Computers & Operations Research 38 (2011) 771–781

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Single CNC machine scheduling with controllable processing times to minimize total weighted tardiness

M. Selim Akturk *, Taylan Ilhan

Department of Industrial Engineering, Bilkent University, Ankara, Turkey

ARTICLE INFO

Available online 22 September 2010

Keywords:
Single machine scheduling
Weighted tardiness
Tool management
Controllable processing time

ABSTRACT

Advanced manufacturing technologies, such as CNC machines, require significant investments, but also offer new capabilities to the manufacturers. One of the important capabilities of a CNC machine is the controllable processing times. By using this capability, the due date requirements of customers can be satisfied much more effectively. Processing times of the jobs on a CNC machine can be easily controlled via machining conditions such that they can be increased or decreased at the expense of tooling cost. Since scheduling decisions are very sensitive to the processing times, we solve the process planning and scheduling problems simultaneously. In this study, we consider the problem of scheduling a set of jobs on a single CNC machine to minimize the sum of total weighted tardiness, tooling and machining costs. We formulated the joint problem, which is NP-hard since the total weighted tardiness problem (with fixed processing times) is strongly NP-hard alone, as a nonlinear mixed integer program. We proposed a DP-based heuristic to solve the problem for a given sequence and designed a local search algorithm that uses it as a base heuristic.

1. Introduction

Buyer–manufacturer relationship plays an important role in business. Buyers desire reliable delivery times for meeting their own schedules. From the perspective of the manufacturers, each buyer has a different priority. All of these require manufacturers to consider weighted tardiness problem in their scheduling decisions. If manufacturers have a flexible manufacturing system, in addition to its other capabilities, they also gain a capability to be more competitive in meeting customer due dates. This capability is to be able to control processing times, which is a readily available feature on modern computer numerical controlled (CNC) machines.

This study deals with scheduling a set of jobs on a single CNC machine to minimize the total weighted tardiness, $1||\sum w_i T_i$. When we analyze the single machine total weighted tardiness problem, there are two important parameters, namely the processing time vector, $\overline{p}$, and due date vector, $\overline{d}$. In the literature, the $\overline{p}$ vector is treated as a hard constraint, i.e., we are not allowed to change it. On the other hand, the $\overline{d}$ vector is considered as a soft constraint that means we are allowed to deviate from the desired due dates but a certain cost penalty is incurred for these deviations. In this study, one of the most important objectives is to show that the processing times can be treated as decision variables as well and their cost impact can be measured in terms of the corresponding machining and tooling costs.

Processing times on a CNC machine are controlled by machining conditions. We can increase or decrease the processing time of a job by changing the machining conditions. However, there is an additional tooling cost which is incurred when we increase the cutting speed and/or feed rate. In the current literature, process planning and scheduling problems are solved sequentially. After calculating locally optimal process parameters that minimize the manufacturing cost, processing times are then passed to the scheduling level. In reality, however, the time it takes to process each part can be controlled (albeit at higher cost). Since it is well known that scheduling problems are very sensitive to processing times, a controllable processing time brings additional solution flexibility in finding solutions to the scheduling problem.

Processing time control and its impact on sequencing decisions and operational performance have been receiving an increasing attention in the scheduling literature. Most of the studies on scheduling with controllable processing times assume that the processing time is a linear function of the amount of resource allocated to the processing of the job as summarized in the recent survey of Shabtay and Steiner [11]. Since the analysis of linear cost functions is tractable, most of the current literature on controllable processing time problems focus on such functions (e.g. Vickson [14], Cheng et al. [3], Tseng et al. [13]). However, using linear cost functions does not reflect the law of diminishing returns. There are some papers, similar to our study, that relax the

* Corresponding author.
E-mail address: akturk@bilkent.edu.tr (M.S. Akturk).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.09.004

linearity assumption by using either a specific or a general type of convex decreasing resource consumption function. Al-Ahmari [1] minimize the sum of completion times in conjunction with determining the optimal machining conditions that specify the processing time of each job. Shabtay and Kaspi [9] and Gurel and Akturk [4] study the problem of minimizing the total weighted flow time on a single machine with controllable processing times using different nonlinear compression cost functions. Yedidsion et al. [17] minimize maximum lateness on a single machine, whereas Xu and Feng [16] develop a local search algorithm to study a single machine scheduling problem with fixed delivery dates.

The value of the controllable processing times becomes even more critical during the current economical crisis, since it allows companies to adjust their production resources more effectively to meet the due date requirements. As far as our problem is concerned, controllable processing times may constitute a flexibility in capacity since the available production time is no longer fixed and can be increased by compressing the processing times of jobs with, of course, an additional amount of cost. Thus, it brings up the trade-off between the revenue gained by satisfying due dates on time and the amount of compression cost.

This study considers the problem of scheduling a set of jobs on a single CNC machine to minimize the sum of total weighted tardiness, tooling and machining costs. This is the first study that uses the total weighted tardiness as a scheduling objective in addition to minimizing the manufacturing cost. The joint problem is NP-hard since the total weighted tardiness problem (with fixed processing times) is strongly NP-hard alone as showed by Lawler [7]. Therefore, no algorithm is likely to be proposed to solve the problem optimally in polynomial time. Hence, it is justifiable to try heuristic approaches to solve our problem.

In this study, we develop an efficient dynamic programming (DP) based algorithm that considers interactions among the jobs in a given sequence. The proposed algorithm minimizes the sum of total weighted tardiness, tooling and machining costs for a given sequence. We employ a problem space genetic algorithm (PSGA) that uses the proposed algorithm as a base heuristic to determine the processing and starting times of each job simultaneously to minimize the stated objective function. At each stage of the base DP algorithm, we generate a set of processing time alternatives for each job (i.e., states of the DP) in a recursive equation. After finding the processing time for the first job in the sequence, it generates the processing times for the rest of the jobs by using the recursive equations and alternative states of each job in a forward DP algorithm.

In the next section, we define the scope of this study and give the mathematical formulation of the problem. In Sections 3 and 4, we introduce our proposed local search and DP-based algorithms, respectively. Computational results are given in Section 5. Finally, we give concluding remarks in Section 6.

2. Problem definition

We are given N jobs, and each job may have a different due date, tardiness penalty and cutting tool type. The problem is scheduling these jobs on a single CNC machine in order to minimize the sum of total weighted tardiness, machining and tooling costs. CNC machine that is continuously available can process one job at a time. Preemption is not allowed and all jobs are ready at time 0. Furthermore, we assume that the CNC machine is equipped with an off-board tool magazine, and tools can be replaced while the machine is working without interrupting the actual cutting operation. Cutting speed, v_i, and feed rate, f_i, of the CNC machine constitute the machining conditions for each job i and they can easily be adjusted to new settings. The pair of (v_i, f_i) determines the processing time of each job.

The notation used for the mathematical formulation is given below:

Parameters

C_0	operating cost of the CNC machine ($/min)
p_i^u, p_i^l	upper and lower bounds for the processing time of job i
w_i	weight of job i
d_i	due date of job i

Decision variables

p_i	processing time of job i
s_i	starting time of job i
T_i	tardiness of job i

If we ignore the scheduling problem, the remaining problem is reduced to determine the optimum machining conditions, i.e. v_i^* and f_i^*, for each job that minimizes the sum of machining and tooling costs. This problem can be solved independently for each job since there is no coupling constraints among the jobs as discussed in Kayan and Akturk [5]. There is a one to one relationship between the machining conditions and processing time. We can calculate the processing time of each job as $p_i^u = (\pi D_i L_i)/(12 v_i^* f_i^*)$, where D_i and L_i are the diameter and length of the generated surface for the job i, respectively. The processing time in the optimal solution of this reduced problem gives an upper bound of the processing time, p_i^u, for our problem since if we increase the processing time above that value, the sum of machining and tooling costs also increase. Furthermore, the weighted tardiness is a regular scheduling measure so that the weighted tardiness cost does not decrease beyond this value as well. Moreover, the available horsepower of the CNC machine and the cutting tool life place upper bounds on the cutting speed, whereas the required surface finish of each job limits the feed rate. Consequently, these parameters specify another bound on the processing time such that there is a lower bound for the processing time of each job, p_i^l, due to technological limitations of the CNC machine along with the job related attributes. In sum, the lower and upper bounds for the processing times in our problem are found by determining optimal machining conditions for each job (i.e., by considering machining and tooling costs only, subject to the tool life, machine power and surface finish constraints).

As we stated before, our overall objective is minimizing the sum of total weighted tardiness, machining, and tooling costs simultaneously. A tardiness penalty is incurred for each time unit, if job i is completed after its due date, d_i, such that $T_i = \max\{0, s_i + p_i - d_i\}$. As discussed in Kayan and Akturk [5], the machining cost for each job, $Mach_i(p_i)$, is a strictly increasing linear function of the processing time and defined as $Mach_i(p_i) = C_0 p_i$. The tooling cost of job i is a strictly decreasing nonlinear function of the processing time and defined as $Tool_i(p_i) = c_{a_i}/((p_i)^{c_{b_i}})$ such that $c_{a_i} > 0$ and $c_{b_i} > 0$. The parameters c_{a_i} and c_{b_i} depend on the diameter, length, CNC machine power and cutting tool type for each job i. We have to make two decisions: an optimal processing time for each job and the sequence of the jobs. By using lower and upper bounds of the processing times defined above, the nonlinear mixed integer programming formulation of the original problem is given below:

$$\text{Minimize} \quad \sum_{i=1}^{N} (w_i T_i + Mach_i(p_i) + Tool_i(p_i))$$

$$\text{subject to} \quad p_i^l \le p_i \le p_i^u, \quad i = 1, \ldots, N \tag{1}$$

$$s_i + p_i \le s_j \vee s_j + p_j \le s_i, \quad i,j = 1, \ldots, N \wedge i \ne j \tag{2}$$

$$p_i > 0, \quad s_i \geq 0, \quad i = 1, \ldots, N \tag{3}$$

The first term in the objective function is the weighted tardiness cost. The second term is machining cost and the last one is tooling cost. The first constraint set is the lower and upper bounds for the processing times. The next set of constraints are the set of disjunctive noninterference constraints such that the CNC machine can process one job at a time.

