
Ranking	$N = 50$	$N = 100$	$N = 200$	$N = 50$	$N = 100$	$N = 200$	$N = 50$	$N = 100$	$N = 200$
COV	41.82	76.08	130.20	21.82	57.18	97.98	22.22	58.30	101.86
ATC	39.94	70.26	112.87	26.21	63.25	101.70	26.79	64.36	103.86
ATC-2	39.36	69.83	111.87	17.12	53.23	90.77	17.37	53.92	92.79
LPT	30.80	49.14	67.24	15.26	42.57	43.00	15.89	43.45	44.05
SPT	38.79	69.02	112.91	37.72	74.51	126.54	38.77	76.48	131.18
WPD	39.44	71.01	111.22	31.19	65.51	100.03	32.09	67.15	104.73
WSPT	38.92	69.79	113.18	37.58	75.19	125.22	38.51	77.03	130.42
EDD	36.21	63.33	97.19	23.52	51.96	63.18	24.32	52.84	64.26
ATC-3	38.42	68.63	110.93	38.14	75.59	123.57	39.06	77.51	128.63
ATC-4	39.52	70.36	113.44	18.98	60.06	91.07	19.34	61.42	95.51
ATC-5	39.94	70.26	112.87	26.22	63.26	101.70	26.88	64.87	107.12
ATC-6	36.42	61.36	93.59	25.36	59.38	90.71	26.12	61.20	95.95

worst results, $\max_r$ and $\min_r$, respectively, achieved by any other algorithms in the same run for the same factor combination, as follows:

$$dev_h = \frac{\max_r - r_h}{\max_r - \min_r}.$$

In addition to the profit and CPU values, we should also evaluate the number of scheduled jobs (or, equivalently, the number of accepted jobs). In table 6, we present the average numbers for the number of scheduled jobs in three different cases, when $N = 50$, $N = 100$ and $N = 200$. We see that the Schedule-ahead algorithm was able to schedule more jobs than the other two algorithms in almost all cases. The reason might be due to the fact that, in the BF algorithms, we could schedule a job at a position earlier than the previously scheduled jobs. Therefore, we have a wider scheduling horizon with respect to a SA schedule, and thus we have more slack until a due date. Consequently, the BF algorithms use this wide scheduling horizon to decrease the manufacturing cost and choose to process the job at a higher processing time. As a result of this myopic choice, the scheduling horizon that we expect to be wide is consumed quickly by jobs with processing times close to their upper bounds. Therefore, in the BF case, we end up with a smaller number of scheduled jobs processed at higher processing times.

Next, we compared the 12 ranking rules in order to find the one that works best with the best initial scheduling algorithm. In table 7, we present the average deviations, best and worst objective function values obtained from 540 runs of each ranking order for the Schedule-ahead algorithm. Since there might be ties, the total number of bests and worsts can be greater than 540. It is clear that the most harmonious ranking rule with the Schedule-ahead algorithm is COVERT, and, hence, we conclude that the best way of scheduling jobs for construction of an initial schedule is ranking the jobs according to the COVERT rule and using the Schedule-ahead algorithm afterwards.

As discussed earlier, the RLI and RDI algorithms can be implemented to improve a given sequence. In table 8 we present the average number of accepted jobs after an

Table 7. Performance of ranking rules under the Schedule-ahead algorithm.

Ranking	Deviation	Best	Worst
COV	0.033	339	0
ATC	0.167	58	0
ATC-2	0.219	2	0
LPT	0.994	0	511
SPT	0.235	34	0
WPD	0.179	58	0
WSPT	0.188	16	0
EDD	0.523	3	23
ATC-3	0.220	10	0
ATC-4	0.200	19	0
ATC-5	0.167	57	0
ATC-6	0.492	0	6

Table 8. Average number of accepted jobs in the schedule after RDI/RLI is applied.

