**International Journal of Advanced Manufacturing Technology manuscript No.**
(will be inserted by the editor)

**S. Afshin Mansouri · S. Hamed Hendizadeh · Nasser Salmasi**

# Bicriteria scheduling of a two-machine flowshop with sequence dependent setup times

Received: date / Accepted: date

**Abstract** A two-machine flowshop scheduling problem is addressed to minimize setups and makespan where each job is characterized by a pair of attributes that entail setups on each machine. The setup times are sequence dependent on both machines. It is shown that these objectives are conflicting so Pareto optimization approach is considered. The scheduling problems considering either of these objectives are $\mathcal{NP}$-hard, so exact optimization techniques are impractical for large size problems. We propose two multi-objective metaheurisctics based on genetic algorithms (MOGA) and simulated annealing (MOSA) to find approximations of Pareto-optimal sets. The performances of these approaches are compared with lower bounds for small problems. In larger problems, performance of the proposed algorithms are compared with each other. Experimentations revealed that both algorithms perform very similar on small problems. Moreover, it was observed that MOGA outperforms MOSA in terms of the quality of solutions on larger problems.

**Keywords** Multicriteria scheduling, Sequence-dependent setups, Flowshop, Pareto-optimal frontier, Genetic Algorithms, Simulated Annealing

S. Afshin Mansouri (corresponding author)
*Brunel Business School, Brunel University, Uxbridge, Middlesex UB8 3PU, UK.*
*Fax: +44-1895-265361*
*E-mail: Afshin.Mansouri@brunel.ac.uk*

S. Hamed Hendizadeh
*Department of Mechanical and Manufacturing Engineering, Faculty of Engineering, University of Manitoba, Winnipeg, Manitoba, Canada R3T 5V6.*
*E-mail: umhendiz@cc.umanitoba.ca*

Nasser Salmasi
*Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran.*
*E-mail: nsalmasi@sharif.edu*

## 1 Introduction

In a general flowshop scheduling problem, $n$ jobs are to be scheduled on $m$ machines in order to optimize some measures of performance. All jobs have the same processing requirements so they need to be processed on all machines in the same order. Two-machine flowshop scheduling problem has been considered as a major subproblem due to its applications in real-life. There are cases where setup times are negligible and therefore could be included in the jobs' processing times. However, in some applications, setups have major impact on the performance measure considered for the scheduling problem so they need to be considered separately. Wang and Cheng [31] and Lee and Jung [12] have addressed, among others, this kind of problem. Two-machine flowshop scheduling has also been considered as a bicriteria problem for instance by Tkindt et al. [30] and Chou and Lee [2].

In this paper we address a flowshop sequence dependent job scheduling problem. The term "sequence dependent" implies that the setup times depend on the sequence in which the jobs are processed on the machines. Each job $J_i$ is characterized by two attributes. The attribute of job $J_i$ on machine $k$ is denoted by $a_{i,k}$. Let $A_1$ and $A_2$ denote the sets of all possible attributes on machines $M_1$ and $M_2$, respectively. If job $J_j$ is processed immediately after job $J_i$, a setup time $s_{ij,k}$ is required on machine $k$, if $a_{i,k} \neq a_{j,k}$. The setup times as in many real world scheduling problems are sequence dependent [32]. The process time of job $J_i$ on machine $k$ is shown by $p_{i,k}$. Scheduling with sequence-dependent setups has received significant attention in recent years, for instance by Lin and Ying [13], Low *et al.* [16], Pugazhendhi *et al.* [23] and Gajpal *et al.* [8].

The goal is to schedule the set of jobs in order to minimize the number of *setups* and the *makespan* (or $C_{max}$). The first objective is usually favored by the production managers to reduce cost and complexity of the production plan while the second one is mostly considered by the customers as a measure of service. There

| | | Machine 2 | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Machine 1 | 1 | - | - | - | $J_1$ | $J_2$ | $J_3$ |
| | 2 | - | - | $J_4$ | $J_5$ | - | - |
| | 3 | - | $J_6$ | - | $J_7$ | - | - |
| | 4 | $J_8$ | - | - | $J_9$ | - | - |

**Table 1** Attributes of jobs. Each job is characterized by its first attribute (row) on the first machine and second attribute (column) on the second machine

| | Attributes | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| Machine 1 | 2 | 2 | 6 | 9 | - | - |
| Machine 2 | 9 | 4 | 9 | 3 | 1 | 7 |

**Table 2** Process times of attributes

| | | Attributes | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| Attributes | 1 | 8 | 4 | 8 | 20 |
| | 2 | 18 | 12 | 5 | 4 |
| | 3 | 1 | 8 | 20 | 7 |
| | 4 | 10 | 6 | 3 | 4 |

**Table 3** Setup times on machine 1 (if the attribute in row $i$ is processed immediately after the attribute in column $j$. The diameter show the setup times of the first attribute in the sequence)

seems to be a natural conflict between these two objectives so one must consider the set of Pareto-optimal solutions equally favorable if preferences of the decision-maker are not known *a priori*. Using the standard three-field notation of multicriteria scheduling problems [29], the bicriteria problem addressed in this research can be referred to as $F_2|S_{sd}|Setups, C_{max}$.

As an example of the above problem, consider a two-stage furniture production system where each stage represents a machine. At stage one, sheets of raw materials (MDF, DDF, plywood, etc.) are cut and subsequently painted in the second stage according to the market demand. The painted pieces are then assembled on an assembly line and delivered to the customers. A setup changeover is needed in the cutting department when the thickness of two successive jobs (furniture parts) differs substantially. In the painting department, a setup is required when the color of two successive jobs changes. The setup times are sequence-dependent. For instance, a setup changeover from black to white in the painting department takes longer than the reverse case.

To show the conflict between the two objectives, consider an instance consisting of nine jobs which are processed on two machines. On the first machine, there are four different attributes while the second machine is capable of handling six attributes. The attributes, process times as well as setup times on the first and second machine are shown in Tables 1, 2, 3 and 4, respectively.

