European Journal of Operational Research 201 (2010) 89–98

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs ☆

Fatih Safa Erenay [a], Ihsan Sabuncuoglu [b], Ayşegül Toptal [b,*], Manoj Kumar Tiwari [c]

[a] Department of Industrial and System Engineering, University of Wisconsin, Madison, WI, USA
[b] Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey
[c] Department of Industrial Engineering and Management, Indian Institute of Technology Kharagpur, Kharagpur 721302, India

ARTICLE INFO

Article history:
Received 8 March 2007
Accepted 12 February 2009
Available online 20 February 2009

Keywords:
Bicriteria scheduling
Average flowtime
Number of tardy jobs
Beam search

ABSTRACT

We consider the bicriteria scheduling problem of minimizing the number of tardy jobs and average flowtime on a single machine. This problem, which is known to be NP-hard, is important in practice, as the former criterion conveys the customer's position, and the latter reflects the manufacturer's perspective in the supply chain. We propose four new heuristics to solve this multiobjective scheduling problem. Two of these heuristics are constructive algorithms based on beam search methodology. The other two are metaheuristic approaches using a genetic algorithm and tabu-search. Our computational experiments indicate that the proposed beam search heuristics find efficient schedules optimally in most cases and perform better than the existing heuristics in the literature.

1. Introduction

Many existing studies on scheduling consider the optimization of a single objective. In practice, however, there are situations in which a decision maker evaluates schedules with respect to more than one measure. Several recent multicriteria scheduling papers address single machine bicriteria scheduling problems. In the vein of this literature, the current study considers the minimization of mean flowtime ($\overline{F}$) and the number of tardy jobs (n_T) on a single machine. Our contribution lies in developing new heuristics that outperform the current approximate solution methodologies and in characterizing the effectiveness of these proposed heuristics in terms of various problem parameters.

The number of tardy jobs and average flowtime are significant criteria for characterizing the behavior of manufacturers who want to meet the due dates of their customers while minimizing their own inventory holding costs. The solution to the single machine problem can be used as an aggregate schedule for the manufacturer, or for generating a more detailed schedule for a factory based on a bottleneck resource. We propose four heuristics to find approximately the *efficient schedules* that minimize n_T and $\overline{F}$ on a single machine. *Efficient schedules* are the set of schedules that cannot be dominated by any other feasible schedule. All other schedules that are not in this set are dominated by at least one of these efficient schedules. Although optimizing either of the objectives, n_T or $\overline{F}$, on a single machine is polynomially solvable, finding efficient schedules that account for them simultaneously is NP-hard (Chen and Bulfin, 1993).

In the literature, most studies on bicriteria scheduling consider a single machine and the minimization of couples of criteria, such as the following: maximum tardiness and flowtime (Smith, 1956; Heck and Roberts, 1972; Sen and Gupta, 1983; Köksalan, 1999; Lee et al., 2004; Haral et al., 2007), maximum earliness and flowtime (Köksalan et al., 1998; Köktener and Köksalan, 2000; Köksalan and Keha, 2003), maximum earliness and number of tardy jobs (Güner et al., 1998; Kondakci et al., 2003), total weighted completion time and maximum lateness (Steiner and Stephenson, 2007), and total earliness and tardiness (M'Hallah, 2007). Extensive surveys of bicriteria single machine scheduling studies are provided by Dileepan and Sen (1988), Fry et al. (1989), and Yen and Wan (2003). Several recent papers investigate bicriteria scheduling problems in other machining environments (Allahverdi, 2004; Toktaş et al., 2004; Arroyo and Armentano, 2005; Gupta and Ruiz-Torres, 2005; Varangharajan and Rejendran, 2005; Vilcot and Billaut, 2008). Nagar et al. (1995), T'kindt and Billaut (1999), and Hoogeveen (2005) review the multicriteria scheduling literature. Other notable studies on multicriteria scheduling investigate the complexity of several problems (e.g., Chen and Bulfin, 1993; T'kindt et al., 2007).

Chen and Bulfin (1993) report that the problem of minimizing n_T while $\overline{F}$ is optimum, on a single machine, can be optimally solved by a polynomial time algorithm, a.k.a. the *adjusted SPT order*. This algorithm uses Moore's Algorithm on the SPT order to break

☆ The appendix of this paper is presented as an online companion at the journal's website.

* Corresponding author. Tel.: +90 (312) 2901702.
E-mail addresses: erenay@wisc.edu (F.S. Erenay), sabun@bilkent.edu.tr (I. Sabuncuoglu), toptal@bilkent.edu.tr (A. Toptal), mkt09@iitkgp.ac.in (M.K. Tiwari).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.02.014

ties among jobs with equal processing times; we refer to the sequence generated according to this algorithm as the *SPT order*. In another study, Emmons (1975) develops an algorithm for the problem of minimizing $\overline{F}$ while n_T is optimum, which is shown to be NP-Hard by Huo et al. (2007).

In the current paper, we seek efficient schedules to minimize the number of tardy jobs and average flowtime on a single machine. The first study on this problem was by Nelson et al. (1986), who proposed a constructive heuristic and an optimal solution based on a branch and bound procedure. In another study, Kiran and Unal (1991) define several characteristics of the efficient solutions. Kondakci and Bekiroglu (1997) present some dominancy rules, which they use to improve the efficiency of the optimal solution procedure by Nelson et al. (1986). Recent studies on the problem propose some general-purpose procedures. Köktener and Köksalan (2000) and Köksalan and Keha (2003) developed heuristic methods based on simulated annealing and a genetic algorithm, respectively. The latter study reports that a genetic algorithm generally outperforms simulated annealing in terms of solution quality, however, a simulated annealing approach is faster than a genetic algorithm.

After reviewing these studies, we observe that only a few solution methodologies (one exact and three heuristics) were proposed for the problem considered in this paper. Moreover, these solution methods are not compared with each other in detail. The only exception is a study by Köksalan and Keha (2003), in which the authors test the performance of their proposed genetic algorithm, relative to the simulated annealing approach of Köktener and Köksalan (2000). A comparison of these two iterative methods with respect to the optimum solution was also made, however, it was limited to a problem size of 20 jobs. In this study, we present four new algorithms: two are constructive algorithms, based on the beam search method, and the other two work iteratively utilizing a genetic algorithm (GA) and tabu-search (TS). We compare these proposed heuristics with each other and with the exact and heuristic solution methods available in the literature.

The organization of this paper is as follows: In Section 2, we present an explicit mathematical formulation for the problem of minimizing the number of tardy jobs and average flowtime on a single machine. In Section 3, we describe Nelson et al.'s (1986) optimal solution method for this problem. The proposed beam search algorithms are presented in Section 4, and GA and TS algorithms are described in Section 5. We discuss the findings of our extensive numerical study in Section 6. Finally, we present general conclusions and future research directions in Section 7.

2. Problem formulation

We consider a single machine environment in which N jobs are to be scheduled with the objective of minimizing the number of tardy jobs and average flowtime. In this environment, jobs have due dates and deterministic processing times. We assume that preemption is not allowed and that there exists no precedence relationship between jobs. P_j and d_j are the processing time and the due date of job j, respectively. Denoting S as a feasible schedule, $\overline{F}(S)$ represents the average flowtime of schedule S, and $n_T(S)$ refers to the number of tardy jobs resulting from schedule S.

