Ann Oper Res (2015) 226:351–377
DOI 10.1007/s10479-014-1667-6

Minimizing total completion time in the flexible flowshop sequence-dependent group scheduling problem

Taha Keshavarz · Nasser Salmasi · Mohsen Varmazyar

Published online: 11 July 2014

Abstract This research considers a flexible flowshop sequence-dependent group scheduling problem with minimization of total completion time. A mixed integer linear mathematical model for the research problem is developed. Since the research problem is shown to be strongly NP-hard, a metaheuristic algorithm based on memetic algorithm (MA) is proposed. A lower bounding method based on the Branch and Price algorithm is also proposed to evaluate the quality of the MA. In order to evaluate the performance of the proposed algorithms, random test problems, ranging in size from small, medium, to large are generated and solved by the MA and the lower bounding method. The results show that the average percentage gap of the MA is 6.03 % compared to the result of the lower bounding method for randomly generated test problems.

Keywords Flexible flowshop · Group scheduling · Sequence-dependent setup time · Branch and price algorithm · lower bound · memetic algorithm

1 Introduction

Flexible fowshops are becoming increasingly popular in industry, primarily due to large workload requirements imposed by jobs on machines representing one or more stages of a flowshop scheduling problem (Logendran et al. 2005). In a flexible flowshop environment, there are stages in series with a number of identical parallel machines at each stage. All jobs

T. Keshavarz
Department of Industrial Engineering, Yazd University, P.O. Box 89195-741, Yazd, Iran
e-mail: keshavarz.taha@chmail.ir

N. Salmasi (✉) · M. Varmazyar
Department of Industrial Engineering, Sharif University of Technology, P.O. Box 11365-8639, Tehran, Iran
e-mail: nsalmasi@sharif.edu

M. Varmazyar
e-mail: varmazyar.mohsen@gmail.com

have to pass through all stages. At each stage, each job should be processed by only one machine.

Usually, the jobs assigned to a manufacturing cell are set to different groups based on their similarities such as similar shape or required setups on machines in order to improve the efficiency. This subject is called group scheduling (GS). GS improve the production efficiency by a reduction in setup time, tooling needs, and work-in-process inventories. Usually, a setup needed on a machine to transfer from one job to another job within a group is negligible. However, it is required to consider a setup time on each machine to transfer from processing one job of a group to a job of another group. Sometimes this setup time is dependent upon the previously processed group on the machine. In this case the problem is called the sequence-dependent group scheduling problem.

Schaller et al. (2000) consider minimization of makespan as the criterion, in the sequence-dependent permutation flowshop group scheduling problem, i.e., $Fm|fmls, s_{hgi}, prmu|C_{\max}$ based on updated scheduling notations by Pinedo (2012) for the first time. They propose several heuristic algorithms as well as a lower bounding method for the research problem. França et al. (2005) develop two metaheuristics based on genetic algorithms (GA) and memetic algorithms (MA) to solve the same problem proposed by Schaller et al. (2000). They show that the MA has a superior performance than the GA as well as the heuristics proposed by Schaller et al. (2000). Logendran et al. (2006b) propose metaheuristics algorithms based on tabu search (TS) for the $F2|fmls, s_{hgi}, prmu|C_{\max}$ problem. They also develop a lower bounding method based on the mathematical model of the research problem to evaluate the performance of the TS algorithms. Salmasi et al. (2011) develop a metaheuristic algorithm based on ant colony optimization (ACO) and a lower bounding method based on the mathematical model of the problem for the $Fm|fmls, s_{hgi}, prmu|C_{\max}$ problem. They show that the ACO has a better performance than the other available metaheuristics in the literature for the proposed research problem. Ravetti et al. (2012) propose and analyze parallel hybrid heuristics for permutation flowshop problem. The $Fm|fmls, s_{hgi}, prmu|\sum C_j$ problem is investigated by Salmasi et al. (2010) for the first time. They develop a mathematical model and several metaheuristics based on TS and ACO for the proposed research problem. They show that their proposed ACO algorithm has a superior performance than the TS. Furthermore, they develop a lower bounding method based on the branch and price (B&P) algorithm. Hajinejad et al. (2011) propose a particle swarm optimization (PSO) algorithm for the $Fm|fmls, s_{hgi}, prmu|\sum C_j$ problem and show that their PSO has a better performance than the ACO algorithm proposed by Salmasi et al. (2010). Naderi and Salmasi (2012) develop two mathematical models for the same problem. They show that one of the mathematical models is so effective that even medium size instances (problems up to 60 jobs in all groups) are solved to optimality in a reasonable time. They also propose a hybrid metaheuristic based on genetic and simulated annealing. They show that their proposed algorithm has a superior performance than the ACO proposed by Salmasi et al. (2010).

Most of prior research are performed on the flowshop group scheduling problem and there are only a few research that consider flexible flowshop environment. Logendran et al. (2005) consider the group scheduling flexible flowshop for the first time. In their research, three constructive heuristic algorithms are developed for the $FFm|fmls|C_{\max}$ problem. Logendran et al. (2006a) develop a TS algorithm for the $FFm|fmls, s_{hgi}|C_{\max}$ problem. For analyzing both the makespan value and computation time, a detailed statistical experiment is performed. Paternina-Arboleda et al. (2008) consider the problem of makespan minimization on a flexible flowshop and propose a heuristic algorithm based on the identification and exploitation of the bottleneck stage. Zandieh et al. (2009) develop two metaheuristic algorithms based on simulated annealing and GA for the $FFm|fmls, s_{hgi}|C_{\max}$ problem. Salmasi et al. (2011)

propose another TS algorithm for the same problem and show that their proposed algorithm has a superior performance than the proposed algorithm by Logendran et al. (2006a). They also propose a mathematical model for the proposed research problem for the first time. Keshavarz and Salmasi (2013) propose another mathematical model as well as a metaheuristic algorithm based on MA for the $FFm|fmls, s_{hgi}|C_{\max}$ problem. They show that both their proposed mathematical model and metaheuristic algorithm have better performance than the ones proposed by Salmasi et al. (2011). They also proposed a lower bounding mechanism for the proposed research problem inspired from Salmasi et al. (2011).

All previous research for the $FFm|fmls, s_{hgi}|\gamma$ problem consider minimization of makespan as the criterion. To the best of our knowledge, in this research the $FFm|fmls, s_{hgi}|\sum C_j$ problem is considered for the first time. A mixed integer linear mathematical model and a metaheuristic algorithm based on MA are proposed for the research problem. Also a lower bounding method based on the B&P algorithm is proposed to evaluate the quality of the proposed MA.

The rest of the paper is organized as follows: The characteristics of the problem are described in Sect. 2. A mathematical model for the research problem is proposed in Sect. 3. The MA for obtaining approximate solutions to the problem is explained in Sect. 4. The Lower bounding method is described in Sect. 5. Results of computational experiments to evaluate the performance of the MA and the lower bounding method are reported in Sect. 6. Finally, the results are discussed in Sect. 7 with providing directions for future research.

2 Problem descriptions

Consider a flexible flowshop environment with m stages. Assume that N groups of jobs should be processed within these stages. Each stage (say stage i) has nm_i identical parallel machines. It is assumed that at least one stage has more than one identical machine in parallel. Each group (say group g) consists of n_g jobs. The goal is to find the best sequence of processing the groups as well as the jobs belonging to each group in order to minimize the total completion time. Based on Pinedo (2012) the flexible flowshop sequence-dependent group scheduling problem (FFSDGSP) with minimization of total completion time can be noted as $FFm|fmls, s_{hgi}|\sum C_j$: Flexible flowshop with m stages (FFm); group (family) scheduling problem (*fmls*); sequence-dependent setups (s_{hgi}); and total completion time minimization ($\sum C_j$).

The setups are assumed to be anticipatory. In other words, the setup on a machine at each stage to process a group can be started before any job belonging to that group physically arrives at that stage. The group scheduling assumptions are valid in this problem. In other words, if processing of a job in a group starts on a machine, all jobs within that group should be processed before switching the machine to process the jobs of another group.

3 The mathematical model

A binary mixed integer linear programming formulation is developed for the research problem. In this model, it is assumed that a dummy group, say group 0, is set as the first group on each machine. This assumption is used to calculate the required setup time for the first group on each machine at each stage. It is also assumed that the completion time of this dummy group is equal to 0 at all stages. The notations, parameters, decision variables, and the mathematical model are as follows:

Parameters and notations:

N	Number of groups
g	Index used for representing groups $g = 1, \ldots, N$
m	Number of stages
i	Index used for representing stages $i = 1, \ldots, m$
nm_i	Number of parallel machines at stage i $i = 1, \ldots, m$
n_g	Number of jobs in group g $g = 1, \ldots, N$
p_{gji}	Process time of job j of group g at stage i $g = 1, \ldots, N; j = 1, \ldots, n_g; i = 1, \ldots, m$
s_{hgi}	Setup time for processing group g after group h on one of the machines of stage i $h = 1, \ldots, N; g = 1, \ldots, N; g \neq h; i = 1, \ldots, m$
s_{0gi}	Setup time for processing group g as the first group on any machine of stage i $g = 1, \ldots, N; i = 1, \ldots, m$
M	A large number

Decision variables:

X_{hgi} $= \begin{cases} 1 & \text{If group } g \text{ is processed immediately after group } h \text{ on one of the machines of stage } i \\ 0 & \text{Otherwise} \end{cases}$

$Y_{gj_1j_2i}$ $= \begin{cases} 1 & \text{If in group } g \text{ job } j_2 \text{is processed after job } j_1 \text{at stage } i \\ 0 & \text{Otherwise} \end{cases}$

C_{gji}	Completion time of job j of group g at stage i
ST_{gi}	Starting time of processing the jobs of group g at stage i
FT_{gi}	Completion time of processing the jobs of group g at stage i