By minimizing the two objectives independently via total enumeration, we find the following solutions:

| | | Attributes | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Attributes | 1 | 4 | 12 | 20 | 3 | 18 | 8 |
| | 2 | 10 | 7 | 16 | 3 | 2 | 5 |
| | 3 | 19 | 13 | 1 | 6 | 2 | 9 |
| | 4 | 1 | 3 | 1 | 9 | 2 | 12 |
| | 5 | 3 | 12 | 19 | 15 | 18 | 14 |
| | 6 | 6 | 15 | 5 | 4 | 20 | 17 |

**Table 4** Setup times on machine 2 (if the attribute in row $i$ is processed immediately after the attribute in column $j$. The diameter show the setup times of the first attribute in the sequence)

$S1: \quad J_4 J_5 J_1 J_3 J_2 J_6 J_7 J_9 J_8$
$S2: \quad J_9 J_4 J_5 J_1 J_3 J_6 J_7 J_8 J_2$

The objective vectors for $S1$ and $S2$ are (11, 83) and (14, 80), where the elements of the vector represent *setups* and *makespan*, respectively. The sequence $S1$ minimizes total setups whilst $S2$ minimizes makespan. Obviously none of these sequences dominates the other one with respect to the both decision criteria. This shows that minimizing the two objective functions are not equivalent. Sequence S1 optimizes number of setups while sequence S2 minimizes the makespan. Figure 1 shows the Gantt charts of $S1$ and $S2$. In this figure, the black blocks represent the setup times.

Agnetis *et al.* [1] show that minimizing total setups in a two stage supply chain where all jobs pass through both stages in the same order (or problem $F_2||Setups$) is $\mathcal{NP}$-hard. This is indeed a special case of the two-machine flowshop scheduling problem and henceforth we conclude that the optimization problem concerning the first objective is $\mathcal{NP}$-hard. Gupta and Darrow [9] proved that minimizing the *makespan* of the two machine sequence dependent job scheduling (SDJS) problem is a $\mathcal{NP}$-hard problem. As such, we conclude that $F_2|S_{sd}|Setups, C_{max}$ is $\mathcal{NP}$-hard and henceforth, exact optimization tools are impractical for large size problem instances.

To the best of our knowledge, this problem has not previously been addressed in the literature. We formulate it in a multi-objective optimization framework. As show earlier, the two objectives are conflicting which entails application of multi-objective search. We propose two multi-objective metaheuristics based on genetic algorithm and simulated annealing to find approximations of Pareto-optimal fronts. An efficient lower bound is also defined to evaluate performance of the solution techniques in small sized problems. The possible effect of an initializing heuristic on performance of the two algorithms is examined. Extensive experiments are carried out using a diverse set of test problems to identify which algorithm is preferred for which class of problem instances.

The rest of the paper is organized as follows: A discussion on multiobjective optimization is given in section 2. In section 3, lower bounds are introduced for the two

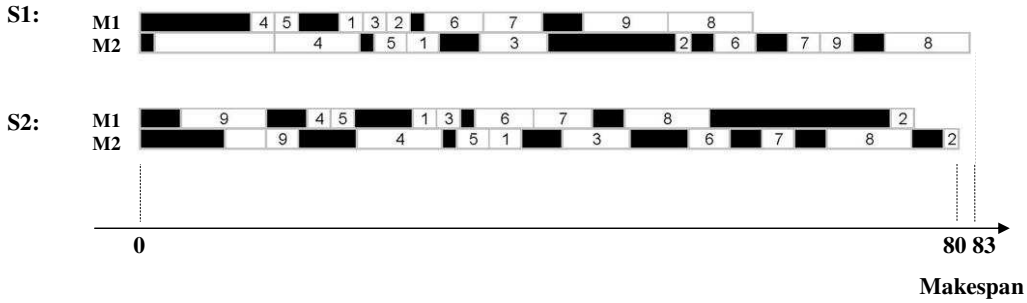


**Fig. 1** Gantt chart of two solutions $S1 = J_4J_5J_1J_3J_2J_6J_7J_9J_8$ and $S2 = J_9J_4J_5J_1J_3J_6J_7J_8J_2$ found via complete enumeration to minimize *setups* and *makespan* (or $C_{max}$). The black segments within the bars represent *setups*. As it can be seen, $S1$ requires the minimum $setups = 11$ and results in $C_{max} = 83$ while $S2$ with more *setups* (14) minimizes $C_{max}$ to as low as 80. This exapmle shows that the two objectives are conflicting and therefore, cannot be optimized at the same time in all cases.

objectives in order to establish ideal solutions for comparison purposes. The MOGA and MOSA approaches are introduced in sections 4 and 5 respectively. Computational results are discussed in section 6. Finally, section 7 concludes.

## 2 Multiobjective Optimization

A Multiobjective Optimization Problem (MOP) can be defined as determining a vector of design variables within a feasible region to minimize a vector of objective functions that usually conflict with each other. Without the loss of generality, a MOP can be formulated as follow:

$$Min\ \{f_1(\tilde{x}), \ldots, f_m(\tilde{x})\}$$

subject to:

$$\begin{aligned} &g_j(\tilde{x}) \leq 0,\ j = 1, \ldots, k; \\ &h_l(\tilde{x}) = 0,\ l = 1, \ldots, p. \end{aligned} \tag{1}$$

where $\tilde{x}$ is the vector of decision variables; $f_i(\tilde{x})$ is the $i$-th objective function; $g_j(\tilde{x})$ is the $j$-th inequality constraint and $h_l(\tilde{x})$ is the $l$-th equality constraint.

A decision vector $\tilde{x}$ is said to dominate a decision vector $\tilde{y}$ (also written as $\tilde{x} \succ \tilde{y}$) iff:

$$f_i(\tilde{x}) \leq f_i(\tilde{y});\ \forall\ i \in \{1, \ldots, m\}; \tag{2}$$

and:

$$\exists\ i \in \{1, \ldots, m\} \mid f_i(\tilde{x}) < f_i(\tilde{y}). \tag{3}$$

All feasible decision vectors that are not dominated by any other feasible decision vector are called nondominated or Pareto-optimal. These are solutions for which no objective can be improved without detracting from at least one other objective.

There are various solution approaches for solving the MOP. Among the most widely adopted techniques are: sequential optimization, $\epsilon$-constraint method, weighting method, goal programming, goal attainment, distance based method and direction based method. For a comprehensive study of these approaches, readers may refer to Szidarovsky *et al.* [28] and Collette and Siarry [4].