Our approach aims at finding efficient schedules for minimizing $\overline{F}$ and n_T. More formally, we are interested in finding a set of schedules where, if S is an element of this set, then there exists no schedule S' satisfying the following constraints, while at least one of these constraints is strict:

$$n_T(S') \leqslant n_T(S), \quad \text{and} \quad \overline{F}(S') \leqslant \overline{F}(S).$$

The solution approach builds on the fact that optimizing either one of the objectives, n_T or $\overline{F}$, on a single machine is polynomially solvable. It is well known in the scheduling literature that the shortest processing time (SPT) rule minimizes the average flowtime and that Moore's Algorithm (Moore, 1968) minimizes the number of tardy jobs. In the rest of the manuscript, we will denote n_T (SPT) and n_T (Moore) as the number of tardy jobs when all jobs are sequenced using the SPT rule and Moore's Algorithm, respectively. Kiran and Unal (1991) showed that for each number of tardy jobs between n_T (SPT) and n_T (Moore), there exists at least one corresponding efficient schedule. The range between n_T (SPT) and n_T (Moore) is referred to as the *efficient range* of the number of tardy jobs. Any schedule having a number value of tardy jobs that is outside the efficient range is dominated by some efficient schedule. Since there exists at least one efficient schedule for every n_T value in this range, the total number of efficient schedules for a given problem is at least n_T (SPT) – n_T (Moore) + 1. Therefore, for a problem with N jobs, we solve the following model for all n such that n_T (SPT) $\geqslant n \geqslant n_T$ (Moore).

$$\begin{aligned} &\text{Min}_{\forall S} \quad \overline{F}(S) \\ &\text{s.t.} \qquad n_T(S) = n. \end{aligned}$$

To present a more detailed formulation of the above problem, let us define X_{ij} and Y_j as follows:

$$X_{ij} = \begin{cases} 1, & \text{if } i\text{th position is held by job } j \\ 0, & \text{o.w.} \end{cases} \quad \text{and}$$

$$Y_j = \begin{cases} 1, & \text{if job } j \text{ is tardy} \\ 0, & \text{o.w.} \end{cases}$$

Also, let M and ξ denote a very large and very small number, respectively. We next present an explicit mathematical model for our problem. Recall that this model should be solved for all $n \in [n_T$ (Moore), n_T (SPT)]

$$\text{Min} \quad \frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}(N-i+1)X_{ij}P_j\right)$$

$$\text{s.t.} \quad \sum_{j=1}^{N} X_{ij} = 1 \quad \text{for all } i \in \{1,2,\ldots N\}, \tag{1}$$

$$\sum_{i=1}^{N} X_{ij} = 1 \quad \text{for all } j \in \{1,2,\ldots N\}, \tag{2}$$

$$d_j - P_j - \sum_{r=2}^{N}\sum_{i=1}^{r-1}\sum_{k=1}^{N} X_{rj}X_{ik}P_k \geqslant -M \times Y_j \quad \text{for all } j \in \{1,2,\ldots N\}, \tag{3}$$

$$d_j - P_j - \sum_{r=2}^{N}\sum_{i=1}^{r-1}\sum_{k=1}^{N} X_{rj}X_{ik}P_k \leqslant M \times (1-Y_j) - \xi \quad \text{for all } j \in \{1,2,\ldots N\}, \tag{4}$$

$$\sum_{j=1}^{N} Y_j = n. \tag{5}$$

In the above formulation, Eq. (1) assures that only one job can be assigned to each position in the schedule. Eq. (2) makes sure that there is no unassigned job. Expressions (3) and (4) jointly identify whether job j is tardy or not, i.e., $Y_j = 0$ or $Y_j = 1$. Finally, Eq. (5) states that only n jobs are tardy. Inequalities (3) and (4) are nonlinear, due to the multiplication of X_{rj} and X_{ik}. Since both variables are binary, however, it is possible to linearize these inequalities by replacing $X_{rj}X_{ik}$ with Z_{rjik} and adding the following three constraints to the model for all $i, j, k, r \in \{1,\ldots,N\}$.

$$\text{a) } X_{rj} \geqslant Z_{rjik}, \quad \text{b) } X_{ik} \geqslant Z_{rjik}, \quad \text{c) } Z_{rjik} \geqslant X_{rj} + X_{ik} - 1.$$

Observe that, the efficient schedule that has n_T (SPT) tardy jobs is the schedule that is formed according to the SPT order. Therefore, the remaining n_T (SPT) – n_T (Moore) efficient schedules need to be

found. Nelson et al. (1986) proposed an efficient branch and bound algorithm to find all these schedules optimally. Yet this algorithm is not computationally efficient for large size problems. Since the heuristics that we propose will take from Nelson et al.'s branch and bound algorithm (B&B Algorithm) and will be compared with it, we next present a brief summary of this algorithm.

3. Optimal solution methodology for minimizing n_T and $\overline{F}$

The B&B Algorithm developed by Nelson et al. (1986) depends on two key points. First is the fact that, given N jobs and a subset of these N jobs, the schedule that gives a minimum value for $\overline{F}$ while keeping the jobs in the given subset non-tardy is found using Smith's Algorithm (see Smith, 1956; Kiran and Unal, 1991). The second point is presented in the following theorem.

Theorem 1. (Nelson et al., 1986). *The jobs that are early in the SPT order are also early in at least one of the efficient schedules with $n_T = n$ for all n s.t. n_T (SPT) $\geqslant n \geqslant n_T$ (Moore).*

This theorem implies that in order to find an efficient schedule with $n_T = n$, it is necessary to determine which other n_T(SPT) – n jobs will be early, besides the early jobs of the SPT order. Therefore, to minimize $\overline{F}$ subject to having n tardy jobs, all subsets with cardinality n_T(SPT) – n that are composed of the tardy jobs in the SPT order should be evaluated using Smith's Algorithm. The schedule that is obtained through this evaluation is the efficient schedule for $n_T = n$.

The B&B method is designed to determine one efficient schedule at every level of the branch and bound tree. More specifically, at the kth level, an efficient schedule for $n_T = n_T$(SPT) – k is found, where k=0,…, n_T(SPT) – n_T(Moore). In this tree, each node stores a set of jobs that need to be kept non-tardy. We will refer to this set as the *early job set*. An early job set at level k is a subset of N jobs with cardinality $N - n_T$(SPT) + k. The nodes at level k cover all possible subsets that have the specified cardinality. Of these jobs, $N - n_T$(SPT) in each early job set are the early jobs of the SPT order, and the remaining k are among the tardy jobs of the SPT order. Smith's Algorithm is run for each node in level k, and the schedule that has the minimum $\overline{F}$ while keeping the corresponding $N - n_T$(SPT) + k jobs non-tardy is found. The schedule that gives the least $\overline{F}$ considering all the nodes at level k, is the efficient schedule for $n_T = n_T$(SPT) − k. The procedure is repeated for each level of the branch and bound tree. The tree starts with the node that stores the early jobs of the SPT order at the level 0 and ends at the level n_T(SPT) – n_T(Moore), after finding the efficient schedule for n_T(Moore).

As stated above, we use Smith's Algorithm to evaluate the nodes of Nelson et al.'s B&B tree. Smith's Algorithm minimizes $\overline{F}$ given T_{max} is zero, where T_{max} is the maximum tardiness. Equivalently, it finds the schedule that minimizes $\overline{F}$ given $n_T = 0$. In order to utilize this algorithm at a node, we first set the due dates of the jobs not included in the corresponding early job set to infinity. That is, for each node k, we solve the following problem using Smith's Algorithm:

$$\begin{aligned} &\text{Min}_{\forall S} \quad \overline{F}, \\ &\text{s.t.} \qquad T_{max}(S) = 0, \\ &d_j = \infty \quad \forall j \notin E_k, \end{aligned}$$

where E_k is the early job set of node k.

