Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 2774–2789
computers & operations research
www.elsevier.com/locate/cor

Scheduling parallel CNC machines with time/cost trade-off considerations

Sinan Gurel, M. Selim Akturk*

Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey

Available online 15 December 2005

Abstract

When the processing times of jobs are controllable, selected processing times affect both the manufacturing cost and the scheduling performance. A well-known example for such a case that this paper specifically deals with is the turning operation on a CNC machine. Manufacturing cost of a turning operation is a nonlinear convex function of its processing time. We also know that scheduling decisions are quite sensitive to the processing times. Therefore, this paper considers minimizing total manufacturing cost (F_1) and total completion time (F_2) objectives simultaneously on identical parallel CNC turning machines. Since decreasing processing time of a job increases its manufacturing cost, we cannot minimize both objectives at the same time, so the problem is to generate non-dominated solutions. We consider the problem of minimizing F_1 subject to a given F_2 level and give an effective formulation for the problem. For this problem, we prove some optimality properties which facilitated designing an efficient heuristic algorithm to generate approximate non-dominated solutions. Computational results show that proposed algorithm performs almost equal with the GAMS/MINOS commercial solver although it spends much less computation time.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Parallel machine scheduling; Controllable processing times; CNC machines; Manufacturing cost; Total completion time

1. Introduction

There are many industry applications where the processing times are controllable. Controllable processing time for a manufacturing operation imply a trade-off between the processing time and the manufacturing cost. Therefore, in order to make appropriate processing time and scheduling decisions, we need to investigate the existing time/cost trade-off between manufacturing cost objective and the scheduling objective under consideration. A well-known example for controllable processing times is a CNC turning machine, which we specifically deal with in this study. Processing time of a turning operation on a CNC machine can be controlled by setting the machining parameters: cutting speed and feed rate. We can decrease its processing time by increasing the cutting speed and/or feed rate, which increases the manufacturing cost of the operation. Most of the studies in the scheduling literature assume fixed processing times, i.e. ignore the manufacturing cost performance, and focus on scheduling performance measures. Differently, we investigate how to make scheduling and process planning decisions simultaneously while considering a scheduling objective and manufacturing cost objective at the same time. In this study, we consider identical parallel CNC turning machines on which we have two objectives to minimize: total completion time and total manufacturing cost.

* Corresponding author. Tel.: +90 312 266 4126; fax: +90 312 266 4054.
E-mail address: akturk@bilkent.edu.tr (M.S. Akturk).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.10.018

There is an extensive literature on process planning decisions for a turning operation. Different objectives were considered such as minimizing processing time and minimizing manufacturing cost. Most of the studies consider the manufacturing cost as a sum of tooling cost and operating cost. Tooling cost increases as the processing time is decreased by increasing the cutting speed and/or feed rate. Operating cost of a job increases by the processing time since it is the cost of running the machine for the given job. Kayan and Akturk [1] show that manufacturing cost of a turning operation can be expressed as a nonlinear convex function of its processing time. They also provide a mechanism to determine the upper and lower bound for the processing time of a turning operation. When manufacturing cost and scheduling cost objectives are considered simultaneously, process planning and scheduling decisions affect each other. Combining the process planning and scheduling decisions for CNC turning machines is an important contribution of this paper.

In the scheduling literature, controllable processing times area was initiated by Vickson [2] where he considered minimizing sum of total completion time and total processing cost on a single machine. He formulated the problem as an assignment problem and showed that it can be solved in polynomial time. Most of the existing studies assumed linear processing or compression cost functions. A nonlinear relation between processing time and used resource is considered by Shabtay and Kaspi [3]. They deal with the problem of minimizing total weighted completion time on a single machine under a maximal resource constraint where processing time of each job is a nonlinear function of allocated resource to it. A review on controllable processing times in multi-objective scheduling is included in the recent review by Hoogeveen [4].

In recent years, many researchers have been dealing with multi-objective scheduling on parallel machines. Some of those studies consider fixed processing times. Gupta and Ruiz-Torres [5] considered the objectives of minimizing total flow time and minimizing total number of tardy jobs simultaneously and proposed heuristic algorithms to generate efficient solutions. Gupta and Ho [6] provided solution methods for the problem of minimizing makespan subject to minimum flow time for two parallel machines. Cao et al. [7] considered the machine selection and scheduling decisions together in order to minimize the sum of machine cost and job tardiness. Alagoz and Azizoglu [8] studied a problem with the objectives of minimizing total completion time and minimizing number of disrupted jobs in a rescheduling environment. Controllable processing times and time/cost trade-offs have also received increasing attention in the recent parallel machine scheduling literature. The first study dealing with controllable processing times on parallel machines is by Alidaee and Ahmadian [9] who solved the problem of minimizing sum of total completion time and total processing cost by extending the Vickson's [2] approach. They considered linear processing cost functions and their approach was extended to nonlinear convex cost function case by Cheng et al. [10]. Jansen and Mastrolilli [11] provided polynomial time approximation schemes for the problem of minimizing two objectives on identical parallel machines: total processing cost and makespan.

In this study, we have two objectives to minimize: total completion time and total manufacturing cost. Since the manufacturing cost of a job increases as the processing time is decreased, we cannot minimize both objectives at the same time. Therefore, our focus will be on finding efficient solutions for this bicriteria problem. Then, we deal with the problem of minimizing total manufacturing cost subject to a total completion time constraint. This problem is more difficult than minimizing the sum of two objective functions which was usually done in the literature. For this problem, we propose an effective formulation which can effectively be solved by commercial nonlinear programming solvers. Using this formulation, we also give useful properties for the problem which allowed us to develop an algorithm that can generate a large set of approximate efficient solutions in a short computation time. Although we specifically discuss the CNC turning machine case, our results are applicable to the cases where nonlinear convex cost functions are considered. Furthermore, our results are also valid for the problems considering linear cost functions which is not also considered in the literature to the best of our knowledge.

In a recent study, Shabtay and Kaspi [12] considered minimizing the total completion time subject to a maximal resource constraint on parallel machines. They assume a nonlinear convex resource consumption function $r_j = w_j p_j^k$ where p_j is the processing time of job j, r_j is the amount of resource allocated to job j, w_j is a job specific constant and k is a negative exponent which is same for all jobs. This resource consumption function corresponds to a special case of our tooling cost term in the manufacturing cost function such that all jobs require the same cutting tool type. However, usually this is not the case in CNC machining, each job may require different cutting tool type and each job could have a different nonlinear manufacturing cost function due to different operational and surface quality requirements. Another difference in our case is that we have an upper bound on the processing time of each job at which the manufacturing cost is minimum. This is because we have an operating cost term which balances the tooling cost at some point. This is

a realistic approach since running a machine longer requires additional cost of energy and labor. Moreover, we have a lower bound on the processing time of each job due to surface quality and CNC machine power requirements. All these complicating issues make our problem more difficult and the analysis of Shabtay and Kaspi [12] cannot be extended to our case.

In the next section, we give the problem definition and provide mathematical formulation for the problem. In Section 3, we prove some useful properties for the problem. In Section 4, we propose an algorithm which generates approximate efficient solutions for the problem. In Section 5, we discuss our findings on a numerical example, and report the computational results in Section 6. Finally, we give concluding remarks in Section 7.