Multicriteria scheduling problems can be classified as a subset of MOPs in manufacturing and service sectors. For a comprehensive survey on theory and applications of multicriteria scheduling, readers may refer to T'kindt and Billaut [29].

## 3 Ideal Solution

An ideal objective vector $\Theta$ could be defined consisting of the lower bounds of the two objectives as follows:

$$\Theta = (LB_1, LB_2) \tag{4}$$

where $LB_1$ and $LB_2$ are lower bounds for the first (minimization of setups) and the second (minimization of makespan) criteria.

Mansouri [17] shows that $LB_1$ could be defined as follows:

$$LB_1 = max\{n, (|A_1| + |A_2|)\} \tag{5}$$

It is obvious that the number of setups could not be less than the number of jobs. However, if the number of jobs ($n$) is less than total attributes ($|A_1| + |A_2|$), the lower bound need to be increased to take into account at least one setup per attribute. These two facts are the basis for deriving the lower bound of setups as stated in the above formula.

Concerning $LB_2$, Logendran et al. [14] proposed a lower bounding technique by minimizing the *makespan* criterion for the two machine group scheduling problem. This model is enhanced by Salmasi [24]. A special case of this lower bound is used for the current research where all groups contain only one job. This model mainly solves the sequence dependent job scheduling problem optimally. In other word, $LB_2$ is defined for a given

problem by solving the appropriate optimization problem whose objective is to minimize *makespan* in a two-machine sequence-dependent problem. We use CPLEX package to find this lower bound for test problems in this research.

## 4 The MOGA Approach

GAs have successfully been applied to the scheduling problems with multiple objectives, for instance by Ponnambalam *et al.* [21], Pasupathy *et al.* [20] and Prasad *et al.* [22]. For comprehensive details of multi-objective optimization by means of GAs, readers may refer to the books of Coello *et al.* [3] and Deb [6].

The main steps of the proposed MOGA approach are presented in Algorithm 1. Moreover, major steps of the algorithm are discussed in more details in the following sub-sections.

**input** : Search parameters
**output**: A nondominated set
Let time counter $t = 0$;
Initialize search parameters;
Let $\{Elite\ Set\} = \emptyset$;
**while** $t < t_{max}$ **do**
  Perform nondominated sorting and niching;
  Select individuals for *mating pool*;
  From *mating pool*, generate *new generation* using genetic operators;
  Let *current generation* = *new generation*;
  Identify $F^1$ = nondominated frontier of *current generation*;
  Let $\{Elite\ Set\} = \{Elite\ Set\} \cup F^1$;
  Refine $\{Elite\ Set\}$;
  Let $t = t + 1$;
**end**
Report $\{Elite\ Set\}$;

**Algorithm 1**: Pseudocode of the MOGA approach

### 4.1 Fitness Assignment

To assign appropriate fitness to the individuals in a population taking into account both objectives, nondominated sorting method proposed by Srinivas and Deb [25] was selected. In this method, the population is ranked on the basis of an individual's non-domination. The nondominated individuals present in the population are first identified by the current population. Then, all these individuals are assumed to constitute the first nondominated frontier in the population and assigned a large dummy fitness value ($DF_i$ for all $i \in F^1$). The same fitness value is assigned to give an equal reproductive potential to all these nondominated individuals. To maintain diversity in the population, these classified individuals are then shared with their dummy fitness values.

Sharing is achieved by dividing the dummy fitness value of an individual by its niche count, i.e. the number of individuals in its niche using the following formula:

$$f_i = \frac{DF_i}{NC_i} \tag{6}$$

where $f_i$, $DF_i$ and $NC_i$ represent, respectively: *fitness*, *dummy fitness* and *niche count* of individual $i$. This formula is used to calculate the fitness of individuals in a population. In order to calculate *niche count* for an individual $i$ (or $NC_i$), its niche boundaries need to be identified first. To calculate the niche dimensions in a given population, the concept of *niche cubicle* proposed by Hyun *et al.* [10] was adopted. A niche cubicle for an individual is a rectangular region around the individual. Dimensions of the niche cubicle in a problem having $m$ objectives are computed as follows:

$$\sigma_{lg} = \frac{Max_{lg} - Min_{lg}}{\sqrt[m]{PopSize}}, l = 1, \ldots, m \tag{7}$$

where $Max_{lg}$ and $Min_{lg}$ are maximum and minimum of the $l$-th objective function at generation $g$. The niche size is calculated at every generation. A solution located in a less dense cubicle is allowed to have a higher probability to survive in the next generation. After sharing, these nondominated individuals are ignored temporarily to process the rest of the population in the same way to identify individuals for the second nondominated frontier. These nondominated solutions are then assigned a new dummy fitness value that is kept smaller than the minimum shared dummy fitness of the previous frontier. In other words, dummy fitness value of the individuals in a given frontier $F^k$ must satisfy this condition: $DF_k < min(f_j), j \in F^{k-1}$. The dummy fitness values are then shared and this process is continued until the entire population is classified into several frontiers and individuals are assigned fitness values.

### 4.2 Selection

Individuals at each generation are selected according to their *count share* ($C$) which is calculated as follows:

$$C_i = f_i \times n;\ i = 1, \ldots, n \tag{8}$$

where $n$ denotes the *Population Size*. Several copies of a given individual might be selected for the *mating pool*. The contribution of individual $i$ to the mating pool is determined by the integer part of $C_i$. The remaining individuals up to the *PopSize* are selected at random from the $\{EliteSet\}$. As an example, consider a population of three individuals with $C_1 = 1.6$, $C_2 = 1.4$ and $C_3 = 0.7$. According to the selection scheme, one copy of individuals 1 and 2 along with an individual from $\{Elite\ Set\}$ constitute the mating pool.

### 4.3 Crossover

Part of selected individuals of *mating pool* are recombined using The *order crossover* [18] according to *Crossover Rate*. The rest of individuals are copied (*i.e.* reproduced) to the next generation.

### 4.4 Diversification

In order to diversify the population, *inversion* and *insertion* operators are implemented.

The *inversion* operator, inverts the order of jobs in a randomly chosen part of the given individual. Individuals are chosen for *inversion* according to *Inversion Rate*. Once an individual is chosen for inversion, it may undergo several successive inversions whose number is determined by *Inversion Numbers*.

