Int J Adv Manuf Technol (2009) 43:767–778
DOI 10.1007/s00170-008-1740-x

ORIGINAL ARTICLE

Robust metaheuristics for group scheduling with sequence-dependent setup times in hybrid flexible flow shops

M. Zandieh · Behrouz Dorri · A. R. Khamseh

Received: 11 February 2008 / Accepted: 1 September 2008 / Published online: 23 September 2008

Abstract This paper considers group scheduling problem in hybrid flexible flow shop with sequence-dependent setup times to minimize makespan. Group scheduling problem consists of two levels, namely scheduling of groups and jobs within each group. In order to solve problems with this context, two new metaheuristics based on simulated annealing (SA) and genetic algorithm (GA) are developed. A design procedure is developed to specify and adjust significant parameters for SA- and GA-based metaheuristics. The proposed procedure is based on the response surface methodology and two types of objective function are considered to develop multiple-objective decision making model. For comparing metaheuristics, makespan and elapsed time to obtain it are considered as two response variables representing effectiveness and efficiency of algorithms. Based on obtained results in the aspect of makespan, GA-based metaheuristic is recommended for solving group scheduling problems in hybrid flexible flow shop in all sizes and for elapsed time SA-based metaheuristic has better results.

Keywords Group scheduling · Hybrid flexible flow shop · Sequence-dependent setup times · Metaheuristics · Simulated annealing · Genetic algorithm · Response surface methodology · Multiple-objective decision making

M. Zandieh (✉) · B. Dorri
Department of Industrial Management,
Management and Accounting Faculty,
Shahid Beheshti University,
Tehran, Iran
e-mail: m_zandieh@sbu.ac.ir

A. R. Khamseh
Department of Industrial and Mechanical Engineering,
Azad University, Qazvin, Iran

1 Introduction

Manufacturing systems have a variety of forms. This depends on jobs that would be processed on system. One of these manufacturing systems which are spreading increasingly in industries is hybrid flexible flow shop. In fact, hybrid flexible flow shops are the extension of flow shops. In a flow shop, there is just one machine at each stage and jobs are processed in a linear fashion from the first stage to the last stage. There may exist multiple machines at each stage to extend the capacity. This results in an extension of flow shop called hybrid flow shop. These parallel machines could be identical, uniform, or unrelated. In a hybrid flow shop, jobs pass all stages and are processed at each stage. If all jobs are not supposed to go through all stages and permitted to skip certain stages, we end up with a new manufacturing system known as hybrid flexible flow shop [30].

A noticeable subject encountered in industries is setup. In many real-world applications, setup times are required when a machine switches from one work order to another. This is very usual in automobile manufacture, printed circuit board autoinsertion lines, or tile industries. Setup times could be sequence independent or sequence dependent. Considering setup times results in higher complexity of scheduling problems and the most difficult case is when the setup times are sequence dependent. In a sequence-dependent setup, the estimated setup time depends on the current job as well as the next job [6, 21, 31].

Another subject applied in some industries in order to improve productivity is cellular manufacturing philosophy. Cellular manufacturing can be defined as an application of group technology that involves grouping machines according to parts that should be processed using them. Group technology is a management philosophy trying to put the products with similar design or manufacturing character-

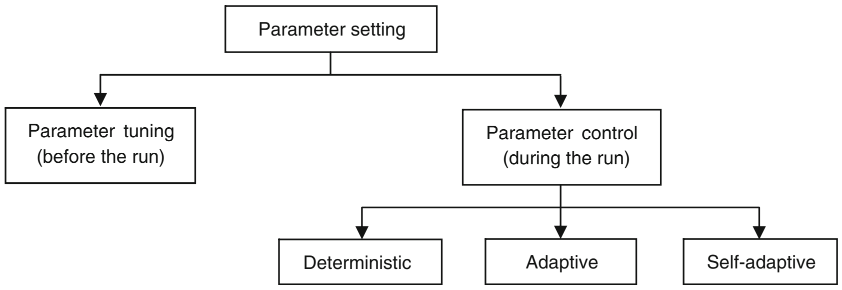
Parameter setting
Parameter tuning (before the run)
Parameter control (during the run)
Deterministic
Adaptive
Self-adaptive

Fig. 1 Global taxonomy of parameter setting in metaheuristics

istics or both in one group. In cellular manufacturing, the main objective is simultaneously identifying machine cells and part families and allocating part families to machine cells. Cellular manufacturing is relatively a recent concept that has been applied successfully in many manufacturing environments and can achieve significant benefits such as setup time reduction, work-in-process inventory reduction, material-handling cost reduction, improvement in machine utilization, improvement in employee moral, and so on [9].

