Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 34 (2007) 2130–2146

computers & operations research

www.elsevier.com/locate/cor

Scheduling with tool changes to minimize total completion time under controllable machining conditions

M. Selim Akturk[a,*], Jay B. Ghosh[b], Rabia K. Kayan[a]

[a]*Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey*
[b]*Department of Information and Operations Management, University of Southern California, USA*

Available online 29 September 2005

Abstract

Scheduling under controllable machining conditions has been studied for some time. Scheduling with tool changes, particularly due to tool wear, has just begun to receive attention. Though machining conditions impact tool wear and induce tool change, the two issues have not been considered together. We address for the first time the problem of scheduling a computer numerically controlled (CNC) machine subject to tool changes and controllable machining conditions; our objective is to minimize the total completion time of a given set of jobs. We establish an important result that helps us identify feasible settings of machining parameters such as feed rate and cutting speed. However, the problem at hand remains intractable, even when a single setting is used. In the general case, we are able to solve the problem exactly for up to 30 jobs using a mixed integer linear programming formulation. For larger problems, we turn to approximate solution via heuristics. We examine a number of different schemes. The best of these schemes are used in a problem space genetic algorithm; this produces quality solutions in a time-efficient manner, as is evidenced from an extensive computational study conducted by us.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Scheduling; Heuristics; CNC machines

1. Introduction

Flexibility is a key to the success of a modern manufacturing system, which is required to deliver diverse products with high quality and short lead times. This flexibility is often achieved through the use of a computer numerically controlled (CNC) machine and is limited only by how effectively resources such as pallets, fixtures and tools are allocated to this machine. Effective tool management, which accounts for a high percentage of the operating costs in an automated manufacturing environment, is of particular importance. Thus, a scheduling model that appropriately incorporates tool management issues can significantly improve the productivity of a flexible manufacturing system.

In this study, we address a single machine scheduling problem under controllable machining conditions and subject to tool changes. Recently, Akturk et al. [1] have focused on a scheduling problem with tool changes caused by wear, but they have assumed that the processing times of the jobs and the tool life are constants. However, by changing the machining conditions, it is possible to control these two values and obtain a better schedule.

* 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.2005.08.014

The cutting speed and the feed rate are the typical machining parameters that constitute the machine settings. An increase in either one of them decreases the processing time of a job but simultaneously increases the tool usage (which in turn decreases the life of a cutting tool and causes an early tool change). Thus, from a scheduling point of view, there is a tradeoff here. Increased cutting speed or feed rate, while expediting the jobs in process, delays the subsequent jobs due to the more frequent tool changes induced by the increased tool usage. Hence, the machining conditions, cutting speed or feed rate, need to be adjusted properly for each job in order to optimize a scheduling objective that involves the job completion times. Considering the processing times and the tool usage rates of the jobs as consequences of the controllable machining conditions rather than constants, a better integration of tool management and scheduling is achieved.

The tool change time is generally long relative to processing times and the tool life is short relative to the planning horizon. Therefore, it makes sense to schedule the jobs with respect to a completion time-related scheduling objective. We focus here on minimizing the total job completion time, as has been done by Akturk et al. [1].

The literatures on tool management and scheduling are unfortunately disjoint. Machining conditions optimization and tool replacement strategies are studied under tool management, whereas issues related to controllable processing times and machine availability are covered under scheduling. To the best of our knowledge, the interaction between these two streams has not been addressed by researchers thus far.

The optimization of the machining conditions for a single operation is a well-known problem, where the decision variables are usually the cutting speed and the feed rate. Petropoulos [2] has used geometric programming, whereas Khan et al. [3] have proposed local search algorithms for machining conditions optimization. These models consider only the contribution of machining time and tooling cost to the total operating cost and usually ignore the contribution of the nonmachining time components. Tool replacement, which also depends on machining conditions, is a typical example of a nonmachining activity. Note further that the machining conditions optimization models do not address scheduling at all.

A tool replacement strategy specifies a tool change schedule based on the economic service levels of the tools. Tang and Denardo [4] have studied the single machine case with given tool requirements, where the tool changes are required due to part mix. Their objective is to minimize the number of tool switches. Hertz et al. [5] have also considered minimizing tool switches on a flexible machine when scheduling different part types. They have proposed several heuristics for the tool loading problem. Despite the studies on tool loading, tools are usually changed more often because of tool wear than part mix, as reported by Gray et al. [6]. Note also that scheduling is not a concern here either.

Processing time control and its impact on sequencing decisions and operational performance have received limited attention in the scheduling literature. Trick [7] has focused on assigning single-operation jobs to identical machines while simultaneously controlling the processing speed of each machine. Zdrzalka [8] has dealt with the problem of scheduling jobs on a single machine in which each job has a release date, a delivery time and a controllable processing time. Karabati and Kouvelis [9] have solved the simultaneous scheduling and optimal processing-times selection problem in a flow line operated under a cyclic scheduling policy. Jansen and Mastrolilli [10] have proposed approximation schemes for parallel machine scheduling problems with controllable processing times. Finally, Shabtay and Kaspi [11] have studied a single machine weighted flow time problem subject to resource consumption and limited resource availability. Note that this line of research considers neither tool usage nor tool replacement.

Machine availability models in scheduling come closest to considering the tool replacement issue. Adiri et al. [12] have considered a flow time scheduling problem where a machine faces breakdowns at stochastic time epochs and the repair time is stochastic. The processing times are assumed constant and the jobs are assumed nonresumable. Lee and Liman [13] have studied the deterministic version of this problem considering only a single scheduled maintenance, whereas Liao and Chen [14] have considered the case where there are several maintenance intervals. Schmidt [15] has reviewed results on scheduling under availability constraints and analyzed the complexity of single and multiple machine problems under completion time and due date-related criteria. Note that this area of research does not address processing time control.

The purpose of this paper is to propose a model linking machining conditions control, tool replacement and job scheduling on a CNC machine. As pointed out earlier, this has not been done before. We present solution strategies to a scheduling problem where the job processing times and the tool usage are controlled by setting the cutting speed and/or the feed rate. There is a single production unit (the CNC machine), and the cutting tools are subject to wear and require replacement. Our objective, once again, is to minimize the total job completion time.

We first provide the notation used by us throughout the paper. We then discuss the issue of machining conditions control, proving an important result along the way, and show how one can identify a set of feasible settings for each job. Next, we give a mixed integer program (MIP), which assigns an optimal machine setting to each job and finds the optimal schedule of the jobs as well. We go on to propose single-pass heuristic algorithms and test their performance on a set of randomly generated problems. We also propose a problem space genetic algorithm (PSGA) to improve the solution quality. In PSGA, a base heuristic (which is called many times within the algorithm) needs to be defined. We use as base heuristics some of our single-pass heuristics that result in high performance at low CPU time.