2. Problem definition

The notation used throughout the paper is as follows:

p_i	processing time of job i
p_i^{l}	processing time lower bound for job i
$f_i(p_i)$	manufacturing cost function of processing time for job i
p_i^{u}	processing time level that gives the minimum manufacturing cost ($f_i(p_i)$) for job i
C_{o}	unit operating cost for CNC turning machines ($/min)
m_i, e_i	tooling cost multiplier and exponent for job i

We have N jobs to be machined on M identical parallel CNC turning machines. Each job corresponds to a different turning operation to be performed on one of the machines so that each job has different cutting properties such as diameter, length, allowable surface roughness and cutting tool. Therefore, each job has a different $f_i(p_i)$, and different p_i^{l} and p_i^{u}. Since the machines are identical, $f_i(p_i)$, p_i^{l} and p_i^{u} for job i are same for all machines. Each CNC machine can perform one job at a time. The problem is to find the best schedule and processing times for each job in order to minimize total manufacturing cost and total completion time.

The problem of selecting optimal machining parameters for a turning operation has been extensively studied in the literature. The objective function to minimize is the sum of tooling and operating costs subject to tool life, surface roughness and machine power constraints. Kayan and Akturk [1] proved that the surface roughness constraint is always tight at the optimal solution and this allowed expressing the manufacturing cost function of job i as a function of p_i as follows:

$$f_i(p_i) = C_{\mathrm{o}} p_i + m_i p_i^{e_i}.$$

The first term is the operating cost which is the cost of running the machine for job i and it is an increasing linear function of p_i. The second term is the tooling cost which is the cost of tool usage for job i and it is a nonlinear decreasing function of p_i. Since $m_i > 0$ and $e_i < 0$ always hold, $f_i(p_i)$ is a nonlinear convex function. However, processing time of job i is subject to a lower bound p_i^{l} due to the technical constraints stated above. The manufacturing cost of a turning operation is minimum at a processing time level p_i^{u}. p_i^{u} is either equal to the processing time level that minimizes $f_i(p_i)$ or equal to p_i^{l} whichever is bigger. For a detailed discussion on how $f_i(p_i)$ is formed and how p_i^{l} and p_i^{u} are determined, we refer to Kayan and Akturk [1].

If we consider the minimization of total completion time as a single objective on identical parallel machines with fixed processing times we can solve the problem by using the following properties:

Property 1. The shortest processing time first (SPT) rule is optimal.

The SPT rule as given in Pinedo [13] is to schedule the smallest job on machine 1 at time zero, schedule the second smallest job on machine 2, and so on; the $(M+1)$th smallest job follows the smallest job on machine 1, $(M+2)$th smallest job follows the second smallest job on machine 2, and so on. The second property, due to Conway et al. [14], is as follows.

Property 2. One can interchange jobs in equivalent positions in sequence on different machines without having any effect on the total completion time.

Two positions on different machines are equivalent if the number of jobs succeeding these positions on the same machines are equal. Property 2 implies the existence of many alternative optimal schedules for the problem.

Using Properties 1 and 2, we can determine how many jobs will be scheduled on each machine in the optimal solution. Then, we can find number of positions on each machine and determine which positions on different machines will be equivalent so that we can form sets of equivalent positions. Using this observation for the total completion time problem with fixed processing times, we can formulate our bicriteria problem with controllable processing times as an assignment problem of jobs to sets of equivalent positions.

In an optimal solution for the total completion time problem with fixed processing times, l machines will have $(\lfloor N/M \rfloor + 1)$ jobs where $l \equiv (N \bmod M)$ with minimal $l \geqslant 0$. The rest of the machines will have $\lfloor N/M \rfloor$ jobs. We define $S(j)$ as the number of positions in set j. We define $N(j)$ as the number of jobs succeeding each job in set j on the same machine plus the job itself. We assume that position 1 is the first position on machine 1, position 2 is the first on machine 2 and M is the first on machine M, position $M+1$ is the second position on machine 1, $M+2$ is the second on machine 2 and so on. If $l > 0$ then there will be $(\lfloor N/M \rfloor + 1)$ sets and set 1 will include position 1 to position l so that $S(1) = l$ and $N(1) = \lceil N/M \rceil$. Set 2 will include position $(l+1)$ to position $(l+M)$ so that $S(2) = M$ and $N(2) = \lceil N/M \rceil - 1$ and, so on. If $l = 0$, then each machine will have equal number of jobs and there will be N/M sets. Each set j will include M positions from position $((j-1)M+1)$ to jM. Then, we can determine $N(j)$ for each set j as follows:

$$N(j) = \left\lceil \frac{N}{M} \right\rceil - j + 1.$$

Then, a mathematical formulation for the bicriteria problem with controllable processing times is as follows, where X_{ij} is the binary variable which controls if job i is placed in one of the positions in set j.

$$\min F_1 \quad \sum_{i=1}^{N} f_i(p_i) = \sum_{i=1}^{N} C_o p_i + m_i p_i^{e_i}$$

$$\min F_2 \quad \sum_{i=1}^{N} \sum_{j} N(j) X_{ij} p_i$$

$$\text{s.t.} \quad \sum_{j} X_{ij} = 1 \quad i = 1, \ldots, N, \tag{1}$$

$$\sum_{i=1}^{N} X_{ij} = S(j) \quad \forall j, \tag{2}$$

$$p_i^{l} \leqslant p_i \leqslant p_i^{u} \quad i = 1, \ldots, N, \tag{3}$$

$$X_{ij} \in \{0, 1\} \quad \forall i, j. \tag{4}$$

In the mixed integer nonlinear programming (MINLP) model above, the first objective function F_1 is the total manufacturing cost. F_1 equals the sum of N nonlinear convex manufacturing cost functions, so it is a convex function. The second objective function F_2 is the total completion time. Total completion time is a nonlinear function since it is a sum of nonlinear terms. Constraint set (1) forces each job to be assigned to a position set. Constraint set (2) fixes the number of jobs to be assigned to each position set. In each set there are certain number of positions and the number of jobs assigned to a set must be equal to the number of positions in the set. Constraint set (3) sets the processing time lower and upper bounds for each job. In the next section, we define a single objective problem and give optimality properties on it.

3. Optimality properties

As discussed earlier, we cannot minimize both objectives at the same time. Therefore, we need to find efficient solutions for the problem. A solution Z to a bicriteria problem is efficient if there exists no other solution which is better than Z in one of the criteria and not worse in the other. Since we have the SPT rule as an optimal strategy for

the total completion time problem, we can determine the efficient solution Z_1 with the minimum total completion time K^{l} and the maximum manufacturing cost by setting $p_i = p_i^{\mathrm{l}}$ for all i and by applying the SPT rule. Similarly, we can find another efficient solution Z_2 with the minimum manufacturing cost by setting $p_i = p_i^{\mathrm{u}}$ for all i and by applying the SPT rule. We denote the total completion time at Z_2 as K^{u}. Two solutions, Z_1 and Z_2, are the points that we can find by just using the SPT rule on processing time lower and upper bounds, respectively. However, a decision maker may find the manufacturing cost for Z_1 too high to pay or may find the total completion time for Z_2 too high. In order to find the efficient solutions other than Z_1 and Z_2, we can consider F_1 as a single objective to minimize subject to an F_2 constraint and formulate a single objective problem which we call SOP as follows:

$$\begin{aligned} \text{SOP} \quad & \min\ F_1 \quad \sum_{i=1}^{N} f_i(p_i) \\ & \text{s.t.} \quad \sum_{i=1}^{N} \sum_{j} N(j) X_{ij}\, p_i \leqslant K, \\ & \text{and} \quad (1), (2), (3), (4). \end{aligned} \tag{5}$$

Constraint (5) guarantees that total completion time (F_2) of the schedule is less than or equal to a predefined value K. An alternative way of modeling the SOP is formulating it as an assignment problem of jobs to individual positions as commonly done in the literature. Such a formulation would not be using the result in Property 2. It would require more variables and constraints in the model. We can solve the SOP model by using MINLP solvers. Then, we can generate a set of n efficient solutions between Z_1 and Z_2 by using the following algorithm denoted as the solver based approach (SBA).