The model:

$$\min \sum_{g=1}^{N} \sum_{j=1}^{n_g} C_{gjm} \tag{1}$$

$$\sum_{\substack{h=0 \\ h \neq g}}^{N} X_{hgi} = 1 \qquad g = 1, ..., N; i = 1, ..., m \tag{2}$$

$$\sum_{\substack{g=1 \\ g \neq h}}^{N} X_{hgi} \leq 1 \qquad h = 1, ..., N; i = 1, ..., m \tag{3}$$

$$X_{hgi} + X_{ghi} \leq 1 \qquad h = 1, ..., N; g = 2, ..., N; g > h; i = 1, ..., m \tag{4}$$

$$\sum_{g=1}^{N} X_{0gi} \leq nm_i \qquad i = 1, ..., m \tag{5}$$

$$ST_{gi} \geq FT_{hi} + s_{hgi} - M(1 - X_{hgi}) \quad h = 0, ..., N; g = 1, \ldots, N; g \neq h; i = 1, ..., m \tag{6}$$

$$C_{gji} \geq ST_{gi} + p_{gji} \qquad g = 1, ..., N; j = 1, \ldots, n_g; i = 1, \ldots, m; p_{gji} \neq 0 \tag{7}$$

$$FT_{gi} \geq C_{gji} \qquad g = 1, .., N;\ j = 1, \ldots, n_g;\ i = 1, \ldots, m;\ p_{gji} \neq 0 \tag{8}$$

$$C_{gj_2i} \geq C_{gj_1i} + p_{gj_2i} - M(1 - Y_{gj_1j_2i})$$
$$g = 1, ..., N;\ j_1 = 1, \ldots, n_g;\ j_2 = 2, \ldots, n_g;\ j_2 > j_1; \quad i = 1, \ldots, m;\ p_{gj_2i} \neq 0 \tag{9}$$

$$C_{gj_1i} \geq C_{gj_2i} + p_{gj_1i} - MY_{gj_1j_2i} \quad g = 1, ..., N;\ j_1 = 1, \ldots, n_g;\ j_2 = 2, \ldots, n_g;$$
$$j_2 > j_1; i = 1, \ldots, m;\ p_{gj_1i} \neq 0 \tag{10}$$

$$C_{gji} \geq C_{gj(i-1)} + p_{gji} \quad g = 1, ..., N;\ j = 1, \ldots, n_g; \quad i = 2, \ldots, m \tag{11}$$

$$C_{gji}, ST_{gi}, FT_{gi} \geq 0$$
$$X_{hgi}, Y_{gj_1j_2i} \in \{0, 1\}$$

The objective function is minimization of total completion time as presented by Eq. (1). Constraint set (2) ensures that each group has exactly one preceding group at each stage. Each group has at most one successor group at each stage. Constraint set (3) is incorporated into the model for this reason. Each two groups such as groups h and g may have three different positions compared to each other at any stage such as stage i: (1) group h is processed immediately before group g (2) group h is processed immediately after group g (3) these two groups are not processed exactly after each other (these two groups are processed either on different machines or on the same machine but not immediately after each other). Constraint set (4) is incorporated into the model for this reason. Constraint set (5) is incorporated into the model to ensure that nm_i machines are scheduled at stage i. At each stage such as stage i, nm_i groups can be scheduled after the dummy group and each of these groups are assigned to one of the parallel machines at that stage. The start time of processing each group at each stage is greater than or equal to the completion time of its immediate predecessor group plus the required setup time of that group. Constraint set (6) is incorporated into the model for this reason. Constraint set (7) is incorporated into the model to make sure that the completion time of a job at a stage is greater than the start time of processing its group, plus its processing time at that stage. The completion time of each group is greater than the completion times of its jobs. Constraint set (8) is incorporated into the model for this reason. The difference between the completion times of two jobs on a machine at each stage should be greater than the processing time of the job is processed later. Constraint sets (9) and (10) are incorporated into the model in order to support this fact. Constraint set (11) ensures that the completion time of each job at each stage is greater than or equal to its completion time at the previous stage plus its processing time at this stage.

The classical $1|s_{hg}|\sum C_j$ and $F2||\sum C_j$ problems are NP-hard (Pinedo 2012). Flexible flowshop is a generalization of these problems, so it can be concluded that the $FFm|fmls, s_{hgi}|\sum C_j$ problem is also NP-hard. Thus, metaheuristic algorithms are required to solve large size problems.

4 Metaheuristic–memetic algorithm

Since the research problem has been shown to be NP-hard, six metaheuristic algorithms based on MA are developed to heuristically solve the problem. MA is an evolutionary algorithm introduced by Moscato (1989) as a hybrid GA combined with an individual learning procedure for local refinement. Previous research show that MA has a good performance in scheduling

and timetabling problems (Hart and Krasnogor 2005; França et al. 2005; Keshavarz and Salmasi 2013). Keshavarz and Salmasi (2013) propose a very efficient MA algorithm for the $FFm|fmls, s_{hgi}|C_{\max}$ problem. Motivated from this, we applied the same algorithm to solve the proposed research problem by several modifications in solution representation, calculating the fitness of a solution, crossover operator, and local search procedure. Other features of the applied MA are the same as the proposed MA by Keshavarz and Salmasi (2013). The characteristics of the proposed algorithm are discussed in the following subsections.

4.1 Solution representation

Generally, solutions are represented by a string of digits called chromosome. Different steps of the algorithm such as crossover and mutation are applied on chromosomes.

The algorithm should determine the sequence of groups as well as jobs within each group on machines at each stage. Any sequence of groups followed by sequence of jobs within each group can be considered as a solution. Each permutation of the digits 1 to N represents a priority list for assigning groups to machines at every stage. Similarly, every permutation of the digits 1 to n_g represents the sequence of jobs that belong to the g^{th} group. So, solution representation for each stage can be presented as a string such as $[G_1, G_2,\ldots, G_N \mid J_{11}, J_{12},\ldots, J_{1n_1} \mid \ldots \mid J_{N1}, J_{N2}, \ldots, J_{Nn_N}]$. The first part of this chromosome represents the priority list of groups. The other parts represent the sequence of jobs within each group sequentially. In this way, chromosomes are divided into $N+1$ independent parts. To reduce the search space of the MA, it is assumed that the sequence of jobs within groups does not change through successive stages. The procedure of assigning groups and jobs to machines at each stage is discussed in Sect. 4.10.

4.2 Calculating the total completion time of a chromosome

Assigning groups to the first available machine is not an efficient sequencing rule for the proposed research problem since it does not consider the effect of sequence-dependent setup time between groups. In other words, keeping some machines idle while a group is waiting for processing may result in a better schedule. Hence, the optimal schedule may belong to a non-delay schedule (A feasible schedule is called non-delay if no machine is kept idle while an operation is waiting for processing).

For calculating the fitness of a chromosome, which is the total completion time value of a chromosome, the groups are assigned to machines at each stage based on the priority list of groups in that chromosome. By considering the setup time that is needed for processing a group on a machine (based on the last processed group on that machine), the group is assigned to a machine with the earliest possible start time. After assigning a group to a machine, the completion time of its jobs are computed based on a job sequence of that group. After computing the completion time of jobs at all stages, the sum of completion time of jobs at the last stage is equal to the total completion time.

As an example, consider $G_1, G_2,\ldots, G_N$ as the priority list of groups and let $nm_1 = 3$. First, group G_1 is assigned to the first machine of stage 1. Then, the completion time of its jobs are calculated based on the job sequence of this group. Assume C_1 as the completion time of group G_1 at stage 1 (C_1 is equal to the completion time of the last job of group G_1). Then, group G_2 is assigned to the second machine of stage 1, if $s_{0G_1} \leq C_1 + s_{G_1G_2}$. Otherwise, it is assigned to the first machine and will be processed after group G_1. Assume that group G_2 is assigned to the second machine and its completion time at stage 1 is C_2. If $\min\{s_{0G_3}, C_1 + s_{G_1G_3}, C_2 + s_{G_2G_3}\} = s_{0G_3}$, then group G_3 is assigned to the third machine.

If $\min\{s_{0G_3}, C_1 + s_{G_1G_3}, C_2 + s_{G_2G_3}\} = C_1 + s_{G_1G_3}$, group G_3 is assigned to the first machine and otherwise if $\min\{s_{0G_3}, C_1 + s_{G_1G_3}, C_2 + s_{G_2G_3}\} = C_2 + s_{G_2G_3}$, group G_3 is assigned to the second machine. Assignment of other groups to the machines is done in the same way.

4.3 Crossover operator

Three different crossover operators, namely "Order Crossover" (OX), "Similar Job Order Crossover" (SJOX), and "Partially Mapped Crossover" (PMX) are employed in this research problem. OX crossover is applied by Keshavarz and Salmasi (2013). OX Crossover determines the piece of the first parent by generating two random numbers to copy over to the offspring. The rest of the offspring is completed according to order of alleles in the second parent. SJOX and PMX crossovers are proposed by Ruiz et al. (2003) and Goldberg and Lingle (1985), respectively. SJOX crossover examines each position of the parents and determines identical alleles at the same positions to copy over to both offspring. Therefore, each offspring directly fills up all alleles from one of the parents up to a randomly chosen cut point. PMX crossover partitions each parent two substrings, and exchanges the two substrings to produce proto-offspring. A mapping relationship based on the selected substrings is determined, and proto-offspring is legalized.

4.4 Local search procedure

Two local search procedures as (1) Swap and the Insertion Moves (SIM) and (2) Path Relinking (PR) are considered as the local search procedures. Keshavarz and Salmasi (2013) propose MA based on SIM local search. In swap moves, the position of two alleles are changed, whereas in insertion moves one allele is removed from its position and inserted in another one. PR is a search technique originally presented by Glover and Laguna (1997) where the goal is to explore the search space or "path" between a given set (usually two) of good solutions. The steps of the proposed MA are presented as a flowchart in Fig. 2 in Appendix 1.

5 Lower bounding method: the branch and price algorithm

The B&P algorithm has been known as an efficient method for solving integer linear programming problems with huge number of variables. The details of the B&P algorithm can be found in Barnhart et al. (1998), Wilhelm (2001), and Wilhelm et al. (2003). Salmasi et al. (2010) and Gelogullari and Logendran (2010) used B&P algorithm to find a lower bound for the flowshop group scheduling problem.

In this research, we propose a B&P algorithm for finding the lower bound of the research problem. The reformulated mathematical model, the column generation approach, and the details of the proposed B&P algorithm are discussed in the following sections.