2. Notation

The notation used throughout the paper is as follows:

α, β, γ	speed, feed, depth of cut exponents for the tool
C_{m}, b, c, e	specific coefficient and exponents of the machine power constraint
C_{s}, g, h, l	specific coefficient and exponents of the surface roughness constraint
C	Taylor's tool life expression parameter
C_{o}	operating cost of the CNC machine ($/min)
C_{t}	cost of the tool ($)
d_i	depth of cut for job i (in)
D_i	diameter of the generated surface for job i (in)
L_i	length of the generated surface for job i (in)
H	maximum available machine power (hp)
S_i	maximum allowable surface roughness for job i ((μin)
v_{ij}	cutting speed for setting j of job i (fpm)
f_{ij}	feed rate for setting j of job i (ipr)
U_{ij}	usage rate of job i using setting j
P_{ij}	machining time of job i using setting j (min)
T_{c}	tool change time (min)
N	number of the jobs
S	number of different settings generated for each job

3. Problem definition

We are given N jobs with a specified depth of cut, length and diameter of the generated surface along with maximum allowable surface roughness attributes. The problem is scheduling these jobs on a CNC machine in order to minimize the total completion time. When the tool life is over, the tool has to be changed. Since there is a constant tool change time, sometimes significant, and it affects the completion time, it is important to consider it in the schedule. The machining conditions of the CNC machine can be changed, and for each job it can be adjusted to different cutting speed and feed rate pair. However, there are some constraints for these settings. The speed and feed rate have to satisfy the machine power, surface finish and tool life constraints. After detecting the feasible region of speed and feed rate, we have to make two decisions, a feasible setting for each job, and the schedule of the jobs. For each job and tool pair, the constraints stated in Akturk and Avci [16] are

$$C'_{\mathrm{t}} v_{ij}^{(\alpha-1)} f_{ij}^{(\beta-1)} \leqslant 1 \quad \text{(Tool life constraint)},$$

$$C'_{\mathrm{m}} v_{ij}^{b} f_{ij}^{c} \leqslant 1 \quad \text{(Machine power constraint)},$$

$$C'_{\mathrm{s}} v_{ij}^{g} f_{ij}^{h} \leqslant 1 \quad \text{(Surface roughness constraint)},$$

$$v_{ij},\ f_{ij} > 0,$$

where $C'_{\mathrm{t}} = \pi D_i L_i d_i^{\gamma}/12C,\ C'_{\mathrm{m}} = C_{\mathrm{m}} d_i^{e}/H,\ C'_{\mathrm{s}} = C_{\mathrm{s}} d_i^{l}/S_i.$

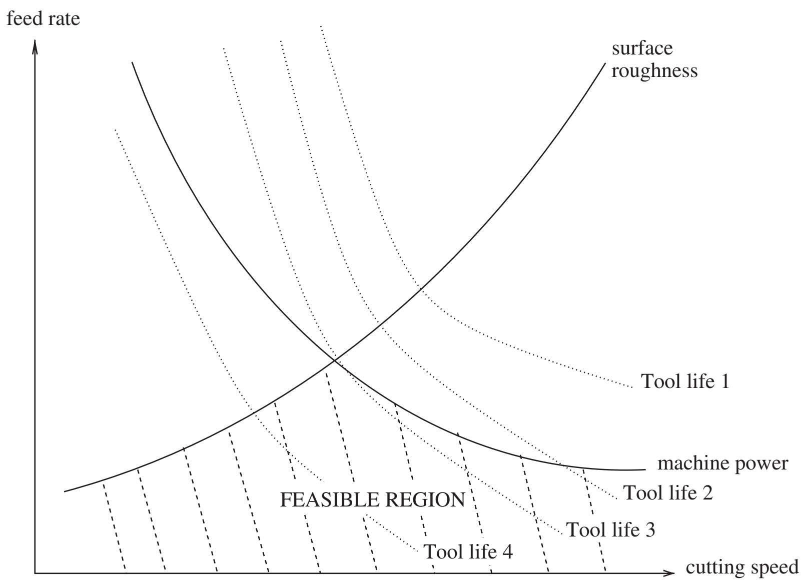

Fig. 1. Feasible region of machine settings.

We first relax the tool life constraint, and solve the relaxed problem. If the optimal solution of the relaxed problem satisfies the tool life constraint, then it is optimal for the overall problem. If it does not satisfy the constraint, then a new optimal solution should be found. The relationship between machine power and surface roughness constraints can be seen in Fig. 1. The tool life constraint may be in one of four situations: redundant, crossing machine power constraint, crossing intersection point of these two constraints, or crossing surface roughness constraint.

Akturk and Avci [16] prove that at least one of the surface roughness and machine power constraints is binding at optimality. In this study, we also prove a useful theorem.

Theorem. *The surface roughness constraint must be tight at optimality.*

Proof. The formulas of processing time and tool usage rate of a job are

$$P = \frac{\pi DL}{12} v^{-1} f^{-1} \quad \text{and} \quad U = \frac{\pi DLd^{\gamma}}{12Cv^{(1-\alpha)} f^{(1-\beta)}}.$$

When the machining power constraint is tight, the setting pair at the intersection gives the minimum processing time under the condition that $b > c > 0$ or $b < c < 0$. b and c cannot be negative since increasing speed or feed rate always require more machine power. Moreover, machine power is more sensitive to the changes in cutting speed than feed rate yielding $b > c$. In the other case where surface roughness constraint is binding, again the setting pair at the intersection gives the minimum processing time under the condition that $h > 0, h > g$ or $h < 0, h < g$. g is always negative since cutting speed and surface roughness are inversely related. Additionally, increasing the feed rate increases the surface roughness, therefore h is a nonnegative coefficient. Consequently, the above inequality conditions are always satisfied. As a result, in both cases, minimum processing time is achieved by the settings at the intersection point. In order to get the minimum usage rate, both v and f values have to be at their lowest permissible values. As a result, the upper edge of the feasible surface roughness line gives minimum processing time and the lower edge gives minimum usage rate for the given job. Therefore, for any time- and cost-related objectives, the optimal pair is on this line. In other words, surface roughness must be binding at optimality. $\square$

Since the surface roughness constraint is a nonlinear function, we can generate discrete points on this function such that each point corresponds to an alternative machining conditions setting for each job, i.e. a feasible (v_{ij}, f_{ij}) pair. Instead of specifying equally spaced discrete points, we can generate these settings for each job separately with respect to its machining parameters of d_i, D_i, L_i, S_i, etc. Let (v^a, f^a) be the optimal solution that minimizes the machining time subject to the machining constraints stated above. Furthermore, we can assume that the contribution of each job to the total completion time objective is proportional to its usage rate, U_{ij}, and its machining time, P_{ij}. We can define the following surrogate measure to approximate this term as $P_{ij} + T_c U_{ij}$. Let (v^b, f^b) be the optimal solution that minimizes this surrogate measure over the same feasible region. Although we do not consider the cost-related measures in this study, the optimal (v^c, f^c) setting that minimizes the manufacturing cost $C_o P_{ij} + C_o T_c U_{ij} + C_t U_{ij}$ can be found as discussed in Akturk and Avci [16]. Finally, let v_l be the minimum allowable cutting speed value, and $r = \lfloor (v^a - v_l)/(S-1) \rfloor$. We use the following procedure to generate S alternative cutting speed values for each job. According to our theorem, the corresponding feed rate can be calculated for a given cutting speed.