4. Proposed beam search algorithms

Beam search is a fast and approximate branch and bound algorithm, where instead of expanding every node to the next level, as in the classical branch and bound tree, only a limited number of promising nodes are expanded. Thus, rather than performing all branch and bound tree operations, beam search efficiently operates on only a small portion of the tree. Examples of beam search applications on various scheduling problems include Sabuncuoglu and Karabuk (1998), Sabuncuoglu and Bayiz (1999), Ghirardi and Potts (2005).

Generally, at a level of beam search tree, the nodes are evaluated via a *global evaluation function*. The nodes with the highest scores are selected to be expanded to the next level. The number of these nodes is fixed and is called *beam width* (b) in the literature. In some beam search applications, a portion of the nodes to be expanded to the next level is chosen randomly. A variation of beam search algorithms uses *local evaluation functions* to eliminate some of the nodes before evaluating them with the global evaluation function. This approach is called a *filtered beam search*. Sabuncuoglu et al. (2008) provide a comprehensive review of beam search algorithms.

There are two types of beam search implementations with respect to the branching procedure: dependent and independent beam search. We applied both of these branching procedures to the problem under consideration.

4.1. Independent beam search (BS-I)

The first two levels (level 0 and level 1) of our beam search tree are the same as Nelson et al.'s search tree. At level 2, however, only b nodes are expanded to the next level. These b nodes have the b smallest $\overline{F}$ values obtained from applying Smith's Algorithm. At the next levels, only one node from the same parent can be expanded to the next level. The schedule implied by the node with the minimum $\overline{F}$ among all the nodes at a level, is the heuristic efficient schedule for that level. Note that the global evaluation function of BS-I is the average flowtime obtained by running Smith's Algorithm for a node. This algorithm utilizes an *independent beam search* because its solution tree has b independent branches. As in Nelson et al.'s B&B Algorithm, BS-I terminates at level n_T(SPT) – n_T(Moore) after finding a heuristic efficient schedule for $n_T = n_T$(Moore).

4.2. Dependent beam search (BS-D)

The dependent beam search algorithm is a slightly modified version of the independent beam search algorithm. In the independent beam search tree, after the second level, only one node is expanded to the next level, among the nodes from the same parent. In the dependent beam search, however, all the nodes at a level are evaluated together, without considering their parent nodes, and b nodes with the smallest $\overline{F}$ values are expanded to the next level. This implies that more than one node that has the same parent node can be expanded to the next level.

Note that the heuristic proposed by Nelson et al. (1986) is based on expanding the node with the minimum flowtime at each level of a given B&B tree. It can be observed that this heuristic is nothing but a special version of our proposed beam search algorithms, with a beam width of 1. In the rest of the text, we refer to this heuristic as *Nelson's Heuristic*.

Next, we illustrate the proposed beam search algorithms over a numerical example.

Example: Consider a single machine scheduling problem with six jobs having the processing time and due-date information as in Table 1.

In order to find the heuristic efficient schedules for the above problem instance, we first need to determine the efficient range. It turns out that we have n_T (Moore) = 1, $\overline{F}$ (Moore) = 19.83, n_T (SPT) = 4, and $\overline{F}$ (SPT) = 13, and, therefore, the efficient range contains four values. Recall that SPT order is the efficient schedule corresponding to $n_T = 4$ and that it has jobs 1 and 2 as early and the remaining jobs as tardy.

Figs. 1a and 1b show the search trees using BS-I and BS-D, respectively, for $b = 2$. Both trees have four levels, each correspond-

Table 1
Problem parameters.

Job (j)	Processing time (P_j)	Due date (d_j)
1	1	40
2	2	3
3	3	5
4	5	7
5	10	20
6	15	32

ing to a different n_T value, with the single node in Level 1 representing the SPT order. The early jobs at each node k are stored in set E_k, and the schedules that give the minimum flowtime while keeping these jobs non-tardy are found using Smith's Algorithm.

5. Proposed iterative algorithms

In this section, we propose two iterative algorithms based on tabu-search and genetic algorithm approaches. Such approaches, in general, are generic metaheuristics for locating a good approximation to the global optimum of a given function, in a large search space. Tabu search belongs to the class of local search techniques and is based on avoiding local optima by using memory structures called tabu lists. These lists temporarily record visited solutions and prevent the algorithm from cycling around these solutions. Genetic algorithms, on the other hand, are among global search heuristics. Solutions are represented as chromosomes with varying gene structures. A typical genetic algorithm is based on changing an initially generated set of solutions using techniques such as mutation and crossover, until a terminating condition is satisfied.

5.1. Proposed tabu-search approach

The proposed TS Algorithm utilizes Theorem 1 and Smith's Algorithm. To find a heuristic efficient schedule with $n_T = n$, subsets of cardinality n_T (SPT) – n that include the tardy jobs of the SPT order are searched. First, a subset with cardinality n_T (SPT) – n is ran-

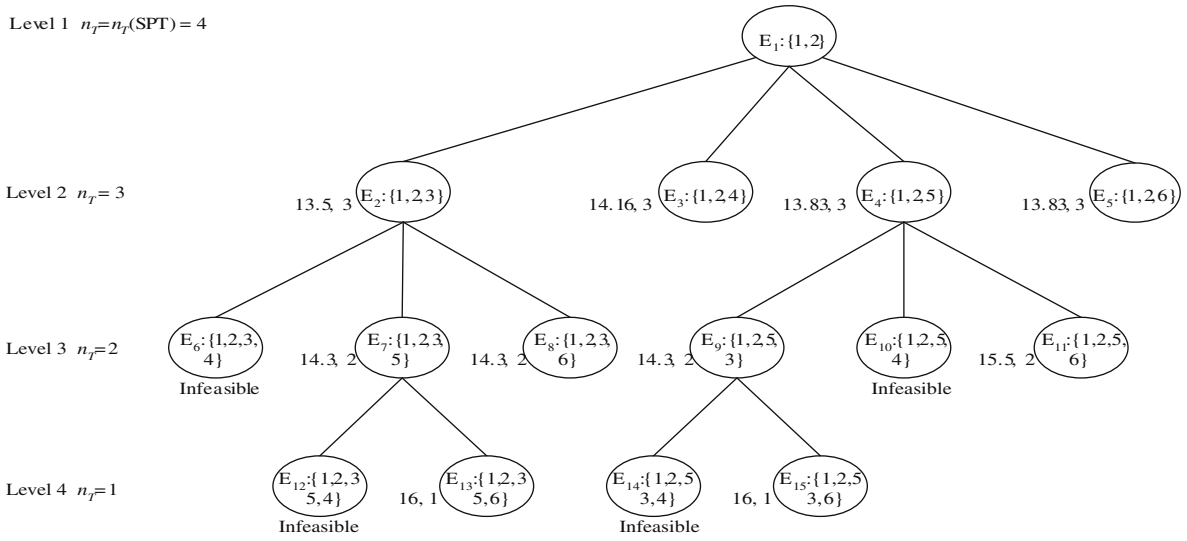

Fig. 1a. Independent beam search tree for the numerical example when $b = 2$.