The *insertion* operator mutates a given individual by picking a single *gene* at random from the position $i$ and inserting it in a random position $j$ along the *chromosome* pushing forward (backward) the segment in between $i$ and $j$ if $i > j$ ($j > i$). The *insertion* operator is applied to all individuals in a population at least once. However, the number of times that this operator is applied on a solution is controlled by *Insertion Number*.

The offspring produced by either operator in the diversification stage, are compared with their parents. The parent is replaced if dominated by its offspring. However, a dominated offspring is still given a chance to replace its parent.

The probability for accepting a dominated offspring, starting from 1.0, is decreased exponentially over the generations to improve convergence. The probability of accepting a dominated offspring resulted via *inversion* or *insertion* operators at a given time $t$ is denoted by $P(A)$ and is calculated using this formula:

$$P(A) = exp\left(\frac{-t}{t_{max} - t}\right),\ t = 0, \ldots, t_{max} \tag{9}$$

where $t_{max}$ is the maximum execution (CPU) time. The idea behind this formula is inspired by the annealing process of simulated annealing in order to improve convergence.

## 5 The MOSA Approach

Simulated annealing is an approach that can provide near-optimal solutions to combinatorial optimization problems. Kirkpatrick *et al.* [11] and Eglese [7] do provide fundamental descriptions of simulated annealing in addition to informative examples. SA has been applied to a vast number of single objective optimization problems over the last two decades. It has also been applied as a tool for multiobjective optimization problems in some applications, for instance by Suresh and Mohanasundaram [27] and Loukil *et al.* [15]. Readers for review of approaches to multiobjective optimization by means of SA may refer to Suman and Kumar [26], Nam and Park [19], and Czyzak and Jaszkiewicz [5].

In this section, we explain the algorithmic steps of the MOSA heuristic in search for Pareto-optimal solutions. The MOSA heuristic is guided by the temperature level, $T$, and the cooling rate, $CR$ up to the freezing temperature, $TF$. Major steps of the proposed MOSA approach are described in Algorithm 2.

```
input  : Search Parameters
output: A Nondominated Frontier
Initialize search parameters;
Generate initial {Elite Set};
Randomly select current solution ∈ {Elite Set};
Let time counter t = 0;
while T > TF and t < t_max do
    for i=1 to Iterations do
        Generate test solution from current solution;
        Compare test solution with {Elite Set};
        Refine {Elite Set};
        if test solution Is Not dominated then
            {Elite Set} = {Elite Set} ∪ test solution;
            Let current solution = test solution;
        end
        else if Metropolis criterion = true then
            Let current solution = test solution;
        end
    end
    Let T = T × CR;
    Let t = t + 1;
end
Report {Elite Set};
```

**Algorithm 2**: Pseudocode of the MOSA approach

### 5.1 Neighborhood Generation

At a given temperature level $T$, a neighboring solution to a current solution is generated via either *inversion* or *insertion* operators, introduced in previous section. The probabilties of *inversion* and *insertion* are complementary, i.e.:

$$P(inversion) = 1 - P(insertion) \tag{10}$$

wherein the probability of inversion is calculated as follows:

$$P(inversion) = \frac{InversionRate \times T}{T_1} \tag{11}$$

This formula gives more chance to inversion to be selected at the beginning of the annealing process; whilst insertion is given more chance as the algorithm converges to the final solutions.

### 5.2 Metropolis Criterion

The *Metropolis* criterion states the probability of accepting a dominated solution as:

$$P(A) = exp\left(\frac{-\Delta E}{K_b T}\right) \tag{12}$$

where:

$$\Delta E = \underset{j}{Max}\left[\underset{i}{Min}\left(\frac{d_{i,j}}{f_{i,j}}\right)\right] \tag{13}$$

and:

$$K_b = \frac{-\Delta E_1}{T_1 ln\left(P(A)_1\right)} \tag{14}$$

Moreover, $\Delta E$ is the minimum distance between a *test solution* and its associated dominating solutions in nondominated frontier, $d_{i,j}$ denotes the distance between the $i$-th objective value of test solution and that of the dominating solution $j$, and $f_{i,j}$ represents the $i$-th objective value of the dominating solution $j$. The value $K_b$ is referred to as the *Boltzman* constant and reflects the probability of accepting a dominated solution (or $P(A)_1$) with $\Delta E_1$ distance from the furthest dominating solution in the current nondominated frontier at the initial temperature $T_1$. This value gives the user some control over the probability of dominated solutions being accepted.

The probability of accepting a dominated solution is a function of both temperature of the system ($T$) and the distance between the dominated solution and the current nondominated frontier ($\Delta E$). Such a distance measure in the proposed MOSA could be considered as the distance between the worse solution and the current solution in a single objective SA. In fact, current nondominated frontier in the MOSA acts the same as the *current solution* in a single objective SA as a reference. As the temperature decreases, the probability of accepting worse moves decreases. Obviously at $T = 0$, no worse move is accepted. $P(A)_1$, $\Delta E_1$ and $T_1$ are among the parameters of the MOSA and need to be determined in the parameter setting stage.

## 6 Computational Experiments

In this research we examine firstly the effect of the initializing procedure on the performance of the proposed MOMHs. Then we compare performance of the two proposed algorithms namely MOGA and MOSA to verify if there is any difference between them. Finally, two problem specific features of the problem sets, i.e. density and balance ness of the problem instances are tested as to their effect on the performance of the preferred solution technique.

### 6.1 Initialization

In order to verify the impact of initial *Elite Set* on the final results of the solution techniques, an initializing heuristic procedure was selected from Mansouri [4] which is described in Algorithm 3.

```
input  : Search Parameters
output: A Nondominated Frontier
while Stopping condition not met do
    Let {Elite Set} = ∅;
    Initialize the sequence S = ∅;
    while A1 ∪ A2 ≠ ∅ do
        Randomly select i ∈ A1 ∪ A2;
        for all (i, x)[or(x, i)] ∈ B do
            Let S = S ∪ (i, x)[or(x, i)];
        end
        Let A1 ∪ A2 = {A1 ∪ A2} \ i;
    end
    Let {Elite Set} = {Elite Set} ∪ S;
    Refine {Elite Set};
end
Report {Elite Set};
```

**Algorithm 3**: Pseudocode of the Initializing Heuristic

The underlying idea of this heuristic procedure is to try to minimize the number of dual setups (or global changeovers as noted in [1]) in switching from one job to another one while keeping a balanced level of setups paid on each machine.