SBA algorithm

Step 1. Find solutions Z_1 and Z_2, and calculate K^{u} and K^{l} by using the SPT rule for p^{u} and p^{l} values, respectively.
Step 2. Set $\Delta = (K^{\mathrm{u}} - K^{\mathrm{l}})/(n+1)$.
Step 3. For $k = 1$ to n, solve the SOP for $K = K^{\mathrm{l}} + k\Delta$.

We found the following useful properties for the SOP formulation above. These properties improved the SBA method and provided a clearer interpretation of the problem.

Lemma 1. *When $K \leqslant K^{\mathrm{u}}$, constraint* (5) *on F_2 must be tight at the optimal solution.*

Proof. When $K > K^{\mathrm{u}}$, total manufacturing cost can be minimized by setting $p_i = p_i^{\mathrm{u}}$ for all i and constraint (5) can be satisfied by applying the SPT rule. In such a case, constraint (5) is loose. When $K \leqslant K^{\mathrm{u}}$, if constraint (5) is loose, then, it is sure that we have at least one job i such that $p_i < p_i^{\mathrm{u}}$ and by increasing p_i we can decrease f_i so that we could improve F_1. Therefore, when $K \leqslant K^{\mathrm{u}}$, a solution cannot be optimal if constraint (5) is loose. □

In our single objective problem, we model the total completion time as a resource constraint to be used to minimize total manufacturing cost. Therefore, Lemma 1 states that when this resource is scarce ($K \leqslant K^{\mathrm{u}}$), we must fully utilize it. Lemma 1 also implies that an optimal solution for the SOP must have the minimum total completion time for the optimal processing times. This implies an optimal solution must satisfy the rules in Properties 1 and 2. To further explore the problem we next consider the relaxed SOP where X_{ij}'s are allowed to take any values in the interval [0,1]. The next property is an extension of Lemma 1 for locally optimal solutions to the relaxed SOP.

Corollary 1. *For the relaxed SOP, when $K \leqslant K^{\mathrm{u}}$ constraint* (5) *on F_2 must be tight at locally optimal solutions.*

Next property is an important one which states that any local optimal solution to the relaxed problem has binary X_{ij}'s. Non-integer local optimal solutions may exist only if there are multiple jobs having identical processing times in the solution. However, such non-integer solutions are alternative solutions for existing integer local optimal solutions.

Lemma 2. *When $K \leqslant K^{\mathrm{u}}$, in a local optimal solution for the relaxed SOP, a job cannot be assigned to multiple position sets, i.e. a local optimal solution must have integer X_{ij}'s.*

Proof. Consider two jobs i_1 and i_2 placed to positions in different sets j_1 and j_2, respectively. Suppose that $p_{i_1} < p_{i_2}$ and $N(j_1) > N(j_2)$. Consider a local optimal solution Z for the relaxed problem. The assignment variables for jobs i_1 and i_2 and positions j_1 and j_2 in Z are as follows:

$$X_{i_1 j_1} = \rho_1 \quad \text{and} \quad X_{i_2 j_1} = \rho_2,$$

$$X_{i_1 j_2} = \xi_1 \quad \text{and} \quad X_{i_2 j_2} = \xi_2 \quad \text{where } \rho_1,\ \rho_2,\ \xi_1 \text{ and } \xi_2 \text{ are positive.}$$

Total completion time calculated for Z is as follows:

$$\begin{aligned} F_2(Z) &= \sum_{i=1}^{N} \sum_{j} N(j) X_{ij} p_i \\ &= \Phi + N(j_1)[\rho_1 p_{i_1} + \rho_2 p_{i_2}] + N(j_2)[\xi_1 p_{i_1} + \xi_2 p_{i_2}] \\ &= \Phi + [N(j_1)\rho_1 + N(j_2)\xi_1] p_{i_1} + [N(j_1)\rho_2 + N(j_2)\xi_2] p_{i_2}, \end{aligned}$$

where Φ is a constant value. Suppose that without changing the processing times, we change the assignment variables to get a new solution Z'. Setting $\delta = \min(\xi_1, \rho_2)$, new values for the assignment variables are as follows:

$$X_{i_1 j_1} = \rho_1 + \delta, \quad X_{i_2 j_1} = \rho_2 - \delta,$$

$$X_{i_1 j_2} = \xi_1 - \delta \quad \text{and} \quad X_{i_2 j_2} = \xi_2 + \delta.$$

By this arrangement, we reallocate these two jobs to position sets j_1 and j_2 such that we increase job i_1's ratio in the preceding position set j_1 without disturbing the assignment constraints.

Total completion time of the solution after this arrangement is

$$F_2(Z') = \Phi + [N(j_1)(\rho_1 + \delta) + N(j_2)(\xi_1 - \delta)] p_{i_1} + [N(j_1)(\rho_2 - \delta) + N(j_2)(\xi_2 + \delta)] p_{i_2}.$$

Then, $F_2(Z') - F_2(Z) = \delta(N(j_1) - N(j_2))(p_{i_1} - p_{i_2}) < 0$, since $N(j_1) > N(j_2)$ and $p_{i_1} < p_{i_2}$. Then, since $K \leqslant K^{\mathrm{u}}$ we can improve total manufacturing cost by increasing processing times. This proves that there exists an improving feasible direction for solution Z so that Z cannot be a local optimal solution. When we generalize this result for all jobs and position sets, we conclude that a solution with non-integer X_{ij}'s cannot be a local optimal solution for the problem. However, for the cases of $p_{i_1} = p_{i_2}$ we may have alternative non-integer local optimal solutions. □

This result is an extremely important one since it shows that although our problem is a MINLP problem, we can employ nonlinear programming (NLP) solvers to solve its relaxed form and achieve integer solutions. This is critical since NLP solvers are computationally more efficient than MINLP solvers. If the objective function is convex, in general, NLP solvers can only guarantee to find local optimal solutions. However, if the feasible region for the problem is a convex set, a local optimal solution is globally optimal. We next check if the feasible region for the problem is a convex set to see whether NLP solvers can guarantee to find the global optimum for the problem.

Lemma 3. *The feasible region for the relaxed SOP is not a convex set.*

Proof. Consider two jobs i_1 and i_2 in a schedule called A_1. They are assigned at positions in sets k and $k+1$, respectively. Suppose that $N(k) = r + 1$ and $N(k + 1) = r$ and the processing times are $p_{i_1} = s_1$ and $p_{i_2} = s_2$, where $s_1 < s_2$.