Step 1: Calculate the number of settings generated between v^a and v^b, $s_1 = 1 + \lfloor (v^a - v^b)/r \rfloor$.
Step 2: Calculate the interval of speed used between points v^a and v^b, $r_1 = \lfloor (v^a - v^b)/s_1 \rfloor$.
Step 3: Calculate cutting speed of job j, $v_j = v^a - (j * r_1)$ for $j = 0, \dots, (s_1 - 1)$.
Step 4: Calculate the number of settings generated between v^b and v^c, $s_2 = 1 + \lfloor (v^b - v^c)/r \rfloor$.
Step 5: Calculate the interval of speed used between points v^b and v^c, $r_2 = \lfloor (v^b - v^c)/s_2 \rfloor$.
Step 6: Calculate cutting speed of job j, $v_j = v^b - [(j - s_1) * r_2]$ for $j = s_1, \dots, (s_1 + s_2 - 1)$.
Step 7: Calculate cutting speed of job j, $v_j = v^c - [(j - s_1 - s_2) * r]$ for $j = (s_1 + s_2), \dots, (S - 1)$.

4. MIP formulation

The following mathematical model can be used to assign a feasible setting to each job and schedule these jobs in order to minimize the total completion time.

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{S}\sum_{k=1}^{N}(N-k+1)P_{ij}X_{ijk} + T_c & \displaystyle\sum_{k=1}^{N-1}(N-k)R_k, \\
\text{s.t.} & \displaystyle\sum_{j=1}^{S}\sum_{i=1}^{N} X_{ijk} = 1, & k = 1, \dots, N, \\
& \displaystyle\sum_{j=1}^{S}\sum_{k=1}^{N} X_{ijk} = 1, & i = 1, \dots, N, \\
& \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{S} U_{ij}X_{ijk} + Y_{k-1} - Y_k \geqslant 0, & k = 1, \dots, N-1, \\
& Y_k - \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{S} U_{ij}X_{ijk} - Y_{k-1} + R_k \geqslant 0, & k = 1, \dots, N-1, \\
& \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{S} U_{ij}X_{ijk+1} + Y_k \leqslant 1, & k = 1, \dots, N-1, \\
& X_{ijk} \in \{0, 1\}, & i = 1, \dots, N, \quad j = 1, \dots, S, \quad k = 1, \dots, N, \\
& R_k \in \{0, 1\} \text{ and } Y_k \geqslant 0, & k = 1, \dots, N-1,
\end{array}
$$

where X_{ijk}= 1 if job i under setting j is scheduled at position k, and 0 otherwise, R_k= 1 if tool is replaced after position k, and 0 otherwise.

The objective function is composed of two parts. The first part gives the aggregated completion times of the jobs without the tool change times, and the second part is the contribution of tool change times to the job completion times. The first constraint set guarantees that only one position is assigned to each job with a single setting. Similarly, the second set guarantees that only one job can assigned to each position. In the third and fourth constraint sets, we use the variable Y_k to keep track of the remaining tool life in each tool k, i.e. $Y_0 = 0$ by definition. The fifth constraint prevents the total usage of the tool exceeding 1.

5. Proposed heuristic algorithms

Akturk et al. [1] proved the NP-hardness of their problem which is a sub-problem of ours. Therefore, no algorithm is likely to be proposed for solving our problem optimally in polynomial time. Hence, it is justifiable to try heuristic methods to solve our problem.

In order to ease the determination of job sequence, the following structural properties showed by Akturk et al. [1] will be used in the heuristic algorithms.

Let p_i be the machining time of job i, t_m be the aggregate machining times of all jobs on block m, and n_m be the number of jobs on block m.

- If jobs i and j are within the same block, then i precedes j if $p_i < p_j$.
- Furthermore, for blocks m and r: m precedes r if $(t_m + T_c)/n_m \leqslant (t_r + T_c)/n_r$.

The first property means that the jobs assigned to the same tool are sequenced in the shortest processing time (SPT) order at optimality. The second property shows that, in an optimal schedule, blocks are sequenced according to their average job times in ascending order.

We basically worked on dispatching rules and then built three-stage single-pass algorithms using these rules. In the first stage, the initial setting conditions of jobs are determined. They are set either to the setting pair giving the minimum machining time, setting 1, or different setting alternatives according to some job specific parameters. Then a dispatching rule, either developed by us or existing in the literature, is used in the second stage to find an initial schedule. Finally, in the third stage, the initial schedule is improved by distributing either tools or jobs one at a time. The illustration of these stages is given in Fig. 2.

5.1. Stage 1: Setting assignment

The jobs with their D, L, d, S attributes are given as an input to the first stage, and after calculating v, f values for each job, the usage rate and machining time are calculated and given as the output. In this stage, we have two alternatives:

- The setting pair, (v_1, f_1) giving the minimum machining time which is at the intersection point (see Fig. 1) is called as the first setting. The values of v_1 and f_1 differ from job to job due to different job attributes, but they are always at the intersection point as proved before. Assigning each job to their first setting pair initially will result in jobs with high usage rates but the lowest feasible machining times. This alternative leads to better results when the first part of the objective function is dominant. However, it may still work in the other case since we change the settings of jobs in the last stage to fit more jobs to the blocks.

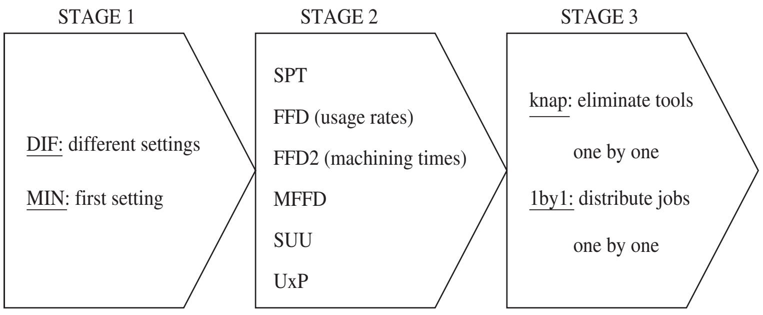

Fig. 2. Three stages of the proposed heuristics.

- Each job uses different rate of tool life per unit time. The jobs with a high depth of cut values will have high usage rates per unit time. Usage rate per unit time is calculated as

$$\frac{U}{P} = \left(\frac{\pi D L d^{\gamma}}{12C}\, v^{(\alpha-1)} f^{(\beta-1)}\right) \Big/ \left(\frac{\pi D L}{12}\, v^{-1} f^{-1}\right) = \frac{d^{\gamma}}{C}\, v^{\alpha} f^{\beta}.$$

  The unit usage rate of the job decreases when sliding down from the first setting to the lower settings. The reason is that the usage rate is lower and machining time is higher in lower settings. First, the jobs are sorted in ascending order of their unit usage rate and they are partitioned into four equal groups. The jobs in the first group are assigned to their first settings. The ones in the second group to their second settings, and so on. Consequently, we reduce the number of alternative settings, which in turn will decrease the computational requirements and guide the search process in stage 3 in areas most likely to contain good solutions. Of course, these settings can be changed in the later stages to improve the objective function.

5.2. Stage 2: Dispatching rule

In this stage, an initial schedule of the jobs with their usage rate and machining time attributes passed from the first stage is found by using some dispatching rules. The settings of the jobs will remain the same, only the distribution of jobs among the blocks will be dealt with.