In the following sections, we will present the local search algorithm for the original problem and the DP-based heuristic for the subproblem.

3. The proposed simultaneous algorithm

In current literature, determination of processing times and scheduling these jobs on a production resource are generally considered as separate problems. These two problems are solved sequentially at different levels of factory management. Processing times are determined at the process planning level, while the scheduling problem is solved afterwards at the operational level. The most significant drawback of any sequential approach is the fact that the interaction between two levels is ignored. By using the flexibilities provided by a CNC machine, the due date requirement can be satisfied better or operation cost due to machining and tooling costs can be decreased. Therefore, we propose a problem space genetic algorithm (PSGA) to exploit the interaction between two levels. Consequently, the machining conditions and the weighted tardiness problems can be solved simultaneously to generate better solutions for the overall problem.

Problem space genetic algorithms (PSGAs) have been shown in previous research to be quite effective for a variety of scheduling problems [12]. PSGAs are basically local search algorithms. To develop a PSGA, it is necessary to define an initial feasible solution, a base heuristic and a neighborhood structure. In order to generate an initial solution, we will employ the apparent tardiness cost (ATC) sequencing rule, which is shown to be superior to other sequencing rules for the $1\|\sum w_i T_i$ problem. Under the ATC rule, jobs are scheduled one at a time; i.e., every time the machine becomes free, a ranking index is computed for each remaining job i. The job with the highest ranking index is then selected to be processed next. The ranking index is a function of the time t at which the machine became free, as well as the p_i, w_i and d_i of the remaining jobs and a look-ahead parameter K as discussed in Pinedo [8].

As outlined in Algorithm 1, we first form a job sequence by utilizing the ATC rule. Afterwards, for a given sequence, we determine the optimum processing times for the overall objective function in Step 2.6. The neighborhood is constructed through perturbation of the ATC priorities and the search is performed in the space of these perturbations. We then employ the basic principles of a genetic algorithm in Steps 3–5 to generate new sequences.

Algorithm 1. Simultaneous algorithm, PSGA.

Require p_i^l, p_i^u, w_i and d_i for each job i, and N.

Step 1. Create an initial population of perturbation vectors, $\overline{\delta}$, at random from a range of $(-\theta,\theta)$.

Step 2. *Sequence generation*: For each perturbation vector l (chromosome) of the population do;

Step 2.1. Set the current time $t = 0$ and number of scheduled jobs to $k=0$, and calculate average of averages of processing times as follows: $\overline{p_{avg}} = \sum_{i=1}^{N}((p_i^u + p_i^l)/2)/N$.

Step 2.2. For each unscheduled job i at time t, calculate the ATC priorities as follows:

$a_i(t) = \frac{w_i}{p_{avg_i}} \exp(-\max(0, d_i - t - p_{avg_i})/K\overline{p_{avg}})$

Step 2.3. ATC priorities are normalized into interval [0,1] yielding $\eta_i(t)$ as follows, let $a_{\min}(t) = \min_i a_i(t)$ and $a_{\max}(t) = \max_i a_i(t)$:

$$\eta_i(t) = \frac{a_i(t) - a_{min}(t)}{a_{max}(t) - a_{min}(t)}$$

Step 2.4. Perturb the priorities of the jobs with this member by adding the perturbation value δ_i^l to the normalized ATC priorities as follows:

$$\eta_i(t) = \eta_i(t) + \delta_i^l$$

Step 2.5. Select the job with the highest perturbed priority and schedule it next in the sequence. Set $t = t + p_{avg_i}$ and $k = k + 1$. If there are any unscheduled jobs, $k < N$, then go to Step 2.2.

Step 2.6. Solve the following nonlinear mathematical model to determine the optimum processing times for the newly generated sequence, and calculate the objective function value for the given perturbation vector, denoted by $V(l)$.

$$\text{Minimize} \quad \sum_{i=1}^{N}(w_i T_i + Mach_i(p_i) + Tool_i(p_i))$$
$$\text{subject to} \quad s_i + p_i \leq s_{i+1} \quad i = 1, \ldots, (N-1) \tag{4}$$

$$p_i^l \leq p_i \leq p_i^u \quad i = 1, \ldots, N \tag{5}$$

$$p_i > 0 \quad \text{and} \quad s_i \geq 0 \tag{6}$$

Step 3. After finishing all members in the population, save the best and worst solutions. If the number of generations reaches the limit, then stop and report the best solution.

Step 4. Compute the *fitness* value, $f(l)$, of each perturbation vector as follows, let $V_{\max}$ be the maximum objective value in the population and the parameter ϕ is the selectivity constant of the algorithm such that:

$$f_l = \frac{(V_{max} - V_l)^{\phi}}{\sum_l (V_{max} - V_l)^{\phi}}$$

As the selectivity constant, ϕ, increases, better solutions will have a greater chance of being selected. If ϕ is too large, the population will converge quickly, which is not desirable, since we are trying to find a diversified set of solutions.

Step 5. Apply crossover and mutation to get the next generation using the fitness distribution and update perturbation vectors, then go to Step 2.

When we reformulate the problem for a given sequence in Step 2.6, the complexity of the problem decreases. We still need to calculate the optimum processing times that minimize the nonlinear objective function (the most time consuming part of the proposed algorithm), but we are free from sequencing problem and consequent binary variables. In the original problem, constraint set (2) includes all permutations of jobs, on the other hand the set (4) includes just one sequence. Therefore, we generate different sequences in PSGA to find the best solution. To increase the computational efficiency of the PSGA, we propose a DP-based heuristic instead of solving the mathematical model in Step 2.6 as discussed below.

4. The proposed DP-based heuristic

The motivation behind this algorithm is that if we can minimize the cost contribution of each job to the total cost, we

can minimize the total cost. Thus, we define a contributed cost function for each job as the sum of its tooling and machining costs (which are calculated independently for each job as a function of its own processing time), and the deviation in the weighted tardiness costs of itself and all the succeeding jobs for the given sequence depending on the selected processing time calculated as $\sum_{j=i}^{N} w_j \Delta Tard_j(p_i)$. In this summation, we measure the impact of p_i on the weighted tardiness cost of each succeeding job including the job i itself as follows:

$$ContCost_i(p_i) = Tool_i(p_i) + Mach(p_i) + \sum_{j=i}^{N} w_j \Delta Tard_j(p_i) \quad (7)$$

In constructing this function, we initially assumed that $p_k = p_k^l$ for $k = 1,\ldots,i-1$, in order to define the contributed cost only on the processing time of job i. In our DP-based heuristic, finding this contributed cost is named as *Graph Generation*, which corresponds to Steps 2 and 3 below, since we generate a graph that shows how the contributed cost of job i changes.

For each job, we can easily construct the function given in Eq. (7). However, we cannot directly calculate the processing time that minimizes this function. Although we initially set the processing times of the previous jobs to their lower bounds as we constructed the contributed cost function, their optimum processing times may differ from the lower bounds. To deal with this issue, we first define Δ_i, which is the total deviation in the sum of processing times of jobs before job i:

$$\Delta_i = \sum_{k=1}^{i-1} p_k - p_k^l \quad (8)$$

Machining and tooling costs of a job are independent from Δ_i. However, $\Delta Tard_i$ is dependent on it. Therefore, our aim is to find the corresponding processing times of each job for all possible values of Δ_i. This corresponds to the *State Generation* step in our algorithm, in which we generate the state space, in other words ranges, of Δ_i and processing times for these states.

The proposed algorithm is similar to a backward DP. Its step by step definition is given in Algorithm 2. Beginning from the last job processed in the given sequence, generating *graphs* and *states* iteratively, it defines a function for each job that gives how the processing times of the jobs processed after that job depend on the deviation of processing time of that job. After finding the processing time of the first job, the algorithm finds the processing time of the second job, third job and so on. It is an approximation algorithm since the defined contributed cost considers the interaction between jobs on the basis of just tardiness costs. There is also an interaction through the tooling cost. However, due to the nonlinearity of tooling cost, developing a practical solution procedure could be very time consuming.

Algorithm 2. DP-based heuristic.

Step 1. Set $i=N$.
Step 2. *Graph Generation:* For $j=i$ to $j=N$ do;
Step 2.1. Set $p_k = p_k^l$ for $k=1,\ldots,i-1$ and construct the function $P_j(p_i)$ that shows how the processing time of job j depends on the processing time of job i.
Step 3. *Graph generation*: Construct the contributed cost for job i, which is defined in Eq. (7).
Step 4. If $i>1$ goto Step 5, else goto Step 7.
Step 5. *State Generation*: Generate the function $P_i(\Delta_i)$, where Δ_i is the total deviation of processing times of jobs 1 to $i-1$ from their corresponding lower bounds as defined in Eq. (8).
Step 6. Generate the function $P_i(p_{i-1})$ from $P_i(\Delta_i)$ by replacing Δ_i with $p_{i-1} - p_{i-1}^l$, then set $i = i-1$ and goto Step 2.
Step 7. Find the minimum of the following total cost function for job 1:

$$ContCost_1(p_1) = \sum_{i=1}^{N} w_j \Delta Tard_i(p_1) + Tool_1(p_1) + Mach_1(p_1)$$

Step 8. Calculate $P_i(p_1^*)$ for $i = 2,\ldots, N$, which give the processing times of all jobs in the sequence.

