Computers & Operations Research 37 (2010) 199 – 212

Contents lists available at ScienceDirect

Computers & Operations Research

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

Total flow time minimization in a flowshop sequence-dependent group scheduling problem

Nasser Salmasi[a], Rasaratnam Logendran[b,*], Mohammad Reza Skandari[a]

[a]Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
[b]School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, 204 Rogers Hall, Corvallis, OR 97331, USA

ARTICLE INFO

Available online 6 May 2009

Keywords:
Sequence dependent scheduling
Group scheduling
Branch and price
Metaheuristics
Lower bound

ABSTRACT

We have developed a mathematical programming model for minimizing total flow time of the flow shop sequence dependent group scheduling (FSDGS) problem, typically classified as $F_m|fmls, S_{plk}, prmu|\sum C_j$. As the problem is shown to be strongly NP-hard, a tabu search (TS) algorithm as well as a hybrid ant colony optimization (HACO) algorithm have been developed to heuristically solve the problem. A lower bounding (LB) method based on the Branch-and-Price algorithm is also developed to evaluate the quality of the metaheuristic algorithms. In order to compare the performance of metaheuristic algorithms, random test problems, ranging in size from small, medium, to large are created and solved by both the TS and the HACO algorithms. A comparison shows that the HACO algorithm has a better performance than the TS algorithm. The results of the heuristic algorithms are also compared with the results of the LB method to evaluate the quality of the solutions. The LB method presented in this paper can be generalized to solve the FSDGS problem with other objective functions.

1. Introduction

In this paper it is assumed that several jobs are assigned to a flow shop in order to be processed. Each machine needs a specific setup to process a job. The required setup time for each machine to process a job depends on the previously processed job on that machine. In this case, grouping the jobs in families (groups) in a way that the jobs of a group need the same setup on machines will increase the efficiency of the production. This problem is known as flow shop sequence dependent group scheduling (FSDGS) problem.

There are several applications of FSDGS problems in industry practice. Paint shop of an automobile manufacturer is an example of such problems. Assume that the paint shop is switching to paint a batch of automobiles in green after painting a batch with another color. The required setup time (including cleaning the nozzles, area, etc.) to start painting the new batch in green is dependent on the previously used color. Clearly, switching from white to green is much easier than switching from black to green since it needs less cleaning activities. Another application which is also observed in automobile industry is the press shop. Items are processed by a series of press machines in several stages in a row. Because of issues such as weight of items and safety, the items are transferred between press machines by conveyers. All parts should be processed in the same order on all machines. The required setup time for each press is dependent upon the part processed previously on the machine. A few studies reported in the recent past also show its applicability in electronics manufacturing. Schaller et al. [1] and Pinedo [2] discussed an industrial-case of the proposed problem in printed circuit boards (PCBs) in which a major setup is required to switch from a group of PCBs to another.

Cheng et al. [3], Allahverdi et al. [4], Zhu and Wilhelm [5], and Allahverdi et al. [6] performed a comprehensive literature review of scheduling problems related to this research. Schaller et al. [1] proposed an efficient heuristic algorithm as well as a lower bounding (LB) method to minimize the makespan of an FSDGS problem. Franca et al. [7] developed two algorithms based on genetic algorithm and memetic algorithm with local search to solve the FSDGS problems for minimization of the makespan criterion. Logendran et al. [8] developed a heuristic algorithm based on tabu search (TS) to solve the two-machine FSDGS problems by considering minimization of makespan. They also developed a lower bounding method to evaluate the quality of their heuristic algorithm. Hendizadeh et al. [9] developed a heuristic algorithm based on TS by applying the concept of "elitism" and the acceptance of worse moves from simulated annealing to improve the intensification and diversification strategies to solve the same problem. To the best of our knowledge, all of the previous research in FSDGS problems addresses only the

* Corresponding author. Tel.: +1 541 737 5239; fax: +1 541 737 2600.
E-mail addresses: nsalmasi@sharif.edu (N. Salmasi), Logen.Logendran@oregonstate.edu (R. Logendran), r.skandari@ie.sharif.edu (M.R. Skandari).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.04.013

minimization of the makespan criterion. In other words, there is no other paper that investigates into the FSDGS problems by considering minimization of total flow time (TFT) as the objective.

Based on Pinedo [2], the problem we investigate can be notated as $F_m|fmls, S_{plk}, prmu|\sum C_j$. The reasons for this notational choice and the assumptions of this research are:

- It is assumed that in a flow shop environment, g groups are assigned to a cell that has m machines (F_m).
- Each group ($p = 1, 2, \ldots, g$) includes b_p jobs ($fmls$).
- The setup time of a group (l) on each machine (k) depends on the immediately preceding group (p) that is processed on that machine (i.e., sequence dependent setup time (s_{plk})).
- The setup time of each group on each machine can be different.
- All jobs and groups are processed in the same sequence on all machines (permutation scheduling ($prmu$)). This is the only way to produce in some industries. For instance, if a conveyer is used to move jobs between machines, then all jobs should be processed in the same sequence on all machines.
- Each job of a group does not need a separate setup time. If it does, the required setup time is assumed to be included in its runtime.
- The setup process of a machine for a group can be started before a job that belongs to the group is available (anticipatory).
- The jobs belonging to each group should be processed without any preemption by other jobs of other groups (group technology assumption).

Garey et al. [10] proved that the multi-stage flow shop job-scheduling problem by considering minimization of TFT criterion is an NP-hard problem. A group scheduling problem can be easily reduced to a multi-stage flow shop job-scheduling problem by assuming zero group setup times and one job per each group. Based on this insight, it is easy to see that the problem investigated in this paper is easily reducible to the one already proven NP-hard. Thus, the fact that the proposed problem is NP-hard, follows immediately.

Since the research problem is NP-hard, heuristic algorithms are developed to solve industry size problems in a reasonable time. Previous research by Logendran et al. [8] has shown TS to be a promising technique for solving similar scheduling problems. Thus, a metaheuristic algorithm based on TS has been developed to solve the research problem. Also among the available heuristics, ant colony optimization (ACO) algorithm, for its capability of solving difficult combinatorial problems, is chosen to be compared with the TS algorithm because of its recent widespread favorable publicity.

We also propose a mathematical model to develop the lower bounding method based on the Branch-and-Price (B&P) algorithm to estimate the quality of the developed heuristic algorithms. Then, we compare the performance of the TS algorithm with the ACO algorithm to uncover any statistically significant difference that might exist between the two.

Heuristic algorithms are presented in Sections 2 and 3. In Section 4, the mathematical model is presented followed by the lower bounding method in Section 5. The test problem specifications are described in Section 6. Finally, the results, and the conclusions along with suggestions for future research are presented in Sections 7 and 8, respectively.

2. Tabu search

A two level TS algorithm is developed for the proposed research problem. The first (outside) level investigates to find the best sequence of groups. The second (inside) level investigates to find the sequence of jobs in each group based on the chosen group sequence by the first level. When a sequence of groups by the outside level is chosen, the second level finds the best sequence of jobs that belongs to each group by considering minimization of TFT. This is done for the inside search by moving from a sequence of jobs in a group sequence to another sequence of jobs in the same group sequence. The relationship between the outside and inside search is that once the outside search is performed to get a new group sequence, the search process is switched to inside search. The inside search is performed to find the best sequence of jobs in groups by considering the proposed group sequence by outside search. When the inside search stopping criteria are satisfied, the best found job sequence is considered. Then, the search returns back to the outside search. The outside search stops when one of the outside search stopping criteria is satisfied. The best found solution, comprises the sequence of groups and the sequence of jobs in each group that provides the best objective function value, is reported as the final solution. The specifications of the developed TS are as follows:

- The stopping criteria are either the specified number of local optima or the maximum number of iterations without improvement, whichever occurs first.
- A random sequence for groups and jobs that belong to each group is applied as an initial solution. The initial solution is considered as the first entry into the outside candidate list.
- During inside search, a neighborhood of a seed is generated by applying swap moves, i.e., changing the order of adjacent jobs that belong to a group. By changing the position of the last and the first job of a group, another neighbor can be generated. The neighbors for an outside seed are derived similar to the inside ones by applying swap moves.
- The tabu list (TL) is constructed based on the recent change in previous best solutions. The TL records these changes or moves in the order they are applied. If a restricted move has a better value than the best global value found so far, called the aspiration level, the tabu restriction is ignored. The best move, after filtering against TL and aspiration criterion, is compared with the current members of the candidate list. If the chosen neighbor does not belong to the current candidate list, it is selected for next perturbation and generation of new neighbor. Otherwise, the next best neighbor is chosen. This move is recorded into the TL.
- The search is terminated by satisfying one of the stopping criteria.
- The long-term memory is used to enhance the search. In this application, the attribute indicates the position of a group or a job within a group. The search is intensified by encouraging the explorations in the neighborhoods of the good solutions with frequently added attributes. This process is performed twice for both levels in this research.
- The TS algorithm parameters should be tuned based on the size of the problem. In order to tune the algorithm, extensive experimentation was performed to evaluate the parameter values. The empirical formulae for the value of parameters used for these research problems are presented in Appendix A. In some cases a formula for a range is generated and in some of them a value for a parameter in a range is offered. If a formula does not provide an integer value for a parameter, it is rounded down. For instance, if a problem with seven groups and 61 jobs is intended to be solved by the algorithm, the size of the TS algorithm should be as follows:

  Outside index list = $7 \times 10 = 70$.
  Outside number of iterations without improvement = 6.
  Outside tabu list size = 2.
  Inside index list = 7.
  Inside number of iterations without improvement = 8.
  Inside tabu list size = 6.