- SPT: This rule orders the jobs in increasing order of their processing times. Whenever a machine is freed, the job with the lowest processing time is scheduled next. It gives an optimal sequence for the total completion time problem, $1//\sum C_j$. However, when tool change times are taken into account, it may not perform as well since it minimizes the first part of the objective function only.
- FFD: When tool change times become more significant according to the machining times, the second part of the objective function dominates and the tool change problem gets closer to the bin packing problem. The first fit decreasing (FFD) rule first orders the jobs in decreasing order of their machining times (in longest processing time order). Then the jobs are assigned to tools in the order of first fit decreasing.
- FFD2: In this rule, we sort the jobs according to their usage rates initially. By this way, the jobs with higher usage rates are placed first. This algorithm tries to use the minimum number of tools, maintaining a balance in the tool usages. When T_{c} value is very high and thus the second part of the objective is more dominant, it is expected that FFD2 would result in good solutions since the second part represents the bin-packing aspect of the problem.
- MFFD: SPT gives better results when the first part of the objective is dominant and the problem resembles to a classical scheduling problem. FFD2 (with usage rates) gives better results when the second part is dominant and the problem is similar to bin-packing problem. However our problem is neither a simple scheduling problem nor a bin-packing one. Therefore using a hybrid approach of these rules should be worthwhile. The first few jobs are placed on tools via SPT and the remaining ones are placed via FFD2. SPT maximizes the number of jobs using the first few tools. It can be beneficial to fill in these tools with the shortest jobs, so that the number of jobs being affected by the tool changes will be minimum. In this algorithm, we first order the jobs in SPT order until 30% of the jobs are placed. Then, the remaining jobs are placed via FFD2.
- SUU: Unit usage of a job is defined in the first stage. Placing the jobs having less usage rates per unit time to earlier tools will slow the wearing process of the tool. Therefore, the jobs are sorted in SUU order first and placed to tools following this order.
- $U \times P$: The jobs with low usage rates and low machining times are preferred to be processed at the beginning. This will cause using less number of tools for more jobs at the beginning of the schedule which will decrease the total completion time. Therefore, minimum $U \times P$ is used as a dispatching rule to sort the jobs and the jobs are placed on tools in this sequence.

In this stage, six different alternatives are presented in order to create an initial schedule. The settings of the jobs assigned in stage 1 remain fixed, only the sequence of the jobs is determined. After placing all the jobs to the tools, the tool blocks and the jobs in these blocks are rearranged according to the two structural properties of optimality mentioned before. In the next stage, we seek to improve this initial schedule by changing both the sequence and the settings of the jobs.

5.3. Stage 3: Improvements

The rules stated above are static; the order of the jobs are determined once at the beginning via a dispatching rule. However, we can improve the objective function by changing the machining conditions of each job as stated earlier. In this stage we propose two alternatives to be applied on the initial sequence found by stage 2. In both of them, the number of the tools to be used is tried to be reduced. Both algorithms stop when no improvements are observed any further.

- KNAP: In this algorithm, the tool blocks are tried to be eliminated from the initial schedule one by one in order to get a better solution. First, we select the tool that will be eliminated then the jobs on this tool are distributed to other tools. Clearly, these tools have no space, i.e. remaining tool life, for these new coming jobs. So, we have to change the settings of the jobs to lower values in order to fit them. Hence, we should find both the jobs whose settings will be changed and their new settings. In order to solve these problems, a mathematical formulation, which is similar to a knapsack formulation, is proposed to minimize the total completion time for each tool separately. For each tool k, change the settings of its jobs solving the knapsack formulation below in order to fit the distributed jobs to the tool.

$$\begin{aligned} \min \quad & \sum_{i=1}^{N_k} \sum_{j=1}^{S} (N_k - i + 1) P_{ij} Z_{ij}, & \\ \text{s.t.} \quad & \sum_{j=1}^{S} Z_{ij} = 1, & i = 1, \ldots, N_k, \\ & \sum_{i=1}^{N_k} \sum_{j=1}^{S} U_{ij} Z_{ij} \leqslant 1, & \\ & Z_{ij} \in \{0, 1\}, & i = 1, \ldots, N_k, \quad j = 1, \ldots, S, \end{aligned}$$

  where N_k = Number of jobs in tool k including the inserted jobs, $Z_{ij} = 1$ if the original setting assigned to job i is changed to the jth setting, and 0 otherwise. The objective is to minimize the total completion time within the tool. The first constraint set guarantees that for each job, only one setting is selected among S alternatives. The second constraint set prevents the total usage of tool k exceeding 1.
- 1by1: In this alternative, like in the first one, a tool is selected to be eliminated. But, instead of assigning all of the jobs to the other tool blocks at the same time, we distribute the jobs one at a time. Each time a job is selected from the tool to be eliminated, and assigned to another tool block by changing the setting of only one job (including the newly inserted job) on this tool. The computation time spent for this alternative is relatively small when compared to the first alternative, since we are not solving any linear programming formulations. The sensitivity of machining time of the job to the changes in usage rate gives the response of that job. First, we can define P in terms of U as follows:

$$P = \left(\left(\frac{\pi D_i L_i}{12} \right)^{(h\alpha - g\beta)} \left(\frac{C_s d_i^l}{S_i} \right)^{\alpha - \beta} \left(\frac{UC}{d_i^\gamma} \right)^{g-h} \right)^{(1/(h(\alpha-1) - g(\beta-1)))}.$$

  This also gives the increase in machining time due to a unit decrease in usage rate. Since some of the coefficients are constant for every job, these can be eliminated and the response of job i is calculated as follows:

$$\text{Response}_i = (D_i L_i)^{(h\alpha - g\beta)} S_i^{\beta-\alpha} d_i^{l(\alpha-\beta) - \gamma(g-h)}.$$

  We select the job that has the minimum response value. The tool usage rate of the selected job is decreased (or equivalently its processing time is increased) as much as necessary to accommodate the newly inserted job. This algorithm focuses on distributing the jobs of the selected tool one by one and evaluating whether a better solution can be found in the intermediate steps. This seems reasonable although it may be thought that we cannot save from the tool change unless we remove all the jobs of the tool block. However, moving the job earlier in the sequence to another tool can decrease the completion time of the job. At a point before distributing all the jobs of the currently selected tool, the tools may get full and changing the setting of the jobs and inserting one more job will not improve the solution any more. In this case the algorithm has the flexibility to stop. We cannot find such solutions in the first alternative.

We explained the three stages of the single-pass heuristics. The representation of a heuristic procedure composed of these stages is done with a descriptor as stage 2 (stage 1, stage 3). For instance SUU(min,knap) means, assign the settings giving the minimum machining times to each job, use the "shortest unit usage" heuristic to find an initial schedule, and improve this schedule by eliminating tools one by one using a knapsack formulation. In order to comprehend the suggested three-stage algorithms, the whole steps of MFFD(dif,1by1) algorithm is presented below as an example.

Step 1: Determine alternative machining conditions settings for each job.

Step 1.1: Generate S alternative settings for each job. The default setting is the one giving the minimum machining time, which is proved to be at the intersection point in Fig. 1.

Step 1.2: Calculate U/P ratio of each job at their default setting and sort the jobs in descending order of their unit usage rates.

Step 1.3: Partition the jobs in four equal groups and assign settings starting from the default one. If min option had been selected, then all jobs would have been assigned to their default settings.

Step 2: Find an initial schedule, i.e. assign jobs to tool blocks.

Step 2.1: Sort the jobs in the SPT order, and assign 30% of the jobs to the tools in this order.

Step 2.2: Resort the remaining jobs in the longest usage rate order, and assign to the tools according to the FFD rule.