5.1 Dantzig–Wolf decomposition model

A decomposition formulation for FFSDGSP by reformulating it as a set-partitioning master problem (MP) by using Dantzig–Wolfe decomposition is proposed. The following parameters and decision variables beside the defined ones in Sect. 3 are required to present this model:

Parameters:

B_i Set of all feasible schedules at stage $i \quad i = 1, \ldots, m$

K_i Number of all feasible schedules at stage $i \quad i = 1, \ldots, m$

b_i^k The k^{th} feasible schedule at stage $i \quad i = 1, \ldots, m; k = 1, \ldots K_i$

C_{gji}^k Completion time of job j of group g at stage i in schedule $b_i^k \quad g = 1, \ldots, N; j = 1, \ldots, n_g; i = 1, \ldots, m$

Decision variables:

$$\lambda_i^k = \begin{cases} 1 & \text{If schedule } b_i^k \in B \text{ is selected at stage } i \\ 0 & \text{Otherwise} \end{cases} \qquad i = 1, \ldots, m; k = 1, \ldots K_i$$

Each column corresponds to a feasible schedule of groups and jobs at a stage.The set of all feasible schedules at stage i is denoted by B_i. Each of these schedules $b_i^k, k = 1, \ldots, K_i$, where $K_i = |B_i|$, is defined by a set of completion times $\left\{C_{gji}^k\right\}$. MP is an integer programming model with a huge number of binary variables. Since there are a huge number of feasible schedules at each stage, it is not possible to include them all in the model. Therefore, column generation approach is used to solve LP relaxation of MP by adding new schedules with negative reduced costs to the model as needed. The linear programming master problem, which includes only a subset of all possible schedules, is called the restricted linear master problem (RLMP) and can be written as below:

RLMP Model:

$$\min Z = \sum_{k=1}^{K_m} \sum_{g=1}^{N} \sum_{j=1}^{n_g} \lambda_m^k C_{gjm}^k \tag{12}$$

s.t.

$$\sum_{k=1}^{K_i} \lambda_i^k C_{gji}^k - \sum_{k=1}^{K_{(i-1)}} \lambda_{(i-1)}^k C_{gj(i-1)}^k \geq p_{gji} \qquad \begin{matrix} i = 2, \ldots, m; \\ g = 1, \ldots, N; \\ j = 1, \ldots, n_g \end{matrix} \quad (\omega_{gji}) \tag{13}$$

$$\sum_{k=1}^{K_i} \lambda_i^k = 1 \qquad i = 1, \ldots, m \; (\alpha_i) \tag{14}$$

$$\lambda_i^k \geq 0 \qquad \begin{matrix} i = 1, \ldots, m; \\ k = 1, \ldots, K_i \end{matrix} \tag{15}$$

The objective is to minimize the total completion time as presented by equation (12). Constraint set (13) ensures that the completion time of a job at a stage (say stage i) is greater than or equal to the completion time of the job at the preceding stage (stage $i-1$) plus the process time of the job at that stage. This constraint set deals with the completion time of jobs at more than one stage. Thus, this constraint set is considered as the linking (complicating) constraint in the model. Constraint set (14) is the convexity constraint and ensure that only one schedule is selected at each stage. Variables λ_i^k are originally binary that are relaxed to continuous decision variables in the RLMP.

ω_{gji} and α_i denote the dual variables corresponding to constraint sets (13) and (14), respectively. The dual problem of RLMP is presented to facilitate the presentation of the sub-problems. Sub-problems are used to identify if there is any column to add to RLMP to improve the objective function value. The dual of RLMP is generated as the following:

The Dual of RLMP Model:

$$\max Z^D = \sum_{i=2}^{m} \sum_{g=1}^{N} \sum_{j=1}^{n_g} p_{gji} \omega_{gji} + \sum_{i=1}^{m} \alpha_i \tag{16}$$

s.t.

$$-\sum_{g=1}^{N}\sum_{j=1}^{n_g}\omega_{gj2}C_{gj1}^{k}+\alpha_1\le 0 \qquad k=1,\dots K_1 \tag{17}$$

$$\sum_{g=1}^{N}\sum_{j=1}^{n_g}\omega_{gji}C_{gji}^{k}-\sum_{g=1}^{N}\sum_{j=1}^{n_g}\omega_{gj(i+1)}C_{gji}^{k}+\alpha_i\le 0 \qquad i=2,\dots,m-1;k=1,\dots,K_i \tag{18}$$

$$\sum_{g=1}^{N}\sum_{j=1}^{n_g}(\omega_{gjm}-1)C_{gjm}^{k}+\alpha_m\le 0 \qquad k=1,\dots,K_m \tag{19}$$

$$\omega_{gji}\ge 0 \qquad i=2,\dots,m;g=1,\dots,N;j=1,\dots,n_g \tag{20}$$

$$\alpha_i \qquad \text{Unrestricted} \qquad i=1,\dots,m \tag{21}$$

Define $\omega_{gj1}=0$ and $\omega_{gj(m+1)}=1$ for $g=1,\dots,N$ and $j=1,\dots,n_g$. Then, in order to find the schedule with the smallest reduced cost at the ith stage, it is needed to solve the following sub-problem (SPi):

SPi Model ($i=1,\dots,m$):

$$\min Z^{SPi}=\sum_{g=1}^{N}\sum_{j=1}^{n_g}\left(\omega_{gj(i+1)}-\omega_{gji}\right)C_{gji}-\alpha_i \tag{22}$$

s.t.

capacity constraint sets for nm_i parallel machines at stage i (i.e., constraints (2)–(10) of the original model for a specific i)

The last terms in (22) i.e., α_i, is constant. Thus, solving each sub-problem is equivalent to solving a parallel machine sequence-dependent group scheduling problem with total weighted completion time as objective function i.e., $Pm|fmls,s_{hgi}|\sum w_jC_j$. Kan (1976) shows that $1|fmls,s_{hgi}|\sum w_jC_j$ is strongly NP-hard. So it can be concluded that $Pm|fmls,s_{hgi}|\sum w_jC_j$ is also strongly NP-hard.

Thus, all sub-problems have similar structure and are strongly NP-hard. If the optimal values of at least one of the sub-problems are negative, there are column(s) that can be added to the master problem and improve the objective function value. The process of solving sub-problems and adding new columns to RLMP continues until all sub-problems have positive optimal objective function values.

In SP2 through m, the objective function coefficient of C_{gji} i.e., $\left(\omega_{gj(i+1)}-\omega_{gji}\right)$can be negative. In this case, the sub-problem would be unbounded. This causes the column generation converges to optimal solution very slowly. In order to resolve this issue, a constraint set is incorporated into the dual of the RLMP model to enforce all coefficients to be positive. The following constraints are incorporated into the dual of RLMP:

$$\left(\omega_{gj(i+1)}-\omega_{gji}\right)\ge 0 \qquad i=2,\dots,m;g=1,\dots,N;j=1,\dots,n_g \tag{23}$$

To achieve this purpose, artificial variables R_{gji} are added to the RLMP. Then, the RLMP model with artificial variables (RLMP-A) would be as follows:

RLMP-A Model:

$$\min Z=\sum_{k=1}^{K_m}\sum_{g=1}^{N}\sum_{j=1}^{n_g}\lambda_m^kC_{gjm}^{k}+\sum_{g=1}^{N}\sum_{j=1}^{n_g}R_{gjm} \tag{24}$$

s.t.

$$\sum_{k=1}^{K_2} \lambda_2^k C_{gj2}^k - \sum_{k=1}^{K_1} \lambda_1^k C_{gj1}^k + R_{gj2} \geq p_{gj2} \qquad g = 1, \ldots, N;\ j = 1, \ldots, n_g \tag{25}$$

$$\sum_{k=1}^{K_i} \lambda_i^k C_{gji}^k - \sum_{k=1}^{K_{(i-1)}} \lambda_{(i-1)}^k C_{gj(i-1)}^k + R_{gji} - R_{gj(i-1)} \geq p_{gji}$$
$$i = 3, \ldots, m;\ g = 1, \ldots, N;\ j = 1, \ldots, n_g \tag{26}$$

$$\sum_{k=1}^{K_i} \lambda_i^k = 1 \qquad i = 1, \ldots, m \tag{27}$$

$$\lambda_i^k \geq 0 \qquad i = 1, \ldots, m;\ k = 1, \ldots, K_i \tag{28}$$

$$R_{gji} \geq 0 \qquad i = 2, \ldots, m;\ g = 1, \ldots, N;\ j = 1, \ldots, n_g \tag{29}$$

Since a few constraints are incorporated into the dual model, the primal model provides a lower bound for the LP relaxation of MP.

5.2 Solving sub-problems

During column generation approach, sub-problems should be solved so many times. Solving sub-problems using integer programming model has shown a very poor performance in Salmasi et al. (2010) for a similar problem. For this reason, in this research two efficient algorithms are developed for solving the sub-problems heuristically and optimally.

It is not necessary that the sub-problems be solved optimally during the intermediate iterations of column generation approach and finding an improving column is adequate. Thus, a metaheuristic is used to solve sub-problems during early iterations. When the metaheuristic is unable to find a suitable column for all sub-problems, the sub-problems are solved optimally. At the end of the column generation approach, all sub-problems must be solved optimally to make sure that the optimal solution of the node is found. In this research, MA is used for solving sub-problems heuristically. After solving the sub-problems using the metaheuristic, if at least one of the sub-problems have a negative objective function, new columns are added to the RLMP-A. By solving updated RLMP-A, new dual values are obtained to update sub-problem objective functions. Adding new columns by solving sub-problems heuristically is continued until all sub-problems are unable to provide a column that improves the objective function value of RLMP-A.

To ensure the optimality of RLMP-A solution, the sub-problems should be solved by an optimal algorithm. A branch and bound (B&B) algorithm is developed to solve the sub-problems optimally. If the optimal solution of one of the sub-problems becomes negative, the new generated column is added to RLMP-A. By solving updated RLMP-A, we obtain new dual values and update sub-problems objective functions. Again the MA is used for solving sub-problems. Finally, the column generation approach stops when the MA cannot find new columns and the optimal solution of all sub-problems (based on the B&B algorithm) are positive. In this case the optimality of the LP relaxation of master problem is proved. After solving RLMP-A, if the values of λ_i^k do not satisfy the integrality condition, branching is occurring. The flowchart of column generation procedure is presented in Appendix 1 as Fig. 3.