Further, suppose that $F_2(A_1) = Q + (r + 1)s_1 + rs_2 = K$, where Q is a constant.

Consider another schedule A_2 which is identical to A_1 except that job i_1 is assigned to the position set $k + 1$ and i_2 is assigned to the position set k with processing times $p_{i_1} = q_1$ and $p_{i_2} = q_2$, where $q_2 < q_1$. Suppose that $F_2(A_2) = Q + (r + 1)q_2 + rq_1 = K$.

Next, define a point A which is a convex combination of A_1 and A_2 as follows:

$A = \lambda A_1 + (1-\lambda)A_2$, where $0 < \lambda < 1$. At point A, $p_{i_1} = \lambda s_1 + (1-\lambda)q_1$ and $p_{i_2} = \lambda s_2 + (1-\lambda)q_2$. Also, $X_{i_1 k} = \lambda$, $X_{i_1(k+1)} = (1-\lambda)$, $X_{i_2 k} = (1-\lambda)$ and $X_{i_2(k+1)} = \lambda$.

$$\begin{aligned} F_2(A) &= Q + [(r+1)\lambda + r(1-\lambda)]p_{i_1} + [(r+1)(1-\lambda) + r\lambda]p_{i_2} \\ &= Q + \lambda^2[(r+1)s_1 + rs_2] + (1-\lambda)^2[(r+1)q_2 + rq_1] \\ &\quad + \lambda(1-\lambda)[(r+1)q_1 + rq_2] + \lambda(1-\lambda)[(r+1)s_2 + rs_1] > K, \end{aligned}$$

since $s_1 < s_2$ and $q_2 < q_1$. This shows that feasible region for the relaxed problem is not a convex set. □

This lemma indicates that NLP solvers may not be able to find global optimum, so they only guarantee to achieve local optimal solutions. Although the complexity of the problem is open, this lemma supports the difficulty of the problem since Murty and Kabadi [15] states that, in general, computing a global minimum in a non-convex NLP is an NP-hard problem.

We showed that NLP solvers guarantee to find integer local optimal solutions for the SOP. A way of solving this problem to global optimum is to solve for processing times for all possible job-position allocations which is computationally inefficient except for small instances. In order to find the global optimal solution for the single objective problem by using MINLP solvers, we next present a linearized single objective problem LSOP model below. In this model, constraint (5) is replaced with constraint set (6)–(10) in which the term $X_{ij}p_i$ is replaced with the variable Y_{ij}. The parameter B in the following model denotes a large positive number:

$$\begin{aligned} \text{LSOP} \quad \min F_1 \quad & \sum_{i=1}^{N} f_i(p_i) \\ \text{s.t.} \quad & \sum_{i=1}^{N}\sum_{j} N(j)Y_{ij} \leqslant K, && (6) \\ & Y_{ij} \geqslant p_i + (X_{ij}-1)B \quad \forall i, j, && (7) \\ & Y_{ij} \leqslant p_i + (1-X_{ij})B \quad \forall i, j, && (8) \\ & Y_{ij} \leqslant BX_{ij} \quad \forall i, j, && (9) \\ & Y_{ij} \geqslant 0 \quad \forall i, j, && (10) \\ \text{and} \quad & (1), (2), (3), (4). \end{aligned}$$

By replacing the terms $X_{ij}p_i$ with the variable Y_{ij} in constraint (5) we obtain constraint (6) in the above model. To assure the equivalence of the term $X_{ij}p_i$ and the variable Y_{ij} we add constraint sets (7)–(10) so that if $X_{ij} = 0$ then $Y_{ij} = 0$ and if $X_{ij} = 1$ then $Y_{ij} = p_i$. However, Lemma 2 is no longer valid for this linearized model since the linearization is only possible for binary X_{ij}'s, so we cannot use NLP solvers to solve the LSOP, instead we can use a MINLP solver to find the global optimal solution.

In this section, we introduced the SOP model and proposed the SBA method to generate a set of efficient solutions. We showed that NLP solvers can also be employed in SBA method to obtain approximate efficient solutions. We also gave the LSOP model which can be solved to global optimum by the MINLP solvers. In the next section, we propose a heuristic method which generates approximate efficient solutions for the bicriteria problem. Proposed method achieves almost equal solution quality compared to commercial NLP and MINLP solvers although it spends much less computation time.

4. A heuristic method to generate approximate efficient solutions

In this section, we first state a very important optimality property for the single objective problem. Different from the properties in Section 3, this optimality property states a relationship that must hold between positions and processing times of the jobs in a local optimal solution. Based on this property we propose a heuristic algorithm which generates a set of approximate efficient solutions. A solution may be viewed as approximately efficient if it is efficient with respect to a large set of known solutions for a given problem. Since we are using a heuristic approach to find the minimum

manufacturing cost value, F_1, for a given total completion time value, we denote these solutions as approximate efficient solutions. That means these solutions do not dominate each other but there can be another solution generated by an exact algorithm that could dominate any one of them.

Lemma 4. *Let i_1, i_2 be a pair of jobs in a local optimum and $n_{i_k} = \sum_j X_{i_k j} N(j)$, $k \in \{1, 2\}$. Then the optimal processing times p_{i_1}, p_{i_2} must satisfy the following conditions*:

(i) *If $p_{i_1} > p_{i_1}^{\mathrm{l}}$ and $p_{i_2} > p_{i_2}^{\mathrm{l}}$ then*

$$\frac{1}{n_{i_1}} \frac{\partial f_{i_1}(p_{i_1})}{\partial p_{i_1}} = \frac{1}{n_{i_2}} \frac{\partial f_{i_2}(p_{i_2})}{\partial p_{i_2}}.$$

(ii) *If $p_{i_1} = p_{i_1}^{\mathrm{l}}$ and $p_{i_2} > p_{i_2}^{\mathrm{l}}$ then*

$$\frac{1}{n_{i_1}} \frac{\partial f_{i_1}(p_{i_1})}{\partial p_{i_1}} \geqslant \frac{1}{n_{i_2}} \frac{\partial f_{i_2}(p_{i_2})}{\partial p_{i_2}}.$$

Proof. Suppose that in a solution $(1/n_{i_1})(\partial f_{i_1}(p_{i_1})/\partial p_{i_1}) < (1/n_{i_2})(\partial f_{i_2}(p_{i_2})/\partial p_{i_2})$ for jobs i_1 and i_2. Then,

$$\lim_{\Delta p \to 0} \left(\frac{f_{i_1}(p_{i_1} + \Delta p) - f_{i_1}(p_{i_1})}{n_{i_1} \Delta p} - \frac{f_{i_2}(p_{i_2}) - f_{i_2}(p_{i_2} - (n_{i_1}/n_{i_2})\Delta p)}{n_{i_2}(n_{i_1}/n_{i_2})\Delta p} \right) < 0,$$

$$\lim_{\Delta p \to 0} \left(\frac{f_{i_1}(p_{i_1} + \Delta p) - f_{i_1}(p_{i_1}) - f_{i_2}(p_{i_2}) + f_{i_2}(p_{i_2} - (n_{i_1}/n_{i_2})\Delta p)}{n_{i_1} \Delta p} \right) < 0.$$

Then, $\exists\ \Delta p > 0$ s.t. $f_{i_1}(p_{i_1} + \Delta p) + f_{i_2}(p_{i_2} - (n_{i_1}/n_{i_2})\Delta p) - f_{i_1}(p_{i_1}) - f_{i_2}(p_{i_2}) < 0$, which means the current solution can be improved by increasing p_{i_1} by Δp and decreasing p_{i_2} by $(n_{i_1}/n_{i_2})\Delta p$. This proves that a local optimal solution must satisfy the conditions above. □

This lemma states that there is a unique solution for the processing times of jobs which is optimal for a given job-position set allocation. When a job-position set allocation is given, we can find the processing times of the jobs by employing a search algorithm and using Lemma 4. Lemma 4 also implies that we cannot move from a local optimal solution to another one without changing the job-position set allocation since there is a unique local optimal solution for a certain job-position set allocation. It implies that we can find the global optimal solution for the problem by trying all possible job-position set allocations and solve each case for optimal processing times.