3. Hybrid ant colony optimization algorithm (HACO)

A hybrid algorithm, resulting from a heuristic inspired by NEH algorithm [11] and ant colony optimization algorithm, is developed for the research problem. The sequence of jobs is calculated by the NEH-inspired algorithm, while the sequence of groups is calculated based on a variant of ACO, termed ant colony system (ACS).

3.1. Sequence of groups

ACO is a population based construction metaheuristic, inspired by the foraging behavior of an ant species. The Argentine ants guide other members of the colony indirectly via dynamic modifications of their environment (trails of pheromones) to propose a better path from nest to food and vice versa, based on their collaborative experience. The more pheromone on a path, the higher is the likelihood of that path being followed by other ants. ACO exploits similar mechanism by deploying artificial ants (agents) to solve combinatorial problems.

Dorigo [12] introduced ACO by using Ant System to solve the Traveling Salesman Problem. Since ACS proposed by Dorigo and Gambardella [13] has shown to have better performance among available versions of ACO, it is used as a part of HACO in this research. The common mechanics behind every ACO algorithm are as follows:

1. A colony of (a number of) ants which constructs solutions independently.
2. The solution construction phase that works using pheromone values and heuristic information.
3. Local search that is applied around the best found solution(s) to improve the solution(s).
4. Updating pheromone values phase that updates the pheromone values based on the quality of the solutions found, so as to guide the algorithm toward promising solutions.

The following are the steps of ACO as pseudocode:

```
Set parameters, initialize pheromone trails
  **while** termination condition not met **do**
    Construct Ant Solutions
    Apply Local Search (*optional*)
    Update Pheromones
  **end while**
```

3.1.1. Solution construction

A group sequence solution is a permutation of groups $\pi_1 = (G_1, G_2, \ldots, G_g)$, in which the absolute positions of groups are also important. A solution can be constructed in two different ways:

(1) Constructing a sequence of groups by determining the relative order of processing them. For instance, the sequence $\pi_1 = \{G_1, G_2, \ldots, G_g\}$ is constructed as follows: G_1 is processed as the first group, G_2 is processed immediately after G_1, and so on.
(2) Defining g slots for g groups, and assigning the groups to the slots. For instance, in the sequence $\pi_1 = \{G_1, G_2, \ldots, G_g\}$, G_1 is assigned to the first slot, G_2 is assigned to the second slot, and so on.

While the two ways seem different, there is a one-to-one accordance between them in interpreting a solution.

3.1.2. Pheromone definition

Two different types of pheromone trails are developed based on the aforementioned construction ways:

(1) Pheromone trail type I τ^1_{pl}: desirability of processing group l immediately after group p, $p = 0, 1, \ldots, g$ $l = 1, 2, \ldots, g$.
(2) Pheromone trail type II τ^2_{ip}: desirability of processing group p in the ith slot of the sequence, p, $i = 1, 2, \ldots, g$.

To elaborate the distinction between the two pheromone trails and the usability of each, two important factors that affect the total flow time of the solutions to a given problem that result from the sequence of groups should be noticed. First the required time for processing jobs on machines (that include the setup time as a major part), and second the sequence of jobs. The setup times are sequence dependant, thus the relative positions of the groups determine the setup times and consequently affect the time required for processing jobs. On the other hand, the absolute position of the groups affects the sequence of jobs indirectly since the jobs belonging to each group should be processed consecutively without being preempted by jobs that belong to other groups.

The first trail type helps in determining the relative position of the groups, while the second one helps to determine the absolute position of the groups (see [14]). Incorporating the two trails in the algorithm alongside helps to move toward promising areas of solution space in a better way.

Although we noted the one-to-one accordance of the construction ways in the previous section, it is vital to acknowledge the difference of utilization of pheromone types during the construction phase. The high value of τ^1_{pl} proposes processing group l immediately after group p, while the high values of τ^2_{ip} proposes processing group p in the ith position of the sequence.

3.1.3. Heuristic information

To bias the construction algorithm toward promising areas, in addition to the pheromone mechanism, heuristic information is deployed in ACO. In ACO literature, heuristic information indicates an intuitive guess of the cost of adding a candidate solution component to the partial solution. For coordinating the direction of pheromone and heuristic information, usually the reverse of the guessed cost (or other related functions) is used instead. In this research, the heuristic information for each candidate group p in the ith step of the construction algorithm (or equally for the ith slot of the sequence) is defined as the differential flow time $\Delta flowtime_{ip}$ produced because of adding the candidate group to the partial sequence (1). Suppose that the partial sequence S_i^{Part} has been constructed until the ith step that has a total flow time of $flowtime(S_i^{Part})$; now if we add the candidate group p to this partial solution, we will have another partial solution $S_{i+1}^{Part} = S_i^{Part} \cup \{p\}$ (or a complete solution) that has a total flow time of $flowtime(S_{i+1}^{Part})$. So, we define heuristic information for the candidate group p in the ith step of construction as $1/(flowtime(S_{i+1}^{Part}) - flowtime(S_i^{Part}))$. In order to calculate the differential flow time, we need to know the sequence of jobs in the groups. Thus, the group sequencing algorithm needs job sequencing algorithm to perform and they go hand in hand.

$$\eta_{ip} = \frac{1}{flowtime(S_{i+1}^{Part}) - flowtime(S_i^{Part})}, \quad S_{i+1}^{Part} = S_i^{Part} \cup \{p\} \tag{1}$$

3.1.4. Stochastic model of solution construction

ACO constructs a solution by adding solution components to a null sequence, one after the other guided by a stochastic mechanism. In HACO, we use the following pseudorandom proportional rule, derived from ACS, to select the next group p to be added to the partial sequence S_i^{Part} from the list of not yet sequenced groups $N(S_i^{Part})$. For the ith group of the sequence, with δ_0 probability, the best group (in terms of pheromone intensity and heuristic information) is selected, and with $(1 - \delta_0)$ probability a random model identifies the next group:

$$p = \begin{cases} \arg \underset{\forall l \in N(S^{Part})}{\mathrm{Max}} & \{(\tau^1_{p'l})^{\alpha_1} \cdot (\tau^2_{il})^{\alpha_2} \cdot \eta^{\beta}_{il}\} \\ P & \text{otherwise} \end{cases} \tag{2}$$

$$\Pr_{ip} = \begin{cases} \dfrac{(\tau^1_{p'p})^{\alpha_1} \cdot (\tau^2_{ip})^{\alpha_2} \cdot \eta^{\beta}_{ip}}{\sum_{\forall l \in N(S^{Part})} (\tau^1_{pl})^{\alpha_1} \cdot (\tau^2_{il})^{\alpha_2} \cdot \eta^{\beta}_{il}} & \text{if } p \in N(S^{Part}_i) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In the above formulae p' is the last sequenced group, i stands for the construction step and P stands for the group selected via the random model for which the distribution function is presented in (3). This model assigns a probability to every group p in the ith step of the algorithm ($\Pr_{ip}$) proportional to the pheromone and heuristic information values. In this model, α_1, α_2, and β are parameters which determine the relative influence of the two pheromone trails and the heuristic information, respectively; or in other words they indicate the relative importance of sequencing group l immediately after group p (the first type of pheromone), the importance of sequencing group p in the ith position of the sequence (the second type of pheromone), and heuristic information importance, respectively.

3.1.5. Local search

Before updating pheromone and after construction of the solutions, a local search is applied to the best solution. The integral part of every local search algorithm is the definition of neighborhood. In HACO, three different neighborhoods are used to improve the quality of solutions, and the efficiency of each was tested:

(1) Swapping all possible pairs of groups (jobs).
(2) Removing a group (a job) and inserting it in all possible positions.
(3) Permutations of all triplets of groups (jobs).

In all scenarios, we tested a random scheme to add power to the local search, i.e., the order of candidate groups (jobs), pairs or triplet of them to undertake local search functions is set by random. For instance, for the first scenario, the order of pairs of groups (jobs) to be swapped in the given sequence is set by random.

3.1.6. Pheromone update

In order to guide the algorithm in ACS, two strategies are used to manipulate the pheromone: local and offline pheromone update. The former diversifies the search by decreasing the pheromone according to the selected candidates by each ant during construction phase (diversification strategy), and the latter intensifies the search toward promising areas at the end of the iteration by the best ant (intensification strategy). The model (4) is used to perform local pheromone update, while the model (5) is used to perform offline pheromone update:

$$\tau_{ab} \leftarrow (1-\varphi).\tau_{ab} + \varphi.\tau_0, \quad \varphi \in (0,1] \tag{4}$$

$$\tau_{ab} \leftarrow \begin{cases} (1-\rho)\cdot\tau_{ab} + \dfrac{1}{C^{BestAnt}} & \text{if best ant uses edge } (a,b) \\ \tau_{ab} & \text{otherwise} \end{cases} \tag{5}$$

The factor φ is the local pheromone decay coefficient, and τ_0 is the initial value of the pheromone, ρ is the offline pheromone, decay coefficient, and $C^{BestAnt}$ is the cost of the best solution. In the formulae, edge (a,b) has two different meanings according to each type of pheromone trail: for the first type, using edge (a,b) equals to sequencing group b after group a, while for the second type, using edge (a,b) means sequencing group b in the ath slot of the sequence. It is important to update the two noted types of pheromone values independently because of their different ways of interpreting a solution.