	BF-RLI			SA-RLI			SA-RDI		
Ranking	$N = 50$	$N = 100$	$N = 200$	$N = 50$	$N = 100$	$N = 200$	$N = 50$	$N = 100$	$N = 200$
COV	22.61	59.56	103.63	42.52	78.76	136.41	45.34	90.76	178.62
ATC	27.37	65.39	104.96	40.76	72.47	117.19	45.87	91.34	178.21
ATC-2	17.67	55.01	94.25	39.56	71.27	115.38	45.49	90.79	177.78
LPT	15.52	43.27	44.01	31.88	55.59	82.13	36.83	71.68	135.57
SPT	39.67	78.35	134.04	40.82	72.90	123.16	45.87	91.02	178.01
WPD	32.67	68.46	106.13	40.72	73.40	117.99	45.63	90.56	177.14
WSPT	39.49	78.79	132.91	41.02	73.24	123.09	46.01	91.27	178.53
EDD	24.16	52.91	63.97	37.46	66.51	104.73	41.97	83.27	159.06
ATC-3	39.94	78.98	129.81	40.97	73.74	120.84	45.97	91.31	178.49
ATC-4	19.63	61.86	93.56	39.94	72.03	117.26	45.36	91.12	177.73
ATC-5	27.37	65.42	104.98	40.52	72.10	116.92	45.87	91.36	178.21
ATC-6	26.42	61.41	93.87	37.34	64.42	100.02	44.30	87.27	170.83

improvement algorithm is utilized. Actually, the number of scheduled jobs is an important measure, since rejection of a job causes loss of goodwill at the customer site and causes a decrease in future sales due to the loss of customers. Therefore, using the SA–RDI combination will be the best way of ensuring continuity of customer orders. The difference between the RLI and RDI algorithms can be seen more clearly by the use of this table, since both algorithms start with the same initial schedules, but end up with schedules that are quite different from each other in terms of the number of accepted jobs.

In order to understand the capabilities of improvement algorithms, we need to investigate the percentage increases in the objective function value and the additional CPU used to obtain the improvement, in percentages. Since we are trying to construct a heuristic algorithm that obtains good solutions while using reasonable computational effort, the ratio of the percentage increase in the objective function value to the additional CPU usage constitutes an important measure.

Given that cpu_i and obj_i are CPU usage and objective function values for the initial scheduling stage and cpu_{i+1} and obj_{i+1} are CPU usage and objective function values for the improvement stage, a ratio can be calculated as

$$\frac{(\text{obj}_{i+1} - \text{obj}_i)/\text{obj}_i}{\text{cpu}_{i+1}/\text{cpu}_i}.$$

In table 9, for each ranking rule, the first column provides the percentage profit increase while the second includes the additional CPU usage in percentages, which are sorted according to the problem size. It is clear that the RDI algorithm is the most efficient algorithm in terms of unit additional CPU spent to obtain a unit improvement in total profit. We see that the RDI uses 35% additional CPU to improve the objective function value about 35% on the average, while the RLI uses 573% additional CPU to improve a BF schedule about 5% in terms of the objective function value, and 426% additional CPU to improve a SA schedule about 10%, on the average.

At the last stage we used the schedules obtained by the algorithm combinations that were discussed above and solved the controllable processing time problem using the MINOS solver. The locally optimal solutions given by the MINOS solver provided an additional profit of 5.7 to 7.9%. According to our computational results, our single-pass heuristic algorithm improves the objective function value at every step. The average increase in the objective function value with respect to the average additional computational effort is greater in earlier steps. However, although this ratio seems to be low when we come to the MINOS stage, the improvement is not negligible in terms of the objective function value. Among the 12 dispatching rules, three initial scheduling algorithms, and two improvement algorithms that we implemented, our single-pass heuristic, composed of COVERT–Schedule-ahead–Ranked-insertion–MINOS solver combination, was shown to perform the best.

7. Concluding remarks

The integration of different literature helps researchers to encounter more realistic problems. In this paper, our main aim is to integrate the related subproblems of scheduling, pricing and process planning to create a problem setting that demonstrates a realistic manufacturing environment. There is no study in the literature considering the machining condition optimization and total weighted earliness and weighted tardiness problems simultaneously in the existence of multiple due dates. From this perspective, our study is the first that considers these issues simultaneously. Since the problem is NP-hard, we developed a single-pass heuristic algorithm that is able to solve large-sized problems in short computation times, due to the proposed lemma that restricts the scheduling of jobs to certain settings. Therefore, the proposed solution properties are crucial, and they should be taken into consideration if any other scheduling algorithm is constructed in the future. Furthermore, we schedule all jobs that provide a profit without taking the amount of it into consideration. However, insertion of jobs that provide small earnings into the schedule may cause some jobs with good profit potential to be rejected. Therefore, finding new dispatching rules that can capture the pricing information

Table 9. Averages for the additional CPU used to obtain the additional profit, in percentages.