Using the information in Lemma 4 we propose an approximation algorithm, which we call as the most profitable job first (MPJ) algorithm, to find a set of approximate efficient solutions other than Z_1 and Z_2. MPJ algorithm starts with the efficient solution Z_1 which has the minimum total completion time but the maximum manufacturing cost. To generate a new approximate efficient solution, we want to select a job and increase its processing time. This will decrease the total manufacturing cost but increase the total completion time. Selecting the job to be perturbed is a very critical decision, and hence we propose a new measure t_i as follows:

$$t_i = \frac{\partial f_i(p_i)/\partial p_i}{N(j)}.$$

We can interpret t_i as the estimated cost change per estimated unit total completion time change when processing time of job i is increased by one unit. This makes sense since we want to find efficient solutions by achieving maximum cost decrease per unit total completion time increase. As we know from Lemma 4, in an optimal solution t_i values must be equal for the jobs whose processing times are higher than their lower bounds. By selecting the job with the minimum t_i value, we want to satisfy or at least be very close to satisfying Lemma 4 at each step. In each iteration of MPJ, we increase the processing time of a selected job by a predefined value Δ so that we can improve total manufacturing cost while giving up from the total completion time. After increasing the processing time of the selected job, we check whether the SPT rule, specified in Property 1, is satisfied for the new schedule or not. If not, the SPT rule is applied. At the end of each iteration, we achieve a new solution with a better manufacturing cost but a higher total completion time

than the previous solution. The proposed MPJ algorithm generates a set of approximate efficient solutions as outlined in the following:

MPJ algorithm

Step 1. Find the non-dominated solutions Z_1 and Z_2.

Step 2. Start with the solution Z_1, set $F_1^{new} = F_1(Z_1)$ and $F_2^{new} = F_2(Z_1)$. While $F_2^{new} < K^u$ do the following:

Step 2.1. Select job i with the minimum t_i. If there are more than one such jobs, select the one with the longest processing time.

Step 2.2. Set $p_i = p_i + \Delta$.

Step 2.3. If the SPT rule is violated then resequence the jobs by the SPT rule.

Step 2.4. Update t_i indices for the perturbed job and for all other jobs whose position in sequence is changed in Step 2.3.

Step 2.5. Update $F_1^{new} = F_1^{new} - [f_i(p_i) - f_i(p_i + \Delta)]$ and recalculate F_2^{new}.

Step 2.6. If $F_2^{new} < K^u$, report the current schedule with F_1^{new} and F_2^{new} as a new approximate efficient solution.

The MPJ algorithm first finds two efficient solutions Z_1 and Z_2 in Step 1, then generates a new approximate efficient solution in between these two points in each iteration as shown in the following.

Lemma 5. *In each iteration of the MPJ algorithm, we generate a new approximate efficient solution.*

Proof. At each iteration of the MPJ algorithm, processing time of a selected job is increased by Δ amount. Since the total completion time is a regular scheduling measure, total completion time of new schedule is strictly higher than the previous one. Similarly, this increase will strictly decrease the total manufacturing cost. This means new solution cannot dominate previously generated solutions. Therefore, no two solutions generated by the MPJ algorithm can dominate each other. □

The MPJ algorithm generates a set of approximate efficient solutions which cannot dominate each other as shown above. We then utilize the set of these discrete points to approximate the efficient frontier. In the MPJ algorithm, Δ is a very critical parameter which affects the quality of the solutions achieved. By using a smaller Δ value, the MPJ algorithm can generate more solutions each of which is more close to satisfy the conditions given in Lemma 4. In the next section, we will discuss the given properties and the algorithms on a numerical example.

5. Numerical example

In this section, we give a five jobs–two machines problem as an example to illustrate the properties and algorithms that we have discussed above. As we have discussed before, due to different job and tool properties each job has different p_i^l and p_i^u levels. In this numerical example we use the following manufacturing cost functions and processing time bounds:

$$f_1(p_1) = 0.25p_1 + 3.30p_1^{-1.29} \quad \text{where } 1.65 \leqslant p_1 \leqslant 3.45,$$
$$f_2(p_2) = 0.25p_2 + 0.02p_2^{-1.71} \quad \text{where } 0.20 \leqslant p_2 \leqslant 0.48,$$
$$f_3(p_3) = 0.25p_3 + 0.20p_3^{-1.22} \quad \text{where } 0.42 \leqslant p_3 \leqslant 0.99,$$
$$f_4(p_4) = 0.25p_4 + 0.03p_4^{-1.22} \quad \text{where } 0.18 \leqslant p_4 \leqslant 0.43,$$
$$f_5(p_5) = 0.25p_5 + 0.20p_5^{-1.40} \quad \text{where } 0.36 \leqslant p_5 \leqslant 1.05.$$

Since we have five jobs and two machines, we have three sets of equivalent positions. The first position is for the shortest job which will be succeeded by two positions on the same machine. The second and third positions are equivalent positions so they form a position set. Each one is succeeded by a single position. The fourth and fifth positions form the last set of equivalent positions and each one is the last position on its machine. We can assume that positions 1, 3 and 5 are on machine 1 and positions 2 and 4 are on machine 2.

We can find the solutions Z_1 and Z_2 given in Table 1 as discussed in Section 2. At Z_1, total completion time is 3.73 (ideal total completion time) and total manufacturing cost is 4.40. At Z_2, total completion time is 8.79 and total

Table 1
Schedules at Z_1 and Z_2

Position	Z_1				Z_2			
	Job i	p_i	$f_i(p_i)$	t_i	Job i	p_i	$f_i(p_i)$	t_i
1	4	0.18	0.29	−0.56	4	0.43	0.19	0.0
2	2	0.20	0.34	−1.22	2	0.48	0.19	0.0
3	5	0.36	0.93	−1.50	3	0.99	0.45	0.0
4	3	0.42	0.68	−1.42	5	1.05	0.45	0.0
5	1	1.65	2.14	−1.10	1	3.45	1.53	0.0
	$F_1(Z_1) = 4.40$				$F_1(Z_2) = 2.81$			