3.1.7. Initialization of the algorithm

To set the initial values of the pheromone and start from a more favorable region in the solution space, a greedy like algorithm is used as follows: in each step of solution construction, the best group (in terms of partial flow time, calculated after adding the candidates to the partial solution) is added until a complete solution is made. Then the initial value for the pheromone τ_0 is calculated as follows:

$$\tau_0 \leftarrow \frac{1}{g \cdot C^{GreedySolution}} \tag{6}$$

In (6), g stands for the total number of groups, $C^{GreedySolution}$ stands for the total flow time of the constructed greedy solution. After initializing the pheromone values, we take this greedy solution as *the best solution* and update pheromone values by using this solution to guide the future search.

Experiments show that the burden of calculating heuristic information during each step of construction is not cost effective and saved time can be used to explore the solution domain in a more efficient way. So we adopted β as zero. Also best results are achieved when adopting α_1, and α_2 as 2 and 1, respectively. This shows the superiority of importance of the relative position of groups. The best values of the parameters φ, ρ, and δ_0 are 0.9, 0.9, and 0.1, respectively. The best results are gained by taking the number of ants equal to 10.

The more extensive the neighborhood, the more effective is the local search, but more time will be spent; so a tradeoff between the power of local search and computational burden should be applied. Results show that only the two first definitions of neighborhood are cost effective, and adding the random scheme boosts the power of the local search noticeably.

3.2. Sequence of jobs in a group

For sequencing jobs inside groups, a heuristic inspired by NEH algorithm [11] is used. Because the original version of NEH is not iterative, and it was originally proposed for makespan objective, we made some changes to the NEH algorithm.

In the original NEH algorithm proposed for the makespan objective, the order of jobs to undertake the second and third steps of the algorithm is defined by sorting the jobs in terms of the sum of the processing times on the machines decreasingly. For that objective, sorting the jobs accordingly might be a good intuition, but for total flowtime objective it is not desirable. Moreover, the original NEH is not iterative and no new solution is constructed unless a new sequence of group is introduced. To counteract the two mentioned problems, the proposed algorithm sorts the job by random in the first step, instead. The steps of the proposed algorithm are as follows:

(1) Order jobs by random.
(2) Take the first two jobs and schedule them in order to minimize the partial flow time as if there were only these two jobs.
(3) For the remaining jobs, try to insert the jobs, one by one, at all possible positions, and select the position which minimizes the partial flow time until all jobs are sequenced.

Experiments show the superiority of the random scheme over the original one.

3.3. Terminating criteria

Experiments show that almost for all problems, the algorithm converges at a maximum of 30 s of CPU time. So, 30 s of CPU time was adopted as the terminating criterion.

4. Mathematical model

We develop a mixed integer linear programming (MILP) model based on the concept of "slots". It is assumed that there are slots for positioning of groups. Each group is assigned to only one slot and each slot is dedicated to receiving only one group. In real world problems, groups have different number of jobs. Because each group can be assigned to any slot, to facilitate the development of the mathematical model, in the model, it is assumed that every group has the same number of jobs, composed of real and dummy jobs. This number is equal to b_{max} which is the maximum number of *real* jobs in a group. If a group has fewer real jobs than b_{max}, the difference, i.e., b_{max}—*number of real jobs*, is assumed to be occupied by dummy jobs for that group. The parameters, decision variables, and the mathematical model are defined as follows:

Parameters:

g	number of groups	
b_p	number of jobs in group p, $p=1,2,\ldots,g$	
m	number of machines	
b_{max}	maximum number of jobs in groups	
t_{pjk}	$\begin{cases}\text{for real jobs; run time of job } j \text{ in group } p \text{ on machine } k\\ \text{for dummy jobs, } 0\end{cases}$	$p=1,2,\ldots,g$ $j=1,2,\ldots,b_{max}$
t'_{pjk}	$\begin{cases}\text{for real jobs; run time of job } j \text{ in group } p \text{ on machine } k\\ \text{for dummy jobs, } -M\end{cases}$	$k=1,2,\ldots,m$ M: a large number
S_{plk}	the setup time for group l on machine k if group p is the preceding group	$p=0,1,2,\ldots,g$ $l=1,2,\ldots,g\,(p\neq l)$ $k=1,2,\ldots,m$
T_{pk}	the summation of run times of jobs in group p on machine k	$T_{pk}=\sum_{j=1}^{b_p} t_{pjk}$

Decision variables:

X_{ijk}	the completion time of job j in ith slot on machine k	$i=1,2,\ldots,g$ $j=1,2,\ldots,b_{max}$ $k=1,2,\ldots,m$
W_{ip}	$\begin{cases}1 & \text{If group } p \text{ is assigned to slot } i\\ 0 & \text{Otherwise}\end{cases}$	$i,\ p=0,1,2,\ldots,g$
Y_{ijq}	$\begin{cases}1 & \text{If job } q \text{ is processed after job } j \text{ in slot } i\\ 0 & \text{Otherwise}\end{cases}$	$i=1,2,\ldots,g$ $j,\ q=1,2,\ldots,b_{max},\ j\neq q$
C_{ik}	the completion time of ith slot on machine k	$i=0,1,2,\ldots,g,\ k=1,2,\ldots,m$
O_{ik}	the setup time for a group assigned to slot i on machine k	$i=1,2,\ldots,g,\ k=1,2,\ldots,m$
A_{ipl}	$\begin{cases}1 & \text{If group } p \text{ is assigned to slot } i \text{ and group } l \text{ is}\\ & \quad\text{assigned to slot } i+1\\ 0 & \text{Otherwise}\end{cases}$	$i=0,1,2,\ldots,g-1$ $p=0,1,2,\ldots,g-1$ $l=1,2,\ldots,g,\ p\neq l$

The Model (*model* 1):

$$\text{Min } Z=\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} X_{ijm} \tag{7}$$

Subject to:

$$\sum_{i=1}^{g} W_{ip}=1, \quad p=1,2,\ldots,g \tag{8a}$$

$$\sum_{p=1}^{g} W_{ip}=1, \quad i=1,2,\ldots,g \tag{8b}$$

$$\sum_{p=0}^{g}\sum_{l=1}^{g} A_{ipl}=1, \quad i=0,1,2,\ldots,g-1\ \ (p\neq l) \tag{9}$$

$$A_{ipl}\le W_{ip}, \quad i=0,1,2,\ldots,g-1,\ \ p,l=1,2,\ldots,g,\ \ p\neq l \tag{10a}$$

$$A_{ipl}\le W_{(i+1)l}, \quad i=0,1,2,\ldots,g-1,\ \ p,l=1,2,\ldots,g,\ \ p\neq l \tag{10b}$$

$$O_{ik}=\sum_{p=0}^{g}\sum_{l=1}^{g} A_{(i-1)pl}S_{plk}, \quad i=1,2,\ldots,g,\ \ k=1,2,\ldots,m,\ \ p\neq l \tag{11}$$

$$X_{ijk}\ge C_{(i-1)k}+O_{ik}+\sum_{p=1}^{g} W_{ip}t_{pjk}{}' \tag{12}$$

$$i=1,2,\ldots,g,\ \ j=1,2,\ldots,b_{max},\ \ k=1,2,3,\ldots,m$$

$$X_{ijk}-X_{iqk}+MY_{ijq}\ge\sum_{p=1}^{g} W_{ip}t'_{pjk}, \quad \begin{matrix} i=1,2,\ldots,g, \\ j,q=1,2,\ldots b_{max}, \quad j<q \\ k=1,2,3,\ldots,m, \end{matrix} \tag{13}$$

$$X_{iqk}-X_{ijk}+M(1-Y_{ijq})\ge\sum_{p=1}^{g} W_{ip}t'_{pqk}, \quad M:\ a \text{ large number} \tag{14}$$

$$C_{i1}=C_{(i-1)1}+O_{i1}+\sum_{p=1}^{g} W_{ip}T_{p1}, \quad i=1,2,3,\ldots,g \tag{15}$$

$$C_{ik}\ge X_{ijk}, \quad i=1,2,\ldots,g,\ \ k=2,3,\ldots,m,\ \ j=1,2,\ldots,b_{max} \tag{16}$$

$$X_{ijk}-X_{ij(k-1)}\ge\sum_{p=1}^{g} W_{ip}t_{pjk}, \quad \begin{matrix} i=1,2,\ldots,g, \\ k=2,3,\ldots,m, \end{matrix} \quad j=1,2,\ldots,b_{max} \tag{17}$$

$$X_{ijk},C_{ik},O_{ik}\ge 0, \quad W_{ip},A_{ipl}\in\{0,1\}, \quad Y_{ijq}\in\{0,1\}$$
$$(j<q),\ \ i=1,2,\ldots,g,\ \ j,q=1,2,\ldots,b_{max}$$

The objective function focuses on minimizing the TFT. There are ‘g’ slots and each group should be assigned to one of them. It is clear that each slot should contain just one group and every group should be assigned to only one slot. Constraints (8a) and (8b) support this assumption. The setup time of a group on a machine is dependent on that group and the group processed immediately preceding it. If group p is assigned to slot i and group l is assigned to slot i+1, then A_{ipl} must be equal to one. Likewise, if group p is not assigned to slot i or group l is not assigned to slot i+1, then A_{ipl} must be equal to zero. Constraints (9), (10a), and (10b) ensure that each is true. Constraint

(11) evaluates the required setup time of groups on machines. The required setup time for a group on a machine is evaluated based on the group assigned to a slot and the group assigned to the preceding slot. Constraint (12) is incorporated to find the completion time of jobs on machines. The completion time of a job that belongs to a group is greater than or equal to the summation of the completion time of the group processed in the previous slot, the setup time for the group, and the run time of the job. Constraints (13) and (14) are incorporated into the model to find the sequence of processing jobs that belong to a group. If job *j* in a group is processed after job *q* of the same group, then the difference between the completion time of job *j* and job *q* on all machines should be greater than or equal to the run time of job *j*. The completion time of the group assigned to a slot on the first machine is evaluated by constraint (15). The completion time of a group assigned to a slot is equal to the summation of the completion time of the group assigned to the preceding slot, the required setup time for the group in this slot, and the sum of the run time of all jobs in the group. Constraint (16) is incorporated into the model to ensure that the completion time of a group on a machine is equal to the completion time of the last job of that group processed on the machine. A machine starts processing a job only if the job is finished on the previous machine. It means that the completion time of a job on a machine must be greater than or equal to the summation of the completion time of the job on the preceding machine and the run time of the job on that machine. Constraint (17) is incorporated for this reason.