		COV		ATC		ATC-2		LPT		SPT		WPD	
	N	Prof	CPU	Prof	CPU	Prof	CPU	Prof	CPU	Prof	CPU	Prof	CPU
BF-RLI	50	1.67	45.58	3.01	256.42	1.28	22.55	1.68	197.87	5.91	59.64	3.68	336.00
	100	3.25	122.85	3.70	520.93	2.56	91.57	3.25	472.15	6.93	196.66	4.55	590.99
	200	7.62	746.61	4.81	1452.68	5.12	381.45	5.61	381.83	10.99	457.67	8.36	333.91
	Total	4.18	305.01	3.84	743.34	2.99	165.19	3.51	350.62	7.94	237.99	5.53	420.30
SA-RLI	50	1.55	9.60	2.03	13.34	0.38	15.38	5.20	208.45	6.46	132.49	3.63	71.30
	100	5.01	39.19	4.82	56.91	2.58	25.71	27.09	810.84	10.06	631.75	4.73	345.29
	200	9.32	235.85	7.97	301.12	5.77	187.23	61.80	2681.98	23.36	1873.57	12.92	1609.40
	Total	5.29	94.88	4.94	123.79	2.91	76.10	31.37	1233.76	13.29	879.27	7.09	675.33
SA-RDI	50	4.31	4.65	8.30	5.40	9.16	3.80	14.96	28.43	12.65	22.24	10.35	22.58
	100	13.38	8.56	23.75	8.61	23.94	6.50	40.53	42.31	29.45	50.66	21.89	53.69
	200	32.06	29.02	60.15	26.38	61.72	17.10	108.08	105.65	70.02	150.34	63.19	155.54
	Total	16.58	14.08	30.74	13.46	31.60	9.13	54.53	58.80	37.37	74.41	31.81	77.27
		WSPT		EDD		ATC-3		ATC-4		ATC-5		ATC-6	
BF-RLI	50	5.00	567.56	3.59	257.45	4.74	242.69	1.54	75.67	2.98	172.83	3.93	111.65
	100	5.89	2131.35	3.02	320.91	5.54	1228.49	3.20	79.25	3.71	717.64	5.40	175.39
	200	10.60	2885.26	2.21	395.20	9.12	2209.45	3.84	186.47	4.82	1847.86	6.50	345.14
	Total	7.16	1861.39	2.94	324.52	6.47	1226.88	2.86	113.80	3.84	912.78	5.28	210.73
SA-RLI	50	5.66	82.77	5.20	118.71	7.74	17.37	0.87	6.92	1.19	10.07	3.97	16.20
	100	7.61	318.77	9.18	532.61	11.42	67.84	3.13	28.93	3.55	50.67	10.36	114.09
	200	18.59	1891.71	17.33	1953.76	19.81	257.78	6.33	165.08	7.01	211.34	18.97	238.56
	Total	10.62	764.42	10.57	868.36	12.99	114.33	3.45	66.97	3.91	90.69	11.10	122.95
SA-RDI	50	11.51	22.89	12.43	22.66	12.99	4.22	8.29	3.73	8.30	4.00	18.05	5.41
	100	25.33	54.87	27.21	37.11	28.02	12.29	23.56	6.07	23.76	6.25	43.27	7.96
	200	59.32	154.39	59.30	113.05	65.82	20.44	57.59	14.99	60.15	23.14	120.36	19.81
	Total	32.05	77.38	32.98	57.61	35.61	12.32	29.82	8.26	30.74	11.13	60.56	11.06

in addition to the regular scheduling parameters is an important issue for future consideration.

References

Akturk, M.S. and Ozdemir, D., An exact approach to minimizing total weighted tardiness with release dates. *IIE Trans*., 2000, **32**, 1091–1101.