5.2.1 Job sequence in sub-problem optimal solution

For each assignment of groups to parallel machines at a stage, the optimal sequence of jobs within each group can be found by using a sorting rule. For a group that assigned to a machine finding its optimal job sequence is equivalent to finding the optimal solution of a $1||\sum w_j C_j$ problem (Pinedo 2012). So in each sub-problem, jobs should be sorted based on the weighted shortest processing time (WSPT) rule. The WSPT rule for each SPi ($i = 1$ through m) can be defined as follows:

WSPT rule for SPi ($i = 1$ **through** m):

Sort jobs within each group in decrease order based on the value of $(\omega_{gj(i+1)} - \omega_{gji})/p_{gji}$

This rule is used in both MA and B&B algorithms for finding the optimal job sequences within each group.

5.2.2 The Branch and Bound algorithm for solving sub-problems

All sub-problems are $Pm|fmls, s_{hgi}|\sum w_j C_j$ and the job sequence within each group can be determined using the WSPT rule. In order to find the optimal sequence of groups, each group is considered as an aggregated job. The process time of this aggregated job is equal to the sum of the process times of the jobs belonging to that group. Assume $j_{(1)}, j_{(2)}, \ldots, j_{(n_g)}$ is the optimal job sequence of group g, $g = 1, \ldots, N$. Assume Ct_g as the completion time of group g. The completion time of each job can be represented as:

$$C_{gj_{(r)}i} = Ct_g - \sum_{t=r+1}^{n_g} p_{gj_{(t)}i} \quad r = 1, \ldots, n_g \tag{30}$$

The objective function of the SPi, $i = 1, \ldots, m$ can be written as the following:

$$\begin{aligned}
Z^{SPi} &= \sum_{g=1}^{N}\sum_{j=1}^{n_g}\left(\omega_{gj(i+1)} - \omega_{gji}\right) C_{gji} - \alpha_i = \sum_{g=1}^{N}\sum_{r=1}^{n_g}\left(\omega_{gj_{(r)}(i+1)} - \omega_{gj_{(r)}i}\right) C_{gj_{(r)}i} - \alpha_i \\
&= \sum_{g=1}^{N}\sum_{r=1}^{n_g}\left(\omega_{gj_{(r)}(i+1)} - \omega_{gj_{(r)}i}\right)\left(Ct_g - \sum_{t=r+1}^{n_g} p_{gj_{(t)}i}\right) - \alpha_i \\
&= \sum_{g=1}^{N}\sum_{r=1}^{n_g}\left(\omega_{gj_{(r)}(i+1)} - \omega_{gj_{(r)}i}\right) Ct_g - \sum_{g=1}^{N}\sum_{r=1}^{n_g}\sum_{t=r+1}^{n_g}\left(\omega_{gj_{(r)}(i+1)} - \omega_{gj_{(r)}i}\right) p_{gj_{(t)}i} - \alpha_i \\
&= \sum_{g=1}^{N}\sum_{r=1}^{n_g}\left(\omega_{gj_{(r)}(i+1)} - \omega_{gj_{(r)}i}\right) Ct_g - constant_i \\
&= \sum_{g=1}^{N}\left(\sum_{j=1}^{n_g}\left(\omega_{gj(i+1)} - \omega_{gji}\right)\right) Ct_g - constant_i
\end{aligned} \tag{31}$$

Thus, the SP1 through m can be considered as $Pm|s_{hg}|\sum w_j C_j$ problems with N aggregated jobs and the weight of each aggregated job g is $\left(\sum_{j=1}^{n_g}\left(\omega_{gj(i+1)} - \omega_{gji}\right)\right)$.

A B&B algorithm is developed to solve the $Pm|s_{hg}|\sum w_j C_j$ problem to find the optimal group sequence for every $Pm|fmls, s_{hgi}|\sum w_j C_j$ problem. The B&B algorithm can also be used to get a valid lower bound for each sub-problem. Every B&B algorithm consists of three main procedures: initialization, branching and bounding. The solution of MA is used as

an initial upper bound for the B&B tree (initialization). Each node of the B&B tree consists of a partial solution. Let $\mathcal{S}$ be the set of scheduled groups and $\mathcal{S}'$ be the complementary set of $\mathcal{S}$ or the set of unscheduled groups. At each level of B&B tree, one of the unscheduled groups is added to the scheduled groups set (Branching). It is clear that each parent node at level l generates$(N-l)$ new nodes at level $(l+1)$. At level 0 of B&B tree, $\mathcal{S}=\varnothing$ and at each node of level $l=1,\ldots,N-2$, $\mathcal{S}$ contains l groups. There are $P(N,l)=N!/(N-l)!$ nodes at level l. $\mathcal{S}$ is a priority list and scheduled groups are assigned to machines based on this priority list. Each group is assigned to a machine with the earliest possible starting time (See Sect. 4.10 for more details about assigning groups to machines).

At each node, the total weighted completion time of scheduled groups ($\mathcal{S}$) and the lower bound of total weighted completion time of unscheduled groups ($\mathcal{S}'$) are calculated (Bounding). The summation of these two values provides the lower bound value associated with that node. For $\mathcal{S}=\left\{g_{(1)},g_{(2)},\ldots,g_{(l)}\right\}$, the total weighted completion time of scheduled groups ($Z^{\mathcal{S}}$) for each SPi ($i=1$ through m) is represented as formula (35):

$$Z^{\mathcal{S}}=\sum_{t=1}^{l}\left(\sum_{j=1}^{n_{g_{(t)}}}\left(\omega_{g_{(t)}j(i+1)}-\omega_{g_{(t)}ji}\right)\right)Ct_{g_{(t)}} \tag{32}$$

where $Ct_{g_{(t)}}, t=1,\ldots,l$ is calculated based on the assignment of groups to nm_i parallel machines at stage i.

For calculating the lower bound of the total weighted completion time of unscheduled groups($\mathcal{S}'$), each unscheduled group such as group g is considered as a job with process time equal to:

$$\left(\min_{h}\left\{s_{hgi}\right\}+\sum_{j=1}^{n_g}p_{gji}\right)\Big/nm_i \tag{33}$$

In other words, the minimum possible setup time for processing group g is added to the processing time of this group and then the result is divided by the number of parallel machines. Then, the unscheduled groups are assigned to a single machine with ready time equal to the minimum ready time of parallel machines after processing the scheduled groups. Ready time of each machine is the completion time of the last group that is processed on that machine. Let r as the minimum ready time. With this assumption, the problem of finding the sequence of unscheduled groups reduces to $1||\sum w_jC_j$. The optimal sequence for $1||\sum w_jC_j$ can be found by using the WSPT rule. Based on the WSPT rule, for each SPi ($i=1$ through m), the unscheduled groups should be sorted in decrease order based on the value of

$$\left(\sum_{j=1}^{n_g}\left(\omega_{gj(i+1)}-\omega_{gji}\right)\right)\Big/\left(\left(\min_{h}\left\{s_{hgi}\right\}+\sum_{j=1}^{n_g}p_{gji}\right)\Big/nm_i\right).$$

Assume that after using the WSPT rule, the sorted unscheduled groups are $g_{(l+1)}, g_{(l+2)},\ldots,g_{(N)}$. Then, the completion time of unscheduled groups can be calculated using the following formulas:

$$Ct_{g_{(l+1)}}=r+\left(\min_{h}\left\{s_{hg_{(l\ell+1)}i}\right\}+\sum\nolimits_{j=1}^{n_{g(l+1)}}p_{g_{(l+1)}ji}\right)\Big/nm_i \tag{34}$$

$$Ct_{g_{(t)}}=Ct_{g_{(t-1)}}+\left(\min_{h}\left\{s_{hg_{(t)}i}\right\}+\sum\nolimits_{j=1}^{n_g}p_{g_{(t)}ji}\right)\Big/nm_i,\quad t=\ell l+2,\ldots,N \tag{35}$$

In this case, the lower bound of total weighted completion time of unscheduled groups ($Z^{S'}$) for each sub-problem can be calculated based on formula (39):

$$Z^{S'} = \sum_{t=\ell l+1}^{N} \left(\sum_{j=1}^{n_{g_{(t)}}} \left(\omega_{g_{(t)}j(i+1)} - \omega_{g_{(t)}ji} \right) \right) Ct_{g_{(t)}} \tag{36}$$

Then, the lower bound of each SPi ($i = 1$ through m) associated with each node can be calculated by adding Z^{S}, $Z^{S'}$, and the constant value related to each sub-problem. These values are represented as formula (40):

$$Z^{S} + Z^{S'} - constant_i = \sum_{t=1}^{l} \left(\sum_{j=1}^{n_{g_{(t)}}} \left(\omega_{g_{(t)}j(i+1)} - \omega_{g_{(t)}ji} \right) \right) Ct_{g_{(t)}} + \sum_{t=\ell l+1}^{N} \times \left(\sum_{j=1}^{n_{g_{(t)}}} \left(\omega_{g_{(t)}j(i+1)} - \omega_{g_{(t)}ji} \right) \right) Ct_{g_{(t)}} - \sum_{g=1}^{N} \sum_{r=1}^{n_g} \sum_{t=r+1}^{n_g} \left(\omega_{gj_{(r)}(i+1)} - \omega_{gj_{(r)}i} \right) p_{gj_{(t)}i} - \alpha_i \tag{37}$$

A node is fathomed if its lower bound be greater than or equal to the current upper bound or be at the level $N-2$ of the B&B tree (in this level the sequence of all groups are determined). Whenever the algorithm reaches to a node at level $N-2$ if its objective function value is lower than the current upper bound, the upper bound is updated. The best first search policy is used for traversing B&B tree.

In order to explain how the B&B algorithmis implemented, an example for $P2|s_{hg}|\sum w_j C_j$ with four jobs is presentedin Appendix 2.