In the following two subsections, we give further explanation about the Steps 2 and 3 and the Step 5 of the proposed algorithm that are named as graph generation and state generation, respectively. These are the critical steps of the algorithm since most of the computational effort in the algorithm is spent in these steps.

4.1. Explanation of graph generation

In Steps 2 and 3, our aim is to find the contributed cost of job i. For that purpose, we first find how processing times of jobs processed after job i depend on the processing time of job i. Then, we find how the tardiness costs of job i to job N deviate depending on the increase in the processing time of job i. By using this information, we construct the contributed cost. As we stated before, the machining and tooling costs of a job are only a function of its own processing time. However, the deviations in tardiness' of the other jobs are dependent on both starting time and their processing times. The problem in constructing the contributed cost of job i arises from calculating how we find the deviations of the tardiness costs of job j for $j=$i, …, N considering the changes in the processing times of job l for $l = i$, …, j.

Since minimizing the total weighted tardiness is a regular scheduling measure, there is no need to insert idle times in the final schedule. Therefore, even a slight change in the processing time of job i changes the starting times of the succeeding jobs such that we need to determine the processing and starting times of each job simultaneously to minimize the joint objective function. To find the $\Delta Tard_j$ for $j=i$, …, N, we have to first construct the function $P_j(p_i)$ that shows how the processing time of job j depends on processing time of job i while $p_k=p_k^l$ for $k=1,\ldots,i-1$. This is a piecewise linear function which can be discontinuous. For kth state of p_i, which is defined as $r_{j,i}^k \le p_i < r_{j,i}^{k+1}$, it returns a value by a function in the form of $c_{j,i}^k - x_{j,i}^k p_i$. In this function, $c_{j,i}^k$ symbolizes a parameter whose superscript k indicates which state it corresponds, and subscript $\{j,i\}$ indicates which P function it corresponds. $x_{j,i}^k$ is a 0 or 1 binary parameter, which indicates p_j either decreases in the same magnitude as p_i increases or it remains the same in the kth *state*. $r_{j,i}^k$ and $r_{j,i}^{k+1}$ are also parameters that show the boundaries of states. The general form of $P_j(p_i)$ is given in Eq. (9). In sum, for a given sequence, each job corresponds to a different stage in a DP format and the maximum number of states at each stage is denoted by $Y^{j,i}$.

$$P_j(p_i) = \{c_{j,i}^k - x_{j,i}^k p_i \text{ for } r_{j,i}^k \le p_i < r_{j,i}^{k+1};\ k = 1,\ldots,Y^{j,i}\} \quad (9)$$

For $i=N$, it is obvious that $P_N(p_N) = p_N$ where $p_N^u \ge p_N \ge p_N^l$. For $i<N$, when we come to this step we have already known $P_j(p_{i+1})$ for $j=(i+2)$, …, N. This function is generated by setting $p_k=p_k^l$ for $k=1$, …, i and making p_{i+1} variable. In other words, $\Delta_{i+1} = p_{i+1} - p_{i+1}^l$. Thus, we can write down P_j depending on Δ_{i+1}, e.g. $P_j(\Delta_{i+1})$.

From the previous iteration, we know $P_{i+1}(p_i)$. Now, in Eq. (8), we put the $P_{i+1}(p_i)$ in place of p_{i+1}, make p_i a variable and set $p_k = p_k^l$ for $k = 1$, …, $i-1$ to obtain the Δ_{i+1} as follows:

$$\Delta_{i+1} = P_{i+1}(p_i) - p_{i+1}^l + p_i - p_i^l \quad (10)$$

When we put the Δ_{i+1} in its place in function $P_j(\Delta_{i+1})$, we find $P_j(p_i)$, $j = (i+2), \ldots, N$.

After finding $P_j(p_i)$ for $j = i, \ldots, N$ (given that $p_k = p_k^l$ for $k = 1, \ldots, i-1$), we can construct $\sum_{j=i}^{N} w_j \Delta Tard_j(p_i)$ as described in Algorithm 3.

Algorithm 3. Graph generation.

Step 1. From functions $P_j(p_i)$ for $j = i, \ldots, N$ collect all $r_{j,i}^k$ for $k = 1, \ldots, \omega_{j,i}+1$, where $\omega_{j,i}$ is the number of states of function $P_j(p_i)$. Call the set of them as Ω_i. Then eliminate duplicate entries in Ω_i and name remaining ones as $r_i^{[k]}$ for $k = 1, \ldots, W_i$ such that $r_i^{[1]} < r_i^{[2]} < \ldots < r_i^{[W_i]}$.

Step 2. Set $p_j^0 = P_j(p_i^l)$, where p_j^0 is the processing time of job j when there is no deviation in p_i from p_i^l. We can define it as follows: $p_j^0 = c_{j,i}^{[0]} - x_{j,i}^{[0]} p_i$ for $r_i^{[0]} \le p_i \le r_i^{[1]}$.

Step 3. For $k = 1$ to $W_i - 1$ do;

Step 3.1. Find the p_j for $j = i+1, \ldots, N$ by using $P_j(p_i)$ over $r_i^{[k]} \le p_i \le r_i^{[k+1]}$, it is defined as follows:
$p_j = c_{j,i}^{[k]} - x_{j,i}^{[k]} p_i$ for $r_i^{[k]} \le p_i \le r_i^{[k+1]}$.

Step 3.2. Set $s_{i-1} = \sum_{t=1}^{i-2} p_t^l$ and for $j = i$ to N do;

Step 3.2.1. Calculate starting time of job j as $s_j = s_{j-1} + p_{j-1}$.

Step 3.2.2. Calculate the deviation in the tardiness cost of job j as

$$\Delta Tard_j(p_i) = \max\{0, s_j + (p_j - p_j^l) - d_j\} \text{ for } r_i^{[k]} \le p_i \le r_i^{[k+1]}.$$

Step 3.3. Sum up all $w_j \Delta Tard_j(p_i)$ and find total weighted deviation in tardiness'

$$TotalTard_i(p_i) = \sum_{j=i}^{N} w_j \Delta Tard_j(p_i) = m_i^k p_i + n_i^k \text{ for } r_i^{[k]} \le p_i \le r_i^{[k+1]}$$

m_i^k and n_i^k are just constants that come from the weighted summation of the constants in the $\Delta Tard_j$ functions. If we define Π as the set of jobs whose $Tard_j > 0$ for j=i, …, N, then

$$m_i^k = \sum_{j \in \Pi} w_j (x_{j,i}^{[k]} - x_{j,i}^{[0]}) \quad \text{and} \quad n_i^k = \sum_{j \in \Pi} (s_j + (c_{j,i}^{[k]} - c_{j,i}^{[0]}) - d_j).$$

At the end of this algorithm, we reach the function that shows the total deviation in tardiness costs of job i to job N depending on p_i. It is a piecewise linear function in the following form as in Fig. 1 and the maximum number of break points is denoted by W:

$$TotalTard_i(p_i) = \{m_i^k p_i + n_i^k \text{ for } r_i^{[k]} \le p_i < r_i^{[k+1]}; k = 1, \ldots, W\} \qquad (11)$$

For example, if all of the consecutive lines form a convex shape, i.e. there is no concavity, then the maximum number of break points is less than or equal to the number of jobs, N. Each such point represents when a job becomes tardy.

When we sum up this function with $Tool_i(p_i)$ and $Mach_i(p_i)$, the *Graph Generation* steps finish with the contributed cost function at hand. The algorithm continues with the *State Generation* step until the processing time of the first job is found.

4.2. Explanation of state generation

Our main aim in this step is, by using the contributed cost function given in Eq. (7), to construct the function that gives the processing time of job i over the different states, or ranges, of Δ_i. In Steps 2 and 3 of the main algorithm, while constructing the contributed cost function of job i, we set $p_k = p_k^l$ for $k = 1, \ldots, i-1$ (although optimum processing times could be higher than their lower bounds). We give the sum of the variations in the processing times in Eq. (10) for job i. We have to include these variations in our total cost function so that we can search over Δ_i. We only need to modify the tardiness cost function as given below. The other cost components, tooling and machining costs, remain constant with respect to Δ_i since they are independent from the deviations in the processing times of the previous jobs in the sequence:

$$TotalTard_i'(p_i, \Delta_i) = \{m_i^k (p_i + \Delta_i) + n_i^k \text{ for } r_i^{[k]} \le p_i + \Delta_i < r_i^{[k+1]}; k = 1, \ldots, W\}$$

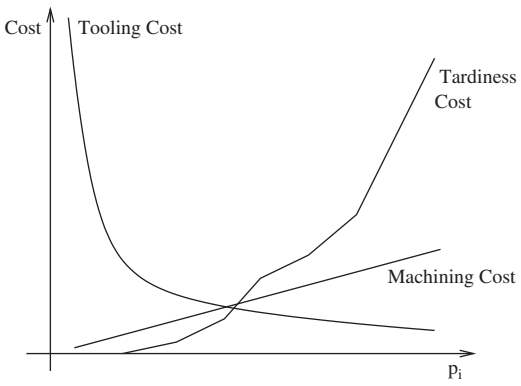

Fig. 1. Cost components for each job.