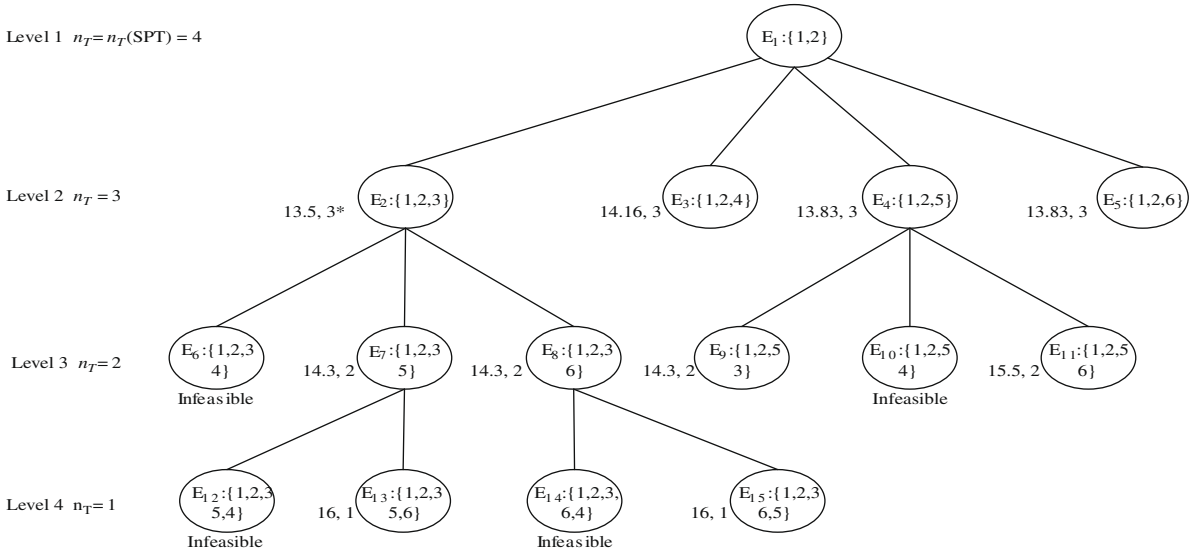

Fig. 1b. Dependent beam search tree for the numerical example when $b = 2$. (*The first entry refers to the minimum flowtime when the jobs in set E2 are nontardy and the second entry is the number of tardy jobs.)

domly selected and taken as the current subset. Then, some neighbors of this current subset with cardinality $n_T(\text{SPT}) - n$ are generated. Next, these neighbors are evaluated using Smith's Algorithm. The neighbor for which Smith's Algorithm gives the least $\overline{F}$ is accepted as the new current subset. After 100 iterations, or if every neighbor appears to be infeasible, the schedule that Smith's Algorithm finds for the current subset is accepted as the heuristic efficient schedule with $n_T = n$.

The procedure described above is a forward search starting from $n_T = n_T(\text{SPT}) - 1$ and continuing towards $n_T = n_T(\text{Moore})$. Our initial runs indicate, however, that the forward search cannot find a heuristic efficient schedule for some $n_T = n$, where $n_T(\text{SPT}) \geqslant n \geqslant n_T(\text{Moore})$. Thus, a backward search is also performed starting from $n_T = n_T(\text{Moore})$. In this backward search, the jobs that are tardy in Moore's Algorithm are allowed to be tardy at every iteration. For each $n_T = n$, which other $n - n_T(\text{Moore})$ jobs will be allowed to be tardy is searched in the same manner as in the forward search. After backward and forward searches are completed, among all the schedules found for $n_T = n$, the one with the smallest $\overline{F}$ is selected as the heuristic efficient schedule. Detailed descriptions of the forward and backward search mechanisms are given in Appendix A.

5.1.1. Neighborhood generation

The neighbors of the current subset are generated by selecting a specific job from the current subset and replacing it with another job that is not an element of the current subset. The selected job is replaced with every possible job, one by one, to generate all possible neighbors. A job is selected to be replaced with a probability that is inversely proportional to the number of times the job has been selected before (i.e., N_j). That is, the probability of selecting job j is given by

$$p_j = \frac{t_j}{\sum_{j=1}^{N} t_j}, \quad \text{where} \quad t_j = \frac{\sum_{j=1}^{N} N_j}{N_j}, \quad j \in \{1, 2, \ldots, N\}.$$

5.1.2. Tabu list and aspiration criterion

The jobs from the current subset that are selected to be replaced are added to the tabu list. Once a job is added to the tabu list, it is kept there for the next five iterations. The aspiration criterion is to override the tabu status of a move if this move yields the best solution so far.

5.1.3. Stopping criteria

As discussed above, the TS Algorithm terminates after a forward search and a backward search are completed. Both of these searches are based on evaluating 100 consecutive neighbors for $n_T = n$ where $n_T(\text{SPT}) \geqslant n \geqslant n_T(\text{Moore})$.

5.2. Proposed genetic algorithm

The proposed genetic algorithm (GA) tries to find the jobs to be tardy in the efficient schedule with $n_T = n$, for all n, such that $n_T(\text{SPT}) \geqslant n \geqslant n_T(\text{Moore})$. It searches on the subset of the N jobs with cardinality n. The proposed algorithm uses binary representation, that is, each of these subsets is represented with chromosomes of N genes having a value 1 or 0. Each gene represents the tardiness state of the corresponding job. For example, if the jth gene has value 1, then the jth job is allowed to be tardy, otherwise the jth job should be non-tardy.

Recall that the schedule that gives minimum $\overline{F}$ while keeping the jobs with gene value zero as non-tardy can be found using Smith's Algorithm. Therefore, finding the right chromosome is equivalent to finding the efficient schedule. The steps of GA can be found in Appendix B.

5.2.1. The fitness function

The fitness function is used to determine the worst two chromosomes in the current population and the second parent chromosome for crossover operations via tournaments. We define the fitness function as

$$w\frac{|n_T(C) - n|}{n_T(\text{SPT}) - n_T(\text{Moore})} + (1 - w)\frac{\overline{F}(C) - \overline{F}(\text{SPT})}{\overline{F}(\text{Moore}) - \overline{F}(\text{SPT})}.$$

This function is quite similar to the one used by Köksalan and Keha (2003). The only difference is that $n_T(C)$ and $\overline{F}(C)$ are obtained by evaluating chromosome C using Smith's Algorithm.

5.2.2. Initial population

Köksalan and Keha (2003) present an algorithm to find the initial schedule with $n_T = n$. Their genetic algorithm starts the search for the efficient schedule with $n_T = n$ at this initial schedule. They refer to this algorithm as the *initial heuristic*. We propose another initialization heuristic. For a given problem having $n_T \leqslant n$, we assign a job to each position starting from the first position. Job j is eligible to be assigned to the current position if scheduling the remaining unassigned jobs according to Moore's Algorithm yields at most n tardy jobs in total. Among the eligible jobs, the one with the shortest processing time will be placed in the current location in the schedule.

The population size is constant and is equal to 30. In forming an initial population of chromosomes, for each n_T value in $\{n, n-1, n+1, n+2\}$, one schedule is generated using Köksalan and Keha (2003)'s initial heuristic, and one schedule is found using our initial heuristic. A total of eight chromosomes are created representing the tardy jobs in these schedules. Similarly, three other chromosomes are generated for the EDD order, the SPT order, and the sequence that results from Moore's Algorithm. The latter two chromosomes are used to form the neighbors. That is, by changing the values of some genes from 1 to 0, five neighbor chromosomes are produced from the chromosome representing the SPT order. Another five are generated by changing the values of some genes from 0 to 1 in the chromosome corresponding to Moore's Algorithm. In both cases, the total gene values of the neighbor chromosomes will be equal to n. Lastly, nine solutions are generated randomly. The initial population consists of all these listed chromosomes.

5.2.3. Crossover and mutation operators

In order to update the population, two chromosomes, called parents, are chosen for crossover. The first parent is generated with a tournament and the second parent is selected randomly. The tournament for the first parent involves determining the chromosome that has the best fitness function value, among five randomly selected ones. Two-point crossover operation is used in the proposed algorithm. Here, two genes on the parent chromosomes are selected randomly, and parts of the chromosomes between these genes are interchanged to generate two new chromosomes. These two chromosomes are added to the population, and the two worst chromosomes according to the fitness function are extracted from the population. This crossover mechanism is similar to the one presented by Köksalan and Keha (2003). Mutation is applied to a randomly selected chromosome in the current population. The selected chromosome's two genes, one with value 1 and the other with value 0, are selected randomly, and their gene values are interchanged.