5. Lower bounding method

Several techniques such as Lagrangian relaxation were tried to get a promising lower bound for the proposed research problem, but none of them had a successful performance. Thus, a generalized lower bounding method based on B&P algorithm is developed for the proposed problem. This model should serve well as a basis for future research in FSDGS problems with other objectives. The details associated with the B&P algorithm can be found in Barnhart et al. [15], Wilhelm [16], and Wilhelm et al. [17].

The goal is to develop a mathematical model whose LP relaxation provides a good lower bound for the original problem. The problem is reformulated with a huge number of variables. Each variable set presents a possible feasible solution (schedule) for a machine. The mathematical model is then decomposed into a master problem and one or more sub-problems (SPs). The number of SPs is equal to the number of machines of the original problem. The sets of columns (variables) are left out of the LP relaxation of master problem because there are too many columns (variables) to handle efficiently and most of them will have their associated variable equal to zero in an optimal solution anyway. At the beginning, the LP relaxation of the master problem is solved by considering a few feasible solutions. Because the master problem at this stage does not include all possible solutions, it is called the Restricted Master Problem (RMP). To check the optimality of the LP solution to the RMP, the SPs, which are a separation problem for the dual LP, are solved to find columns to enter the master problem. If such columns exist, the LP is re-optimized. If there is no column to enter and the LP solution does not satisfy the integrality conditions, then branching is applied for the optimal solution of the LP problem.

In the following sections, in Section 5.1, the mathematical model for solving the problem by B&P algorithm is discussed. Then, in Sections 5.2–5.4, the rules applied to simplify solving SPs are explained. In Section 5.5, the rules applied for solving SPs are presented. In Sections 5.6 and 5.7, branching rules and stopping criteria are presented, respectively. And finally, in Section 5.8, the rules of evaluating the lower bound for the original problem are presented.

5.1. Mathematical model for solving the problem by B&P algorithm

In the original mathematical model which was presented in Section 4, constraints (8)–(11) deal with finding the setup times for groups. Constraints (12)–(14) deal with finding the completion time of jobs on each machine. Constraints (15) and (16) are also related to evaluating the completion time of each group on different machines. These constraints can be applied for each machine separately. Constraint (17) is the only constraint that deals with the completion time of jobs on more than one machine. Thus, this constraint is considered as the linking (complicating) constraint in the model. The RMP includes constraint (17), which is a relational constraint between the completion times of jobs on machines. A convexity constraint for each SP is also incorporated into the RMP. The parameters, decision variables, and mathematical model of the *LP relaxation* of the RMP, herein after referred to as LRMP, for an *m* machine problem are presented as follows:

Parameters:

h_k	the number of columns that exist in LRMP related to the *k*th machine	$k=1,2,\ldots,m$
$X_{ijk}^{h_k}$	the completion time of *j*th job of *i*th slot on machine *k* in h_kth existing solution in LRMP related to machine *k*	$i=1,2,\ldots,g$ $j=1,2,\ldots,b_{max}$ $k=1,2,\ldots,m$
$W_{ipk}^{h_k}$	$\begin{cases}1 & \text{If group } p \text{ is assigned to slot } i \text{ in } h_k\text{th existing solution in LRMP related to machine } k\\ 0 & \text{Otherwise}\end{cases}$	$i=0,1,2,\ldots,g$ $p=0,1,2,\ldots,g$ $k=1,2,\ldots,m$

Decision variable:

λ_h^k the decision variable of the *h*th existing solution of machine *k* in LRMP $k=1,2,\ldots,m$

The LRMP Model (*model* 2):

$$\operatorname{Min} Z=\sum_{h=1}^{h_m}\lambda_h^m\sum_{i=1}^{g}\sum_{j=1}^{b_{max}}X_{ijm}^{h} \tag{18}$$

Subject to:

$$\sum_{h=1}^{h_k}\lambda_h^k\left(X_{ijk}^h-\sum_{p=1}^{g}w_{ipk}^h t_{pjk}\right)-\sum_{h1=1}^{h_{k-1}}\lambda_{h1}^{k-1}X_{ij(k-1)}^{h1}\geq 0$$
$$i=1,2,\ldots,g,\ \ j=1,2,\ldots,b_{max},\ \ k=2,3,\ldots,m \tag{19}$$

$$\sum_{h=1}^{h_k}\lambda_h^k=1,\quad k=1,2,3,\ldots,m \tag{20}$$

$$\lambda_h^k\geq 0,\quad k=1,2,\ldots,m,\ \ h=1,2,\ldots,h_m$$

In this model, λ_h^k's are the LRMP decision variables. The objective is to minimize total flow time. Constraint (19) is the relational constraint, incorporated to the model based on constraint (17) of the original model. Constraint (20) is the convexity constraint. For each SP a convexity constraint is incorporated to the model. If the d_{ijk}'s and α_k denote the dual variables of constraint (19) and (20), respectively, the dual problem of the LRMP can be formulated as follows:

$$\text{Max } Z' = 0 * \left(\sum_{k=2}^{m} \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ijk} \right) + \sum_{M=1}^{m} \alpha_M \tag{21}$$

Subject to:

$$-\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ij2}(X_{ij1}) + \alpha_1 \leq 0 \tag{22}$$

$$\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ijk} \left(X_{ijk} - \sum_{p=1}^{g} W_{ip} t_{pjk} \right) + \alpha_k - \left(\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ij(k+1)} X_{ijk} \right) \leq 0 \quad k = 2,3,\ldots,m-1 \tag{23}$$

$$\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ijm} \left(X_{ijm} - \sum_{p=1}^{g} W_{ip} t_{pjm} \right) + \alpha_m - \left(\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} X_{ijm} \right) \leq 0 \tag{24}$$

$d_{ijk} \geq 0$, α_k Unrestricted, $i = 1,2,\ldots,g$, $j = 1,2,\ldots,b_{max}$, $k = 2,3,\ldots,m$

A feasible solution of LRMP is optimal, if its dual can satisfy the constraints of the dual problem. Thus, to check the optimality of LRMP, the constraints of the dual problem are checked. Each dual problem constraint deals with the completion time of jobs on a specific machine. Thus, to check the optimality, the SPs, in which their objective functions are the left-hand side of the dual constraints, are solved. If the value of the objective function is greater than zero, the column is added to the LRMP. Decomposition leads to an independent SP for each machine. When the LRMP is solved, it generates new dual variables for SPs. The SPs are:

$$\text{SP}_1 \quad \text{Max} \quad W_1 = -\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ij2}(X_{ij1}) + \alpha_1$$
$$\text{S.t.} \quad \text{Constraints (8)–(15) of the original problem} \tag{25}$$

$$\text{SP}_2 \text{ through SP}_{m-1} \quad \text{Max} \quad W_k = \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ijk} \left(X_{ijk} - \sum_{p=1}^{g} W_{ip} t_{pjk} \right) + \alpha_k - \left(\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ij(k+1)} X_{ijk} \right)$$
$$k = 2,3,\ldots,m-1$$
$$\text{S.t.} \quad \text{Constraints (8)–(14), and (16) of the original problem} \tag{26}$$

$$\text{SP}_m \quad \text{Max} \quad W_m = \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} d_{ijm} \left(X_{ijm} - \sum_{p=1}^{g} W_{ip} t_{pjm} \right) + \alpha_m - \left(\sum_{i=1}^{g} \sum_{j=1}^{b_{max}} X_{ijm} \right)$$
$$\text{S.t.} \quad \text{Constraints (8)–(14), and (16) of the original problem} \tag{27}$$

The decomposed model has the following characteristics:

- For large size problems the time required to solve the SPs is very high. The experiments that we performed show that the SPs become more complicated when the maximum number of jobs in a group is increased.

Table 1
The coefficient of X_{ijk}'s in sub-problems and the required constraints.

Sub-problem	X_{ijk} coefficient	Required inequalities	Ranges
SP_1	$-d_{ij2}$	$d_{ij2} \geq 0$	$i = 1,2,\ldots,g$, $j = 1,2,\ldots,b_{max}$
SP_{k1}, $k1 = 2,\ldots,m-1$	$d_{ijk} - d_{ij(k+1)}$	$d_{ijk} - d_{ijn} \leq 0$	$i = 1,2,\ldots,g$, $j = 1,2,\ldots,b_{max}$ $k = 2,3,\ldots,m-2$, $n = k+1$
SP_m	$d_{ijm} - 1$	$d_{ijm} \leq 1$	$i = 1,2,\ldots,g$, $j = 1,2,\ldots,b_{max}$

- The coefficient of X_{ijk}'s in the objective function of SPs, except SP_1, can be positive. Because of the maximization of the objective function as well as the inability of any constraint in SPs to limit the value of X_{ijk}'s, it is possible that an SP be unbounded. For instance, in SP2, the coefficient of X_{ij2}'s is equal to $(d_{ij2} - 1)$. By considering the maximization of the objective function, it is possible that if $d_{ij2} \geq 1$, SP_2 be unbounded. Thus, it is required to add an upper bound to SP_2. In this case the quality of solution and the efficiency of the algorithm are highly dependent on the value of the chosen upper bound for the SPs.