Each piece in the function above is called as a 'line'. This line and the corresponding contributed cost are defined as follows:

$$L_i^k(p_i, \Delta_i) = m_i^k (p_i + \Delta_i) + n_i^k \text{ for } r_i^{[k]} \le p_i + \Delta_i < r_i^{[k+1]}; \text{ and } \infty, o.w.$$

$$CL_i^k(\Delta_i, p_i) = L_i^k(p_i, \Delta_i) + Tool(p_i) + Mach(p_i)$$

To find the $P_i(\Delta_i)$, we have to solve the following minimization problem:

$$P_i(\Delta_i) = \{p_i \in \text{argmin}\{TotalTard_i'(p_i, \Delta_i) + Tool_i(p_i) + Mach_i(p_i) : p_i^l \le p_i \le p_i^u\}\}$$

For $\Delta_i = 0$, $TotalTard_i'(p_i, \Delta_i)$ is equivalent to $TotalTard_i(p_i)$ function (given in Eq. (11)) and its example shape is also given in Fig. 1.

For each Δ_i value, there is only one minimum value of the total cost function. Moreover there is a range of $x \le \Delta_i \le y$ such that minimum point still remains as the minimum. Our purpose is to find the minimum values and corresponding "Δ−ranges" defined in the function $P_i(\Delta_i)$ so that we can solve the minimization problem above. To find the minimum of the total cost, we present Algorithm 4. In this algorithm, we take two adjacent pieces and analyze the contributed cost function that corresponds to these two pieces. In Step 2.2, if these two lines form a convex shape (as in Fig. 2), to find $R_i(\Delta)$ we use *UseConvex* subroutine, else we use *UseConcave* subroutine. For readability, in the next two subsections, we will drop the subscript i from all parameters, functions and variables except for p_i and Δ_i.

Algorithm 4. Cost function.

Step 1. Set $P_i(\Delta_i) = 0$ for $0 \le \Delta_i \le \Delta_i^{\max}$ where $\Delta_i^{\max} = \sum_{k=1}^{i-1} (p_k^u - p_k^l)$.

Step 2. For $k = 1$ to $W_i - 1$ do;

Step 2.1. Generate the function $R_i(\Delta_i)$ for kth and $(k+1)$th pieces in the $TotalTard_j'(p_i, \Delta_i)$ as follows:

$$R_i(\Delta_i) = \{p_i \in \text{argmin}\{\min\{CL_i^k(\Delta_i, p_i), CL_i^{k+1}(\Delta_i, p_i)\} : p_i^l \le p_i \le p_i^u\}\}.$$

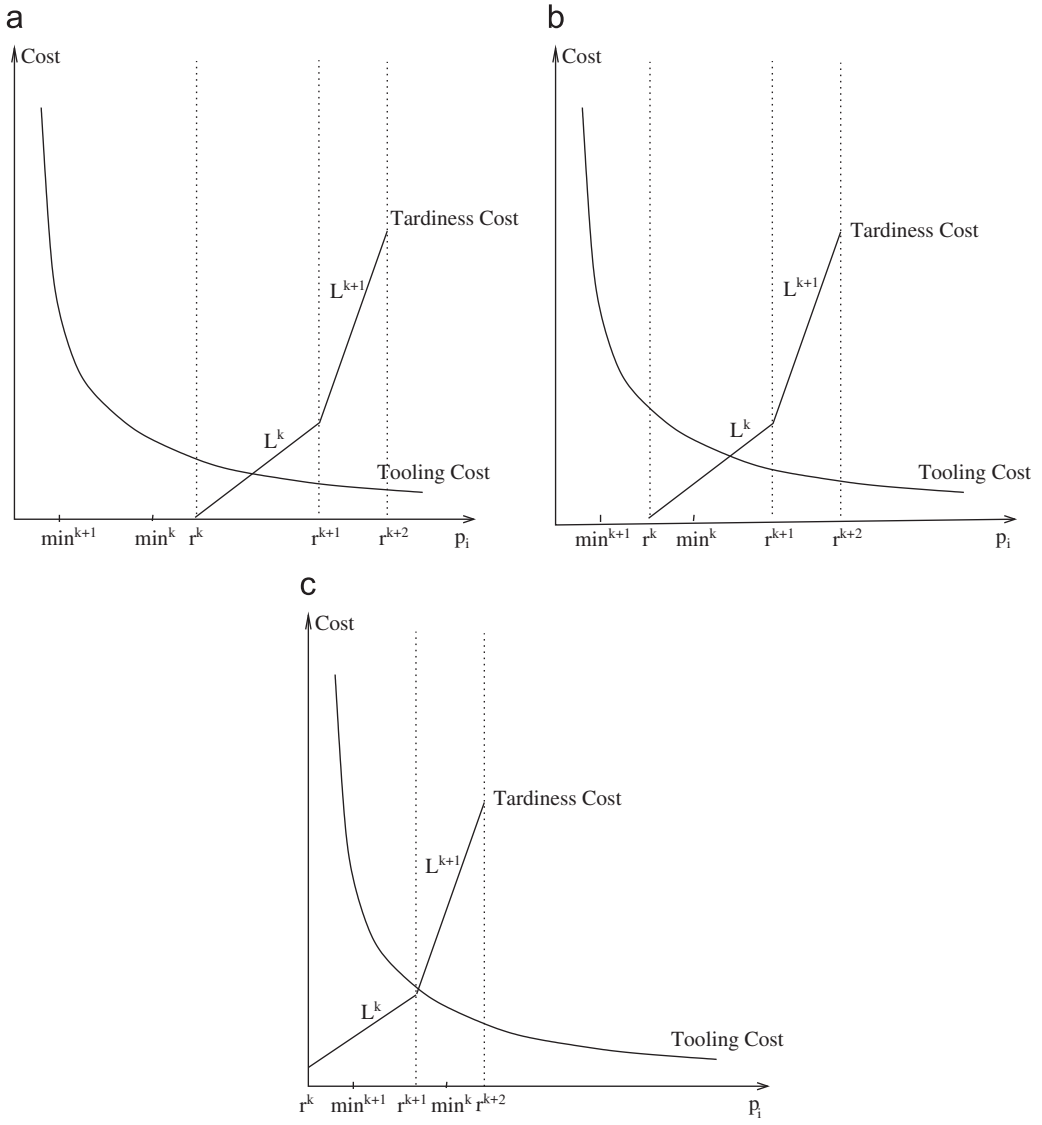

Fig. 2. Illustration of possible locations of breakpoints of tardiness lines and minimum cost points that corresponds to those lines.

Step 2.2. Find $P_i'(\Delta_i)$ by combining $R_i(\Delta_i)$ and $P_i(\Delta_i)$ as follows:

$$P_i'(\Delta_i) = \{p_i \in \text{argmin}\{\min\{ContCost_i(R_i(\Delta_i)), ContCost_i(P_i(\Delta_i))\}\}\}$$

Step 2.3. Set $P_i(\Delta_i) = P_i'(\Delta_i)$.

4.2.1. UseConvex subroutine

Two adjacent tardiness cost lines may form a convex shape as in Fig. 2. The reason for this is obvious, when a job becomes tardy at a point without affecting the other jobs, the slope of the function after that point increases. $R(\Delta_i)$ has a direct formulation in this case which we call the *UseConvex* subroutine. For given L^k and L^{k+1}, $R(\Delta_i)$ is formulated as follows:

$$R(\Delta_i) = \begin{cases} r^{[k]} - \Delta_i & \text{for } 0 \le \Delta_i < r^{[k]} - \min^k \\ \min^k & \text{for } r^{[k]} - \min^k \le \Delta_i < r^{[k+1]} - \min^k \\ r^{[k+1]} - \Delta_i & \text{for } r^{[k+1]} - \min^k \le \Delta_i < r^{[k+1]} - \min^{k+1} \\ \min^{k+1} & \text{for } r^{[k+1]} - \min^{k+1} \le \Delta_i < r^{[k+2]} - \min^{k+1} \\ r^{[k+2]} - \Delta_i & \text{for } r^{[k+2]} - \min^{k+1} \le \Delta_i \le \Delta_i^{\max} \end{cases}$$

where

$$\min^k = \arg\min\{CL^k(0, p_i) : p_i^l \le p_i \le p_i^u\}$$

and

$$\min^{k+1} = \arg\min\{CL^{k+1}(0, p_i) : p_i^l \le p_i \le p_i^u\}$$

In this formulation $r^{[k]}$ is greater than $\min^k$. In other cases, for example $r^{[k]} \le \min^k$, $r^{[k+1]} \le \min^{k+1}$ and so on, the formulation is modified by deleting ranges in which both sides of Δ_i are non-positive and by changing left side of the Δ_i, whose left side is negative but whose right side is positive, to zero.