5.2.4. Stopping criteria

In order to find a heuristic efficient schedule for $n_T = n$ where $n_T(\text{SPT}) \geqslant n \geqslant n_T(\text{Moore})$, varying values of the weight w are used in the fitness function (see Step 7 of the algorithm in Appendix B). For

Table 2
Due date ranges.

Due date type	Due date range
I	[0,0.4SP]
II	[0.1SP,0.3SP]
III	[0.25SP,0.45SP]
IV	[0.3SP,1.3SP]

a given w value, the search is complete after 100 crossovers or if the best chromosome does not change for 20 consecutive crossovers.

6. Computational results

In order to evaluate the performances of the proposed heuristics, we conducted experiments on randomly generated problems with sizes of 20, 30, 40, 60, 80, 100, and 150 jobs. The processing times are taken as uniformly distributed in the ranges [0,25] and [0,100], representing low and high processing time variability, respectively. The due dates are also uniformly distributed on the four different ranges summarized in Table 2. Here, SP denotes the sum of the processing times of the N jobs. The same due date and processing time distributions are used by Köksalan and Keha (2003).

Before performing an extensive numerical study, we conducted a preliminary analysis to decide on a beam width value. We observed that the quality of the solution for a problem is very sensitive to a marginal increase in the beam width at its smaller values. As the beam width increases, however, its impact on the solution quality diminishes. We also observed that the range of the beam width values that improve the solution is smaller for small size problems. Therefore, we focused on the largest sized problems within our consideration to decide on a single value for the beam width and used it for all problems. Namely, for each processing time and due-date combination, we generated five sample problems with 150 jobs (i.e., 40 problems overall).

Recall that the total number of efficient schedules for a given problem is at least $n_T(\text{SPT}) - n_T(\text{Moore}) + 1$. Our sample problems yielded 756 efficient schedules, each corresponding to a different n_T value for an instance. In order to measure the impact of increasing beam width, we considered the number of heuristic efficient schedules that had a better solution quality at the new beam width value, compared to that at $b = 1$. Fig. 2 shows a plot of how the number of such cases changes with increasing beam width, when BS-I is used. The behavior of how the performance of BS-D changes with increasing beam width is similar. As seen in Fig. 2, the performance of the beam search does not change much after $b = 20$; therefore, we took $b = 20$ for using BS-I and BS-D.

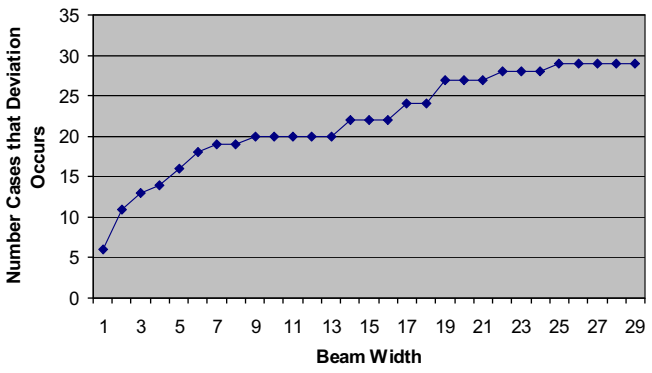

Fig. 2. Number of heuristic efficient schedules in which BS-I with beam width b yields a better solution than BS-I with beam width 1.

6.1. Comparison with the optimal solution

Since the B&B Algorithm developed by Nelson et al. (1986) requires long computational times (e.g., up to four days for a problem with 60 jobs), the performance of the proposed heuristics, relative to the optimum solution, was tested only for *small size* problems (i.e., 60 jobs or less). A detailed comparison of all the heuristics among themselves was made over *large size* problems (with more than 60 jobs), and the results will be presented in the next section.

Comparison of the heuristics with the optimal solution was made over 320 small size problems, resulting from 10 instances for each combination of job size, due date, and processing time distribution. These problems were solved using the seven heuristics (i.e., Nelson's Heuristic, BS-I, BS-D, Köksalan and Keha (2003)'s genetic algorithm (GA(K&K)), Köktener and Köksalan (2000)'s simulated annealing (SA(K&K)), the proposed tabu-search (TS), and the proposed genetic algorithm (GA)), and were compared with Nelson's optimal solution procedure. Köksalan and Keha (2003) state that tournament size does not affect the performance of GA(K& K) considerably. Therefore, we took the tournament size as 5 for all genetic algorithm applications.

The following four measures were considered in our experiments.

(i) *Average percentage deviation*: Average percentage deviation illustrates the average gap between the heuristic and the optimal solution over all efficient schedules and test problems for which this gap is positive. These cases will be referred to as *deviation instances* in the rest of the manuscript. The average percentage deviation of a heuristic from the optimum is defined as

$$\frac{\sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})} 100 \times \frac{\bar{F}(m,n)-\bar{F}_{\text{OPT}}(m,n)}{\bar{F}_{\text{OPT}}(m,n)}}{\sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})}\varphi_{m,n}}.$$

Here, M is the total number of problems; $\bar{F}(m,n)$ and $\bar{F}_{\text{OPT}}(m,n)$ are, respectively, the mean flowtime values of the heuristic and the optimum solutions of the mth problem, given $n_T = n$. $n_T(m,\text{Moore})$ and $n_T(m,\text{SPT})$ are the number of tardy jobs for the mth problem, when jobs are sequenced according to Moore's Algorithm and the SPT order. Finally, $\varphi_{m,n}$ is defined as

$$\varphi_{m,n} = \begin{cases} 1, & \text{if } \bar{F}(m,n) > \bar{F}_{\text{OPT}}(m,n) \\ 0, & \text{o.w.} \end{cases}$$

(ii) $\textit{Maximum Percentage Deviation} = \max_{(m,n)}\left(100 \times \frac{\bar{F}(m,n)-\bar{F}_{\text{OPT}}(m,n)}{\bar{F}_{\text{OPT}}(m,n)}\right)$, where $n_T(\text{SPT}) \geqslant n \geqslant n_T(\text{Moore})$ for a test problem. We use maximum percentage deviation as an indicator of the worst-case performance of a heuristic.

(iii) ND/N_{Total}, where ND (total number of deviation instances) and N_{Total} (total number of efficient schedules) are defined as

$$ND = \sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})}\varphi_{m,n} \quad \text{and}$$

$$N_{\text{Total}} = \sum_{m=1}^{M}[n_T(m,\text{SPT}) - n_T(m,\text{Moore}) + 1].$$

ND/N_{Total} is the proportion of deviation instances among all efficient schedules for the M problems; therefore, $(1 - ND/N_{\text{Total}})$ is the proportion of optimally found efficient schedules by a heuristic.

(iv) *Average CPU time*: The average computation time spent in finding all heuristic efficient schedules for a test problem, using a Pentium 3.00 GHz processor.

Table 3 summarizes the results of our experiments with problem sizes of 20, 30, 40, and 60 jobs. The results indicate that Nelson's Heuristic, BS-I and BS-D perform better than GA(K&K) and SA(K&K), according to all performance measures. The average percentage deviation value for the problems with 60 jobs is the only exception for which GA(K&K) outperforms BS-I. The reason behind this is that BS-I resulted in only five deviation instances, and one of the deviation values was high. For all other performance measures, BS-I performs better than GA(K&K).