Table 2
Results of the first seven iterations of MPJ algorithm

Iteration	Job i	p_i	Sequence	t_1	t_2	t_3	t_4	t_5	F_2	F_1
0			4 2 5 3 1	−1.1	−1.22	−1.42	−0.47	−1.50	3.73	4.40
1	5	0.46	4 2 3 5 1	−1.1	−1.22	−0.71	−0.47	−1.56	3.89	4.18
2	5	0.56	4 2 3 5 1	−1.1	−1.22	−0.71	−0.47	−0.88	3.99	4.07
3	2	0.30	4 2 3 5 1	−1.1	−0.32	−0.71	−0.47	−0.88	4.19	3.93
4	1	1.75	4 2 3 5 1	−0.93	−0.32	−0.71	−0.47	−0.88	4.29	3.83
5	1	1.85	4 2 3 5 1	−0.79	−0.32	−0.71	−0.47	−0.88	4.39	3.75
6	5	0.66	4 2 3 5 1	−0.79	−0.32	−0.71	−0.47	−0.51	4.49	3.68
7	1	1.95	4 2 3 5 1	−0.67	−0.32	−0.71	−0.47	−0.51	4.59	3.61

manufacturing cost is 2.81 (ideal cost). Z_1 and Z_2 have different job-position allocations. In Table 1, we also give the t_i values for each job. At Z_1, $(\partial f_5(p_5^{\mathrm{l}})/\partial p_5) = -3.0$ and it is succeeded by a single job, then $t_5 = \frac{-3.0}{2} = -1.5$. Since $\partial f_i(p_i^{\mathrm{u}})/\partial p_i = 0$ for all i at Z_2, we see that $t_i = 0$ for all i.

In Table 2, we give the first seven iterations of the MPJ algorithm for the example problem. In the first iteration we consider the efficient solution Z_1. At Z_1 in Table 2, t_5 is minimum so we select job 5 to perturb (Step 2.1). We increase p_5 by $\Delta = 0.1$ from 0.36 to 0.46 (Step 2.2). Since the SPT order is violated we resequence the jobs and this results pairwise interchanging jobs 3 and 5 (Step 2.3). We update the t_i's for jobs 3 and 5 (Step 2.4). For the new sequence, $t_5 = -1.56$. Next, we report the achieved solution at the end of iteration 1 as given in Table 2 (Step 2.5). Still t_5 is the minimum, we increase p_5 again in iteration 2 and achieve a new schedule. The algorithm continues until it meets the total completion time level of K^{u}.

We use the estimated cost change and total completion time data to make decisions, so using smaller Δ would always give a better approximation. If we use a smaller Δ in our example, the sequence change that occurred in iteration 1 might occur in the next iteration or a different job may be selected in iteration 2 which may imply achieving different schedules. Δ also affects the number of iterations such that smaller Δ implies more iterations which means generating more solutions.

To compare the solution that MPJ achieved at the end of iteration 1, we solved the single objective problem by the NLP solver GAMS/MINOS for $K = 3.89$. The result achieved by MINOS and the result achieved by MPJ are given in detail in Table 3. Total manufacturing cost achieved by MINOS is 4.20. From Table 2, we know that MPJ achieved the total manufacturing cost of 4.18. This is a case where MPJ performed better than MINOS. The reason for this situation can be seen in Table 3. Two solutions are different in terms of job sequence and processing times. This means MINOS stuck to a local optimal solution, but MPJ achieved a better solution at a different job sequence. If the sequences were the same, MINOS would achieve a better solution since it can change all jobs' processing times while MPJ changes the processing time of a single job at a time. Furthermore, we solved the LSOP model for the example for $K = 3.89$ by using the MINLP solver GAMS/BARON and achieved the global optimum which is the same as the solution achieved by MPJ. We solved LSOP model for all the F_2 levels of solutions achieved by MPJ. In Fig. 1, we present the set of efficient solutions (including Z_1 and Z_2) found by the MPJ and corresponding efficient (global optimal) solutions

Table 3
Schedules generated by different methods at iteration 1

Schedule by MPJ					Schedule by MINOS				
Position	Job i	p_i	t_i	$f_i(p_i)$	Position set	Job i	p_i	t_i	$f_i(p_i)$
1	4	0.18	−0.47	0.29	1	4	0.18	−0.47	0.29
2	2	0.20	−1.22	0.36	2	2	0.21	−1.10	0.34
3	3	0.42	−0.71	0.68	2	5	0.41	−1.10	0.8
4	5	0.46	−1.56	0.71	3	3	0.46	−1.10	0.63
5	1	1.65	−1.10	2.14	3	1	1.65	−1.10	2.14

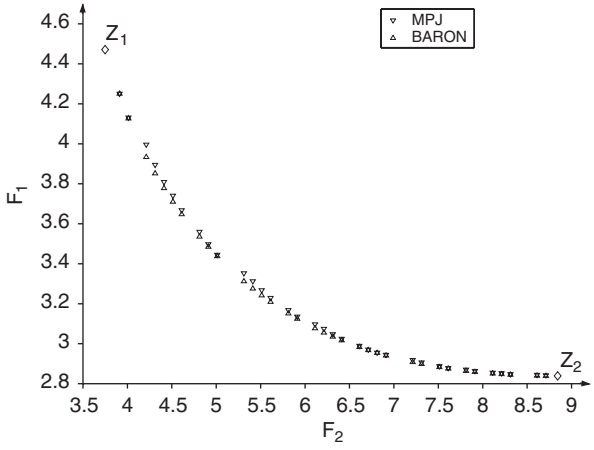

Fig. 1. A set of non-dominated solutions for the numerical example.

achieved by GAMS/BARON for the example problem. For this small example, F_1 levels of solutions achieved by MPJ are very close to global optimum and there are many cases in which they are equal.

In Table 3, we see that solution by MINOS satisfies Lemma 1 and gives an integer local optimal solution to the NLP model as stated in Lemma 2. This example further shows that it is possible to achieve better solutions than MINOS since MINOS cannot guarantee global optimality due to Lemma 3 which states the non-convex nature of the problem. When we check the MINOS solution in Table 3, we see that t_i values are equal for all jobs that have $p_i > p_i^1$, except t_1 for which $p_1 = p_1^1$. This is an example that illustrates the optimality conditions stated in Lemma 4.

6. Computational analysis

In this paper, we presented an efficient formulation for the single objective problem of minimizing total manufacturing cost subject to a total completion time constraint. We proposed two approximation algorithms SBA and MPJ to generate a set of approximate efficient solutions. We coded the SBA algorithm in GAMS 2.5 and implemented it by using the solver MINOS 5.51. GAMS is a well-known commercial mathematical modeling tool with many different solvers attached to it (Brooke et al. [16]). MINOS is the oldest NLP solver available with GAMS and is still the NLP solver that is used the most frequently. MINOS has been developed at the Systems Optimization Laboratory at Stanford University and it is still being improved. We coded the MPJ algorithm in C language and compiled with Gnu C compiler version 3.2. All codes were run on the operating system Mandrake 10.0 with Linux 2.6.3 on a computer with 1294 MB memory and Pentium III 1133 MHz. CPU. In this section, we discuss the results of the computational study.

We considered four experimental factors which are given in Table 4. The first two factors, number of jobs and number of machines, define the problem size. The third factor is the machine type. CNC turning machines may have different technical characteristics. An important one is the maximum applicable cutting power H which affects the p^1 levels.