Step 2.3: Arrange all tools and jobs on the tools according to the structural properties stated before and calculate the total completion time of the initial schedule.

Step 3: Improve the initial schedule by eliminating tool blocks.

Step 3.1: Select the tool having the minimum total usage of the initial schedule.

Step 3.2: Starting from the first job and following the sequence within the tool, select the job to be moved from this tool. Select the tool having the minimum total usage among the tools other than the pre-selected. On this tool, find the job that has the minimum Response$_i$ value, and decrease its tool usage rate as much as necessary to accommodate the newly inserted job. If there are no more jobs to be moved from the tool, eliminate this tool block and go to step 3.1 to select another tool. If knap alternative had been selected, we would assign the first job to the tool having the minimum total usage among the tools other than the pre-selected, and continue distributing the jobs to the tools in this fashion until all the jobs on the tool are assigned to other tools. For each tool k, we calculate the new settings of its jobs solving the knapsack formulation described above to fit the distributed jobs to each tool block, and go to step 3.1.

Step 3.3: Arrange this new sequence with respect to the structural properties and calculate the new total completion time objective. The same arrangement is also done in the knap case.

Step 3.4: If the new objective is better than the previous one, go to step 3.2 and select another job to be moved. If it is worse, there is still a likelihood of improvement by moving all remaining jobs and eliminating the tool. To test this possibility, go to the next step. If knap alternative had been selected, the stopping criterion would have been a worse solution. This means that eliminating one more tool will not improve the current solution.

Step 3.5: Distribute all of the remaining jobs on the tool as a last chance. If we cannot improve the objective function value, then report the best solution found so far. If not, go to step 3.1 to select the next tool to be eliminated. This step can be considered a look-ahead step since distributing the remaining jobs at once and eliminating the tool may decrease the objective function drastically by eliminating one instance of tool change. We cannot foresee such possibilities with incremental improvements.

5.4. The problem space genetic algorithm

To develop a PSGA for a problem, it is necessary to define a base heuristic and a neighborhood structure. At each iteration of the search, it is important to use a relatively fast base heuristic which provides a means to incorporate problem-specific knowledge into the search. The computational effort required at each iteration and the overall quality of the base heuristic influence the general effectiveness of the problem space approach. By perturbing the problem data, the neighborhood is constructed in the problem space and in the space of these perturbations, the search is done. To search the neighborhood for improvement, the genetic algorithm (GA) is used as a search algorithm. In problem space search, the chromosomes represent the perturbation vectors and genes represent the perturbation amount for a single job. The original problem data are perturbed and the problem with perturbed data is solved by using the base heuristic to obtain a solution. Although the heuristic is applied on the perturbed values, the objective is calculated with the original data values as expected. The new perturbation values are generated from the previous population by using the GA instead of a random generation. Asexual and sexual reproduction and mutation operators can be used in generations. For a more detailed discussion on PSGA, refer to Storer et al. [17].

In this study, cutting speed is taken as the problem data to be perturbed. Since feed rate, machining time and usage rate values depend on the cutting speed, we can calculate the perturbed values of these attributes by using perturbed cutting speed data. A definition of PSGA parameters is given in Table 4. We present below a single-start PSGA which proceeds until the number of generations reaches MAXGEN. However, in the multi-start PSGA, the algorithm restarts itself from the first step for NUMSTART times.

Step 1: Initially generate POPSIZE individuals. For each perturbation vector i (or chromosome) of the population, the number of genes is equal to the number of jobs. Each gene θ_k corresponds to a perturbation amount for each job k that is selected randomly from a range of UN$(-\theta, \theta)$.

Step 2: For each member (chromosome) of the population, perturb the cutting speed of the jobs with this member and calculate the corresponding perturbed feed rate, machining time and usage rate values of the jobs. We also apply mutation to each chromosome. MUTPROB is the probability of mutation, and each gene has this probability to be mutated. If the gene is selected, then it is replaced by a newly generated random value taken in UN$(-\theta, \theta)$. Let v_k be the cutting speed passed from stage 1, then the perturbed cutting speed will be $v_k + \theta_k$. For each generation, insert these perturbed data as an input to the base heuristic and calculate the objective value of the solution using the original cutting speeds, v_k. After finishing all members in the population, save the best and worst solutions. If the number of generations reaches the MAXGEN then stop and report the best solution, else go to step 3 to generate a new population.

Step 3: All of the dispatching rules discussed in stage 2 are a function of v_k such that each perturbation vector could lead to a different sequence even for the same dispatching rule. Compute the fitness values of each member i. The fitness value shows the probability that the population member will be selected for breeding. Let $C_{\max}$ be the maximum objective value in the population, and C_i be the objective value of the ith member. Then, fitness of this member, f_i, is calculated as

$$f_i = \frac{(C_{\max} - C_i)^{\pi}}{\sum_i (C_{\max} - C_i)^{\pi}}.$$

The parameter π determines the selectivity of the algorithm. The selection probability of better solutions increase as π increases. We use these fitness values in selections of asexual and sexual reproduction. %SEXUAL · POPSIZE number of members are generated by sexual reproduction while the remaining are of asexual.

6. Computational results

The single-pass algorithms and PSGA are coded in C language and compiled with Gnu C compiler. The MIP formulations for the original problem and the knapsack formulation used in the third stage of the single-pass heuristics are solved using the callable library routines of CPLEX MIP solver. There are four experimental factors that can affect the efficiency of the algorithms, which can be seen in Table 1 . The experimental design is a 2^4 full factorial design. In addition, we consider two different problem sizes, N being equal to 100 and 200 jobs. Moreover, we take five replications for each factor combination resulting in 160 different randomly generated runs.

H is the maximum available machine horse power. The increase in the value of H increases the feasible speed and feed rate values, and this increases usage rate more than decreasing the machining time. The number of tool changes increases due to an increase in usage rates. L_i is the length of the generated surface for job i. High L means long jobs that have high usage rates and machining times. Owing to these jobs, the tool is changed more frequently, which increases

Table 1
Experimental design factors

Factors	Definition	Level 1	Level 2
H	Machine power	5	10
L	Surface length	UN[2,4]	UN[6,8]
S	Surface roughness	UN[50,100]	UN[300,400]
T_c	Tool change time	Mean[L]	$2 \cdot$ Mean[L]

Table 2
Summary results of heuristics for 100 jobs

Heuristics	Objective			Run times (s)		
	Min	Average	Max	Min	Average	Max
FFD(min,knap)	1530	10 203	37 954	0.15	11.73	68.35
FFD(min,1by1)	1505	12 041	65 867	0.05	0.08	0.13
FFD(dif,knap)	1381	8249	31 460	0.14	5.63	41.80
FFD(dif,1by1)	1381	9443	40 828	0.04	0.08	0.11
MFFD(min,knap)	1414	10 602	46 177	0.18	9.75	66.56
MFFD(min,1by1)	1447	10 452	36 581	0.05	0.08	0.11
MFFD(dif,knap)	1417	8129	34 655	0.31	4.99	34.31
MFFD(dif,1by1)	1380	9114	39 226	0.04	0.08	0.13
FFD2(min,knap)	1493	9540	37 580	0.28	13.61	109.04
SPT(min,knap)	1388	11 075	48 709	0.16	13.83	88.88
SUU(min,knap)	1633	11 171	40 501	0.14	13.58	82.30
$U \times P$(min,knap)	1373	11 463	43 420	0.15	9.16	73.40

the contribution of tool change time in the objective function value. S_i is the maximum allowable surface roughness for job i. If the allowable roughness increases, lower cutting speed and higher feed rates can be used and this decreases the usage and machining time values causing a decrease in the number of tool changes. T_c is the tool change time. When it increases, the second part of the objective function gains more importance. In order to observe the effect of tool change time compared to machining times, we relate the levels of T_c with L, since L is the most important determinant of the magnitude of the machining times. The fixed parameters are $D_i = \text{UN}[1, 4]$, $d_i = \text{UN}[0.05, 0.30]$, $C_o = 0.5$, $C_t = 0.5$, and $S = 10$.