For problems with 20, 30, and 40 jobs, BS-I and BS-D find all efficient schedules optimally, and they consume the same amount of computational time. For problems with 60 jobs, BS-D and BS-I both yield some deviation instances; however, the number of such instances is smaller than for the other heuristics. When BS-I and BS-D are compared to Nelson's Heuristic, we observe that their solution quality is better; however, they require more computational time. Although the performances of GA(K&K), SA(K&K), and Nelson's Heuristic worsen as the problem size increases, the performances of the beam search heuristics are quite stable.

Another observation is that both GA and TS perform better than GA(K&K) and SA(K&K) but not as good as the beam search algorithms. GA outperforms TS, according to the average percentage deviation criterion, for three out four different job sizes. For the problems with 40 and 60 jobs, however, TS finds a greater number of efficient schedules optimally than GA does. Their average deviation values do not exhibit a pattern according to the job size. As job size increases, however, ND/N_{Total} increases for both TS and GA.

Table 3 further shows that Nelson's Heuristic is the fastest of all seven approximate solution methods. Although GA(K&K)'s solution quality is better than that of SA(K&K), it is much slower than SA(K&K). GA is the slowest heuristic. BS-I and BS-D perform much better than GA(K&K), SA(K&K), GA, and TS in terms of the average CPU time.

Tables 4 and 5 summarize the average percentage deviation values for low and high processing time distributions, respectively, under each due date distribution type and problem size combination. These tables illustrate that BS-I, BS-D provide better solutions than do GA, SA, SA(K&K), and GA(K&K) with respect to each job size, processing time, and due date distribution type. In fact, BS-I and BS-D deviate from the optimal solution only in the problem sizes of 60 jobs.

Tables 4 and 5 further indicate that the problems generated using Type IV due date distribution are solved quite effectively by the beam search heuristics and by Nelson's Heuristic. Recall that this distribution type represents problems with loose due dates. Although these algorithms also work well for problems with tighter due dates (i.e., due date distribution types I, II, or III), most deviation instances occur in these problem types. Processing time variability, on the other hand, does not affect the solution quality of BS-I and BS-D. For Nelson's Heuristic, deviation from optimality

Table 3
Comparison of the heuristics with the optimum solution.

Problem size	Performance measure	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
20 Jobs	Average deviation (%)	0.28	0	0	0.64	4.57	0.14	0.44
	ND/N_{Total}	1/183	0/183	0/183	63/183	177/183	7/183	37/183
	Maximum deviation (%)	0.28	0	0	7.17	34.38	0.33	4.57
	CPU time (seconds)	0.01	0.01	0.01	0.26	0.22	0.50	0.30
30 Jobs	Average deviation (%)	1.26	0	0	0.54	2.52	0.34	0.32
	ND/N_{Total}	5/260	0/260	0/260	158/260	248/260	42/260	52/260
	Maximum deviation (%)	3.00	0	0	8.04	22.48	2.27	3.83
	CPU time (seconds)	0.01	0.02	0.02	1.08	0.49	2.06	1.15
40 Jobs	Average deviation (%)	0.34	0	0	0.60	2.71	0.24	0.54
	ND/N_{Total}	7/365	0/365	0/365	125/365	358/365	115/365	85/365
	Maximum deviation (%)	1.08	0	0	3.82	26.76	1.94	5.0
	CPU time (seconds)	0.02	0.04	0.04	3.10	0.73	6.10	2.87
60 Jobs	Average deviation (%)	0.39	0.90	0.93	0.53	2.58	0.29	0.67
	ND/N_{Total}	15/582	5/582	2/582	486/582	573/582	308/582	186/582
	Maximum deviation (%)	2.22	2.22	0.182	4.05	3.35	25.15	7.55
	CPU time (seconds)	0.05	0.14	0.14	15.53	1.70	30.79	12.95

Table 4
Average deviation from optimum in problems with low processing time variability (%).

Problem size	Due date type	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
20 Jobs	I	0.28	0	0	0.50	3.16	0.18	0.34
	II	0	0	0	0.68	5.41	0	0.97
	III	0	0	0	0.22	4.64	0.33	1.66
	IV	0	0	0	0.32	7.22	0	0.07
30 Jobs	I	0	0	0	0.78	1.62	0.47	0.13
	II	0	0	0	0.23	3.31	0.11	0.58
	III	0	0	0	0.16	2.46	0	0.22
	IV	1.95	0	0	0.61	3.54	0.35	0.23
40 Jobs	I	0.19	0	0	1.06	1.52	0.38	0.86
	II	1.06	0	0	0.22	2.11	0.08	0.22
	III	0.04	0	0	0.22	3.33	0.08	0.03
	IV	0	0	0	0.35	4.68	0.23	0.25
60 Jobs	I	0.27	0.61	0.91	0.77	1.64	0.42	0.68
	II	0.01	0.01	0.01	0.25	1.96	0.08	0.04
	III	0	0	0	0.13	3.04	0.03	0.05
	IV	0	0	0	0.25	3.55	0.08	0.13

Table 5
Average deviation from optimum in problems with high processing time variability (%).

Problem size	Due date type	Nelson's Heuristic	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
20 Jobs	I	0	0	0	0.92	3.09	0.08	0.46
	II	0	0	0	0.38	3.67	0	0.22
	III	0	0	0	0.24	5.67	0	0.03
	IV	0	0	0	0.87	6.11	0	0.22
30 Jobs	I	0.19	0	0	0.89	1.41	0.42	0.30
	II	2.39	0	0	0.14	2.01	0.02	0.20
	III	0.005	0	0	0.08	4.03	0	1.92
	IV	0	0	0	0.40	3.75	0.18	0.23
40 Jobs	I	0	0	0	0.87	1.37	0.23	1.14
	II	0.88	0	0	0.30	2.14	0.04	0.13
	III	0	0	0	0.20	3.99	0.03	0.01
	IV	0	0	0	0.37	3.82	0.05	0.13
60 Jobs	I	0	0	0	1.08	1.77	0.48	1.02
	II	0.62	0	0	0.22	1.90	0.05	0.02
	III	0.001	0	0	0.17	2.89	0.05	0.02
	IV	0	0	0	0.29	4.08	0.12	0.27

mostly occurs in the problems with low processing time variability combined with Type I due dates and in problems with high processing time variability combined with Type II due dates. It can also be observed that GA performs better in problems with high processing time variability.

It is important to note that, as the problem size increases, Nelson's Heuristic, BS-I, BS-D, SA(K&K), and TS may fail to find a solution for some of the efficient schedules. As stated before, for a given problem, there are n_T (SPT) – n_T (Moore) + 1 efficient schedules. In some of the 320 test problems, however, these heuristics cannot find an approximate solution specifically for the efficient schedule with n_T (Moore) number of tardy jobs (see Table 6). The number of such problems is very small, and most of the cases that cannot be solved by Nelson's Heuristic are solved by BS-I and BS-D. Nevertheless, the number of these instances seems to increase as the problem size increases. In order to see whether this trend will continue for larger problem sizes and to better observe the performance of our heuristics, we performed experiments on problems with 80, 100, and 150 jobs.

6.2. Experiments on larger problem sizes

We generated larger size problems with 80, 100, and 150 jobs using the same processing time and due date distributions discussed earlier. For each job size, processing time, and due date distribution type, we generated 10 problems and obtained 240 problems in total. In our experiments with larger problems, the first measure that we consider is the average percentage difference of a heuristic's solution from that of Nelson's Heuristic, which is given by

$$\frac{\sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})} 100 \times \frac{\overline{F}_{\text{Nelson}}(m,n)-\overline{F}_{\cdot}(m,n)}{\overline{F}_{\text{Nelson}}(m,n)}}{\sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})}\psi_{m,n}}, \tag{6}$$

Table 6
Number of efficient schedules for which no solution is found.