The construction of this formula is based on graphical observations. First of all, if we take the partial derivatives of CL^k and CL^{k+1} over their defined regions for Δ_i, we see that $\min^{k+1} \le \min^k$ since:

$$\frac{\partial}{\partial p_i} CL^k(\Delta_i, p_i) = m^k - \frac{c_a c_b}{p_i^{c_b+1}} + C_0 = 0 \Longrightarrow p_i = \left(\frac{c_a c_b}{m^k C_0}\right)^{1/(c_b+1)} = \min^k$$

$$\frac{\partial}{\partial p_i} CL^{k+1}(\Delta_i, p_i) = m^{k+1} - \frac{c_a c_b}{(p_i)^{c_b+1}} + C_0 = 0 \Longrightarrow p_i = \left(\frac{c_a c_b}{m^{k+1} C_0}\right)^{1/(c_b+1)} = \min^{k+1}$$

$$m^k < m^{k+1} \Longrightarrow \min^{k+1} \le \min^k$$

Initially, let $r^{[k+2]} \ge r^{[k+1]} \ge r^{[k]} \ge min^k \ge \min^{k+1}$ as in Fig. 2a. In this case, $R(\Delta_i) = r^{[k]} - \Delta_i$, and this case is valid for $0 \le \Delta_i < r^{[k]} - \min^k$.

When $r^{[k]} - \min^k \le \Delta_i < r^{[k+1]} - \min^k$, the tardiness cost function becomes as in Fig. 2b. In this case, the minimum for L 1 is the minimum of total cost function (considering only L 1 and L 2).

When $r^{k+1} - \min^k \le \Delta_i < r^{[k+1]} - min^{k+1}$, the tardiness cost function becomes as in Fig. 2c. In this case, the breakpoint between two lines, $r^{[k+1]} - \Delta_i$, is the minimum of total cost function. The rest of the function $R(\Delta_i)$ can be derived in a similar way.

4.2.2. UseConcave subroutine

Sometimes, two consecutive tardiness cost lines form a concave shape as in Fig. 3. To explain the reason behind this, let us consider three jobs, namely i, j, and k, whose processing order is job i, job j, and job k. Furthermore, let job k be tardy and have a constant processing time. While p_j is constant and as p_i increases, tardiness of job k increases. However, if at a point, say t, p_j may start to decrease as p_i increases, at that point, the tardiness of job k equals to a constant value and remains at that value as long as p_j decreases. This causes concavity because the slope of the total tardiness cost function falls down after point t. In such a case, $R(\Delta_i)$ requires a more detailed analysis. The same subroutine can also be used for non-consecutive lines, which might occur at Step 2.3 of the *State Generation* algorithm.

Now consider the following simplified versions of L^k and L^s with $r^{[s]} \ge r^{[k+1]}$:

$$L^k(p_i) = \begin{cases} m^k * p_i + (n^k)', & b_1 \le p_i < b_2 \\ \infty & o.w. \end{cases} \quad L^s(p_i) = \begin{cases} m^s * p_i + (n^s)', & b_3 \le p_i < b_4 \\ \infty & o.w. \end{cases}$$

where

$$b_1 = r^{[k]} - \Delta_i, \quad b_2 = r^{[k+1]} - \Delta_i, \quad b_3 = r^{[s]} - \Delta_i$$

$$b_4 = r^{[s+1]} - \Delta_i, \quad (n^k)' = n^k + m^k \Delta_i, \quad (n^s)' = n^s + m^k \Delta_i.$$

Corresponding total cost functions CL^k and CL^s are similar to the ones in Fig. 4. The possible locations of b_1, b_2, b_3, b_4, $\min^k$, and $\min^s$ relative to each other change depending on the value of Δ_i and values of $r^{[k]}$, $r^{[s]}$, $r^{[k+1]}$, and $r^{[s+1]}$. The possible ordering of them for each line is as follows:

$$\text{for } L^k \begin{cases} (A)\ \min^k \le b_1 < b_2 & \text{for } 0 \le \Delta_i \le r^{[k]} - \min^k \\ (B)\ b_1 \le \min^k \le b_2 & \text{for } r^{[k]} - \min^k \le \Delta_i \le r^{[k+1]} - \min^k \\ (C)\ b_1 < b_2 \le \min^k & \text{for } r^{[k+1]} - \min^k \le \Delta_i \le \Delta_i^{max} \end{cases}$$

$$\text{for } L^s \begin{cases} (1)\ \min^s \le b_3 < b_4 & \text{for } 0 \le \Delta_i \le r^{[s]} - \min^s \\ (2)\ b_3 \le \min^s \le b_4 & \text{for } r^{[s]} - \min^s \le \Delta_i \le r^{[s+1]} - \min^s \\ (3)\ b_3 < b_4 \le \min^s & \text{for } r^{[s+1]} - \min^s \le \Delta_i \le \Delta_i^{max} \end{cases}$$

Graphically, (A), (B) and (C) indicate the position of the line L^k, as Δ_i increases, whereas (1), (2) and (3) for L^s. The positions of L^k and L^s are important for us. We can explain the reason of that with an example, consider the case that both of these lines position over minimums of contributed cost lines corresponding to them for a defined range of Δ_i. This provides us an important advantage: now we know that for these two pieces of contributed cost, the processing time of job j that minimizes the *ContCost*′ is either m^s or m^k over this range of Δ_i. All of the combinations of these cases could be very helpful while constructing $R(\Delta)$.

In our algorithm we construct the $R(\Delta_i)$ for each possible combinations of A, B, C and 1, 2, 3 over defined ranges of Δ_i. A schematic representation of possible combinations and the flow between these combinations and some information about cost components are given in Fig. 5. For example, $(A,1)$ corresponds to $min^k \le b_1 < b_2$ and $\min^s \le b_3 < b_4$; $(A,2)$ corresponds to $\min^k \le b_1 < b_2$ and $b_3 \le \min^s < b_4$; and so on. The outgoing arcs from $(A,1)$ means that from $(A,1)$ we can go either $(A,2)$ or $(B,1)$. If $r^{[k]} - \min^k > r^{[s]} - \min^s$ then we go from $(A,1)$ to $(A,2)$, else we go from $(A,1)$ to $(B,1)$.

4.3. Illustrative example

We will solve a small scheduling problem for a given sequence to illustrate how the proposed DP-based heuristic works. We construct an example for three jobs with the sequence of 1, 2, and 3 (i.e. job 1 is processed first). Machining cost, C_0, is 0.5 and the cost of tooling (used for calculating p^l and p^u for each job in Kayan and Akturk [5]) is 4.5. The required data for each job is given in Table 1.

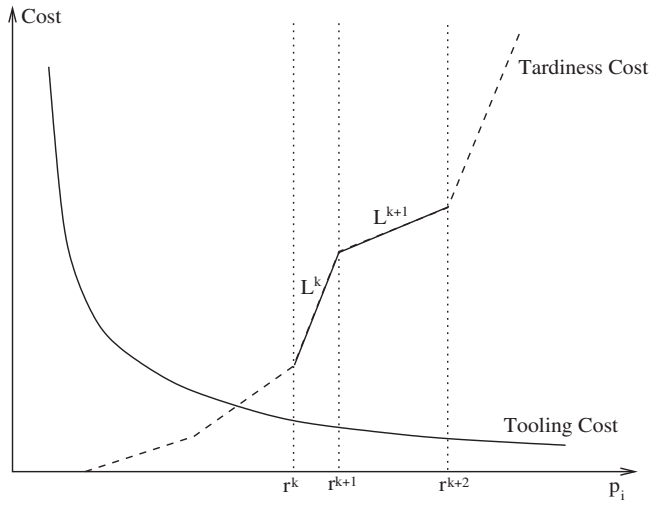

Fig. 3. Two lines that form concave shape.

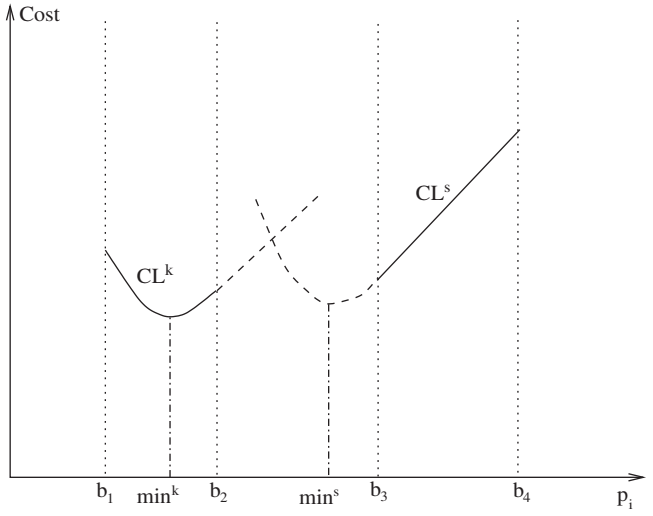

Fig. 4. The illustration of possible total costs for two tardiness lines.

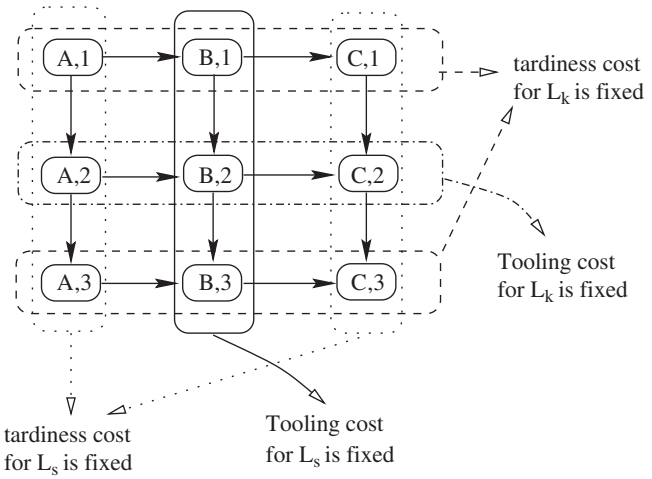

Fig. 5. Possible combinations of locations of ranges for two tardiness lines and information about cost components corresponding to each line.