5.3 Branching method for branch and price algorithm

After solving the RLMP-A optimally, it is not guaranteed that the values of λ_i^k be integer (0 or 1). In this case the branching is occurred in order to enhance the value of the lower bound. Applying a standard B&B procedure to the RLMP-A with its existing columns will not guarantee an optimal (or even feasible) solution (Barnhart et al. 1998). Barnhart et al. (1994) and Desrosiers et al. (1995) suggest to branch on the decision variables of the original problem. The decision variables related to the sequence of groups or X_{hgi} are selected for branching in this research. If X_{hgi} is selected for branching,in one of the generated nodes in the next level of B&P tree, group h is processed exactly before group g (on one of the parallel machines at stage i) and in the other node, group g is not processed exactly after group h (the two groups are processed on different machines or on the same machine but group$g$is not processed exactly after group h).

To find the best decision variable for branching, for each two groups h and g and each stage i, a branching index is calculated. The index is equal to the sum of λ_i^k which in its corresponding schedule, group h is processed exactly before group g. This index may vary from 0 to 1. If this index is close to 0, then it means that in most schedules of stage i, group h is not processed exactly before group g. If this index is close to 1, then it means that in most schedules of stage i, group h is processed exactly before group g. So, the decision variable with closer branching index to 0.5 is selected for branching.

5.4 Branch and price algorithm 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, a time limitation criterion is applied. The B&P algorithm terminates after

15 minutes runtime. The B&P algorithm terminates when the algorithm solves the problem or the time limitation reached. If the B&P algorithm solves the problem, it provides a lower bound for the original problem. If in a problem, all nodes are not solved 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. In other words the B&P tree is traversed based on breadth first search order and the minimum value of the nodes which are solved optimally, but their branches are not solved optimally is reported as the lower bound for the original problem. A simple flowchart of B&P algorithm is presented in Appendix 1 as Fig. 4.

6 Computational Results

Salmasi et al. (2010) generate a set of test problems for the $Fm|fmls, s_{hgi}, prmu|\sum C_j$ problem with two, three, and six-stage problems, separately. These test problems are generalized for this research problem. To use these test problems for the FFSDGSP, motivated from Logendran et al. (2006a), the number of machines at each stage are considered randomly between one to three machines. The specifications of the test problems are as follows:

- The number of groups is between two to 16 in three different categories: small (problems with 2–5 groups), medium (problems with six to 10 groups), and large (problems with 11 to 16 groups).
- The number of jobs in each group is between two to 10.
- The ratio of setup times of processing groups on machines in consecutive stages is defined by three levels. The setup time of processing a group at a stage can be less, equal, or greater than the setup time of processing the group at the next stage.

These specifications are the ones used to generate a test problem. Then, each test problem is solved by the three crossovers as an algorithm factor by applying one of the two local search procedures. Based on this explanation, each experimental unit of the first three factors (which generate a test problem) is split into six different parts to be solved by one of the combinations of the crossovers and the local search procedures. Based on Salmasi and Logendran (2008), the split plot design is the most appropriate model to compare the results. The model is a mixed model since it includes fixed factors (groups, jobs, ratio of setup times, algorithms, and local search procedures) as well as a random factor (problem instances). The model of the experiment for a three stage problem can be represented as:

$$Y_{ijklmnr} = \mu + G_i + J_j + R1_k + R2_l + T_{t(ijkl)} + \alpha_m + L_n + \varepsilon_{ijklmnr} \tag{38}$$

where μ is the overall mean, G_i is the effect of group factor ($i = 1, 2, 3$), J_j is the effect of job factor ($j = 1, 2, 3$), $R1_k$ is the ratio of set-up time of M_1/M_2 factor ($k = 1, 2, 3$), $R2_l$ is the ratio of set-up time of M_2/M_3 factor ($l = 1, 2, 3$), T_t is the block factor (a random factor) t=1, …, 162 (162 is the number of test problems), α_m is the effect of algorithm factor ($m = 1, 2, 3$), L_n is the effect of local search procedures factor ($n = 1, 2$) and $\varepsilon_{ijklmnr}$ is the error term,

The goals of performing the experimental design are:

- Find the algorithm with the best performance.
- Identify if there is a statistically significant difference between the performances of local search procedures.

Table 1 The average of the objective function values for the test problems by using different local search procedures

Local search procedures	2 Stage problems	3 Stage problems	6 Stage problems
SIM	11661.46	16467.85	39517.65
PR	11688.83	16489.05	39571.42

Table 2 The average of the objective function values by using different algorithms

Algorithms	OX, SIM	PMX, SIM	SJOX, SIM
2-Stage problems	11658.35	11662.13	11663.93
3-Stage problems	16463.18	16469.58	16470.79
6-Stage problems	39517.56	39515.5	39519.91
Average	22546.36	22549.07	22551.54

The hypothesis tests to investigate for these goals are:

$$\begin{cases} H_0 : \alpha_1 = \alpha_2 = \alpha_3 \\ H_1 : \text{if any of the } \alpha's \text{ is different from the others} \end{cases}$$
$$\begin{cases} H_0 : L_1 = L_2 \\ H_1 : L_1 \neq L_2 \end{cases}$$

The Microsoft visual C++ 2008 is used to code the different versions of the MA. The proposed B&P algorithm is also coded with visual C++ 2008 using ILOG CPLEX (version 12.1) concert technology. The test problems are run on a laptop with 2.1 GHz CPU and 2 GB RAM. The experimental design is coded with Statistical Analysis System, SAS, Release 9.1, to find the best proposed MA. A significance level of 5 % is used in this experiment. Each problem is solved for 30 s by each algorithm.

The results of the experiment are presented in Appendix 3 for two, three, and six stage problems, separately. The results show that there is not a significant difference among the algorithms (the p-values of A (Algorithm) are equal to 0.9431, 0.9728, and 0.9422 for two, three, and six stage problems, separately). But there is a significant difference between the local search procedures (The p-values of L (local search procedures) are less than 0.0001 for two, three, and six stage problems in Tables 15, 16 and 17). Thus, there is a significant difference between using the local search procedures. The average of the objective function values for the test problems by using the SIM and PR local search procedures are presented in Table 1.

Since the SIM local search algorithm provides solutions with the lower average, it can be considered as the better one. Since there is not any significant difference among the performance of the algorithms, the average of the objective function values of the algorithms by considering SIM as the local search algorithm procedure is calculated as presented in Table 2.

In order to compare the performance of the algorithms based on the required time to reach to the best solution in test problems, the time spent to find the best solution in the test problems by using SIM as the local search algorithm procedure is reported in Table 3 and Fig.1. As it is shown in Table 3, the average time of finding the best solution by OX crossover with SIM local search is lower than the other ones. The details of this comparison are presented

Table 3 The average time of finding the best solution by using different algorithms

Algorithms	OX, SIM	PMX, SIM	SJOX, SIM
2-Stage problems	10.75	9.55	14.96
3-Stage problems	14.34	16.3	22.77
6-Stage problems	13.21	14.01	18.11

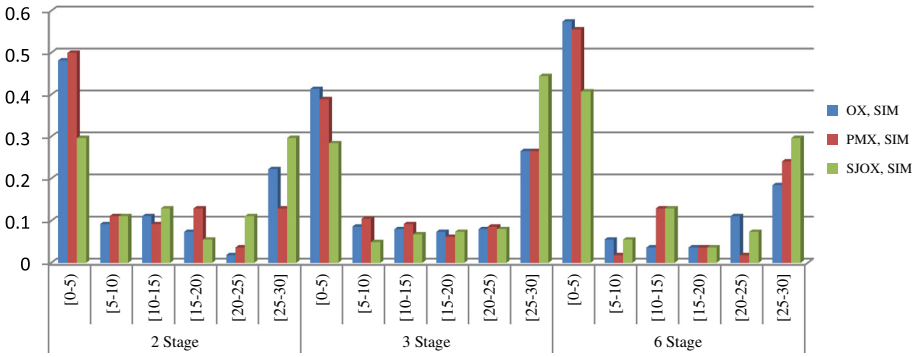

Fig. 1 Runtime percentage histogram

Table 4 The average percentage gap for the MA from the lower bound

Problem	Number of test problems	Average percentage gap
2-Stage	54	5.16
3-Stage	162	5.53
6-Stage	54	8.91
Total	270	6.03

in Fig.1 by presenting the distribution of the time required to find the best solution in the 30 seconds time interval applied to solve each problem.

In order to evaluate the performance of the lower bounding method as well as the proposed MA (MA with OX crossover and SIM local search), all test problems are solved by these algorithms. The percentage gap of the MA is calculated based on this formula for test problems:

$$(\text{The MA solution} - \text{The lower bound solution})/(\text{The lower bound solution})$$

The average percentage gap of the MA is presented in Table 4. The average percentage gap for all test problems is 6.03 %. It shows that both upper and lower bounding methods are efficient. The average percentage gap of the MA based on group size for two, three, and six-stage problems, separately are presented in Tables 5, 6 and 7.

The distribution of percentage gap for two, three, and six-stage problems separately are also presented in Tables 8, 9 and 10.

To speed up the column generation convergence, artificial variables are added to the master problem. It restricts the dual space and lead to a lower bound for the LP relaxation. The average percentage gap of the MA shows that the lower bounding method is efficient and the effect of artificial variables is not considerable.