Heuristic	20 Jobs	30 Jobs	40 Jobs	60 Jobs
Nelson's Heuristic	0/183	0/260	5/365	15/582
BS-I	0/183	0/260	2/365	4/582
BS-D	0/183	0/260	1/365	3/582
GA(K&K)	0/183	0/260	0/365	0/582
SA(K&K)	3/183	3/260	1/365	8/582
GA	0/183	0/260	0/365	0/582
TS	5/183	16/260	20/365	48/582

where $\overline{F}(m,n)$ is the minimum flowtime for the m^{th} problem, when $n_T = n$, given by a heuristic other than Nelson's. $\overline{F}_{\text{Nelson}}(m,n)$ is the minimum flowtime resulting from Nelson's Heuristic for the same problem. $n_T(m,\text{Moore})$ and $n_T(m,\text{SPT})$ are the number of tardy jobs using Moore's Algorithm and SPT order, respectively. Finally, $\psi_{m,n}$ is defined as

$$\psi_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) \neq \overline{F}_{\text{Nelson}}(m,n), \\ 0, & \text{o.w.} \end{cases}$$

Average percentage difference illustrates the average gap between the solution of a heuristic (i.e., BS-I, BS-D, GA(K&K), SA(K&K), GA, TS) and that of Nelson's Heuristic over all efficient schedules and test problems, where the two solutions differ. In our experimentation with larger size problems, we take Nelson's Heuristic as a benchmark, because, among the heuristic approaches proposed in the literature, Nelson's Heuristic performs the best, as discussed in Section 6.1. Note that, according to Expression (6), larger values of average percentage difference indicate better performance for a heuristic. Other measures we consider in the experiments with larger problems are as follows.

(i) N+: Number of solutions for efficient schedules over all test problems that a specific heuristic performs better than Nelson's Heuristic. That is,

$$\text{N+} = \sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})} \eta_{m,n}, \quad \text{where}$$

$$\eta_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) < \overline{F}_{\text{Nelson}}(m,n) \\ 0, & \text{o.w.} \end{cases}$$

(ii) N−: Number of solutions for efficient schedules over all test problems that a specific heuristic performs worse than Nelson's Heuristic. That is,

$$\text{N}- = \sum_{m=1}^{M}\sum_{n=n_T(m,\text{Moore})}^{n_T(m,\text{SPT})} \mu_{m,n}, \quad \text{where}$$

$$\mu_{m,n} = \begin{cases} 1, & \text{if } \overline{F}(m,n) > \overline{F}_{\text{Nelson}}(m,n), \\ 0, & \text{o.w.} \end{cases}$$

Note that taking Nelson's Heuristic as a point of reference, a large value of N+ and a small value of N− are desirable for a heuristic.

(iii) Maximum Percentage Difference $= \max_{(m,n)}\left(100 \times \frac{\overline{F}_{\text{Nelson}}(m,n)-\overline{F}_{\cdot}(m,n)}{\overline{F}_{\text{Nelson}}(m,n)}\right)$. A negative value of maximum percentage difference implies that Nelson's Heuristic performs better than the current heuristic, in

Table 7
Comparison of the other heuristics with Nelson's Heuristic.

Problem size	Performance measure	BS-I	BS-D	GA(K&K)	SA(K&K)	GA	TS
80 Jobs	Average % difference	0.124	0.126	−0.532	−2.694	−0.223	−0.228
	$N+/N_{Total}$	31/754	31/754	4/754	0/754	7/754	5/754
	$N-/N_{Total}$	0/754	0/754	670/754	683/754	502/754	169/754
	Max % difference	0.908	0.908	0.526	−0.007	0.526	0.473
	Min % difference	0.000	0.000	−4.362	−21.319	−2.306	−4.133
100 Jobs	Average % difference	0.069	0.069	−0.514	−2.946	−0.275	−0.278
	$N+/N_{Total}$	39/990	39/990	4/990	0/990	8/990	5/990
	$N-/N_{Total}$	0/990	0/990	922/990	912/990	768/990	509/990
	Max % difference	0.685	0.685	0.534	−0.012	0.534	0.333
	Min % difference	0.000	0.000	−4.600	−27.639	−3.527	−5.151
150 Jobs	Average % difference	0.047	0.041	−0.480	−3.723	−0.259	−0.268
	$N+/N_{Total}$	74/1531	77/1531	3/1531	0/1531	7/1531	7/1531
	$N-/N_{Total}$	0/1531	0/1531	1480/1531	1310/1531	1332/1531	1026/1531
	Max % difference	1.225	1.225	1.458	−0.037	1.458	1.222
	Min % difference	0.000	0.000	−3.325	−38.030	−3.342	−5.941

all cases considered. Maximum percentage difference can be considered as a measure of the best-case performance of a heuristic, and its higher values are desirable.

(iv) Minimum Percentage Difference $= \min_{(m,n)}\left(100 \times \frac{\bar{F}_{Nelson}(m,n) - \bar{F}_{(m,n)}}{\bar{F}_{Nelson}(m,n)}\right)$. A positive value of minimum percentage difference implies that the current heuristic performs better than Nelson's Heuristic. Minimum percentage difference can be considered a measure of the worst-case performance of a heuristic, and its higher values are desirable.

As Table 7 shows, the proposed beam search heuristics and Nelson's Heuristic perform better than SA(K&K), GA(K&K), GA, and TS, in larger size problems as well, with respect to all the measures. As implied by the $N+/N_{Total}$ and $N-/N_{Total}$ measures, in almost all cases, Nelson's Heuristic performs better than or equal to these iterative algorithms. Yet, BS-I and BS-D perform better than Nelson's Heuristic. Furthermore, as job size increases, the number of instances where the proposed heuristics outperforms Nelson's Heuristic increases. We also observe that BS-D performs slightly better than BS-I on larger size problems. In most cases, however, their solution qualities are almost the same. As seen in Table 7, for problems with 150 jobs, BS-D outperforms Nelson's Heuristic in a few more instances than does BS-I.

GA and TS perform better than GA(K&K) and SA(K&K) for almost all measures. The performances of TS and GA are nearly the same for the cases in which they both find a solution. In these cases, the overall average difference from Nelson's Heuristic is nearly the same. The best-case and worst-case performances of GA, as measured by the maximum and minimum percentage differences, respectively, are better than those of TS. The real handicap of TS is that there are a considerable number of efficient schedules for which it cannot find a heuristic solution (see Table 8). GA, on the other hand, finds heuristic efficient schedules for every instance.

Table 8 illustrates that for all the heuristics, excluding GA and GA (K&K), the number of efficient schedules for which no heuristic solution is found increases with an increase in problem size. It can also be observed that Nelson's Heuristic cannot find any solution for a larger number of efficient schedules than BS-I and BS-D. In fact, if a heuristic efficient schedule cannot be found either by BS-I or BS-D, it cannot be found by Nelson's Heuristic either. There are some n_T values for which Nelson's Heuristic cannot arrive at a heuristic efficient schedule, but BS-I or BS-D can. Heuristic efficient schedules that cannot be found frequently coincide with problems that have a Type I due date distribution, and less frequently coincide with those that have Types II and III.

Table 8
Number of efficient schedules for which no Heuristic solution is found.

Heuristic	80 Jobs	100 Jobs	150 Jobs
Nelson's Heuristic	13/754	29/990	48/1531
BS-I	7/754	13/990	32/1531
BS-D	7/754	13/990	34/1531
GA (K&K)	0/754	0/990	0/1531
SA (K&K)	23/754	63/990	206/1531
GA	0/754	0/990	0/1531
TS	57/754	96/990	189/1531

Table 9
Average CPU time per problem in seconds.