Consider the m-machine problem. In order to prevent an unbounded solution in any of the SPs, the coefficient of X_{ijk}, which are shown in Table 1, in any SP should be negative. Thus, the inequalities in this table should hold true in order to prevent unbounded solutions.

To support these rules, a set of artificial variables, $S1$, are added to the relational constraint (19) for the jth job of slot i of the constraint between M_1 (the first machine) and M_2 (the second machine) with the coefficient equal to 1. The same set of artificial variables is also added to the relational constraint between M_2 and M_3 with the coefficient equal to -1, respectively. Another artificial variable set, $S2_{ij}$, is also added to the relational constraint of the jth job of slot i between M_2 and M_3 with the coefficient equal to 1. The same set of artificial variables is also added to the relational constraint between M_3 and M_4 with the coefficient equal to -1, respectively. These artificial variables are added, respectively, to consecutive constraints from M_2–M_3 through M_{m-1}–M_m. The new model and its dual problem are as follows. This model will provide a lower bound for model 2.

$$\text{Min } Z = \sum_{h=1}^{h_m} \lambda_h^m \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} X_{ijm}^h + \sum_{i=1}^{g} \sum_{j=1}^{b_{max}} SM_{ij} \tag{28}$$

Subject to:

$$\sum_{h=1}^{h_2} \lambda_h^2 \left(X_{ij2}^h - \sum_{p=1}^{g} W_{ip2}^h t_{pj2} \right) - \sum_{h1=1}^{h_1} \lambda_{h1}^1 X_{ij1}^{h1} + S1_{ij} \geq 0 \quad i = 1,2,\ldots,g \quad j = 1,2,\ldots,b_{max} \tag{29}$$

$$\sum_{h=1}^{h_k} \lambda_h^k \left(X_{ijk}^h - \sum_{p=1}^{g} W_{ipk}^h t_{pjk} \right) - \sum_{h1=1}^{h_{k-1}} \lambda_{h1}^{k-1} X_{ij(k-1)}^{h1} + Sk_{ij} - S(k-1)_{ij} \geq 0 \quad i = 1,2,\ldots,g \quad j = 1,2,\ldots,b_{max} \quad k = 3,4,\ldots,m \tag{30}$$

$$\sum_{h=1}^{H} \lambda_h^k = 1, \quad k = 1,2,\ldots,m \tag{31}$$

$$\lambda_h^k \geq 0, \quad k = 1,2,3,\ldots,m \quad h = 1,2,\ldots,h_m$$

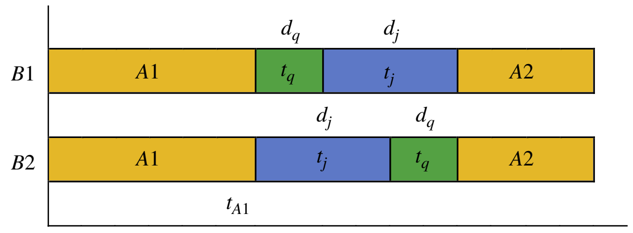

Fig. 1. The Gantt chart of processing two different sequences.

Table 2
The completion times of jobs in B_1 and B_2.

Job	Dual variable	Completion time in B_1	Completion time in B_2
Job q	d_q	$t_{A1}+t_q$	$t_{A1}+t_q+t_j$
Job j	d_j	$t_{A1}+t_q+t_j$	$t_{A1}+t_j$

The dual problem of the new model:

$$\text{Max } Z' = 0 * \sum_{k=2}^{m}\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} d_{ijk} + \sum_{M=1}^{m} \alpha_M \tag{32}$$

Subject to:

$$-\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} d_{ij2}(X_{ij1}) + \alpha_1 \leq 0 \tag{33}$$

$$\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} d_{ijk}\left(X_{ijk} - \sum_{p=1}^{g} W_{ip}t_{pjk}\right) + \alpha_k - \left(\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} d_{ij(k+1)}X_{ijk}\right) \leq 0$$
$$k = 2,3,\ldots,m-1 \tag{34}$$

$$\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} d_{ijm}\left(X_{ijm} - \sum_{p=1}^{g} W_{ip}t_{pjm}\right) + \alpha_m - \left(\sum_{i=1}^{g}\sum_{j=1}^{b_{max}} X_{ijm}\right) \leq 0 \tag{35}$$

$$d_{ij2} \leq 1, \quad \begin{matrix} i = 1,2,\ldots,g \\ j = 1,2,\ldots,b_{max} \end{matrix} \tag{36}$$

$$d_{ijk} - d_{ij(k+1)} \leq 0, \quad i = 1,2,\ldots,g, \quad j = 1,2,\ldots,b_{max} \quad k = 2,3,\ldots,m-1 \tag{37}$$

$$d_{ijm} \geq 0, \quad \alpha_k \text{ Unrestricted}, \quad i = 1,2,\ldots,g, \quad j = 1,2,\ldots,b_{max} \tag{38}$$

As mentioned, the SPs get harder to be solved if the maximum number of jobs in a group is increased. There are rules that can relax the job sequence constraints of SPs and simplify solving them. These rules for each SP are discussed in the following sections. Thus, the complexity of the SPs will be based on only finding the sequence of groups.

5.2. *The relaxing rule for* SP$_1$ *in the multiple-machine problem*

The objective function of SP$_1$ is shown in (25). In this equation α_1 is a constant. Assume that B_1 and B_2 are two different sequences of processing jobs for SP$_1$, which are shown in Fig. 1. The only difference between these two sequences is that the sequences of processing job q and job j, in which both of them belong to the same group, are changed. The completion times of the preceding jobs before these jobs are the same, and are shown by t_{A1} in both sequences in Fig. 1. Variables d_q and d_j are the dual values of these jobs, respectively. Assume that the objective function value of B_1 is less than B_2. The completion times of these jobs in B_1 and B_2 are shown in Table 2.

By substituting these values in the objective function of SP$_1$, because B_1 has a smaller objective function value, inequality (39) holds true:

$$-d_q(t_{A1}+t_q) - d_j(t_{A1}+t_q+t_j) \leq -d_j(t_{A1}+t_j) - d_q(t_{A1}+t_q+t_j) \tag{39}$$

Simplifying the inequality leads to

$$\frac{d_q}{t_q} \leq \frac{d_j}{t_j} \tag{40}$$

Based on this fact, at the optimal solution of SP$_1$, the sequence of jobs that belong to a group should respect inequality (40). Thus, by incorporating constraints in SP$_1$ which can guarantee that inequality (40) holds true, the sequence of jobs in a group can be calculated easily. By incorporating the constraints below to SP$_1$, for any given group sequence, the sequence of jobs in a group can be calculated by the model. Thus, the SP$_1$ can be solved optimally easier. Based on these constraints job q is processed after job j if

$$Y_{ijq} \geq \sum_{p=1}^{g} W_{ip}\left(\frac{d_{ij2}}{t_{pj1}} - \frac{d_{iq2}}{t_{pq1}}\right), \quad \begin{matrix} i = 1,2,\ldots,g \\ j,q = 1,2,\ldots,b_{max}, \ \ j<q \end{matrix} \tag{41}$$

$$Y_{ijq} \leq 1 + \sum_{p=1}^{g} W_{ip}\left(\frac{d_{ij2}}{t_{pj1}} - \frac{d_{iq2}}{t_{pq1}}\right), \quad \begin{matrix} i = 1,2,\ldots,g \\ j,q = 1,2,\ldots,b_{max}, \ \ j<q \end{matrix} \tag{42}$$

In these constraints, if job q is processed after job j in slot i, then the value of $\sum_{p=1}^{g} W_{ip}((d_{ij2}/t_{pj1}) - (d_{iq2}/t_{pq1}))$ is positive. In this case, by applying constraint (41) is SP$_1$, Y_{ijq} will be equal to 1. Constraint (42) supports this value as well. On the other hand, if job q is processed before job j in slot i, then the value of $\sum_{p=1}^{g} W_{ip}(d_{ij2}/t_{pj1} - d_{iq2}/t_{pq1})$ is negative. By applying constraint (41) in SP$_1$, Y_{ijq} will have a value greater than a negative number. In this case, constraint (42) forces Y_{ijq} to take a value equal to zero.