Table 1
Job data for the illustrative example.

Job #	D	L	w	d	p^l	p^u	c_a	c_b
1	7.2	6	1	1	0.70	2.42	2.06	1.35
2	7.1	8	3	2	1.23	3.04	5.18	1.41
3	7.1	5	2	3	0.56	2.08	2.92	1.24

Job 3 : We start with generating total tardiness function of the last job in the given sequence, job 3, because its *TotalTard*′ function includes only its own tardiness cost:

$$s_3 = p_1^l + p_2^l = 0.70 + 1.23 = 1.93$$

$$\Delta_3^{\max} = (2.42-0.70)+(3.04-1.23) = 3.53$$

$$P_3(p_3) = p_3$$

$$TotalTard_3(p_3) = \begin{cases} 0 & \text{for } 0.56 \le p_3 < 1.07 \\ 2(p_3-1.07) & \text{for } 1.07 \le p_3 \le 2.08 \end{cases}$$

$$TotalTard_3'(p_3,\Delta_3) = \begin{cases} 0 & \text{for } 0.56 \le p_3+\Delta_3 < 1.07 \\ 2(p_3+\Delta_3-1.07) & \text{for } 1.07 \le p_3+\Delta_3 \le 2.08+3.53 = 5.61 \end{cases}$$

This function gives how tardiness cost value changes according to the changes in the processing times of jobs 1, 2 and 3. Now, we have to generate $P_3(\Delta_3)$. The total contributed cost lines that correspond to each tardiness cost range is as follows:

$$CL^1(\Delta_3,p_3) = \begin{cases} 0.5p_3+2.92/(p_3)^{1.24} & \text{for } 0.56 \le p_3+\Delta_3 < 1.07 \\ \infty & o.w. \end{cases}$$

$$CL^2(\Delta_3,p_3) = \begin{cases} 2.5p_3-2.14+2\Delta_3+2.92/(p_3)^{1.24} & \text{for } 1.07 \le p_3+\Delta_3 \le 5.61 \\ \infty & o.w. \end{cases}$$

The minimums of lines CL^1 and CL^2 are $\min^1 = 2.08$ and $\min^2 = 1.05$, respectively. We can construct the $R(\Delta_3)$ by using the *UseConvex* subroutine. Since there is no more line $P_3(\Delta_3) = R(\Delta_3)$:

$$P_3(\Delta_3) = R(\Delta_3) = \begin{cases} 1.07-\Delta_3 & \text{for } 0.00 \le \Delta_3 < 0.02 \\ 1.05 & \text{for } 0.02 \le \Delta_3 \le 5.61 \end{cases}$$

Afterwards, we can reduce $P_3(\Delta_3)$ to $P_3(p_2)$ by setting $\Delta_3 = p_2 - 1.23$:

$$P_3(p_2) = \begin{cases} 2.30-p_2 & \text{for } 1.23 \le \Delta_3 < 1.25 \\ 1.05 & \text{for } 1.25 \le \Delta_3 \le 3.04 \end{cases}$$

Job 2 : We have to generate *TotalTard* function of job 2. The only job after job 2 is job 3 and we already know how its processing time changes with respect to the processing time of job 3 from $P_3(p_2)$. We calculate the tardiness cost for each job and sum them up:

$$s_3 = p_1^l + p_2 = 0.70 + p_2$$

$$Tard_3(p_2) = \max\{0, p_2 + P_3(p_2) - 2.30\} = \begin{cases} 0 & \text{for } 1.23 \le p_2 < 1.25 \\ p_2 - 1.25 & \text{for } 1.25 \le p_2 \le 3.04 \end{cases}$$

Let $s_2 = p_1^l = 0.70$, such that $\Delta_2^{\max} = 2.42 - 0.70 = 1.72$. Consequently,

$$Tard_2(p_2) = \max\{0, p_2 - 1.28\} = \begin{cases} 0 & \text{for } 1.23 \le p_2 < 1.28 \\ p_2 - 1.28 & \text{for } 1.28 \le p_2 \le 3.04 \end{cases}$$

$$TotalTard(p_2) = w_2 Tard_2(p_2) + w_3 Tard_3(p_2) = \begin{cases} 0 & \text{for } 1.23 \le p_2 < 1.25 \\ 2(p_2-1.25) & \text{for } 1.25 \le p_2 < 1.28 \\ 5(p_2-1.28) & \text{for } 1.28 \le p_2 \le 3.04 \end{cases}$$

$$TotalTard_2'(\Delta_2,p_2) = \begin{cases} 0 & \text{for } 1.23 \le p_2+\Delta_2 < 1.25 \\ 2(p_2+\Delta_2-1.25) & \text{for } 1.25 \le p_2+\Delta_2 < 1.28 \\ 5(p_2+\Delta_2-1.28) & \text{for } 1.28 \le p_2+\Delta_2 \le 4.76 \end{cases}$$

Now, we have to calculate $P_2(\Delta_2)$. There are three distinct piecewise lines in total cost function of job 2 as follows:

$$CL^1(\Delta_2,p_2) = \begin{cases} 0.5p_2+5.18/(p_2^{1.41}) & \text{for } 1.23 \le p_2+\Delta_2 < 1.25 \\ \infty & o.w. \end{cases}$$

$$CL^2(\Delta_2,p_2) = \begin{cases} 2.5p_2-2.50+2\Delta_2+5.18/(p_2^{1.41}) & \text{for } 1.25 \le p_2+\Delta_2 < 1.28 \\ \infty & o.w. \end{cases}$$

$$CL^3(\Delta_2,p_2) = \begin{cases} 5.5p_2-6.40+5\Delta_2+5.18/(p_2^{1.41}) & \text{for } 1.28 \le p_2+\Delta_2 \le 4.76 \\ \infty & o.w. \end{cases}$$

The processing times that minimize CL^1, CL^2 and CL^3 are $\min^1 = 3.04$, $\min^2 = 1.56$, and $\min^3 = 1.23$, respectively. As it is seen, all consecutive lines form a convex shape. A graphical representation is given in Fig. 6. Thus, we can use the *UseConvex* subroutine and find the corresponding $P_2(\Delta_2)$ easily:

$$P_2(\Delta_2) = \begin{cases} 1.28-\Delta_2 & \text{for } 0.00 \le \Delta_2 < 0.05 \\ 1.23 & \text{for } 0.05 \le \Delta_2 \le 1.72 \end{cases}$$

Job 1 : The *TotalTard* function for job 1 can be found in a similar way as follows:

$$TotalTard(p_1) = \begin{cases} 2(p_1-0.73) & \text{for } 0.73 \le p_1 < 0.77 \\ 5(p_1-0.77) & \text{for } 0.77 \le p_1 < 1.00 \\ 6(p_1-0.80) & \text{for } 1.00 \le p_1 \le 2.42 \end{cases}$$

Since we reach to the first job in the given sequence, we skip the state generation step and directly find the point that minimize the contributed cost function. Its minimum is 0.83, and hence $p_1 = 0.83$. We can now calculate the processing times of the jobs 2 and 3 in a backward manner. Since $p_1 = 0.83$, then Δ_2,

$2.42-0.83 = 1.59$, correspond to the second state of $P_2(\Delta_2)$. Thus, $p_2 = 1.23$. Now, we know both p_1 and p_2 and can calculate $\Delta_3 = 2.42-0.83+1.23-1.23 = 1.59$. For this value of Δ_3, $p_3 = 1.05$. In this particular instance, the mathematical formulation also gives the same solution, and hence it is an optimal solution.

When we compare different processing time alternatives in Table 2 for a given sequence of $\{1-2-3\}$, we can easily observe that the processing time upper bounds, p^u, give the minimum tooling cost, but the maximum machining and weighted tardiness costs. On the other hand, the processing time lower bounds, p^l, give the minimum machining and weighted tardiness costs, but the maximum tooling cost. Therefore, the controllable processing times, denoted by p^*, provide an important tradeoff. It is important to note that the p^u values (that minimize the sum of machining and tooling costs) are the processing time values used in the scheduling literature. By utilizing the inherent flexibility of the CNC machines, we could reduce the overall cost by 58%. The cost of implementing the controllable processing time idea is virtually negligible; it only requires few changes in the CNC part program.

5. Experimental results

In this section, the experimental factors of the problem are specified and the performance of both the proposed DP-based heuristic and local search algorithm are compared with their corresponding bench-mark algorithms. Both local search and DP-based heuristics are coded in C language and compiled with GNU compiler. The nonlinear model of the original problem for a given sequence is formulated in GAMS 2.25 and solved by MINOS 5.3. All problems are solved on a sparc station (Sun Enterprize 4000) under SunOS 5.7.