Table 4
Experimental design factors

Level	Experimental factors			
	N	M	Machine type	Tool cost (C_t)
1	50	3	$C_o = 1$, $H = 5$	U[6, 10]
2	100	6	$C_o = 2$, $H = 10$	U[15, 19]
3	150		$C_o = 4$, $H = 20$	
4	200			

Table 5
Performance measures for different step size levels

Δ	Measure	Mean	Min	Max	Std. dev.
0.01	R	0.000282	−0.002119	0.001213	0.000318
	MINOS CPUs	78.92	1.96	323.35	91.37
	MPJ CPUs	0.08	0.01	0.20	0.05
	MPJ set size	5559.16	1323	12459	3012.14
0.03	R	0.002764	0.000178	0.010918	0.002654
	MINOS CPUs	79.45	2.26	327.91	92.13
	MPJ CPUs	0.03	0.00	0.08	0.02
	MPJ set size	1857.03	443	4159	1005.58

A machine with higher H level allows lower p_i^1's. Usually, machines with higher cutting power are more expensive which means they have higher operating cost level (C_o). We consider three types of machines with different H and C_o levels. The last experimental factor is the tooling cost level C_t, which affects the multiplier m_i in the tooling cost term of the manufacturing cost function. We generated C_t for each tool type as given in Table 4 where $U[a, b]$ is a uniform distribution in interval [a,b]. For each experimental setting, we solved for five replications resulting in 240 randomly generated problems. For each replication we generated cutting specifications (diameter, length, depth of cut and required surface roughness) of jobs randomly. For each job we randomly used one of the tool types out of 10 types of cutting tools with different technical coefficients given in Kayan and Akturk [1]. Furthermore, to test the effect of Δ on the results we tried two levels of Δ, namely 0.01 and 0.03.

For each replication we first apply the MPJ method to the problem and generate a set of efficient solutions. Out of this set, we select 50 solutions, other than Z_1 and Z_2. To be able to compare solution quality of MPJ and SBA, we applied the SBA method for the F_2 levels of the selected solutions. In order to test the algorithms on different regions of the efficient frontier, we selected those 50 points such that each successive point pair has equal or almost equal separation. We considered the performance measure $R = (F_1^{MPJ} - F_1^{SBA})/F_1^{SBA}$ which is the relative difference between F_1 level achieved by the MPJ algorithm and F_1 level achieved by the SBA method for the same K value.

In Table 5, we give the results for R, CPU time required by SBA to solve 50 instances and CPU time required by MPJ to generate a set of efficient solutions for both levels of Δ. We also include the size of the solution set generated by MPJ. We would like to emphasize the mean R level, which is less than 0.0003 and indicates that MPJ achieves almost equal cost performance as MINOS. Even for $\Delta = 0.03$, the mean R is around 0.3%. The minimum R is negative for $\Delta = 0.01$ which shows that MPJ can achieve better results than MINOS due to Lemma 3. The maximum level and standard deviation show that R values do not deviate much from the mean, and even the worst R is 0.1%. The next important criterion to compare two approaches is the computational requirements. Despite employing MINOS for just 50 points out of thousands generated by MPJ, in Table 5, we see that CPU time requirement of MINOS is incomparably high with respect to MPJ. On the average MPJ produces 5559 solutions, so MPJ can generate many alternative solutions in a very short computation time. When we increase Δ, we see that R is increased but CPU time required by MPJ is decreased as expected. However, even for $\Delta = 0.01$, the required CPU time is 0.08 s on the average, so we can decrease Δ even further and it would still not require much CPU time and would give a better approximation in terms of solution quality and number of alternative solutions.

Table 6
Average performance measures for different N and M levels when $\Delta = 0.01$

N	M	R	MINOS CPU	MPJ CPU
50	3	0.000216	6.16	0.03
	6	0.000308	3.62	0.03
	Total	0.000262	4.89	0.03
100	3	0.000298	34.44	0.06
	6	0.000285	16.76	0.05
	Total	0.000291	25.60	0.05
150	3	0.000294	108.88	0.09
	6	0.000266	48.47	0.09
	Total	0.000280	78.67	0.09
200	3	0.000298	292.62	0.14
	6	0.000291	120.43	0.14
	Total	0.000294	206.53	0.14

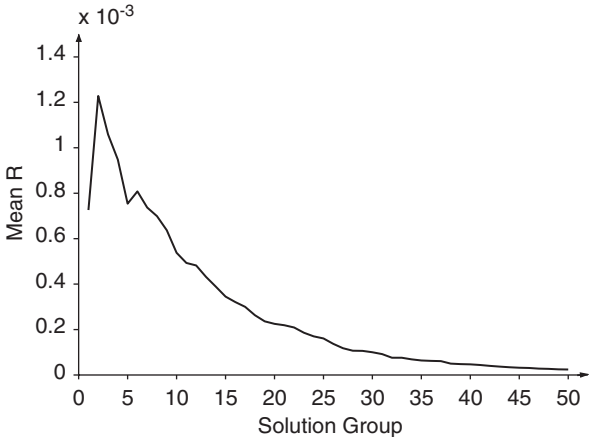

Fig. 2. Behavior of R on different regions of the efficient frontier.

Table 6 gives R and CPU time measures for different levels of number of jobs and number of machines. As N increases, we do not observe a significant decrease or increase on R. The CPU time required by MINOS is strongly affected by N, which increases rapidly from 4.89 to 206.53 (almost 40 times), when N is increased from 50 to 200. On the other hand, the CPU seconds required by MPJ increased slightly from 0.03 to 0.14. This shows that as N increases, CPU time increase rate for MINOS is higher than CPU time increase rate for MPJ. Although we tried 200 jobs at most, we can solve larger instances by using MPJ within acceptable CPU time levels. When we check the change on R with respect to M, we see that it increases for 50 jobs case and decreases for the others. Also, the CPU time required by MPJ is not affected by the level of M. However, CPU time required by MINOS is significantly affected by M. As M is increased for a given N, CPU time required by MINOS is reduced. This is because as we increase the number of machines, the number of equivalent position sets decrease and this makes the problem easier for MINOS since it copes with less alternatives of job-position set allocations.

Another observation we obtain from the computational results is related to the relative performance of MPJ on different parts of the efficient frontier. In our experiments, we evaluated the R values for 50 efficient solutions except Z_1 and Z_2 for each replication. Let us denote the first found efficient solutions right after Z_1 as Group 1, and so on, then, we have 50 solution groups for each replication. When we compare the average R value for each group, we observe that the average R is decreasing as we go from the first group to the last group as shown in Fig. 2. For example, R is higher

Table 7
Comparison with the global optimal solutions

N	Measure	Min	Max	Mean	Std. dev.
7	R(MPJ–BARON)	0.000007	0.009724	0.001373	0.002429
	R(MINOS–BARON)	0.0	0.004609	0.000610	0.001310
	BARON CPUs	11.86	29.24	19.48	4.40
10	R(MPJ–BARON)	0.000025	0.004111	0.000693	0.000853
	R(MINOS–BARON)	0.0	0.009073	0.001682	0.002630
	BARON CPUs	875.87	2377.93	1412.34	404.34

for the total completion time levels closer to Z_1. As we get closer to Z_2, R gets smaller and smaller. This relationship is also shown to be statistically significant by the ANOVA results. We think that this is related to the behavior of the manufacturing cost functions. As we get closer to Z_2, we work on flatter parts of the cost functions where the decision making based on slope information as in the MPJ algorithm is more reliable.