5.3. *The relaxing rule for* SP$_2$ *through* SP$_{m-1}$ *in the multiple-machine problem*

The objective function of any sub-problem, except the first and the last SP, SP$_k(k = 2,3,\ldots,m-1)$, is shown in (26). In this equation α_k is a constant. To find a rule to relax the job sequence constraints, consider two different sequences of processing jobs which are shown in Fig. 1. Assume that the objective function value of B_1 is less than B_2. The completion time of the jobs which make the difference between B_1 and B_2 are shown in Table 2. In this case, by substituting the value of completion times in the objective function of SP$_k$, inequality (43) holds true:

$$\begin{aligned} & d_{kq}(t_{A1}+t_q-t_q) - d_{(k+1)q}(t_{A1}+t_q) + d_{kj}(t_{A1}+t_q+t_j-t_j) \\ & \quad - d_{(k+1)j}(t_{A1}+t_q+t_j) \\ & \quad\quad \leq d_{kj}(t_{A1}+t_j-t_j) - d_{(k+1)j}(t_{A1}+t_j) + d_{kq}(t_{A1}+t_j+t_q-t_q) \\ & \quad - d_{(k+1)q}(t_{A1}+t_q+t_j) \end{aligned} \tag{43}$$

Simplifying the inequality leads to

$$\frac{d_{kj} - d_{(k+1)j}}{t_j} \leq \frac{d_{kq} - d_{(k+1)q}}{t_q} \tag{44}$$

Thus, at the optimal solution of SP$_k$, if there is no idle time among the processing time of jobs, the sequence of jobs of a group respect inequality (44). By incorporating the constraints below to SP$_k$, for any given group sequence, the sequence of jobs in a group can be calculated by the model. The explanation for the constraints below is similar to the ones presented for constraints (41) and (42):

$$Y_{ijq} \geq \sum_{p=1}^{g} W_{ip}\left(\frac{d_{iq(k+1)} - d_{iqk}}{t_{pqk}} - \frac{d_{ij(k+1)} - d_{ijk}}{t_{pjk}}\right) \quad i = 1,2,\ldots,g \quad j,q = 1,2,\ldots,b_{max},\ j < q \quad k = 2,3,\ldots,m-1 \tag{45}$$

$$Y_{ijq} \leq 1 + \sum_{p=1}^{g} W_{ip}\left(\frac{d_{iq(k+1)} - d_{iqk}}{t_{pqk}} - \frac{d_{ij(k+1)} - d_{ijk}}{t_{pjk}}\right) \quad i = 1,2,\ldots,g \quad j,q = 1,2,\ldots,b_{max},\ j < q \quad k = 2,3,\ldots,m-1 \tag{46}$$

5.4. *The relaxing rule for* SP$_m$ *in the multiple-machine problem*

The objective function of SP$_m$ is given in (27). In this equation α_m is a constant. Consider two different sequences of processing jobs (B_1 and B_2) which are shown in Fig. 1. The completion time of the jobs which make the differences between B_1 and B_2, i.e., job q and job j are shown in Table 2. Assume that the objective function value of B_1 is less than B_2. In this case, by substituting the value of completion times of the two sequences for B_1 and B_2 in the objective function of SP$_m$, inequality (47) holds true:

$$d_q(t_{A1} + t_q - t_q) - (t_{A1} + t_q) + d_j(t_{A1} + t_q + t_j - t_j) - (t_{A1} + t_q + t_j) \leq d_j(t_{A1} + t_j - t_j) - (t_{A1} + t_j) + d_q(t_{A1} + t_j + t_q - t_q) - (t_{A1} + t_j + t_q) \tag{47}$$

Simplifying the above inequality leads to

$$\frac{d_j - 1}{t_j} \leq \frac{d_q - 1}{t_q} \tag{48}$$

Since the objective is maximizing the SP$_m$, at the optimal solution of SP$_m$, the sequence of jobs of a group respects inequality (48). Thus, by incorporating the constraints below to SP$_m$, for any given group sequence, the sequence of jobs in a group can be calculated by the model:

$$Y_{ijq} \geq \sum_{p=1}^{g} W_{ip}\left(\frac{d_{iqm} - 1}{t_{pqm}} - \frac{d_{ijm} - 1}{t_{pjm}}\right), \quad i = 1,2,\ldots,g \quad j,q = 1,2,\ldots,b_{max},\ j < q \tag{49}$$

$$Y_{ijq} \leq 1 + \sum_{p=1}^{g} W_{ip}\left(\frac{d_{iqm} - 1}{t_{pqm}} - \frac{d_{ijm} - 1}{t_{pjm}}\right), \quad i = 1,2,\ldots,g \quad j,q = 1,2,\ldots,b_{max},\ j < q \tag{50}$$

The explanation for the constraints is similar to the ones presented for constraints (41) and (42). These constraints help to solve SP$_m$ easier since they restrict the completion time of each group.

5.5. *Solving SPs*

At each level of solving a node, by solving each SP, a column is added to LRMP. Since the SPs are very complicated, it is better to avoid solving them optimally as long as possible. It is not necessary that the SPs be solved optimally during the intermediate levels of solving a node to choose a column. Thus, a heuristic algorithm based on TS is applied to solve SPs until it provides a column with positive objective function value as a solution. When the TS is unable to find such a column for all SPs, the SPs are solved optimally. This process is performed until none of the SPs can provide a column with positive objective function. At this time, the node is optimally solved by solving the mathematical models of the SPs optimally. In other words, at the end of each node, all SPs must be solved optimally to make sure that the optimal solution of a node is found.

5.6. *Branching*

The LRMP, which is solved by column generation, will not necessarily provide an integral solution. Thus, applying a standard branch-and-bound procedure to the LRMP with its existing columns will not guarantee an optimal (or feasible) solution [15]. Barnhart et al. [18] and Desrosiers et al. [19] suggested to branch on the original variables of the problem. Since the sequence of jobs in a group can be calculated by the rules discussed, the branching is only performed on the group binary variables, i.e., A_{ipl}'s or W_{ip}'s. In this case, to find the best variable to branch on, all A_{ipl} variables related to all machines are considered for branching.

Each column that exists in LRMP relates to a decision variable (λ_h^k) of each node. To find the best decision variable for branching, for each A_{ipl} related to each machine, a branching index is calculated. The value of this index is the sum of the coefficients of the existing columns in LRMP in which $A_{ipl} = 1$. Wilhelm et al. [17] suggested to branch on the original variable in which its branching index has the nearest value to "0.5" compared to the other variables. Thus, branching is performed on the variable in which its branching index has the closest value to 0.5.

Suppose that A_{ipl}^k (A_{ipl} that belongs to the kth SP) has the closest value to 0.5 among all variables. In this case, the parent node is branched on two new nodes. In one node, the constraint $A_{ipl} = 1$ is added to SP$_k$ of the parent node and in the other node the constraint $A_{ipl} = 0$ is added to SP$_k$ of the parent node. All existing columns related to all machines but the kth machine are added to both child nodes. The columns related to the kth machine of the parent node are separated into two parts. The ones in which $A_{ipl} = 1$ are added to the first node, and the remaining are added to the one which includes the $A_{ipl} = 0$ constraint.

5.7. *Stopping criteria*

The branching process is continued until all nodes provide an integer solution, be infeasible, or are fathomed. Since this process requires a considerable amount of time, especially for large size problems, and considering the required amount of time for solving SPs optimally which are NP-hard, a time limitation is applied for solving problems. During solving the problems to obtain lower bounds, if the time spent for solving a problem exceeds 4 h, the SPs of the current node started is solved optimally once. After solving all SPs optimally, the algorithm stops and the best lower bound obtained so far is reported as the lower bound of the problem. The maximum time spent on solving an SP is set to at most 2 h. If an SP cannot be solved optimally in 2 h, solving the SP is stopped and the lower bound of the SP is considered as the objective function value of the SP. During solving the nodes, the breadth first procedure is used.

5.8. *The lower bound for the original problem*

The B&P algorithm terminates when one of the following two conditions is applicable in each problem:

- The B&P algorithm solves the problem optimally.
- The B&P algorithm is unable to solve the problem optimally because of the imposed time limitation.

If the B&P algorithm solves the problem optimally, it provides a lower bound for the original problem. If the B&P algorithm is unable to solve the problem optimally, the following rules are used to calculate the lower bound of the original problem.

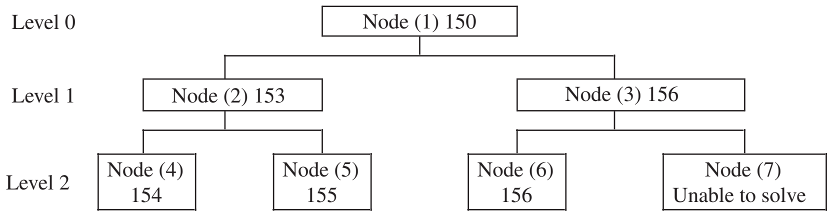

Fig. 2. The objective function value of nodes for an incomplete problem.

If in a problem, all nodes are not solved optimally because of time limitation, the lower bound of the original problem is the minimum value of the nodes which are solved optimally, but their branches are not solved optimally yet. For instance, consider the nodes of a problem in Fig. 2. Suppose that the B&P algorithm is stopped at the middle of the 7th node because of time limitation. In this case, since node 7 is not optimally solved yet, the lower bound of the original problem is the minimum objective function value of nodes 3, 4, and 5, which is equal to 154.

- In some problems, the SPs cannot be solved optimally in their time limitation of 2 h. In such cases, the algorithm stops solving the SP after 2 h and the lower bound of the SP is considered as the objective function value of the SP.
- If in a problem, a node cannot be solved optimally, the lower bound of the problem is equal to the objective function value of the recent LRMP minus the summation of the objective function values of the SPs [20].

The B&P algorithm is coded by concert technology concept of CPLEX 9.0 [21] version, by applying the beta version of MAESTRO library function developed for the B&P algorithm.