We developed a four-factorial experimental design to test both proposed DP-based heuristic and PSGA. The first factor was the number of jobs, N, and we generated a variety of problems with 40 and 80 jobs. The difficulty of the problem increases exponentially when we increase the N. Furthermore, as the number of job increases, the decision on the processing times of the jobs becomes more critical since an increase in the processing time of a job has greater effect on the total tardiness cost value. In our computational setting, the operating cost of the CNC machine was a fixed parameter, and $C_0 = 0.5$. On the other hand, the cost of tooling, C_t, for each job was the second experimental factor, and were selected randomly from the interval UN[0.5,1.5] and UN[3.5,4.5] for the low and high levels, respectively, where UN stands for an uniform distribution. As tooling cost increases, the tradeoff between tooling cost and the sum of tardiness and machining costs increases. This increases the difficulty of the problem as well. Due dates, d_i, were randomly generated from the interval $UN[(1-TF-RDD/2),(1-TF+RDD/2)] * \Sigma\overline{p}_i$, where TF is the tardiness factor, RDD is the range of due dates, $\overline{p}_i$ is the average processing time of job i and $\Sigma\overline{p}_i$ is the sum of average processing times of all jobs. Both TF and RDD were set to 0.2, 0.5 and 0.8. In sum, the number of jobs and tooling cost can take values in two levels, whereas tardiness factor and range of due date can take values in three levels. Thus, the experimental design is a 2*2*3*3 full factorial design.

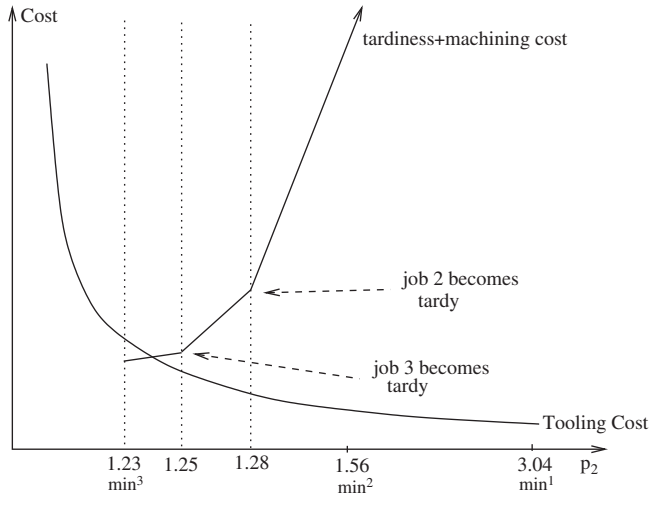

Fig. 6. Processing time variation of Job 2.

Table 2
Processing time alternatives for the illustrative example.

Processing time	Machining cost	Tooling cost	Weighted tardiness	Total
p^u=(2.42, 3.04, 2.08)	3.77	2.831	20.88	27.481
p^l=(0.7, 1.23, 0.56)	1.245	12.89	0.0	14.135
p^*=(0.83, 1.23, 1.05)	1.555	9.516	0.4	11.471

5.1. Computational results for a given sequence

By using the full factorial design, we first tested our DP-based heuristic. For the benchmark, we solved the mathematical formulation given in Step 2.6 of the Algorithm 1 (we have a nonlinear objective function with a linear set of constraints). For each factor combination, we took 25 replications resulting in 900 randomly generated runs. The summary of the results is listed in Table 3 showing the minimum, maximum and average results for each algorithm.

The averages of 25 replications for each factor combination are given in Table 4. Under the 'Deviations' column, we report the deviation from the optimal solution under 'Obj' and what percent the CPU time (in seconds) of our heuristic is less than the one of the mathematical formulation under 'CPU'. The results show that our proposed heuristic deviates from the optimal solution only 2% on the average. However, we gain about 82% improvement in the CPU time on the average. The most difficult case is the factor combination of (1 1 0 2), where the number of jobs, tooling cost and the range of due date are at their highest level and the tardiness factor is at its lowest level. Even in the worst case, our algorithm deviates 17% on the average. Both of these approaches can be used as a basic heuristic in Step 2.6 of the proposed PSGA. Even if the average CPU times for the mathematical model in Table 4 may seem small, since this step is called hundreds of times, its effect in the main algorithm becomes very significant.

5.2. Local search parameters and results

We now compare the proposed simultaneous algorithm with a two-stage sequential algorithm. We first generate different sequences using a PSGA based local search algorithm. For a given

Table 3
Summary results for the problem with given sequence.

Algorithms	Objective			Runtimes		
	Min	Average	Max	Min	Average	Max
DP-based heuristic	25.54	134.41	410.16	0.03	0.14	0.82
Math. model in GAMS	25.12	131.40	410.05	0.40	0.73	1.48

Table 4
Comparison of DP-based heuristic with Math model in GAMS.

N	C_t	TF	RDD	DP-based Algo.		Math. model		Deviations	
				Obj	CPU	Obj	CPU	Obj	CPU
0	0	0	0	31.68	0.09	30.90	0.54	0.02	0.83
0	0	0	1	32.03	0.11	31.15	0.57	0.03	0.81
0	0	0	2	35.23	0.13	33.74	0.59	0.04	0.79
0	0	1	0	46.91	0.06	46.78	0.52	0.00	0.88
0	0	1	1	47.18	0.06	47.03	0.52	0.00	0.88
0	0	1	2	49.65	0.07	49.41	0.53	0.00	0.87
0	0	2	0	72.05	0.06	72.05	0.46	0.00	0.88
0	0	2	1	72.40	0.06	72.39	0.47	0.00	0.88
0	0	2	2	71.35	0.06	71.34	0.49	0.00	0.89
0	1	0	0	77.34	0.14	73.51	0.62	0.05	0.77
0	1	0	1	77.86	0.15	73.61	0.63	0.05	0.77
0	1	0	2	82.31	0.16	75.53	0.65	0.08	0.75
0	1	1	0	94.39	0.08	93.71	0.56	0.01	0.86
0	1	1	1	95.10	0.08	93.92	0.58	0.01	0.86
0	1	1	2	98.87	0.10	95.91	0.58	0.03	0.83
0	1	2	0	121.10	0.06	121.07	0.53	0.00	0.89
0	1	2	1	121.30	0.06	121.25	0.53	0.00	0.89
0	1	2	2	120.31	0.06	119.97	0.54	0.00	0.88
1	0	0	0	68.59	0.18	67.50	0.82	0.02	0.78
1	0	0	1	72.04	0.26	70.16	0.91	0.03	0.71
1	0	0	2	83.20	0.43	78.78	1.02	0.05	0.58
1	0	1	0	119.83	0.13	119.74	0.68	0.00	0.80
1	0	1	1	121.96	0.14	121.87	0.73	0.00	0.81
1	0	1	2	131.53	0.15	131.43	0.79	0.00	0.82
1	0	2	0	210.87	0.10	210.87	0.61	0.00	0.83
1	0	2	1	212.40	0.11	212.40	0.60	0.00	0.82
1	0	2	2	210.29	0.11	210.28	0.61	0.00	0.81
1	1	0	0	177.69	0.31	166.90	1.15	0.06	0.73
1	1	0	1	189.32	0.35	167.90	1.27	0.11	0.72
1	1	0	2	211.56	0.46	176.29	1.31	0.17	0.65
1	1	1	0	230.80	0.15	229.31	1.01	0.01	0.85
1	1	1	1	234.51	0.15	231.20	1.05	0.01	0.85
1	1	1	2	244.08	0.17	239.93	1.09	0.02	0.84
1	1	2	0	324.23	0.11	324.15	0.81	0.00	0.86
1	1	2	1	325.58	0.11	325.48	0.81	0.00	0.86
1	1	2	2	323.13	0.12	322.96	0.87	0.00	0.87

sequence, in order to find the processing times we can either use the DP-based heuristic or solve the mathematical model using a commercial solver. Therefore, we compare the results of the following three algorithms:

- *The proposed simultaneous algorithms*: This is the proposed algorithm which is explained in Section 3. We could run the PSGA for two different base heuristics:
  - *PSGA*[*DP-based*]: It uses the DP-based heuristic explained in Section 4.
  - *PSGA* [*Math Model*]: It uses the math formulation of the problem for given sequences which is modeled in GAMS and solved by MINOS.
- *Sequential algorithm:* As discussed earlier, this is the algorithm that reflects the general approach in the current scheduling literature. The sequential algorithm consists of two stages as outlined in Algorithm 5. Although the sequential algorithm solves the subproblems in each stage optimally or almost optimally, it ignores the interaction between these two optimization problems. In Stage 2, we could use any algorithm to solve the $1||\sum w_iT_i$ problem using the locally optimum processing times found in Stage 1 for each job independently. There are several promising approaches in the literature to solve the $1||\sum w_iT_i$ problem quite effectively, such as Avci et al. [2], Kellegoz et al. [6], Maheswaran et al. [10], and Wang and Tang [15]. We have used the algorithm proposed by Avci et al. [2] in Stage 2. Since they also rely on the problem space genetic algorithm idea, this gives us a fair comparison to assess the relative advantage of using a simultaneous approach over a sequential approach.

Algorithm 5. Sequential algorithm.

Stage 1. Calculate the locally optimum machining conditions, which are the cutting speed and feed rate of the CNC machine (or equivalently corresponding processing times), to minimize the sum of machining and tooling costs for each job independently as discussed in Kayan and Akturk [5].
Stage 2. Solve the $1||\sum w_iT_i$ problem for the given processing times using the problem space genetic algorithm proposed by Avci et al. [2].

The parameters used in these PSGAs are given in Table 5. For the sequential PSGA, we assigned parameters to values that give the best results for the total weighted tardiness problem stated by Avci et al. [2]. As choosing the values of parameters for our proposed PSGAs, we used the same values for %SEXUAL, ϕ, θ, MUTPROB, and CROSSOVER. We chose the values of MAXGEN, POPSIZE and NUMSTART by considering the computational requirements.