Avci, S., Akturk, M.S. and Storer, R.H., A problem space algorithm for single machine weighted tardiness problems. *IIE Trans*., 2003, **35**, 479–486.

Azizoglu, M., Kondakci, S. and Kirca, O., Bicriteria scheduling problem involving total tardiness and total earliness penalties. *Int. J. Prod. Econ*., 1991, **23**, 17–24.

Chand, S. and Chhajed, D., A single machine model for determination of optimal due dates and sequence. *Oper. Res*., 1992, **40**, 596–602.

Charnsirisakskul, K., Griffin, P.M. and Keskinocak, P., Order selection and scheduling with leadtime flexibility. *IIE Trans*., 2004, **36**, 697–707.

Cheng, T.C.E., Chen, Z.L. and Li, C.-L., Parallel-machine scheduling with controllable processing times. *IIE Trans*., 1996, **28**, 177–180.

Cheng, T.C.E., Chen, Z.L., Li, C.-L. and Lin, B.M.T., Scheduling to minimize the total compression and late costs. *Nav. Res. Logist*., 1998, **45**, 67–82.

Daniels, R.L. and Sarin, R.K., Single machine scheduling with controllable processing times and number of jobs tardy. *Oper. Res*., 1989, **37**, 981–984.

Engels, D.W., Karger, D.R., Kolliopoulos, S.G., Sengupta, S., Uma, R.N. and Wein, J., Techniques for scheduling with rejection. *J. Algor*., 2003, **49**, 175–191.

Fry, T.D., Armstrong, R.D. and Blackstone, J.H., Minimizing weighted absolute deviation in single machine scheduling. *IIE Trans*., 1984, **19**, 445–450.

Geunes, J., Romeijn, H.E. and Taaffe, K., Requirements planning with pricing and order selection flexibility. *Oper. Res*., 2006, **54**, 394–401.

Gurel, S. and Akturk, M.S., Considering manufacturing cost and scheduling performance on a CNC turning machine. *Eur. J. Oper. Res*., 2007, **177**, 325–343.

Hassin, R. and Shani, M., Machine scheduling with earliness, tardiness and non-execution penalties. *Comput. Oper. Res*., 2005, **32**, 683–705.

Kaminsky, P. and Lee, Z., Analysis of on-line algorithms for due date quotation. Working Paper, Department of Industrial Engineering, University of California, Berkeley, 2001.

Kayan, R.K. and Akturk, M.S., A new bounding mechanism for the CNC machine scheduling problems with controllable processing times. *Eur. J. Oper. Res*., 2005, **167**, 624–643.

Keskinocak, P., Ravi, R. and Tayur, S., Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Mgmt. Sci*., 2001, **47**, 264–279.

Ng, C.T.D., Cheng, T.C.E., Kovalyov, M.Y. and Lam, S.S., Single machine scheduling with a variable common due date and resource-dependent processing times. *Comput. Oper. Res*., 2003, **30**, 1173–1185.

Ow, P.S. and Morton, T.E., The single machine early/tardy problem. *Mgmt. Sci*., 1989, **35**, 177–191.

Panwalkar, S.S. and Rajagopalan, R., Single-machine sequencing with controllable processing times. *Eur. J. Oper. Res*., 1992, **59**, 298–302.

Ray, S. and Jewkes, E.M., Customer lead-time management when both demand and price are lead time sensitive. *Eur. J. Oper. Res*., 2004, **153**, 769–781.

Slotnick, S.A. and Morton, T.E., Order acceptance with weighted tardiness. *Comput. Oper. Res*., 2007, **34**, 3029–3042.

So, K.C., Price and time competition for service delivery. *Mfg. Serv. Oper. Mgmt*., 2000, **2**, 392–409.

Vickson, R.G., Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine. *Oper. Res*., 1980, **28**, 1155–1167.

Wester, F.A.W., Wijngaard, J. and Zijm, W.H.M., Order acceptance strategies in a production-to-order environment with setup times and due-dates. *Int. J. Prod. Res.*, 1992, **30**, 1313–1326.

Yang, B. and Geunes, J., A single resource scheduling problem with job-selection flexibility, tardiness costs and controllable processing times. *Comput. Ind. Engng.*, 2007, in press.