Table 5 The average percentage gap for 2-stage problems

Group size	Number of test problems	Average percentage gap
Small	18	3.46
Medium	18	6.26
Large	18	5.76
Total	54	5.16

Table 6 The average percentage gap for 3-stage problems

Group size	Number of test problems	Average percentage gap
Small	54	4.78
Medium	54	5.73
Large	54	6.07
Total	162	5.53

Table 7 The average percentage gap for 6-stage problems

Group size	Number of test problems	Average percentage gap
Small	18	6.82
Medium	18	10.95
Large	18	7.47
Total	54	8.91

Table 8 The distribution of percentage gap for 2-stage problems

Gap (%)	[0–1]	(1–5]	(5–10]	(10–15]	(15–100]
Probability (%)	11.11	44.44	29.63	12.96	1.85
Cumulative Probability (%)	11.11	55.56	85.19	98.15	100.00

Table 9 The distribution of percentage gap for 3-stage problems

Gap (%)	[0–1]	(1–5]	(5–10]	(10–15]	(15–100]
Probability (%)	9.88	41.98	31.49	14.20	2.47
Cumulative Probability (%)	9.88	51.85	83.33	97.53	100.00

Table 10 The distribution of percentage gap for 6-stage problems

Gap (%)	[0–1]	(1–5]	(5–10]	(10–15]	(15–100]
Probability (%)	31.48	16.67	20.37	3.70	27.78
Cumulative Probability (%)	31.48	48.15	68.52	72.22	100.00

Table 11 The comparison among the optimal solution, the memetic algorithm, and the lower bound for small size problems

Problem	No. solved problems	Largest solved problem		Average of $100(MA - Opt.)/Opt.$	Average of $100(Opt. - LB)/Opt.$
		No. group	No. job		
2-Stage	9	5	16	0.00	2.46
3-Stage	22	5	17	0.00	4.54
6-Stage	7	4	21	0.06	2.05

Table 12 The comparison among the average percentage gap of B&P and CPLEX

Group size	2-stage		3-stage		6-stage	
	B&P (15 min)	CPLEX (60 min)	B&P (15 min)	CPLEX (60 min)	B&P (15 min)	CPLEX (60 min)
Small	3.69	11.27	5.95	20.67	10.03	2.37
Medium	6.45	19.28	7.08	47.09	15.90	62.27
Large	7.44	90.54	6.49	87.20	6.16	88.19
Total	5.94	36.95	6.55	47.73	10.71	45.79

The proposed mathematical model is used to solve small size problems optimally. The number of small size problems that are solved optimally, the largest solved problem, the average percentage error of the MA, and the average percentage error of the lower bounding method using the B&P algorithm are presented in Table 11.

The quality of solutions provided by the proposed lower bound is compared with a standard lower bound that is obtained by CPLEX. Totally 36 Random test problems (12 in each category of 2, 3 and 6-stage problems) are solved by CPLEX and the percentage gaps are reported after 1 hour. The average percentage gap of CPLEX is compared with the average percentage gap of the proposed lower bounding method in Table 12. The results show that the average percentage gap of the proposed B&P is significantly lower than the lower bounds provided by the CPLEX and so the proposed method outperforms CPLEX.

There is not any available research for the proposed research problem in the literature. Thus, the proposed methods cannot be compared with the result of any other research. However, Salmasi et al. (2010) develop several upper bounding methods and a lower bounding method based on the B&P algorithm for the $Fm|fmls, s_{hgi}, prmu|\sum C_j$ problem. Their problem is much easier than the research problem in this paper, since the flexible flowshop problem is a generalization of the flowshop problem.Their algorithms are run on a Power Edge 2650 with 2.4 GHz Xeon, and 4 GB RAM. They consider 4 hours as their stopping criterion and only solved 43 out of 270 test problems. The average percentage gap of their proposed algorithms (the ACO as the upper bound and the B&P algorithm as the lower bound) for these 43 test problems is 14.76 %. However, all of these test problems are solved for the flexible flowshop problem using the proposed methods in this research in 15 min time limit and the average percentage gap 6.03 % is reached. The superiority of the proposed algorithm in this research is mainly due

to more efficient algorithms for solving sub-problems and the more balanced branching rule.

7 Conclusions and suggestions for future research

In this research, a mathematical model, a metaheuristic algorithm and a lower bounding method based on the B&P algorithm are proposed for the FFSDGSP. Some experiments are performed based on test problems ranging in size from small, medium, to large for two, three, and six-stage problems. The performance of the MA is compared with the result of the lower bounding method. The average percentage gap of the MA is 6.03% for randomly generated test problems. To reduce the search space and hence the run time of the metaheuristic algorithm, it is assumed that the sequence of jobs within groups does not change through successive stages in the MA. However, the results show this assumption can speed up solving the problem by the algorithms without significant effect on the quality of solution.

Recognizing the industrial relevance of FFSDGSP, further research can be performed by considering other optimization criteria such as minimization of total tardiness and earliness. Other exact, heuristic and lower bounding algorithms can be developed for the proposed research problem.

Appendices

Appendix 1: Flow charts for the proposed algorithms

See Figs. 2, 3 and 4.