Two sets of experiments were conducted using MOGA and MOSA for the largest problem with optimal objective values and also each set was run for 20 times. In the first set, randomly generated initial *Elite Set*s were used while in the second set, the above initializing procedure was used. The results of these experiments show that the application of the initializing procedure has not a major positive impact on the quality nor the convergence speed of the algorithms. For both MOGA and MOSA aproaches, the average difference between the quality of the final solutions with and without initializing heuristic was just 0.002.

One explanation for this finding might be that the proposed initializing procedure only favors the first objective. In addition, it seems that larger Pareto-optimal frontiers in the current research compared to that of a previous study [17] could be another reason why the initializing procedure does not contribute to the MOMHs in this research. The initializing procedure forces the MOMHs to start search from a solution in the neighborhood of one extreme solution along the frontier. This would in turn forces the algorithm to spend too efforts to diversify the frontier in search for a good approximation of true Pareto-optimal frontier.

### 6.2 Test Problems

Two groups of test problems were generated to evaluate the performance of proposed solution approaches. The first group includes 11 small size problems with 9 to 38 jobs. For this group, the ideal solution $\Theta$ (Equation 4) could be defined for comparisons wherein, optimal values for the second objective ($LB_2$) could be calculated in a reasonable CPU time using CLPEX solver. The second group includes 32 larger problems with up to 2560 jobs. For this group, the $LB_2$ could not be found using optimization tools in a reasonable time and henceforth, the ideal solution $\Theta$ could not be defined.

### 6.3 Parameter Setting

Comprehensive experiments were done to find a good set of parameter values to make the solution approaches more efficient in finding true Pareto-optimal frontiers. Most of those experiments were applied to the largest problem whose optimal objective values were available. Finally, the following values were selected for MOGA: *Population Size* = 50, *Crossover Rate* = 0.05, *Insertion Number* = 1, *Inversion Rate* = 0.80, and *Inversion Number* = 5. Also selected values for MOSA parameters are: $\Delta E_1 = 0.01$, $P(A)_1 = 0.01$, $CR = 0.99999$, $T_1 = 10$, $TF = 1$, *Iterations* = 15 and *Inversion Rate* = 0.6.

One interesting observation made during the parameter setting, was on the effect of low *Crossover Rate* (i.e. 0.05) on performance of the MOGA. The other rates that were considered in our experiments include: 0.02, 0.50 and 0.90. The results show that the selected rate is the most efficient one from among the examined values.

### 6.4 Comparison Method

Two Quality Indices ($QI$) were defined for the two groups of test problems. For small size problems, the $QI$ is calculated with reference to lower bounds of the two objectives using the following formula:

$$QI = 1 - Min\left(\frac{f_1 - LB_1}{LB_1}, \frac{f_2 - LB_2}{LB_2}\right) \quad (15)$$

For large size problems, where $LB_2$ could not be found, the union of the final frontiers of MOGA and MOSA is used as the reference set (called $R$) discarding dominated solutions. For a given frontier $F$, the $QI$ is calculated as follows:

$$QI = \frac{\sum_{j \in G} \underset{i \in R}{Min}\left\{\frac{(f_1^j - f_1^i) + (f_2^j - f_2^i)}{f_1^j + f_2^j}\right\}}{|G|} \quad (16)$$

where $f_1^x$ ($f_2^x$) denote the first (second) objective value of the solution $x$ and $|G|$ represents cardinality of the set $G$.

| | MOGA | MOSA |
|---|---|---|
| Mean | 0.930 | 0.933 |
| Variance | 0.002 | 0.003 |
| Observations | 14 | 14 |
| *Pearson* correlation | 0.963 | |
| Hypothesized mean difference | 0 | |
| Degree of freedom | 13 | |
| $t$ Stat | -0.710 | |
| $t$ Critical one tail | 1.771 | |

**Table 8** $t$-test: paired two sample for $QI$ on small problems ($\alpha = 0.05$). The results indicate that there is no significant difference between the two algorithms in terms of $QI$ when applied to small problems. This has been concluded from: ($t$ Stat) < ($t$ Critical one tail). Therefore it could be stated that at $\alpha = 0.05$ level of confidence, both MOGA and MOSA perform equivalently on small problems

### 6.5 Hardware and Software

The MOGA and MOSA algorithms were coded in C++ and executed on a Pentium III, 800 MHz processor with 128 MB RAM. The ILOG CPLEX (version 9.0) was used to solve the lower bounding model of $LB_2$ on a Power Edge 2650 with 2400 MHz Xeon processor and 4000 MB RAM.

### 6.6 Comparative Results

Both MOGA and MOSA algorithms were run 20 times on each problem for 60 seconds. Average (AVRG) and standard deviation (STDV) of these runs for small problems are reported in Table 5. Large problems are categorized in two group as balanced and unbalanced whose results are reported in Tables 6 and 7, respectively. A balanced problem is characterized by equal number of attributes on both machines. The CPLEX CPU times to find the optimal solution ($LB_2$) for small size problems are also reported in Table 5. It should be noted here that although CPLEX was run on a more powerful machine than the one used by MOGA and MOSA, it was not able to find the optimal solution for a 40-job problem after two days. As such, the largest problem with optimal solution ($LB_2$) contains 38 jobs in this table.

### 6.7 Comparisons on small problems

Table 5 shows that the $QI$ indices for MOGA and MOSA on small problems are close. The average $QI$ for MOGA and MOSA for this group are 0.930 and 0.933, respectively. To investigate whether this difference is significant, a $t$-test was made at $\alpha = 0.05$ level of significance whose results are presented in Table 8. It reveals that the performance of the two algorithms on small problems is not significantly different.