We next analyzed the absolute performance of MPJ and SBA against the global optimal solutions achieved by the MINLP solver BARON 7.2.3 by solving the LSOP models given in Section 3. Branch And Reduce Optimization Navigator (BARON) is a well-known MINLP solver and it is hooked up to GAMS modeling system. BARON combines constraint propagation, interval analysis, and duality for efficient range reduction with rigorous relaxations constructed by enlarging the feasible region and/or underestimating the objective function. Due to CPU time requirement we could only solve 7 jobs and 10 jobs cases for three machines for the same experimental settings. For each replication, we selected five efficient points out of the set generated by the MPJ algorithm and solved the LSOP and the relaxed SOP models by the solvers BARON and MINOS, respectively. By this way, we tested these three approaches on 300 points. Table 7 shows the relative difference values for the MPJ method versus SBA using BARON, and SBA using MINOS versus SBA using BARON. Results show that both MPJ and MINOS find solutions very close to global optimum. There are cases where MINOS achieves the global optimal solution. We also see from Table 7 that when we increase the number of jobs from 7 to 10, the CPU time required by BARON increases by a factor of 140.

Up to this point we have discussed the pointwise solution quality and computational requirements for both methods. At this point we analyze the results with a multi-objective optimization point of view to compare two approaches. We want to compare the performance of two approximate efficient solution sets generated by different methods. We used the performance measure proposed by Hansen and Jaszkiewicz [17], which consists of measuring the probability $P(A, B)$ that an algorithm A, gives a better solution than some other algorithm B. It is calculated as $P(A, B) = \int_{u\in[0,1]} C(A(u), B(u))\,\mathrm{d}u$, where

$$C(A(u), B(u)) = \begin{cases} 1 & f(A(u)) < f(B(u)), \\ \frac{1}{2} & f(A(u)) = f(B(u)), \\ 0 & f(A(u)) > f(B(u)), \end{cases}$$

and $f(A(u)) = \min_{x\in A}\{\max(uF_1'(x), (1-u)F_2'(x))\}$, where $F_1'(x) = (F_1(x) - F_1(Z_2))/(F_1(Z_1) - F_1(Z_2))$ which is a normalization of F_1. Here u is the weight of the normalized objective function F_1 for the decision maker. The method basically tries a number of u values between 0 and 1 and estimates the decision maker's probability to choose a solution generated by method A.

Table 8 includes $P(MPJ, MINOS)$ results for different values of Δ. When $\Delta = 0.01$ we see that probability of decision maker to prefer a solution generated by MPJ is 98% on the average. This is due to the fact that the MPJ method generates significantly higher number of efficient solutions than the GAMS/MINOS solver. When Δ is increased this falls to 82% since pointwise solution quality and number of generated points decreases. The results show that MPJ is a powerful method from the multi-objective optimization point of view.

In this section, we gave the computational results of proposed approaches and discussed the results from several points of view. We saw that by using the optimality properties for the problem we could develop an algorithm which can compete with a commercial software in terms of solution quality but gives better results in terms of computational

Table 8
Comparison of the approximation algorithms

Δ	P (*MPJ*, *MINOS*)		
	Mean	Min	Max
0.01	0.975	0.870	0.999
0.03	0.816	0.430	0.970

requirements. This highlights the importance of extracting and using the problem specific information when dealing with a problem.

7. Conclusions

In this study, we proposed alternative approaches to provide efficient solutions for the problem of minimizing the objectives of total manufacturing cost and total completion time on identical parallel machines. We specifically considered the nonlinear manufacturing cost functions on CNC turning machines but the results are applicable to any case with convex processing cost functions. Our results are important to any decision maker who deals with controllable processing times. The results are easily applicable especially on CNC turning machines where the processing time of an operation can be easily set by adding a single line of code to its CNC program. We first considered the single objective problem of minimizing total manufacturing cost subject to a total completion time constraint. For this MINLP problem, we provided an efficient formulation whose relaxed form can be solved to integer by NLP solvers. We showed that NLP solvers can just guarantee to find local optimal solutions for the problem. We also proved optimality conditions that must hold between processing times of jobs at optimality. By the help of these properties, we developed a heuristic algorithm which generates a set of approximate efficient solutions for the problem. Computational results proved that the heuristic algorithm performs as good as well-known commercial GAMS/MINOS NLP solver with significantly less computational effort.

Acknowledgements

The authors would like to acknowledge an anonymous referee for his/her helpful guidance on improving the presentation and the contents of the paper. This work was supported in part by the Scientific and Technical Research Council of Turkey under Grant #2211.

References

[1] Kayan RK, Akturk MS. A new bounding mechanism for the CNC machine scheduling problems with controllable processing times. European Journal of Operational Research 2005;167:624–43.
[2] Vickson RG. Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine. Operations Research 1980;28:1155–67.
[3] Shabtay D, Kaspi M. Minimizing the total weighted flow time in a single machine with controllable processing times. Computers and Operations Research 2004;31:2279–89.
[4] Hoogeveen H. Multicriteria scheduling. European Journal of Operational Research 2005;167:592–623.
[5] Gupta JND, Ruiz-Torres AJ. Generating efficient schedules for identical parallel machines involving flow-time and tardy jobs. European Journal of Operational Research 2005;167:679–95.
[6] Gupta JND, Ho JC. Minimizing makespan subject to minimum flow time on two identical parallel machines. Computers and Operations Research 2001;28:705–17.
[7] Cao D, Chen M, Wan G. Parallel machine selection and job scheduling to minimize machine cost and job tardiness. Computers and Operations Research 2005;32:1995–2012.
[8] Alagoz O, Azizoglu M. Rescheduling of identical parallel machines under machine eligibility constraint. European Journal of Operational Research 2003;149:523–32.
[9] Alidaee B, Ahmadian A. Two parallel machine sequencing problems involving controllable job processing times. European Journal of Operational Research 1993;70:335–41.

[10] Cheng TCE, Chen ZL, Lee C-L. Parallel-machine scheduling with controllable processing times. IIE Transactions 1996;28:177–80.
[11] Jansen K, Mastrolilli M. Approximation schemes for parallel machine scheduling problems with controllable processing times. Computers and Operations Research 2004;31:1565–81.
[12] Shabtay D, Kaspi M. Parallel machine scheduling with a convex resource consumption function. European Journal of Operational Research 2006; to appear.
[13] Pinedo M. Scheduling: theory, algorithms and systems. New Jersey: Prentice-Hall; 2002.
[14] Conway RW, Maxwell WL, Miller LW. Theory of scheduling. Massachusetts: Addison-Wesley; 1967.
[15] Murty KG, Kabadi SN. Some NP-complete problems in quadratic and nonlinear programming. Mathematical Programming 1987;39:117–29.
[16] Brooke A, Kendrick D, Meeraus A, Raman R. GAMS: a user's guide. GAMS Development Corporation; 2004.
[17] Hansen MP, Jaszkiewicz A. Evaluating the quality of approximations to the non-dominated set. IMM Technical Report, Technical University of Denmark, IMM-REP-1998-7; 1998.