The same factorial design in the previous section is used for all PSGAs listed above. However, this time, we take five replications for each factor combination since CPU times are much higher for PSGAs when compared to the algorithms in the previous section. Therefore, for each algorithm we took 180 randomly generated runs. The summary of the results are given in Table 6. On the average, the proposed PSGA [Dp-based] provides 62% improvement in solution quality over the sequential algorithm. The time-wise loss that corresponds to this improvement is only 86 CPU seconds. The results of PSGA[Dp-based] and PSGA [Math Model] are as expected. Since the only difference between these two PSGAs are their base heuristics, their time and solution quality comparative results are not much different from the comparative results of DP-based heuristic and mathematical formulation for a given sequence. On the average, the loss in solution quality in PSGA [Dp-based] is just 3% compared to PSGA [Math Model]. However, time-wise gain is about 361 s or 75%. In most studies in the literature, the processing times are assumed to be fixed at and equal to the most economical processing times in terms of manufacturing costs. As pointed out previously, this might not be the best alternative for scheduling-related criteria as demonstrated in Table 6. Controllable processing times provide an important flexibility in finding solutions with better overall objective function values.

The averages of five replications for each factor combination can be seen in Table 7. For all factor combinations, there is a significant cost-wise improvement. The runtime of PSGA[Dp-based] can be as low as 10 s and as high as 476 s, while runtime of

Table 5
Definitions and levels of PSGA parameters for the sequential and the proposed simultaneous algorithms.

Definitions of parameters	Sequential	Simultaneous
POPSIZE: size of the population in a generation	50	20
MAXGEN: number of generations	100	30
% SEXUAL: probability of sexual reproduction	0.8	0.8
CROSSOVER: crossover type in sexual reproduction	Single	Single
MUTPROB: mutation probability for each gene	0.05	0.05
ϕ: selectivity of the algorithm	4	4
θ: perturbation magnitude	1	1
NUMSTART: number of restarts	5	1

Table 6
Summary results of PSGA's for the problem.

Algorithms	Objective			Runtimes		
	Min	Average	Max	Min	Average	Max
Two-stage PSGA	26.39	246.40	984.15	8.48	33.71	79.83
PSGA[DP-based]	20.75	93.36	282.33	10.15	119.29	476.29
PSGA[GAMS]	20.75	90.42	282.87	300.55	480.23	859.81

Table 7
Comparison of the sequential and the proposed simultaneous algorithms.

N	C_t	*TF*	*RDD*	Sequential		PSGA[DP-based]		PSGA[GAMS]	
				Obj	CPU	Obj	CPU	Obj	CPU
0	0	0	0	49.15	10.89	24.96	83.31	24.92	296.97
0	0	0	1	64.50	10.70	23.73	71.88	23.35	304.92
0	0	0	2	70.37	9.11	22.48	77.61	22.19	308.27
0	0	1	0	72.86	10.55	32.21	41.96	31.93	275.96
0	0	1	1	81.74	10.62	30.49	46.12	29.80	285.26
0	0	1	2	95.46	9.52	29.87	46.12	28.71	274.66
0	0	2	0	106.97	10.37	47.00	28.56	46.80	251.13
0	0	2	1	106.18	10.14	45.56	28.74	45.58	250.20
0	0	2	2	118.00	9.52	46.37	29.47	45.03	249.41
0	1	0	0	138.71	10.95	58.90	88.38	57.96	322.38
0	1	0	1	148.32	10.83	59.02	71.08	56.47	325.26
0	1	0	2	166.80	9.88	55.63	72.49	54.40	327.03
0	1	1	0	170.78	11.07	70.66	52.35	70.38	306.79
0	1	1	1	172.67	10.96	70.44	57.65	68.14	307.30
0	1	1	2	198.70	9.82	69.88	48.69	67.48	303.50
0	1	2	0	210.21	9.92	89.54	29.20	89.26	282.63
0	1	2	1	202.72	9.38	89.23	31.43	87.68	279.76
0	1	2	2	223.46	9.02	89.03	32.85	87.14	276.51
1	0	0	0	114.11	54.61	52.90	197.85	51.41	475.91
1	0	0	1	134.36	40.38	50.55	229.91	47.64	540.17
1	0	0	2	142.78	45.14	51.24	313.44	45.70	584.54
1	0	1	0	185.31	51.63	75.16	80.60	73.34	410.46
1	0	1	1	201.67	37.00	70.33	86.83	69.07	405.22
1	0	1	2	226.78	47.83	73.00	99.29	71.68	397.48
1	0	2	0	294.62	47.70	128.13	56.08	127.46	326.97
1	0	2	1	299.13	39.32	135.96	56.19	132.90	330.88
1	0	2	2	341.28	40.46	140.09	56.58	138.90	339.80
1	1	0	0	374.20	47.29	137.15	226.80	125.96	653.18
1	1	0	1	377.78	41.37	133.31	232.28	121.27	658.71
1	1	0	2	427.82	47.03	135.17	268.00	119.05	661.15
1	1	1	0	476.72	46.88	166.16	99.82	161.54	568.34
1	1	1	1	469.41	41.83	169.34	94.78	160.84	570.60
1	1	1	2	541.59	45.45	177.17	85.90	169.00	498.64
1	1	2	0	606.89	47.12	233.04	57.85	229.17	457.64
1	1	2	1	586.60	36.38	237.79	58.30	234.60	446.81
1	1	2	2	671.74	39.67	239.37	61.71	238.47	461.38

PSGA[Math Model] is between 300 and 859 s and seem more stable. According to our computational results, both the sequential algorithm and PSGA[Math Model] do not exploit the problem structure to improve the solution quality, or to reduce the run time. However, the DP-based heuristic exploits the problem characteristics, such as position of a job in a given sequence, to improve the solution quality and computational requirements.

6. Conclusion

The integration of scheduling and process planning issues creates more realistic problems. In this paper, we proposed a joint algorithm to determine the processing time of each job and to schedule these jobs on a single CNC machine considering the total weighted tardiness and manufacturing cost. To the best of our knowledge, the single CNC machine scheduling problem that considers job-dependent tardiness penalties and controllable processing times simultaneously is not studied in the literature. We first compared the proposed DP-based heuristic with an exact algorithm for a given sequence and show that the deviation from the optimal solution is very small compared to the gain from the computational time. Afterwards, we compared the proposed PSGA with a two-stage sequential algorithm. The experimental results indicate that there is a significant interaction between machining conditions and weighted tardiness problems and solving these two problems together increases the cost effectiveness of the system greatly. Our results clearly indicate that the proposed PSGA that uses proposed DP-based heuristic gives the best results in terms of time-cost ratio. As a future research, for the CNC machines with an on-board tool magazine (that means machine has to be stopped in order to replace a cutting tool due to tool wear or part change due to limited tool magazine size), tool replacement decisions should be integrated to the joint problem in addition to the process planning and part sequencing problems since they also affect the part completion times.

Acknowledgments

The authors thank anonymous referees for their constructive comments.

References

[1] Al-Ahmari AMA. Computer aided optimization of scheduling and machining parameters in job shop manufacturing systems. Production Planning and Control 2002;13:401–6.
[2] Avci S, Akturk MS, Storer RH. A problem space genetic algorithm for single machine weighted tardiness problems. IIE Transactions 2003;35(5):479–86.
[3] Cheng TCE, Chen ZL, Chung-Lun L. Parallel machine scheduling with controllable processing times. IIE Transactions 1996;28(2):177–80.
[4] Gurel S, Akturk MS. Considering manufacturing cost and scheduling performance on a CNC turning machine. European Journal Operational Research 2007;177(1):325–43.
[5] Kayan RK, Akturk MS. A new bounding mechanism for the CNC machine scheduling problems with controllable processing times. European Journal Operational Research 2005;167(3):624–43.
[6] Kellegoz T, Toklu B, Wilson J. Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. Applied Mathematics and Computation 2008;199:590–8.
[7] Lawler EL. A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness. Annals of Discrete Mathematics 1977;1:331–42.
[8] Pinedo M. Scheduling: theory, algorithms and systems. 2nd ed. New Jersey: Prentice-Hall; 2002.
[9] Shabtay D, Kaspi M. Minimizing the total weighted flowtime in a single machine with controllable processing times. Computers & Operations Research 2004;31:2279–89.
[10] Maheswaran R, Ponnambalam SG, Jawahar N. Hybrid heuristic algorithms for single machine total weighted tardiness scheduling problems. International Journal of Intelligent Systems Technologies and Applications 2008;4:34–56.
[11] Shabtay D, Steiner G. A survey of scheduling with controllable processing times. Discrete Applied Mathematics 2007;155(13):1643–66.
[12] Storer RH, Wu SD, Vaccari R. New search spaces for sequencing problems with application to job shop scheduling. Management Science 1992;38(10): 1495–509.
[13] Tseng CT, Liao CJ, Huang KL. Minimizing total tardiness on a single machine with controllable processing times. Computers & Operations Research 2009;36:1852–8.
[14] Vickson RG. Choosing the job sequence and processing times to minimize processing plus flow cost on a single machine. Operations Research 1980;28: 1155–67.
[15] Wang X, Tang L. A population-based variable neighborhood search for the single machine total weighted tardiness problem. Computers & Operations Research 2009;36:2105–10.
[16] Xu K, Feng Z, Jun K. A tabu-search algorithm for scheduling jobs with controllable processing times on a single machine to meet due-dates. Computers & Operations Research 2010;37(11):1924–38.
[17] Yedidsion L, Shabtay D, Kaspi M. A bicriteria approach to minimize maximal lateness and resource consumption for scheduling a single machine. Journal of Scheduling 2007;10:341–52.