6. Test problem specifications

In industry, the goal is to decompose the production line to small, but independent manufacturing cells. Thus, a flow shop cell with more than six machines is highly unlikely in industry. In this research we consider the flow shop cells in which there are at most six machines. In the interest of time, an experiment which includes the minimum and the maximum number of machines is considered. Thus, the comparison is performed for two-, three-, and six-machine problems separately by solving the test problems generated. Three factors are considered to generate test problems for this research as follows:

- *Number of groups*: Schaller et al. [1] performed their experiments by considering at most 10 groups in a cell. Using this as a guideline and yet accommodating the possibility of investigating larger number of groups in industry settings, the maximum number of groups in a cell is set equal to 16 in this research. Thus, test problems are generated in three different categories: small, medium, and large, randomly from a uniform discrete distribution *DU* [1,5], *DU* [6,10], and *DU* [11,16] for small, medium, and large size problems, respectively.
- *Number of jobs in a group*: Problems including 2–10 jobs in a group are considered in this research. This number is the same as that used by Schaller et al. [1]. Our experience showed that the maximum number of jobs that belongs to a group in a problem is very important. Thus, this factor is chosen to classify the test problems based on their number of jobs in a group. For instance, if in a group scheduling problem with three groups, groups have five, seven, and nine jobs, respectively, then the problem is classified as a 9-job problem. The test problems based on their maximum number of jobs in a group are classified into three different sizes, small, medium, and large, respectively. The problems include at most 2–4 jobs in a group are classified as small size, problems with at most 5–7 jobs in a group are classified as medium size, and finally if one of the groups of a problem includes 8–10 jobs, then the problem belongs to the large size problems based on its number of jobs. Thus, the number of jobs in a group is generated based on a random integer from a discrete uniform distribution *DU* [2,4], *DU* [5,7], and *DU* [8,10] for small, medium, and large size problems, respectively.
- *The ratio of setup times*: The experiments performed indicate that the quality of solutions strongly depends on the ratio of setup times of groups on *consecutive machines*. Three levels are defined for this factor. In a sequential machine pair, if the setup time of the first machine is significantly less than the setup time of the second machine, the problem belongs to the first level. If both machines have almost the same setup times, the problem belongs to the second level. Finally, if the setup time of the first machine is significantly greater than the second machine, the problem is classified as the third level of this factor. This factor is applied to all sequential machine pairs. For instance, in a three machines problem, this ratio for "M_1/M_2" and "$M_2/\ M_3$" is compared. Thus, this is considered as two separate factors in this problem.

If this technique is applied for problems with more than three machines, the number of test problems which should be investigated will increase significantly. For instance, for a six-machine problem, because the number of whole-plot factors will increase to 7 (group factor, job factor, and 5 factors for ratios of sequenced machines), if in each cell only two replicates are applied, then it is required to solve $3^7 \times 2 = 4374$ problems. By considering that there are three versions of TS and two different initial solution generator mechanisms, $4374 \times 3 \times 2 = 26{,}244$ problems should be solved. This is the correct way to perform the experiment, but in the interest of time, it was not practical for this research. Thus, the experiment for problems with more than three machines is performed by just applying one factor for the ratio of setup times for all machine pairs. In this case, only a factor is defined for the ratio of setup times of machine pairs with three levels. Level 1 indicates the problems in which the required setup times for each machine are increased sequentially. The second level investigates the problems in which the setup times of all machines are almost equal. And finally, level three investigates the problems in which the setup times of machines are decreased from the first machine to the last machine.

Based on above definitions we have different class of problems to test. For instance for two-machine problems we have 27 different classes (3 (classes for groups)×3(classes for jobs)×3(classes for setup ratios)). This number is equal to 81 for three-machine problems since there are two different classes for setup ratios. If two sample problems are generated for each class of problems, for two-, three-, and six-machine problems, then the number of generated test problems is equal to 54, 162, and 54, respectively. The test problems are generated based on the specifications below:

- The runtime of jobs on machines is a random integer from a *DU* [1,20].

Table 3
The setup time of each machine on two-machine problems.

Machine	Level 1	Level 2	Level 3
M_1	*DU* [1,50]	*DU* [1,50]	*DU* [17,67]
M_2	*DU* [17,67]	*DU* [1,50]	*DU* [1,50]

Table 4
The setup time of each machine on three-machine problems.

Machine	Level 1	Level 2	Level 3
M_1	*DU* [1,50]	*DU* [1,50]	*DU* [45,95]
M_2	*DU* [17,67]	*DU* [1,50]	*DU* [17,67]
M_3	*DU* [45,95]	*DU* [1,50]	*DU* [1,50]

Table 5
The setup time of each machine on six-machine problems.

Machine	Level 1	Level 2	Level 3
M_1	*DU* [1,50]	*DU* [1,50]	*DU* [300,350]
M_2	*DU* [17,67]	*DU* [1,50]	*DU* [170,220]
M_3	*DU* [45,95]	*DU* [1,50]	*DU* [92,142]
M_4	*DU* [92,142]	*DU* [1,50]	*DU* [45,95]
M_5	*DU* [170,220]	*DU* [1,50]	*DU* [17,67]
M_6	*DU* [300,350]	*DU* [1,50]	*DU* [1,50]

- The setup time of groups on each machine for the two-machine problem is shown in Table 3. As discussed, in the first the setup time of groups on M_1 should be less than M_2. If these setup times are generated based on *DU* [1,50] and *DU* [17,67] for M_1 and M_2, respectively, then the average ratio of setup times is equal to 0.607 ([(1+50)/2]/[(17+67)]/2 = 0.607), which satisfies the condition. The setup times for other levels are generated similar to this rule to satisfy the required ratio of each level as well.
- The setup times of groups on each machine for problems with three and six machines are shown in Tables 4 and 5. The setup times shown in Table 4 for the three-machine problem can be applied for setup time ratio factors of M_1/M_2 as well as M_2/M_3. The distribution to generate random setup time for each machine at each level is chosen based on the required ratio among setup times.

7. Results

Both TS and HACO algorithms are coded in C programming language and the test problems are solved by them. The performances of the algorithms for two-, three-, and six-machine problems are compared separately, as a paired *t*-test experiment. The results are shown in Appendix B. Since the *p*-value for all three experiments are almost equal to zero (0, 0, and 0.0007, respectively), we can conclude that there is a statistically significant difference between the results of these algorithms. Since the average of the total flow time of the HACO algorithm is lower than the TS, we can conclude that the HACO has a better performance compared to the TS algorithm. The time required to reach the best solution for the HACO algorithm is shown in Fig. 3. For instance, for two-machine problems, almost 50% of the problems attain their best solutions in less than 5 s.

The lower bounding method is applied to estimate the quality of solutions. The ILOG CPLEX (version 9.0) is used to solve the lower bounding model. The heuristic algorithms (TS and HACO) and the lower bounding problems are run on a Power Edge 2650 with 2.4 GHz Xeon, and 4 GB RAM.

If the B&P algorithm is applied to solve large size problems, it requires a considerable amount of time to identify an LB. Thus, in the interest of time, only a few of the test problems are considered to be solved for estimating the quality of solutions as a sample. The size of the sample used for two-, three-, and six-machine problems is 11, 21, and 11, respectively. As shown in Table 6 the average percentage errors of the problems for the HACO algorithm are 14.4%, 16.3%, and 13.9% for two-, three-, and six-machine problems. For large size problems, the SPs cannot be solved optimally during their 2-h time limitation, and it explains the reason for not attaining a high quality LB for SPs. In each of these problem structures, there are a few problems with high percentage error (more than 50%). The number of these problems is one, two, and one for two-, three-, and six-machine problems, respectively. If these problems with high percentage error are removed from the sample, the percentage error of the problems is reduced drastically to result in satisfactory percentage errors. Table 6 also shows the reduced percentage errors by removing the problems with more than 50% error. More detailed results about each test problem instances are given in Salmasi [22]. In order to investigate the reason for low percentage error, the test problems with less than five groups as well as at most four jobs in a group are solved optimally by CPLEX based on the developed mathematical model. In all of these test problems the optimal solution was equal to the solutions developed by HACO algorithm. Thus, we can conclude that the low percentage error is due to the performance of the lower bounding technique. Since the research problem is too complex, this method is currently the only available method to determine the LB for the proposed research problem. The percentage error is calculated based on the formula:

(The heuristic algorithm solution − The lower bound)/The lower bound

The results based on different levels of the group size factor are presented in Table 7 for two-, three-, and six-machine problems, separately. Among the 11 problems chosen to be solved for two-machine problems, four of them belong to small size, four test problems belong to medium size, and three of them belong to large size problems. As presented in this table, the percentage errors for these levels are 5.8%, 9.2%, and 32.9%, respectively. One of these 11 test problems, which belongs to the large size factor, has a percentage error of more than 50%. By removing this problem, the revised percentage errors are reported in the last column of this table. The same policy is used to provide the information in Table 7 for three- and six-machine problems.

Also in order to evaluate the effect of different levels of setup time ratios, the percentage errors are evaluated and presented based on different levels of setup time ratio factors in Table 8–10 for two-, three,- and six-machine problems, respectively. In these tables, for each level of the setup time ratio factors (one ratio factor for two- and six-machine problems and two ratio factors for three-machine problems), the number of problems, the original percentage error, and the percentage error after removing the test problems with more than 50% percentage error are presented.

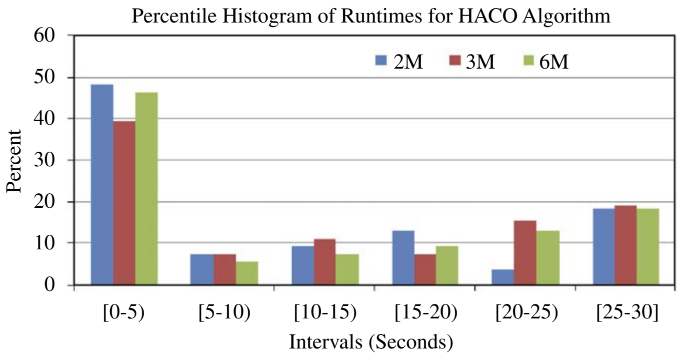

Fig. 3. Runtime percentage histogram.