Another $t$-test was also made at the same level of significance to investigate the difference of the two algorithms on the size of their final frontiers as a measure

| Problem | | | CPLEX CPU | *Quality index* (*QI*) | | | | *Frontier Size* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | MOGA | | MOSA | | MOGA | | MOSA | |
| $n$ | $\|A_1\|$ | $\|A_2\|$ | Time (sec) | AVRG | STDV | AVRG | STDV | AVRG | STDV | AVRG | STDV |
| 8 | 4 | 4 | 1.06 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 |
| 8 | 4 | 5 | 0.97 | 0.995 | 0.000 | 0.995 | 0.000 | 3.000 | 0.000 | 3.000 | 0.000 |
| 9 | 5 | 5 | 1.72 | 0.965 | 0.000 | 0.965 | 0.000 | 4.000 | 0.000 | 4.000 | 0.000 |
| 9 | 4 | 6 | 0.75 | 0.920 | 0.000 | 0.920 | 0.000 | 3.000 | 0.000 | 3.000 | 0.000 |
| 10 | 5 | 5 | 2.81 | 0.916 | 0.000 | 0.916 | 0.000 | 3.000 | 0.000 | 3.000 | 0.000 |
| 10 | 4 | 7 | 3.31 | 0.955 | 0.000 | 0.955 | 0.000 | 2.000 | 0.000 | 2.000 | 0.000 |
| 11 | 6 | 6 | 4.53 | 0.948 | 0.000 | 0.948 | 0.000 | 3.000 | 0.000 | 3.000 | 0.000 |
| 11 | 3 | 9 | 7.70 | 0.958 | 0.000 | 0.959 | 0.003 | 2.000 | 0.000 | 2.300 | 0.470 |
| 12 | 6 | 6 | 19.09 | 0.916 | 0.210 | 0.953 | 0.009 | 3.350 | 0.489 | 3.400 | 0.754 |
| 22 | 17 | 13 | 1679 | 0.968 | 0.010 | 0.964 | 0.011 | 3.100 | 1.165 | 3.300 | 0.979 |
| 28 | 17 | 13 | 4142 | 0.885 | 0.012 | 0.876 | 0.010 | 7.450 | 1.572 | 7.300 | 1.895 |
| 30 | 19 | 20 | 19000 | 0.892 | 0.204 | 0.921 | 0.019 | 7.300 | 2.055 | 7.250 | 1.251 |
| 35 | 19 | 20 | 33182 | 0.866 | 0.010 | 0.859 | 0.013 | 8.053 | 2.368 | 8.100 | 1.447 |
| 38 | 19 | 20 | 16056 | 0.837 | 0.015 | 0.826 | 0.015 | 8.000 | 2.224 | 8.100 | 1.210 |
| Average | | | | 0.930 | | 0.933 | | | | | |

**Table 5** Comparative results on small problems.

| Problem | | | | MOGA *QI* | | MOSA *QI* | |
|---|---|---|---|---|---|---|---|
| $n$ | $\|A_1\|$ | $\|A_2\|$ | Density | AVRG | STDV | AVRG | STDV |
| 40 | 20 | 20 | 0.1 | 0.989 | 0.003 | 0.986 | 0.005 |
| 80 | 20 | 20 | 0.2 | 0.986 | 0.004 | 0.984 | 0.008 |
| 120 | 20 | 20 | 0.3 | 0.983 | 0.005 | 0.977 | 0.009 |
| 160 | 20 | 20 | 0.4 | 0.986 | 0.006 | 0.975 | 0.009 |
| 90 | 30 | 30 | 0.1 | 0.983 | 0.005 | 0.977 | 0.009 |
| 180 | 30 | 30 | 0.2 | 0.990 | 0.005 | 0.983 | 0.005 |
| 270 | 30 | 30 | 0.3 | 0.987 | 0.005 | 0.982 | 0.005 |
| 360 | 30 | 30 | 0.4 | 0.991 | 0.004 | 0.986 | 0.004 |
| 250 | 50 | 50 | 0.1 | 0.988 | 0.005 | 0.978 | 0.005 |
| 500 | 50 | 50 | 0.2 | 0.989 | 0.006 | 0.979 | 0.007 |
| 750 | 50 | 50 | 0.3 | 0.992 | 0.005 | 0.991 | 0.002 |
| 1000 | 50 | 50 | 0.4 | 0.995 | 0.001 | 0.996 | 0.002 |
| 640 | 80 | 80 | 0.1 | 0.994 | 0.002 | 0.965 | 0.012 |
| 1280 | 80 | 80 | 0.2 | 0.994 | 0.002 | 0.982 | 0.009 |
| 1920 | 80 | 80 | 0.3 | 0.994 | 0.002 | 0.988 | 0.006 |
| 2560 | 80 | 80 | 0.4 | 0.989 | 0.003 | 0.988 | 0.006 |
| Average | | | | 0.989 | | 0.982 | |

**Table 6** Comparative results on large balanced problems

| Problem | | | | MOGA *QI* | | MOSA *QI* | |
|---|---|---|---|---|---|---|---|
| $n$ | $\|A_1\|$ | $\|A_2\|$ | Density | AVRG | STDV | AVRG | STDV |
| 22 | 17 | 13 | 0.1 | 0.977 | 0.009 | 0.971 | 0.011 |
| 42 | 19 | 11 | 0.2 | 0.895 | 0.003 | 0.984 | 0.006 |
| 48 | 7 | 23 | 0.3 | 0.978 | 0.011 | 0.968 | 0.011 |
| 80 | 10 | 20 | 0.4 | 0.966 | 0.015 | 0.954 | 0.023 |
| 90 | 27 | 33 | 0.1 | 0.985 | 0.005 | 0.976 | 0.011 |
| 160 | 20 | 40 | 0.2 | 0.984 | 0.007 | 0.977 | 0.013 |
| 263 | 35 | 25 | 0.3 | 0.979 | 0.010 | 0.964 | 0.011 |
| 312 | 41 | 19 | 0.4 | 0.986 | 0.005 | 0.979 | 0.004 |
| 233 | 37 | 64 | 0.1 | 0.990 | 0.004 | 0.984 | 0.003 |
| 495 | 45 | 55 | 0.2 | 0.992 | 0.004 | 0.984 | 0.003 |
| 630 | 70 | 30 | 0.3 | 0.991 | 0.004 | 0.989 | 0.004 |
| 968 | 48 | 59 | 0.4 | 0.989 | 0.006 | 0.987 | 0.006 |
| 562 | 72 | 78 | 0.1 | 0.992 | 0.003 | 0.952 | 0.013 |
| 1109 | 84 | 66 | 0.2 | 0.995 | 0.002 | 0.967 | 0.008 |
| 1361 | 42 | 108 | 0.3 | 0.994 | 0.002 | 0.981 | 0.008 |
| 2000 | 50 | 100 | 0.4 | 0.993 | 0.001 | 0.992 | 0.006 |
| Average | | | | 0.980 | | 0.976 | |