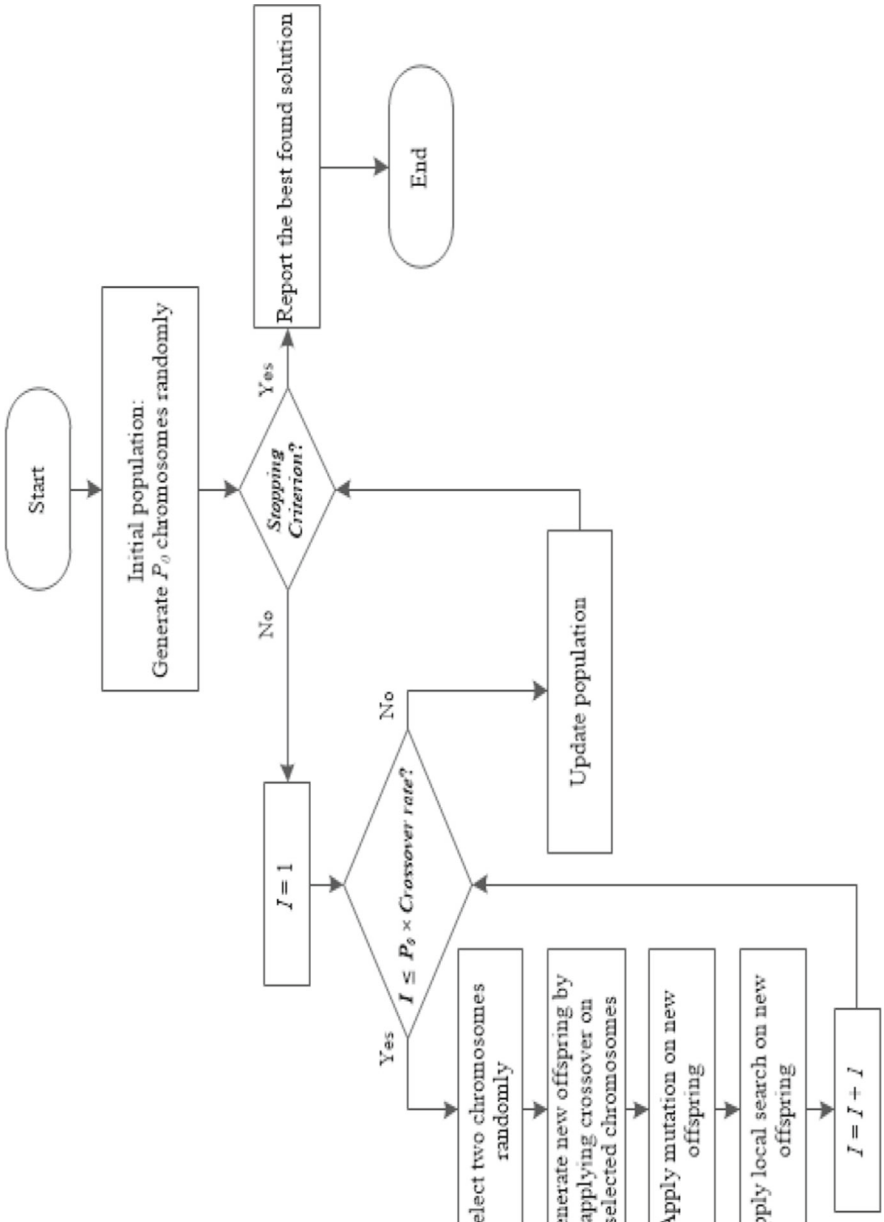

Fig. 2 The flow chart of the proposed memetic algorithm

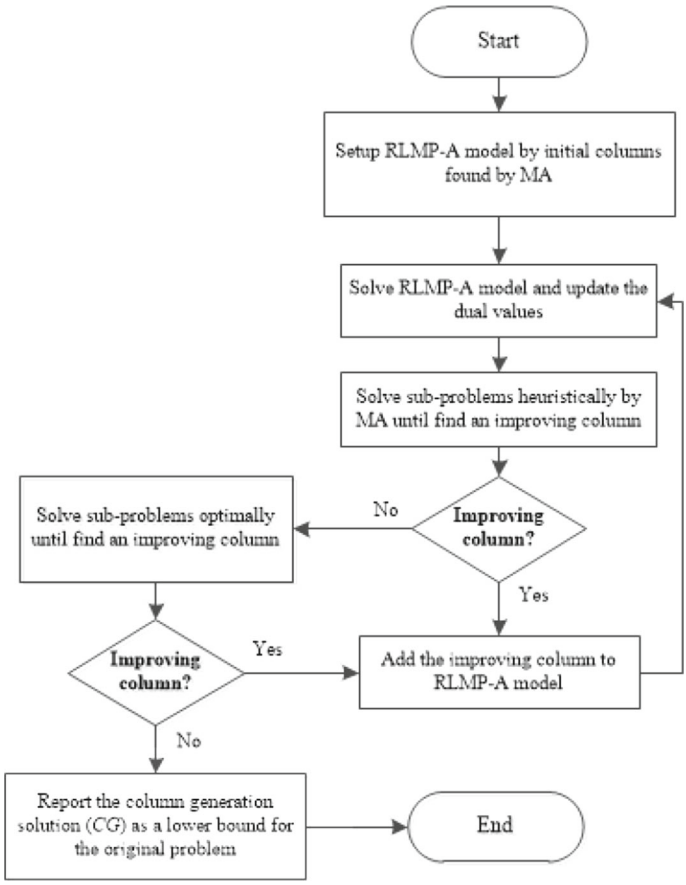

Fig. 3 The flow chart of column generation procedure

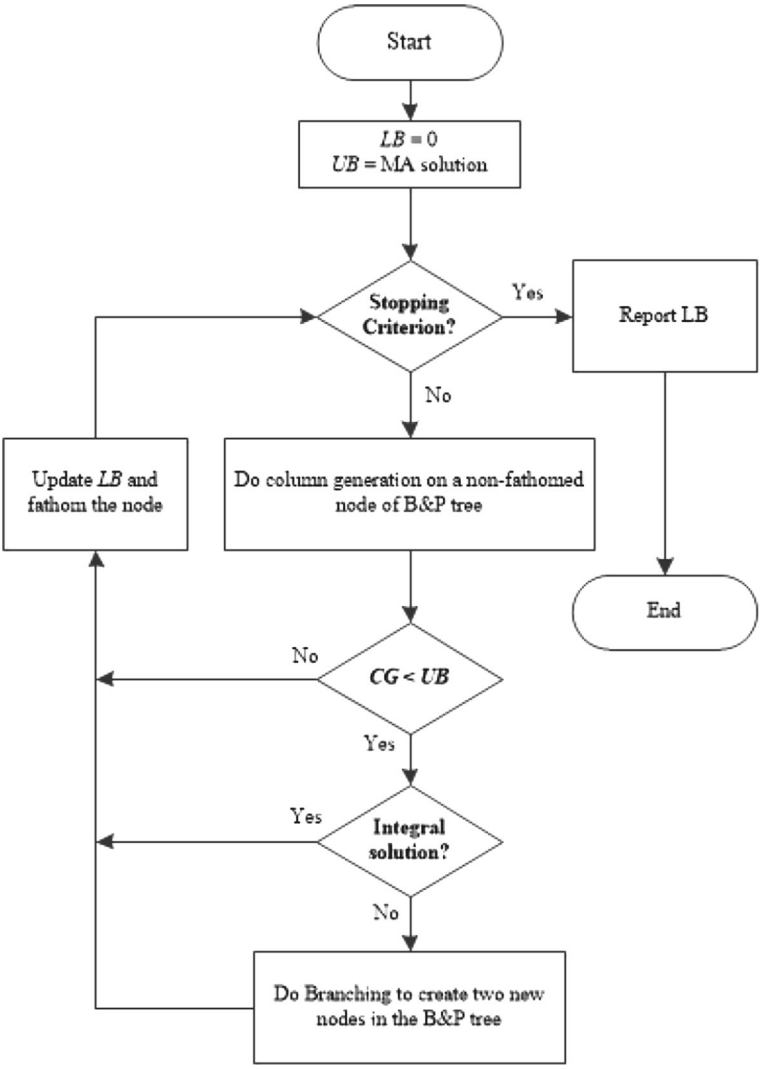

Fig. 4 The flow chart of Branch and Price algorithm

Appendix 2

Consider a $P2|s_{gh}|\sum w_j C_j$ problem with four jobs. The processing time, the weight of jobs, and the sequence-dependent setup times between jobs are provided in Tables 13 and 14.

Table 13 The processing time and the weight of jobs

Job (j)	1	2	3	4
Process time (p_j)	10	15	12	20
weight (w_j)	1	2	5	3

Table 14 The sequence-dependent setup times between jobs (s_{ij})

Job (i)	Job (j)			
	1	2	3	4
0	10	14	7	5
1	–	18	25	22
2	19	–	2	8
3	12	7	–	8
4	24	8	12	–

The B&B tree is presented in Fig. 5. The Initial upper bound is $UB = 311$. This upper bound is obtained by assigning $J_3 - J_2$ to machine 1 and $J_4 - J_1$ to machine 2.

At node 0 of B&B tree, $\mathcal{S} = \emptyset$, $\mathcal{S}' = \{1, 2, 3, 4\}$ and $r = 0$. For calculating the lower bound of this node, the process time of each job is calculated by using the following equations:

$$p'_1 = \left(\min_h \{s_{h1}\} + p_1\right) \Big/ nm = (s_{01} + p_1)/2 = (10 + 10)/2 = 10$$

$$p'_2 = \left(\min_h \{s_{h2}\} + p_2\right) \Big/ nm = (s_{32} + p_2)/2 = (7 + 15)/2 = 11$$

$$p'_3 = \left(\min_h \{s_{h3}\} + p_3\right) \Big/ nm = (s_{23} + p_3)/2 = (2 + 12)/2 = 7$$

$$p'_4 = \left(\min_h \{s_{h4}\} + p_4\right) \Big/ nm = (s_{04} + p_4)/2 = (5 + 20)/2 = 12.5$$

$$\frac{w_1}{p'_1} = \frac{1}{10}; \frac{w_2}{p'_2} = \frac{2}{11}; \frac{w_3}{p'_3} = \frac{5}{7}; \frac{w_4}{p'_4} = \frac{3}{12.5}$$

Using the WSPT rule, the sequence of unscheduled jobs would be $J_3 - J_4 - J_2 - J_1$. So the lower bound would be:

$$lb_0 = 5 \times 7 + 3 \times (7 + 12.5) + 2 \times (7 + 12.5 + 11) + 1 \times (7 + 12.5 + 11 + 10) = 195$$

At node 1 of B&B tree, $\mathcal{S} = \{1\}, \mathcal{S}' = \{2, 3, 4\}$, and $r = \min\{C_1 = s_{01} + p_1, 0\} = \min\{20, 0\} = 0$. Using the WSPT rule, the sequence of unscheduled jobs would be $J_3 - J_4 - J_2$and the lower bound would be:

$$lb_1 = \max\{lb_0, 1 \times (C_1) + 5 \times (r + 7) + 3 \times (r + 7 + 12.5) + 2 \times (r + 7 + 12.5 + 11)\} = 195$$

At node 2 of B&B tree, $\mathcal{S} = \{2\}$, $\mathcal{S}' = \{1, 3, 4\}$, and $r = \min\{C_2 = s_{02} + p_2, 0\} = \min\{29, 0\} = 0$. Using the WSPT rule the sequence of unscheduled jobs would be $J_3 - J_4 - J_1$ and the lower bound would be:

$$lb_2 = \max\{lb_0, 2 \times (C_2) + 5 \times (r + 7) + 3 \times (r + 7 + 12.5) + 1 \times (r + 7 + 12.5 + 10)\} = 195$$

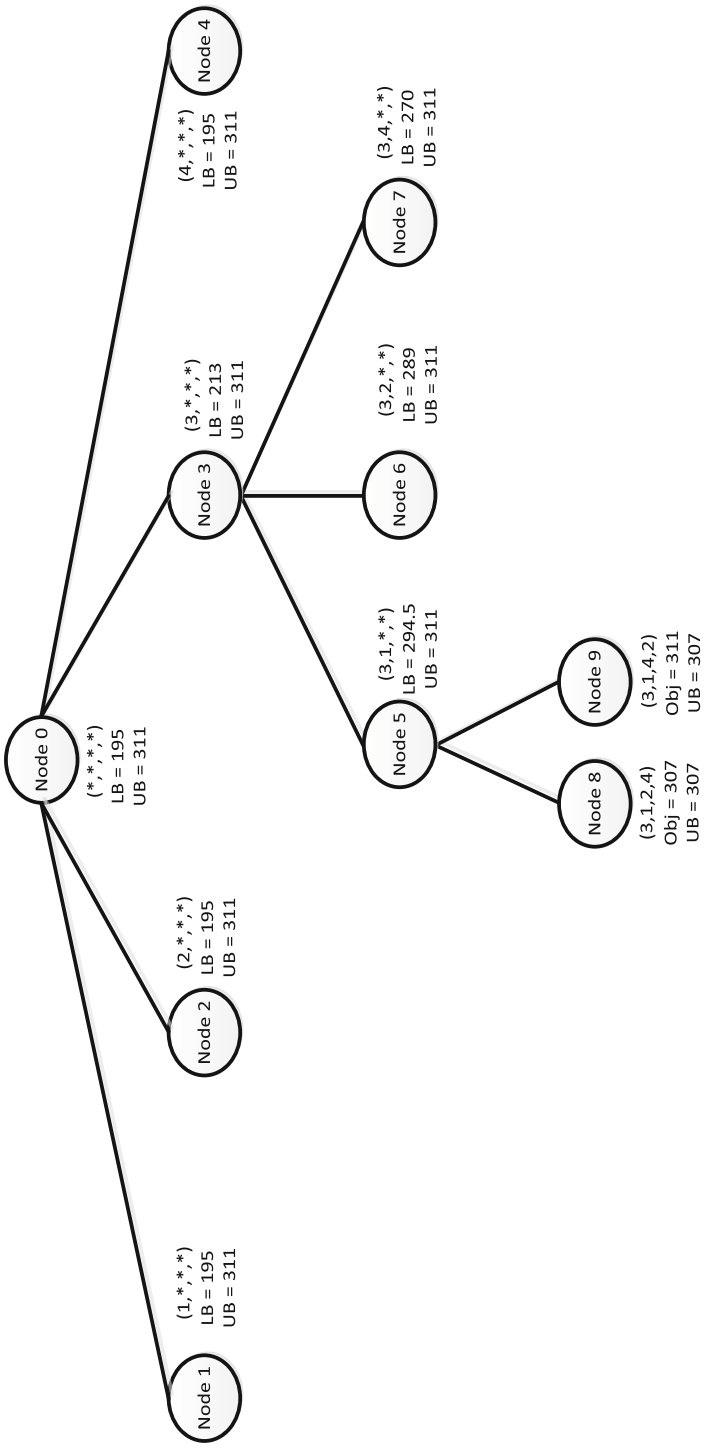

Fig. 5 The Branch-and-Bound tree

At node 3 of B&B tree, $\mathcal{S} = \{3\}$, $\mathcal{S}' = \{1, 2, 4\}$, and $r = \min\{C_3 = s_{03} + p_3, 0\} = \min\{19, 0\} = 0$. Using the WSPT rule, the sequence of unscheduled jobs would be $J_4 - J_2 - J_1$and the lower bound would be:

$$lb_3 = \max\{lb_0, 5 \times (C_3) + 3 \times (r + 12.5) + 2 \times (r + 12.5 + 11) + 1 \times (r + 12.5 + 11 + 10)\} = 213$$

At node 4 of B&B tree, $\mathcal{S} = \{4\}$, $\mathcal{S}' = \{1, 2, 3\}$, and $r = \min\{C_4 = s_{04} + p_4, 0\} = \min\{25, 0\} = 0$. Using the WSPT rule, the sequence of unscheduled jobs would be $J_3 - J_2 - J_1$and the lower bound would be:

$$lb_4 = \max\{lb_0, 3 \times (C_4) + 5 \times (r + 7) + 2 \times (r + 7 + 11) + 1 \times (r + 7 + 11 + 10)\} = 195$$