Table 6
The percentage error for the hybrid ant colony optimization algorithm from the lower bound.

Problem	Average percentage error	Average percentage error by removing more than 50% percentage error problems
2-Machine	14.4	8.4 (by removing one test problem)
3-Machine	16.3	11.8 (by removing two test problems)
6-Machine	13.9	9.1 (by removing one test problem)

Table 7
The average percentage error for the hybrid ant colony optimization algorithm from the lower bound based on levels of group factor.

Problem	Group size	Number of test problems	Percentage error	Percentage error by removing more than 50% percentage error problems
2-Machine	Small	4	5.8	5.8
	Medium	4	9.2	9.2
	Large	3	32.9	11.9 (by removing one test problem)
	Total	11	14.4	8.4 (by removing one test problem)
3-Machine	Small	7	6.6	6.6
	Medium	7	10.3	10.3
	Large	7	32.1	21.5 (by removing two test problems)
	Total	21	16.3	11.8 (by removing two test problems)
6-Machine	Small	4	8.0	8.0
	Medium	4	6.1	6.1
	Large	3	31.3	16.0 (by removing one test problem)
	Total	11	13.9	9.1 (by removing one test problem)

Table 8
The average percentage error for the hybrid ant colony optimization algorithm from the lower bound based on levels of ratio of setup times for 2-machine problems.

Level of setup ratio factor	Number of test problems	Percentage error	Percentage error by removing more than 50% percentage error problems
Level 1	3	6.8	6.8
Level 2	4	28.9	13.5 (by removing one test problem)
Level 3	4	5.7	5.7
Total	11	14.4	8.4 (by removing one test problem)

Table 9
The average percentage error for the hybrid ant colony optimization algorithm from the lower bound based on levels of ratio of setup times for 3-machine problems.

Level of setup ratio factor (M_1/M_2)	Level of setup ratio factor (M_2/M_3)	Number of test problems	Percentage error	Percentage error by removing more than 50% percentage error problems
Level 1	Level 1	3	19	2.3 (by removing one test problem)
	Level 2	2	14	14
	Level 3	2	21	21
Level 2	Level 1	2	5.6	5.6
	Level 2	3	21.9	21.9
	Level 3	2	37.3	9.4 (by removing one test problem)
Level 3	Level 1	2	10	10
	Level 2	2	2.9	2.9
	Level 3	3	12.6	12.6
Total		21	16.3	11.8 (by removing two test problems)

Table 10
The average percentage error for the hybrid ant colony optimization algorithm from the lower bound based on levels of ratio of setup times for 6-Machine problems.

Level of setup ratio factor	Number of test problems	Percentage error	Percentage error by removing more than 50% percentage error problems
Level 1	3	0.0	0.0
Level 2	4	35.2	26.3 (by removing two test problems)
Level 3	4	3.1	3.1
Total	11	13.9	9.1 (by removing one test problem)

8. Conclusions and suggestions for future research

To the best of our knowledge, this is the first work on FSDGS problems by considering minimization of TFT criterion. In this research, a metaheuristic algorithm based on TS as well as a hybrid heuristic algorithm based on ant colony algorithm are developed for solving FSDGS problems. Then, the TS algorithm is compared with the HACO algorithm. The result of paired *t*-test shows that the HACO algorithm has a better performance for all problem sizes than the best TS algorithm. These comparisons are performed based on test problems ranging in size from small, medium, to large for two-, three-, and six-machine problems.

A generalized lower bounding method based on the branch-and-price method is also developed for FSDGS problems to estimate the quality of solutions. The comparison of the result of the heuristic algorithms with the optimal solutions for small size problems shows that the HACO algorithm can provide a good quality solution for the problems investigated.

All of the previous work on FSDGS problems is based only on minimization of makespan criterion. Recognizing the industrial relevance of FSDGS problems, further research can be performed by considering other optimization criteria such as minimization of total tardiness and minimization of weighted tardiness, amongst others.

Acknowledgment

The research reported in this paper is funded in part by the National Science Foundation (USA) Grant no. DMI-0010118. Their support is gratefully acknowledged.

Appendix A. The tabu search parameter values for minimization of total flow time criterion

See Table A1.

Appendix B. The result of paired *t*-tests for the tabu search and hybrid ant colony optimization algorithm comparison

```
Paired t-test for two-machine problems
data: x: V1 in SDF64, and y: V2 in SDF64
t = −4.5692, df = 53, p-value = 0
Alternative hypothesis: true mean of differences is
not equal to 0
95 percent confidence interval: −476.9937 −185.9693
sample estimates: mean of x − y: −331.4815
```

```
Paired t-test for three-machine problems
data: x: V3 in SDF64, and y: V4 in SDF64
t = −4.7069, df = 161, p-value = 0
alternative hypothesis: true mean of differences is
not equal to 0
95 percent confidence interval: −500.2285 −204.5369
sample estimates: mean of x − y: −352.3827
```

```
Paired t-test for six-machine problems
data: x: V5 in SDF64, and y: V6 in SDF64
t = −3.5835, df = 53, p-value = 0.0007
alternative hypothesis: true mean of differences is
not equal to 0
95 percent confidence interval: −613.1127 −173.0725
sample estimates: mean of x − y: −393.0926
```

References

[1] Schaller JE, Gupta JND, Vakharia AJ. Scheduling a flowline manufacturing cell with sequence dependent family setup times. European Journal of Operational Research 2000;125:324–39.
[2] Pinedo M. Scheduling theory, algorithms, and systems. 3rd ed., Englewood Cliffs, NJ: Prentice-Hall; 2008.

Table A1

Outside parameters								
Index list			Iterations without improvement			Tabu list size		
Number of groups (G)		Parameter value/ formula	Number of groups (G)		Parameter value/formula	Number of groups (G)		Parameter value
From	To		From	To		From	To	
2	4	2	2	3	G/2	2	5	1
5	6	G∗2	4	5	2	6	7	2
7	7	G∗10	6	6	3	8	9	3
8	12	G∗20	7	7	6	10	14	4
13	16	250	8	14	8	15	16	6
			15	16	11			
Inside parameters								
Number of jobs (J)		Parameter value	Number of jobs (J)		Parameter value	Number of jobs (J)		Parameter value
From	To		From	To		From	To	
2	20	2	2	30	2	2	20	1
21	30	3	31	40	3	21	30	2
31	40	5	41	50	5	31	39	3
41	50	6	51	120	8	40	50	4
51	90	7				51	60	5
91	120	8				61	75	6
						76	85	8
						86	95	9
						96	100	10
						101	120	13

[3] Cheng TCE, Gupta JND, Wang G. A review of flowshop scheduling research with setup times. Production and Operations Management 2000;9(3):262–82.
[4] Allahverdi A, Gupta JND, Aldowaisian T. A review of scheduling research involving setup considerations. Omega, International Journal of Management Science 1999;27:219–39.
[5] Zhu X, Wilhelm WE. Scheduling and lot sizing with sequence-dependent setups: a literature review. IIE Transactions 2006;38:987–1007.
[6] Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY. A survey of scheduling problems with setup times or costs. European Journal of Operational Research 2008;187(3):985–1032.
[7] Franca PM, Gupta JND, Mendes PM, Veltink KJ. Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. Computers and Industrial Engineering 2005;20:1–16.
[8] Logendran R, Salmasi N, Sriskandarajah C. Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. Journal of Computers and Operations Research 2006;33:158–80.
[9] Hendizadeh H, Faramarzi H, Mansouri SA, Gupta JND, Elmekkawy TY. Meta-heuristics for scheduling a flowshop manufacturing cell with sequence dependent family setup times. International Journal of Production Economics 2008;111:593–605.
[10] Garey MD, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1976;1(2):117–29.
[11] Nawaz M, Enscore E, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, International Journal of Management Science 1983;11(1):91–5.
[12] Dorigo M. Optimization, learning, and natural algorithms. PhD thesis, Politecnico di Milano; 1992.
[13] Dorigo M, Gambardella LM. Ant colonies for the traveling salesman problem. BioSystems 1997;43(2):73–81.
[14] Gajpal Y, Rajendran C. An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. International Journal of Production Economics 2006;101:259–72.
[15] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH. Branch and price: column generation for solving huge integer programs. Operations Research 1998;46(3):316–29.
[16] Wilhelm WE. A technical review of column generation in integer programming. Optimization and Engineering 2001;2:159–200.
[17] Wilhelm WE, Damodaran P, Li J. Prescribing the content and timing of product upgrade. IIE Transactions 2003;35:647–63.
[18] Barnhart C, Hane CA, Johnson EL, Sigimondi G. A column generation and partitioning approach for multi-commodity flow problem. Telecommunication Systems 1995;3:239–58.
[19] Desrosiers J, Dumas Y, Solomon MM, Soumis F. Time constrained routing and scheduling. In: Ball ME, Magnanti TL, Monma C, Nemhauser GL, editors. Handbooks in operations research and management science. Amsterdam: Elsevier; 1995.
[20] Lubbecke ME, Desrosiers J. Selected topics in column generation. Operations Research 2005;53(6):1007–23.
[21] CPLEX, Release 9.0, ILOG Institute, Paris, France.
[22] Salmasi N. Multi-stage group scheduling problems with sequence dependent setups. Doctoral Dissertation, Oregon State University, Corvallis, Oregon; 2005.