6.1. Experimental results of single-pass heuristics

With three stages of the single-pass heuristics, there are 24 combinations. We took 12 of these combinations, each with 160 runs and a total of 1920 runs. All are solved on a Sparc station 10 under SunOS 5.4. In the initial trial runs, the ones using FFD and MFFD in stage 2 seemed to be more promising, and hence all combinations using FFD and MFFD are taken. The other dispatching rules, FFD2, SPT, SUU and $U \times P$ are combined with “min” and “knap” alternatives of stage 1 and 3. The average of these runs is listed in Tables 2 and 3 showing the minimum, maximum and average results for 100 and 200 jobs, respectively.

In the “knap” alternative in stage 3, a knapsack formulation is solved many times using CPLEX, which increases the computation times. We plot these values for 100-jobs case as an example in Fig. 3. As can be seen from this figure, SUU(min, knap) and SPT(min, knap) have worse objective function values and long CPU times. FFD2(min, knap) is relatively better but still require long runtimes. Selection of “1by1” decreases the computation time drastically. Moreover, the “dif” alternative is always better than “min” in terms of objective function values and CPU times. For both cases, MFFD(dif, knap) and MFFD(dif, 1by1) dominate all other alternatives. One is better in terms of solution quality and the other is better in terms of CPU time. In PSGA, the base heuristic is called many times; therefore, the one with low computation time is more likely to be selected. Also, the solution quality of base heuristic is effective in

Table 3
Summary results of heuristics for 200 jobs

Heuristics	Objective			Run times (s)		
	Min	Average	Max	Min	Average	Max
FFD(min,knap)	6828	46 615	194 400	0.35	28.66	268.32
FFD(min,1by1)	6828	50 375	266 956	0.10	0.17	0.40
FFD(dif,knap)	6367	35 355	130 563	0.41	13.30	149.99
FFD(dif,1by1)	6367	38 768	161 604	0.12	0.16	0.25
MFFD(min,knap)	6576	46 803	188 234	0.39	24.91	220.80
MFFD(min,1by1)	6576	44 061	147 921	0.11	0.18	0.46
MFFD(dif,knap)	6354	34 079	144 952	0.38	13.30	155.66
MFFD(dif,1by1)	6276	37 670	154 818	0.12	0.15	0.19
FFD2(min,knap)	6597	44 243	205 851	0.62	37.14	340.11
SPT(min,knap)	6422	48 755	155 182	0.33	30.76	255.08
SUU(min,knap)	7569	49 834	161 356	0.32	36.42	297.06
$U \times P$(min,knap)	6388	48 781	225 392	0.32	27.45	236.15

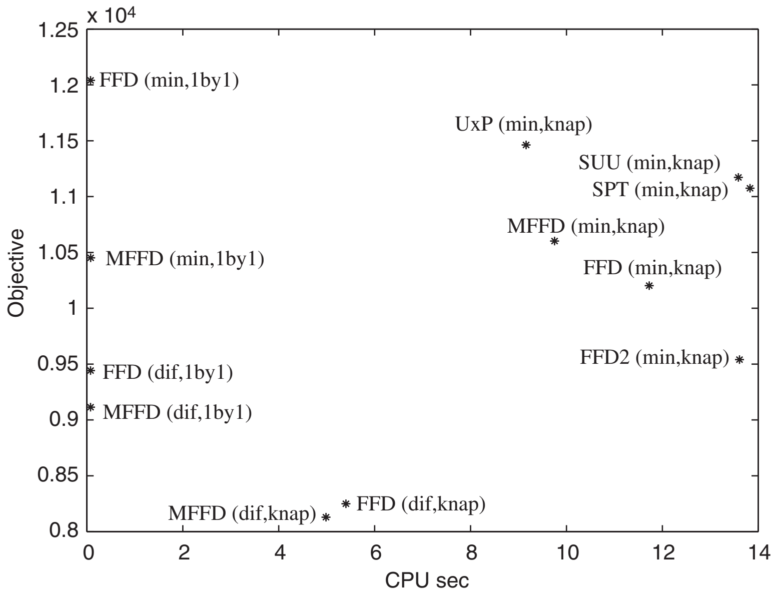

Fig. 3. Summary results of heuristics for 100 jobs.

the performance of PSGA. As a result, the algorithms MFFD(dif, knap) and MFFD(dif, 1by1), which dominate others in terms of time and solution quality, are chosen as base heuristic alternatives in local search.

6.2. Local search parameters and results

To test the efficiency of the proposed PSGA, the required algorithm is coded in C language and runs are taken on a 400 Mhz UltraSPARC station. It was tested on a series of parameter settings to find the appropriate level for each parameter. These parameters and their tested values are given in Table 4. Trial runs are taken for many parameter combinations. We decided on the levels of the parameters by comparing the objective function values and computation times via paired samples t-test statistics using SPSS 10. Since perturbation magnitude shows the deviation from the original data and original sequence, it is important to decide on this parameter. We took three different levels. $\theta = 40$ gave good results but high CPU times when $\pi = 2$, and $\theta = 80$ gave both good results and low CPU times when $\pi = 4$.

Table 4
Definitions and levels of PSGA parameters

Parameters	KNAP			1by1		
POPSIZE: size of the population in a generation	15	20	50		50	
MAXGEN: number of generations	25		30	150		250
%SEXUAL: probability of sexual reproduction	0.5	0.8	1	0.5	0.8	1
CROSSOVER: crossover type in sexual reproduction	Single		Uniform	Single		Uniform
MUTPROB: mutation probability for each gene	0.01		0.05	0.01		0.05
π : selectivity of the algorithm	2		4	2		4
θ : perturbation magnitude	40	60	80	40	60	80
NUMSTART: number of restarts	1		3	1		5

Table 5
Selected parameter combinations for two base heuristics

Base heuristic	MFFD(dif,1by1)	MFFD(dif,knap)
POPSIZE	50	20
MAXGEN	150	30
%SEXUAL	0.5	0.5
CROSSOVER	Single	Single
MUTPROB	0.01	0.01
π	4	4
θ	80	80
NUMSTART	5	3

When $\theta = 80$, $\pi = 4$ pair is compared with $\theta = 40$, $\pi = 2$ pair, the former was better in terms of both solution quality and CPU time. For two base heuristic selections, the fixed parameter values are given in Table 5.