At node 5 of B&B tree, $\mathcal{S} = \{3, 1\}$, $\mathcal{S}' = \{2, 4\}$. First, J_3 is assigned to machine 1 and $C_3 = s_{03} + p_3 = 19$. Then, J_1 is assigned to machine 2 (because $s_{01} < C_3 + s_{31}$) and $C_1 = s_{01} + p_1 = 20$. So $r = \min\{C_3, C_1\} = 19$. Using the WSPT rule, the sequence of unscheduled jobs would be $J_4 - J_2$ and the lower bound would be:

$$lb_5 = \max\{lb_3, 5 \times (C_3) + 1 \times (C_1) + 3 \times (r + 12.5) + 2 \times (r + 12.5 + 11)\} = 294.5$$

At node 6 of B&B tree, $\mathcal{S} = \{3, 2\}$, $\mathcal{S}' = \{1, 4\}$. First, J_3 is assigned to machine 1 and $C_3 = s_{03} + p_3 = 19$. Then J_2 is assigned to machine 2 (because $s_{02} < C_3 + s_{32}$) and $C_2 = s_{02} + p_2 = 29$. So $r = \min\{C_3, C_2\} = 19$. Using the WSPT rule, the sequence of unscheduled jobs would be $J_4 - J_1$ and the lower bound would be:

$$lb_6 = \max\{lb_3, 5 \times (C_3) + 2 \times (C_2) + 3 \times (r + 12.5) + 1 \times (r + 12.5 + 10)\} = 289$$

At node 7 of B&B tree, $\mathcal{S} = \{3, 4\}$, $\mathcal{S}' = \{1, 2\}$. First, J_3 is assigned to machine 1 and $C_3 = s_{03} + p_3 = 19$. Then J_4 is assigned to machine 2 (because $s_{04} < C_3 + s_{34}$) and $C_4 = s_{04} + p_4 = 25$. So $r = \min\{C_3, C_4\} = 19$. Using the WSPT rule, the sequence of unscheduled jobs would be $J_2 - J_1$ and the lower bound would be:

$$lb_7 = \max\{lb_3, 5 \times (C_3) + 3 \times (C_4) + 2 \times (r + 11) + 1 \times (r + 11 + 10)\} = 270$$

At node 8 of B&B tree, $\mathcal{S} = \{3, 4, 1\}$, $\mathcal{S}' = \{2\}$. First, J_3 is assigned to machine 1 and $C_3 = s_{03} + p_3 = 19$. Then, J_4 is assigned to machine 2 (because $s_{04} < C_3 + s_{34}$) and $C_4 = s_{04} + p_4 = 25$. Then, J_1 is assigned to machine 1 (because $C_3 + s_{31} < C_4 + s_{41}$) and $C_1 = C_3 + s_{31} + p_1 = 41$. J_2 is assigned to machine 2 (because $C_4 + s_{42} < C_1 + s_{12}$) and $C_2 = C_4 + s_{42} + p_2 = 48$. This node is at level 3, so the objective function of this node can be calculated:

$$obj_8 = 5 \times 19 + 3 \times 25 + 1 \times 41 + 2 \times 48 = 307$$

So the new upper bound would be $UB = 307$. This node is fathomed.

At node 9 of B&B tree, $\mathcal{S} = \{3, 4, 2\}$, $\mathcal{S}' = \{1\}$. First, J_3 is assigned to machine 1 and $C_3 = s_{03} + p_3 = 19$. Then, J_4 is assigned to machine 2 (because $s_{04} < C_3 + s_{34}$) and $C_4 = s_{04} + p_4 = 25$. Then, J_2 is assigned to machine 1 (because $C_3 + s_{32} < C_4 + s_{42}$) and $C_2 = C_3 + s_{32} + p_2 = 41$. J_1 is assigned to machine 2 (because $C_4 + s_{41} < C_2 + s_{21}$) and $C_1 = C_4 + s_{41} + p_1 = 59$. This node is at level 3, so this node is fathomed. The objective function of this node can be calculated:

$$obj_9 = 5 \times 19 + 3 \times 25 + 2 \times 41 + 1 \times 59 = 311$$

By traversing other nodes that are not fathomed, the optimal solution can be found.

Table 15 ANOVA table for two-stages

Type 3 tests of fixed effects for two-stages

Num Effect	Den DF	DF	F Value	Pr>F
G	2	267	3.38E+06	<.0001
J	2	267	1025332	<.0001
R1	2	267	73664.2	<.0001
A	2	267	0.06	0.9431
L	1	267	16.95	<.0001
T(G*J*R1)	47	267	101549	<.0001

Table 16 ANOVA table for three-stages. Type 3 tests of fixed effects for three-stages

Num Effect	Den DF	DF	F Value	Pr>F
G	2	807	1.03E+07	<.0001
J	2	807	3016099	<.0001
R1	2	807	156115	<.0001
R2	2	807	316270	<.0001
T(G*J*R1*R2)	153	807	90127.2	<.0001
A	2	807	0.03	0.9728
L	1	807	16.41	<.0001

Table 17 ANOVA table for six-stages. Type 3 tests of fixed effects for six-stages

Num Effect	Den DF	DF	F Value	Pr>F
G	2	267	1.02E+07	<.0001
J	2	267	3119373	<.0001
R1	2	267	3952215	<.0001
A	2	267	0.06	0.9422
L	1	267	15.66	<.0001
T(G*J*R1)	47	267	416778	<.0001

Appendix 3: The ANOVA table for two, three, and six-stage problems

References

Barnhart, C., Hane, C. A., Johnson, E. L., & Sigismondi, G. (1994). A column generation and partitioning approach for multi-commodity flow problems. *Telecommunication Systems*, *3*(3), 239–258.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, *46*(3), 316–329.

Desrosiers, J., Dumas, Y., Solomon, M. M., & Soumis, F. (1995). Time constrained routing and scheduling. *Handbooks in operations research and management science*, *8*, 35–139.

França, P. M., Gupta, J. N., Mendes, A. S., Moscato, P., & Veltink, K. J. (2005). Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers & Industrial Engineering*, *48*(3), 491–506.

Gelogullari, C. A., & Logendran, R. (2010). Group-scheduling problems in electronics manufacturing. *Journal of Scheduling*, *13*(2), 177–202.

Glover, F., & Laguna, M. (1997). *Tabu search*. Norwell, MA: Kluwer Academic Publishers.

Goldberg, D. E., & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications* (pp. 154–159). Mahwah, NJ: Lawrence Erlbaum Associates, Publishers.

Hajinejad, D., Salmasi, N., & Mokhtari, R. (2011). A fast hybrid particle swarm optimization algorithm for flow shop sequence dependent group scheduling problem. *Scientia Iranica*, *18*(3), 759–764.

Hart, W. E., & Krasnogor, N. (2005). *Recent advances in memetic algorithms* (Vol. 166). Heidelberg: Springer.

Kan, A. H. G. R. (1976). *Machine scheduling problems: classification, complexity and computations*. Birmingham: Stenfert Kroese.

Keshavarz, T., & Salmasi, N. (2013). Makespan minimisation in flexible flowshop sequence-dependent group scheduling problem. *International Journal of Production Research*, *51*(20), 6182–6193.

Logendran, R., Carson, S., & Hanson, E. (2005). Group scheduling in flexible flow shops. *International Journal of Production Economics*, *96*(2), 143–155.

Logendran, R., Deszoeke, P., & Barnard, F. (2006a). Sequence-dependent group scheduling problems in flexible flow shops. *International Journal of Production Economics*, *102*(1), 66–86.

Logendran, R., Salmasi, N., & Sriskandarajah, C. (2006b). Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups. *Computers & Operations Research*, *33*(1), 158–180.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P. *Report*, *826*, 1989.

Naderi, B., & Salmasi, N. (2012). Permutation flowshops in group scheduling with sequence-dependent setup times. *European Journal of Industrial Engineering*, *6*(2), 177–198.

Paternina-Arboleda, C., Montoya-Torres, J., Acero-Dominguez, M., & Herrera-Hernandez, M. (2008). Scheduling jobs on a k-stage flexible flow-shop. *Annals of Operations Research*, *164*(1), 29–40.

Pinedo, M. (2012). *Scheduling: Theory, algorithms, and systems*. Berlin: Springer.

Ravetti, M., Riveros, C., Mendes, A., Resende, M. C., & Pardalos, P. (2012). Parallel hybrid heuristics for the permutation flow shop problem. *Annals of Operations Research*, *199*(1), 269–284.

Ruiz, R., Maroto, C., & Alcaraz, J. (2003). New genetic algorithms for the permutation flowshop scheduling problems. In *Proceedings of the fifth metaheuristic international conference, Kyoto* (p. 63-1).

Salmasi, N., & Logendran, R. (2008). A heuristic approach for multi-stage sequence-dependent group scheduling problems. *Journal of Industrial Engineering International*, *4*(7), 48–58.

Salmasi, N., Logendran, R., & Skandari, M. R. (2010). Total flow time minimization in a flowshop sequence-dependent group scheduling problem. *Computers & Operations Research*, *37*(1), 199–212.

Salmasi, N., Logendran, R., & Skandari, M. R. (2011). Makespan minimization of a flowshop sequence-dependent group scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *56*(5–8), 699–710.

SAS Release, 9.1, (2002–2003), SAS Institute Inc., Cary, North Carolina, USA.

Schaller, J. E., Gupta, J. N., & Vakharia, A. J. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*, *125*(2), 324–339.

Shahvari, O., Salmasi, N., Logendran, R., & Abbasi, B. (2012). An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. *International Journal of Production Research*, *50*(15), 4237–4254.

Wilhelm, W. E. (2001). A technical review of column generation in integer programming. *Optimization and Engineering*, *2*(2), 159–200.

Wilhelm, W. E., Damodaran, P., & Li, J. (2003). Prescribing the content and timing of product upgrades. *IIE Transactions*, *35*(7), 647–663.

Zandieh, M., Dorri, B., & Khamseh, A. R. (2009). Robust metaheuristics for group scheduling with sequence-dependent setup times in hybrid flexible flow shops. *The International Journal of Advanced Manufacturing Technology*, *43*(7–8), 767–778.