**Table 7** Comparative results on large unbalanced problems

| | MOGA | MOSA |
|---|---|---|
| Mean | 4.161 | 4.196 |
| Variance | 5.930 | 5.790 |
| Observations | 14 | 14 |
| *Pearson* correlation | 0.999 | |
| Hypothesized mean difference | 0 | |
| Degree of freedom | 13 | |
| $t$ Stat | -1.228 | |
| $t$ Critical one tail | 1.771 | |

**Table 9** $t$-test: paired two sample for *Frontier Size* on small problems ($\alpha = 0.05$). The results indicate that there is no significant difference between the two algorithms in terms of *Frontier Size* when applied to small problems. This has been concluded from: ($t$ Stat) < ($t$ Critical one tail). Therefore it could be stated that for small problems, at $\alpha = 0.05$ level of confidence, diversity of both MOGA and MOSA is equivalent

| | MOGA | MOSA |
|---|---|---|
| Mean | 0.985 | 0.979 |
| Variance | 0.000 | 0.000 |
| Observations | 32 | 32 |
| *Pearson* correlation | 0.106 | |
| Hypothesized mean difference | 0 | |
| Degree of freedom | 31 | |
| $t$ Stat | 1.731 | |
| $t$ Critical one tail | 1.696 | |

**Table 10** $t$-test: paired two sample for $QI$ on large problems ($\alpha = 0.05$). The fact that ($t$ Stat > $t$ Critical one tail) implies that the difference between the two algorithms is significant. Hence, it is concluded that at $\alpha = 0.05$ level of confidence, MOGA performs better than MOSA on large problems

of diversity. The results, as shown in Table , show that there is no significance difference between MOGA and MOSA in this respect at = 0.05 level of significance. In other words, both MOGA and MOSA perform equally on small size problems.

### 6.8 Comparisons on large problems

To investigate performance of the two algorithms on large size problems, the following research questions need to be addressed:

*i.* Is there any difference, in terms of the quality of the solutions, between the two algorithms on large size problems? If yes, which algorithm performs better?
*ii.* Is performance of the algorithms influenced by features of the problems, i.e., balance ness and density?

In order to answer the first question, a $t$-test is made at $\alpha = 0.05$ level of significance whose results are given in Table 10. According to this table, the average $QI$ metric of MOGA is 0.985 while the same figure for MOSA is 0.979. The test reveals that this difference is significant. In other words, it could be stated that at $\alpha = 0.05$ level of confidence, MOGA performs better than MOSA on large size problems.

Concerning the second question, two ANOVA tests were made for MOGA and MOSA separately to investigate whether their performance is influenced by the balance of attributes on the two machines and density of jobs in the attribute matrix of the problems. Each test was made at $\alpha = 0.05$ and $\alpha = 0.10$ levels of significance. The results for MOGA and MOSA are presented in Tables 11 and 12, respectively. The results of experiments at $\alpha = 0.05$ show that performance of both MOGA and MOSA are not influenced by neither balance nor density of the problems. The same result was observed for MOGA at $\alpha = 0.10$ as shown in Table 11. However, the experiments at $\alpha = 0.10$ show that performance of MOSA is influenced by balance of the attributes on two machines in different problems. The result at $\alpha = 0.10$ level of significance shown in Table 12 together with the result of experiments reported in Tables 6 and 7 indicate that MOSA performs better on balanced instances. Again at this level of significance, the results show no evidence as to the impact of density on performance of MOSA.

## 7 Conclusion

We address a two machine flowshop scheduling problem in which each job is featured by two attributes (called attributes 1 and 2) which makes it different from other jobs. A setup need to be done on the first (second) machine if two consecutive jobs have different attribute 1 (2). The scheduling objectives are the minimization of the number of setups as well as minimization of makespan. It was shown that these objectives are conflicting so they cannot be optimized at the same time. So the set of Pareto-optimal solutions need to be found and presented to the production planner so he can select the appropriate schedule based on his preferences.

The problem is $\mathcal{NP}$-hard so exact optimization tools are not applicable for large size problems. Two multi-objective metaheuristic algorithms based on genetic algorithms and simulated annealing, namely MOGA and MOSA were proposed. Statistical experimentations show that both algorithms are equally performing on small size problems while MOGA performs better than MOSA on large size instances.

Moreover, the effect of density and balance of the attributes matrix, as two problem specific features were investigated for their possible influence on performance of the two algorithms through ANOVA tests. The results show no evidence as to the impact of density on performance of the algorithms at $\alpha = 0.05$ level of significance. In the meantime, an experiment at $\alpha = 0.10$ show that performance of the MOSA approach is influenced by the balance of problems.

Application of other metaheuristics to the above problem besides application of the proposed solution techniques to the problems with different set of criteria, e.g. due date based objectives provide a broad area for further research.

| Source of variation | SS | DF | MS | $F$(test) | $F$-critical ($\alpha = 0.05$) | $F$-critical ($\alpha = 0.10$) |
|---|---|---|---|---|---|---|
| Balance of attributes on two machines | 0.001 | 1 | 0.001 | 1.979 | 4.260 | 2.927 |
| Density of jobs in the attribute matrix | 0.000 | 3 | 0.000 | 0.495 | 3.009 | 2.327 |
| Interactions | 0.000 | 3 | 0.000 | 0.571 | 3.009 | 2.327 |
| Within combinations | 0.008 | 24 | 0.000 | | | |
| Total | 0.001 | 23 | | | | |

SS: *Sum of squares*, DF: *Degrees of Freedom*, MS: *Mean square*

**Table 11** ANOVA test for MOGA on large problems. The results indicate that performance of MOGA is not influenced by neither balance nor density at $\alpha = 0.05$ and $\alpha = 0.10$ due to the fact that $F(test) < Fcritical$ at these levels.

| Source of variation | SS | DF | MS | $F$(test) | $F$-critical ($\alpha = 0.05$) | $F$-critical ($\alpha = 0.10$) |
|---|---|---|---|---|---|---|
| Balance of attributes on two machines | 0.000 | 1 | 0.000 | 3.385 | 4.260 | 2.927 |
| Density of jobs in the attribute matrix | 0.000 | 3 | 0.000 | 1.006 | 3.009 | 2.327 |
| Interactions | 0.000 | 3 | 0.000 | 1.024 | 3.009 | 2.327 |
| Within combinations | 0.003 | 24 | 0.000 | | | |
| Total | 0.002 | 23 | | | | |

SS: *Sum of squares*, DF: *Degrees of Freedom*, MS: *Mean square*

**Table 12** ANOVA test for MOSA on large problems. The results indicate that performance of MOSA is not influenced by neither balance nor density at $\alpha = 0.05$ for $F(test) < Fcritical$ at this level. However, the results at $\alpha = 0.10$ indicate that performance of MOSA is influenced by balance of the problem.