Before comparing two base heuristics used in PSGA, their relative performance with respect to the optimal solution have to be examined. However, the MIP formulation using CPLEX can solve up to 30 jobs. In fact heuristics, especially local search heuristics, usually perform better for large problem sizes. Therefore, comparing for 30-job size may not be enough for our algorithm to prove itself. You can see this fact from Tables 6 and 7. PSGA[1] and PSGA[K] stand for problem space genetic algorithms which use MFFD(dif, 1by1) and MFFD(dif, knap) as base heuristics, respectively. In Table 6, each row is the average of five runs. In some combinations, results of local search are very close to optimal while in others it may deviate significantly. Although POPSIZE and MAXGEN values for the PSGA[K] are lower, i.e. less number of solution alternatives are searched, it gives better results in terms of solution quality. As we mentioned before, the base heuristic solution quality is effective in the performance of local search. The base heuristic used in this algorithm was the best one in terms of solution quality among the single-pass algorithms. The PSGA[1]'s computation times are significantly lower than both the optimal procedure and the PSGA[K] due to the fast base heuristic. As seen from these tables, CPU times of the optimal procedure are significantly higher than the PSGA[K] in which a knapsack formulation is solved many times.

The average results of the PSGA for both base heuristic alternatives are given in Table 8. These results are also compared using paired samples t-test statistics (see Tables 9 and 10). For example, pairs 5 and 6 indicate the difference of two alternatives in terms of solution quality and computation times, respectively. In pairs 1 and 2, the evolution of the PSGA with MFFD(dif, 1by1) is observed. MFFD(dif) assigns settings to every job via "dif" and provides an initial schedule at the end of stage 2. MFFD(dif, 1by1) is the single-pass heuristic composed of all three stages. All steps of this evolution gives significant improvements as seen from these tables. In pairs 3 and 4, we report the evolution of the knap alternative, and again significant improvements are recorded. Pair 7 is the difference of two base heuristics and MFFD(dif, knap) gives better results as expected. This also proves the fact that base heuristic with high solution quality affects the performance of the local search. In the last pair, PSGA[1] is compared with a single-pass algorithm, MFFD(dif, knap). The reason is the computation times. PSGA[K] gives good results but in high computation times. It may be asked that, instead of using local search with a poor (in terms of solution quality) heuristic, using a strong

Table 6
Comparison of PSGA with optimal for 30 jobs

H	L	S	T_c	Optimal		PSGA[1]		PSGA[K]	
				Objective	CPU time	Objective	CPU time	Objective	CPU time
0	0	0	0	225	6963	247	213	253	948
1	0	0	0	226	12652	367	127	289	1297
0	1	0	0	801	177899	1091	120	1021	1376
0	0	1	0	116	319	118	107	131	211
0	0	0	1	275	9051	364	111	287	1059
1	1	0	0	804	418	1200	97	1051	358
1	0	1	0	102	1943	139	90	121	958
1	0	0	1	277	5342	356	68	303	1094
0	1	1	0	389	7751	400	108	398	177
0	1	0	1	1008	2637	1725	125	1203	404
0	0	1	1	117	26	123	122	122	59
1	1	1	0	384	24464	695	100	412	1025
1	1	0	1	1009	4471	1975	117	1361	1787
1	0	1	1	116	4619	231	85	151	587
0	1	1	1	454	4719	533	96	476	335
1	1	1	1	458	4636	540	93	477	2069
Mean				423	16744	632	111	504	859
Std. dev.				315	43386	572	32	413	592

Table 7
Paired samples statistics for optimal and PSGA comparison

Pairs		Mean	Std. dev.	95% CI of difference		t	Sig.
				Lower	Upper		
Pair 1	Optimal-PSGA[1]	−209	277	−356	−61	−3.0	0.009
Pair 2	Optimal-PSGA[K]	−81	108	−139	−23	−3.0	0.009
Pair 3	PSGA[1]-PSGA[K]	128	187	28	228	2.7	0.015
Pair 4	Time(optimal-PSGA[1])	16 633	43 384	−6484	39 751	1.5	0.146
Pair 5	Time(optimal-PSGA[K])	15 885	43 230	−7150	38 921	1.5	0.162
Pair 6	Time(PSGA[1]-PSGA[K])	−748	593	−1064	−432	−5.0	0.000

single-pass heuristic may yield better results? The test results show that this argument is not true. PSGA[1] gives significantly better results showing that using local search algorithm even with a poor heuristic can yield good results. In Fig. 4, we present an efficient frontier of the proposed algorithms. Local search algorithms have high CPU times compared to single-pass heuristics, therefore the CPU times are located on the X-axis logarithmically.

The two local search algorithms are also compared for different combinations of H and T_c. H is an important factor since it circumscribes the search space. Settings, usage rates and processing times of the jobs are directly affected by the level of H. T_c changes the nature of the problem. When T_c is high, i.e. tool change time is more significant, the problem resembles a bin-packing problem. In Table 11, the test results for four different cases are represented, such as in the first four pairs, $H = 0$ and $T_c = 0$ case is considered. In every case, local search is significantly better than its base heuristic. Except for the first case (although the difference is insignificant), PSGA[K] is significantly better than PSGA[1]. In this case, PSGA[1] is recommended to be used since its computation time is low. We also compare PSGA[1] with MFFD(dif, knap) as above. In the first case, single-pass heuristic is significantly worse than the local search. For cases 2 and 3, this difference is not significant. However, in the last case, MFFD(dif, knap) gives significantly better results than PSGA[1]; hence we recommend to use MFFD(dif, knap) for this case.

Table 8
Comparison of two base heuristics of PSGA

H	L	S	T_c	PSGA[1]		PSGA[K]	
				Objective	CPU time	Objective	CPU time
0	0	0	0	3183	118	3167	765
1	0	0	0	4808	129	4773	1110
0	1	0	0	11 807	111	11 337	1731
0	0	1	0	2048	633	1792	390
0	0	0	1	4355	116	4231	841
1	1	0	0	17 242	346	14 938	3898
1	0	1	0	1644	133	1592	883
1	0	0	1	8412	131	5213	7823
0	1	1	0	5766	130	5555	602
0	1	0	1	18 251	114	18 083	990
0	0	1	1	2223	669	1699	418
1	1	1	0	6956	110	6976	703
1	1	0	1	29 557	354	28 349	3706
1	0	1	1	2466	137	2059	748
0	1	1	1	7045	154	6768	566
1	1	1	1	11 741	111	9980	1381

Table 9
Paired samples statistics for different comparisons

Results	Mean	N	Std. deviation	Std. error mean
MFFD(dif)	9789	80	9706	1085
MFFD(dif,1by1)	9718	80	9579	1071
MFFD(dif,knap)	9475	80	9413	1052
PSGA[1]	8594	80	7440	832
PSGA[K]	6788	80	4834	540
TIME(PSGA[1])	216	80	179	20
TIME(PSGA[K])	1660	80	2121	237

Table 10
Paired samples test results for different comparisons

Pairs		Mean	Std. dev.	95% CI of difference		t	Sig.
				Lower	Upper		
Pair 1	MFFD(dif)-MFFD(dif,1by1)	71	214	23	118	2.9	0.004
Pair 2	MFFD(dif,1by1)-PSGA[1]	1124	2393	591	1656	4.2	0.000
Pair 3	MFFD(dif)-MFFD(dif,knap)	314	916	110	517	3.0	0.003
Pair 4	MFFD(dif,knap)-PSGA[K]	2687	5467	1471	3904	4.3	0.000
Pair 5	PSGA[1]-PSGA[K]	1806	3296	1073	2539	4.9	0.000
Pair 6	TIME(PSGA[1])-TIME(PSGA[K])	−1444	2136	−1919	−968	−6.0	0.000
Pair 7	MFFD(dif,1by1)-MFFD(dif,knap)	243	816	61	424	2.6	0.009
Pair 8	PSGA[1]-MFFD(dif,knap)	−881	2331	−1400	−362	−3.3	0.001