Heuristic	80 Jobs	100 Jobs	150 Jobs
Nelson's Heuristic	0.08	0.12	0.34
BS-I	0.35	0.82	3.95
BS-D	0.36	0.84	4.00
GA(K&K)	48.63	124.87	723.57
SA(K&K)	13.33	21.32	49.31
GA	94.77	245.22	1445.85
TS	36.43	88.52	420.13

While the number of no solution cases is quite high for SA(K&K) and TS, GA and GA(K&K) find a solution for every n_T value. As seen in Table 9, however, the computation time requirements for GA and GA(K&K) are much longer than those of the beam search algorithms. Therefore, for large size problems, in order to find heuristic efficient schedules for all n_T values in the efficient range, BS-D or BS-I should first be used to minimize the number of instances in which no solution is found. Then, a genetic algorithm should be utilized to solve the remaining instances.

7. Conclusions

As a result of our experiments, we conclude that BS-D and BS-I perform quite well for the multicriteria scheduling problem of minimizing the average flowtime and number of tardy jobs. In most cases, these two algorithms find the efficient schedules optimally. The only disadvantage of our beam search algorithms is that, although rarely, they fail to find heuristic efficient solutions for some n_T values in the efficient range. For such cases, we propose that GA or GA(K&K) be used. The proposed GA and TS algorithms also yield better results than GA(K&K) and SA(K&K), even though they are poor, relative to BS-D and BS-I.

With the insights gained from this study, we propose to extend our current research to solve other multicriteria scheduling problems. This work can be extended to more complex settings, such

as parallel machine environments. It would also be interesting to study robustness and stability measures in dynamic and stochastic manufacturing settings.

Acknowledgements

The authors thank Süleyman Kardaş and Mustafa Aydoğdu for their assistance in obtaining the numerical results for the genetic algorithm and the tabu-search applications in this paper.

Appendix. Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2009.02.014.

References

Allahverdi, A., 2004. A new heuristic for *m*-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness. Computers and Operations Research, 31; 157–180.
Arroyo, J.E.C., Armentano, V.A., 2005. Genetic local search for multi-objective flowshop scheduling problems. European Journal of Operational Research 167, 717–738.
Chen, C.L., Bulfin, R.L., 1993. Complexity of single machine multi-criteria scheduling problems. European Journal of Operational Research 70, 115–125.
Dileepan, P., Sen, T., 1988. Bicriterion static scheduling research for a single machine. Omega 16, 53–59.
Emmons, H., 1975. One machine sequencing to minimize mean flowtime with minimum tardy. Naval Research Logistics Quarterly 22, 585–592.
Fry, T., Armstrong, R., Lewis, H., 1989. A framework for single machine multiple objective scheduling research. Omega 17, 595–607.
Ghirardi, M., Potts, C.N., 2005. Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. European Journal of Operational Research 165 (2), 457–467.
Güner, E., Erol, S., Tani, K., 1998. One machine scheduling to minimize maximum earliness with minimum number of tardy jobs. International Journal of Production Economics 55, 213–219.
Gupta, J.N.D., Ruiz-Torres, A.J., 2005. Generating efficient schedules for identical parallel machines involving flow-time and tardy jobs. European Journal of Operational Research 167, 679–695.
Haral, U., Chen, R.-W., Ferrell, W.G., Kurz, M.B., 2007. Multiobjective single machine scheduling with nontraditional requirements. International Journal of Production Economics 106, 574–584.
Heck, H., Roberts, S., 1972. A note on the extension of a result on scheduling with secondary criteria. Naval Research Logistics Quarterly 19, 403–405.
Hoogeveen, J.A., 2005. Multicriteria scheduling. European Journal of Operational Research 167, 592–623.
Huo, Y., Leung, J.Y.T., Zhao, H., 2007. Complexity of two-dual criteria scheduling problems. Operations Research Letters 35, 211–220.
Kiran, A.S., Unal, A.T., 1991. A single-machine problem with multiple criteria. Naval Research Logistics 38, 721–727.
Kondakci, S.K., Bekiroglu, T., 1997. Scheduling with bicriteria: Total flowtime and number of tardy jobs. International Journal of Production Economics 53, 91–99.
Kondakci, S., Azizoglu, M., Köksalan, M., 2003. Single machine scheduling with maximum earliness and number tardy. Computers and Industrial Engineering 45, 257–269.
Köksalan, M., 1999. A heuristic approach to bicriteria scheduling. Naval Research Logistics 46, 777–789.
Köksalan, M., Keha, A.B., 2003. Using genetic algorithms for single-machine bicriteria scheduling problems. European Journal of Operational Research 145, 543–556.
Köksalan, M., Azizoglu, M., Kondakci, S., 1998. Minimizing flowtime and maximum earliness on a single machine. IIE Transactions 30, 192–200.
Köktener, E.K., Köksalan, M., 2000. A simulated annealing approach to bicriteria scheduling problems on a single machine. Journal of Heuristics 6, 311–327.
Lee, W.-C., Wu, C.-C., Sung, H.-J., 2004. A bi-criterion single-machine scheduling problem with learning considerations. Acta Informatica 40, 303–315.
M'Hallah, R., 2007. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. Computers and Operations Research 34, 3126–3142.
Moore, J.M., 1968. An n job, one machine sequencing algorithm for minimizing the number of late jobs. Management Science 15, 102–109.
Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: A literature survey. European Journal of Operations Research 81, 88–104.
Nelson, R.T., Sarin, R.K., Daniels, R.L., 1986. Scheduling with multiple performance measures: The one machine case. Management Science 32, 464–479.
Sabuncuoglu, I., Bayiz, M., 1999. Job shop scheduling with beam search. European Journal of Operational Research 118, 390–412.
Sabuncuoglu, I., Karabuk, S., 1998. A beam search algorithm and evaluation of scheduling approaches for FMSs. IIE Transactions 30, 179–191.
Sabuncuoglu, I., Gockun, Y., Erel, E., 2008. Backtracking and exchange of information: Methods to enhance a beam search algorithm for assembly line scheduling. European Journal of Operational Research 186, 915–930.
Sen, T., Gupta, S.K., 1983. A branch and bound procedure to solve a bicriterion scheduling problem. IIE Transactions 15, 84–88.
Smith, W.E., 1956. Various optimizers for single stage production. Naval Research Logistics Quarterly 3, 1–2.
Steiner, G., Stephenson, P., 2007. Pareto optimal for total weighted completion time and maximum lateness on a single machine. Discrete Applied Mathematics 155, 2341–2354.
T'kindt, V., Billaut, J.-C., 1999. Some guidelines to solve multicriteria scheduling problems. IEEE International Conference on Systems, Man and Cybernetics Proceeding 6, 463–468.
T'kindt, V., Bouibede-Hocine, K., Esswein, C., 2007. Counting and enumeration complexity with application to multicriteria scheduling. Annals of Operations Research 153, 215–234.
Toktaş, B., Azizoğlu, M., Köksalan, S.K., 2004. Two-machine flow shop scheduling with two criteria: Maximum earliness and makespan. European Journal of Operational Research 157, 286–295.
Varangharajan, T.K., Rejendran, C., 2005. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. European Journal of Operational Research 167, 772–795.
Vilcot, G., Billaut, J.C., 2008. A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. European Journal of Operational Research 190, 398–411.
Yen, B.P.C., Wan, G., 2003. Single machine bicriteria scheduling: A survey. International Journal of Industrial Engineering 10, 222–231.