## 8 Acknowledgment

The authors would like to thank the anonymous referees for their constructive comments which improved the quality and presentation of this paper. The work of Afshin Mansouri was partially supported by UK EPSRC under grant EP/D050863/1.

## References

1. A. Agnetis, P. Detti, C. Meloni, and D. Pacciarelli. Set-up coordination between two stages of a supply chain. *Annals of Operations Research*, 107(1-4):15–32, 2001.
2. F. D. Chou and C. E. Lee. Two-machine flowshop scheduling with bicriteria problem. *Computers & Industrial Engineering*, 36(3):549–564, 1999.
3. C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.
4. Y. Collette and P. Siarry. *Multiobjective Optimization: Principles and Case Studies*. Springer, 2004.
5. P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.
6. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.
7. R. W. Eglese. Simulated annealing: A tool for operational research. *European Journal of Operational Research, EJOR, Vol. 46, :271-281*, 1990.
8. Y. Gajpal, C. Rajendran, and H. Ziegler. An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. *The International Journal of Advanced Manufacturing Technology*, 30(5-6):416–424, 2006.
9. J. N. D. Gupta and W. P. Darrow. The Two-Machine Sequence Dependent Flowshop Scheduling Problem. *European Journal of Operational Research*, 24(3):439–446, 1986.
10. C. J. Hyun, Y. Kim, and Y. K. Kim. A Genetic Algorithm for Multiple Objective Sequencing Problems in Mixed Model Assembly Lines. *Computers & Operations Research*, 25(7/8):675–690, 1998.
11. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 13 1983.
12. Y. H. Lee and J. W. Jung. New heuristics for no-wait flowshop scheduling with precedence constraints and sequence dependent setup time. In O. Gervasi, M. L. Gavrilova, V. Kumar, A. Laganà, H. P. Lee, Y. Mun, D. Taniar, and C. J. K. Tan, editors, *ICCSA (4)*, volume 3483 of *Lecture Notes in Computer Science*, pages 467–476. Springer, 2005.
13. S.-W. Lin and K.-C. Ying. Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics. *The International Journal of Advanced Manufacturing Technology*, to appear.
14. R. Logendran, N. Salmasi, and C. Sriskandarajah. Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. *Computers & Operations Research*, 33:158–180, 2006.
15. T. Loukil, J. Teghem, and P. Fortemps. A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research*, 179(3):709–722, 2007.
16. C. Low, T.-H. Wu, and C.-M. Hsu. Mathematical modelling of multi-objective job shop scheduling with dependent setups and re-entrant operations. *The International Journal of Advanced Manufacturing Technology*, 27(1-2):181–189, 2005.
17. S. A. Mansouri. Coordination of setups between two stages of a supply chain using multi-objective genetic algorithms. *International Journal of Production Research*, 43(15):3163–3180, 2005.
18. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed.* Artificial Intelligence. Springer-Verlag, Berlin, 1996.
19. D. Nam and C. H. Park. Multiobjective Simulated Annealing: A Comparative Study to Evolutionary Algorithms. *International Journal of Fuzzy Systems*, 2(2):87–97, 2000.
20. T. Pasupathy, C. Rajendran, and R. K. Suresh. A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. *The*

*International Journal of Advanced Manufacturing Technology*, 27(7-8):804–815, 2006.

21. S. G. Ponnambalam, H. Jagannathan, M. Kataria, and A. Gadicherla. A TSP-GA multi-objective algorithm for flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 23(11-12):909–915, 2004.
22. S. D. Prasad, O. K. Chetty, , and C. Rajendran. A genetic algorithmic approach to multi-objective scheduling in a kanban-controlled flowshop with intermediate buffer and transport constraints. *The International Journal of Advanced Manufacturing Technology*, 29(5):564–576, 2006.
23. S. Pugazhendhi, S. Thiagarajan, C. Rajendran, and N. Anantharaman. Generating non-permutation schedules in flowline-based manufacturing sytems with sequence-dependent setup times of jobs: a heuristic approach. *The International Journal of Advanced Manufacturing Technology*, 23(1-2):64–78, 2004.
24. N. Salmasi. *Multi-Stage Group Scheduling Problems with Sequence Dependent Setups*. PhD thesis, Oregon State University, 2005.
25. N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.
26. B. Suman and P. Kumar. Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *Journal of the Operational Research Society*, 57:1143–1160, 2006.
27. R. K. Suresh and K. M. Mohanasundaram. Pareto archived simulated annealing for job shop scheduling with multiple objectives. *The International Journal of Advanced Manufacturing Technology*, 29(1-2):184–196, 2006.
28. F. Szidarovsky, M. E. Gershon, and L. Dukstein. *Techniques for multiobjective decision making in systems management*. Elsevier, New York, 1986.
29. V. T'kindt and J.-C. Billaut. *Multicriteria Scheduling. Theory, Models and Algorithms*. Springer, Berlin, 2006.
30. V. T'kindt, N. Monmarché, F. Tercinet, and D. Laügt. An Ant Colony Optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research*, 142(2):250–257, Oct. 2002.
31. X. Wang and T. C. E. Cheng. Heuristics for two-machine flowshop scheduling with setup times and an availability constraint. *Computers & Operations Research*, 34(1):152–162, 2007.
32. X. Zhu and W. E. Wilhelm. Scheduling and lot sizing with sequence-dependent setups: A literature review. *IIE Transactions*, 38:987–1007, 2006.