7. Conclusion

We have addressed a single machine scheduling problem where job processing times and tool usage can be controlled by selecting the appropriate machining conditions; our objective has been to assign a machine setting to each job and to schedule the jobs such that the total job completion time is minimized. In the process, we have proved an important

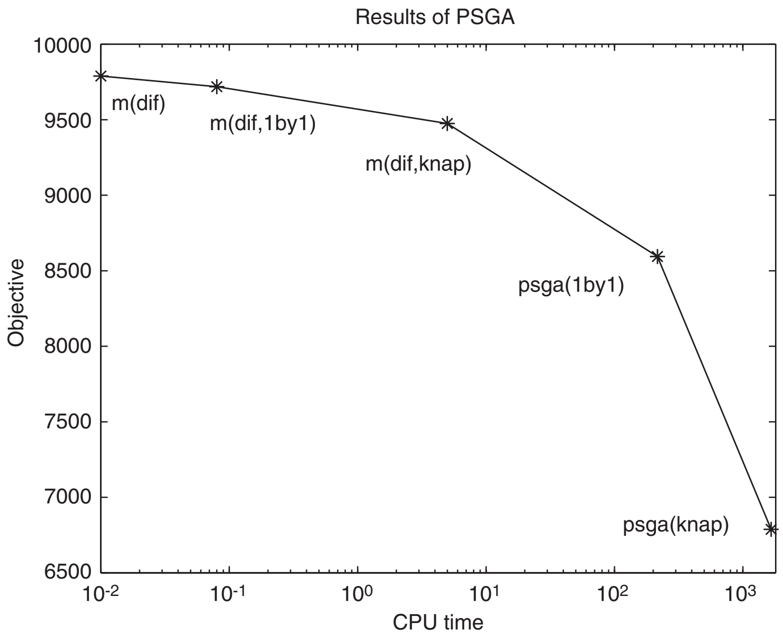

Fig. 4. Efficient frontier of the proposed algorithms.

Table 11
Paired samples test results for two base heuristics under four cases

Pairs		Case	Mean	95% CI of difference		Upper	t	Sig.
				Std. dev.	Lower			
Pair 1	MFFD(dif,knap)-PSGA[K]	(00)	167	171	87	247	4.4	0.000
Pair 2	MFFD(dif,1by1)-PSGA[1]	(00)	192	180	107	276	4.8	0.000
Pair 3	PSGA[K]-PSGA[1]	(00)	9	61	−19	38	0.7	0.499
Pair 4	MFFD(dif,knap)-PSGA[1]	(00)	176	152	105	247	5.2	0.000
Pair 5	MFFD(dif,knap)-PSGA[K]	(01)	588	751	237	939	3.5	0.002
Pair 6	MFFD(dif,1by1)-PSGA[1]	(01)	320	396	135	505	3.6	0.002
Pair 7	PSGA[K]-PSGA[1]	(01)	-533	821	−918	−149	−2.9	0.009
Pair 8	MFFD(dif,knap)-PSGA[1]	(01)	55	330	−100	209	0.7	0.467
Pair 9	MFFD(dif,knap)-PSGA[K]	(10)	1557	1971	634	2479	3.5	0.002
Pair 10	MFFD(dif,1by1)-PSGA[1]	(10)	1263	1934	357	2168	2.9	0.009
Pair 11	PSGA[K]-PSGA[1]	(10)	−1686	2048	−2645	−728	−3.7	0.002
Pair 12	MFFD(dif,knap)-PSGA[1]	(10)	−130	962	−580	321	−0.6	0.554
Pair 13	MFFD(dif,knap)-PSGA[K]	(11)	3579	3408	1984	5173	4.7	0.000
Pair 14	MFFD(dif,1by1)-PSGA[1]	(11)	2721	3947	874	4568	3.1	0.006
Pair 15	PSGA[K]-PSGA[1]	(11)	−5016	4925	−7320	−2711	−4.6	0.000
Pair 16	MFFD(dif,knap)-PSGA[1]	(11)	−1437	2666	−2685	−189	−2.4	0.026

result that has helped us identify the feasible settings for a job in terms of the cutting speed alone. Given the feasible settings for each job, we have proposed a MIP formulation for finding the optimal setting for each job and the optimal job schedule. As a practical solution strategy, we have then offered a number of single-pass heuristic algorithms that execute in three stages. Finally, we have given a PSGA, which uses one of the single-pass algorithms as its base heuristic, to improve upon the solution quality. Based on an extensive computational study, we have found that MFFD(dif,1by1) and MFFD(dif,knap) have had the best performance in terms of computation time and solution quality, respectively. We have, therefore, implemented them as the base heuristics in PSGA. The results have been encouraging.

References

[1] Akturk MS, Ghosh JB, Gunes ED. Scheduling with tool changes to minimize total completion time: a study of heuristics and their performance. Naval Research Logistics 2003;50:15–30.

[2] Petropoulos P. Optimal selection of machining variables using geometric programming. International Journal of Production Research 1973;11:305–14.

[3] Khan Z, Prasad B, Singh T. Machining condition optimization by genetic algorithms and simulated annealing. Computers and Operations Research 1997;24:647–57.

[4] Tang CS, Denardo EV. Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches. Operations Research 1988;36:767–77.

[5] Hertz A, Laporte G, Mittaz M, Stecke KE. Heuristics for minimizing tool switches when scheduling part types on a flexible machine. IIE Transactions 1998;30:689–94.

[6] Gray AE, Seidmann A, Stecke KE. A synthesis of decision models for tool management in automated manufacturing. Management Science 1993;39:549–67.

[7] Trick MA. Scheduling multiple variable-speed machines. Operations Research 1994;42:234–48.

[8] Zdrzalka S. Scheduling jobs on a single machine with release dates, delivery times and controllable processing times: worst case analysis. Operations Research Letters 1991;10:519–23.

[9] Karabati S, Kouvelis P. Flow-line scheduling problem with controllable processing times. IIE Transactions 1997;29:1–14.

[10] Jansen K, Mastrolilli M. Approximation schemes for parallel machine scheduling problems with controllable processing times. Computers and Operations Research 2004;31:1565–81.

[11] 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.

[12] Adiri I, Bruno J, Frostig E, Rinnooy Kan AHG. Single machine flow-time scheduling with a single breakdown. Acta Informatica 1989;26: 679–96.

[13] Lee CY, Liman SD. Single machine flow-time scheduling with scheduled maintenance. Acta Informatica 1992;29:375–82.

[14] Liao CJ, Chen WJ. Single machine scheduling with periodic maintenance and nonresumable jobs. Computers and Operations Research 2003;30:1335–47.

[15] Schmidt G. Scheduling with limited machine availability. European Journal of Operational Research 2000;121:1–15.

[16] Akturk MS, Avci S. Tool allocation and machining conditions optimization for CNC machines. European Journal of Operations Research 1996;94:335–48.

[17] Storer RH, Wu SD, Vaccari R. New search spaces for sequencing problems with application to job shop scheduling. Management Science 1992;38:1495–509.