After allocating groups of jobs to cells, groups and jobs within each group must be scheduled. Scheduling of groups in each cell results in a problem called group scheduling and the objective is to identify the sequence of parts belonging to each group as well as the sequence of groups themselves to optimize some measure of performance. Thus, group scheduling problem is comprised of two levels. At level 1, the sequence of parts that belong to each group is determined and, at level 2, the sequence of groups themselves is determined [17].

There are many different objectives that can be considered for a scheduling problem. One that is used in many researches is the completion time of the scheduling, namely, makespan that is also the objective in this research. Thus, group scheduling problem with sequence-dependent group setup times in hybrid flexible flow shop in order to minimize makespan is examined in this research.

Because different constraints and assumptions can result in different scheduling problems in hybrid flexible flow shops, for better definition, we introduce the following assumptions:

- All data are known deterministically.
- Machines are available at all times and there is no breakdown.
- No preemption is allowed.
- There are infinite buffers between stages.
- There is no travel time between stages.
- All jobs are available at the beginning of scheduling.
- Machines are idle during setup.
- Parallel machines are identical.

The remainder of the paper is organized as follows. “Section 2” contains a literature review. In “Section 3,” the proposed metaheuristics are described. The way of generating test problems is described in “Section 4.” For adjusting parameters, we provide a new approach in “Section 5.” In “Section 6,” the metaheuristics are compared. “Section 7” concludes the paper with a look at future works.

Current solution: G_1 (J_{11}, J_{12}, J_{13}) – G_2 (J_{21}, J_{22}, J_{23}) – G_3 (J_{31}, J_{32}, J_{33})
New neighborhood solution: G_1 (J_{11}, J_{12}, J_{13}) – G_3 (J_{33}, J_{32}, J_{31}) – G_2 (J_{22}, J_{21}, J_{23})

Fig. 2 Neighborhood-solution-generation mechanism

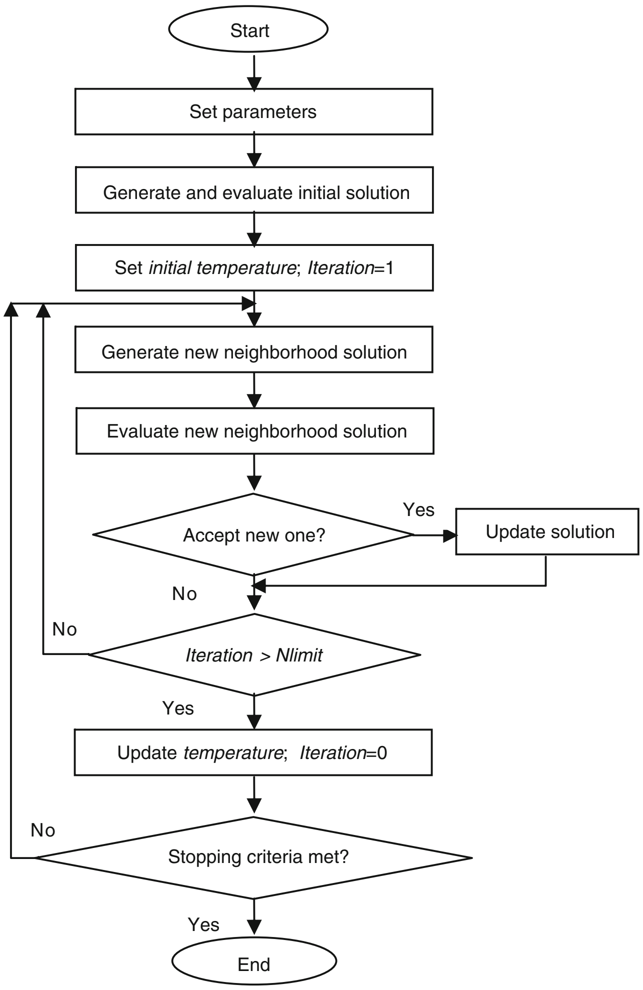
Start
Set parameters
Generate and evaluate initial solution
Set initial temperature; Iteration=1
Generate new neighborhood solution
Evaluate new neighborhood solution
Accept new one?
Yes
Update solution
No
No
Iteration > Nlimit
Yes
Update temperature; Iteration=0
No
Stopping criteria met?
Yes
End

Fig. 3 Flow chart of the SA-based algorithm

Fig. 4 Representation of solutions in GA

	M_1	J_1	J_2	J_3
G_1	1.22	0.34	0.92	-
G_2	2.31	0.45	0.21	0.72
G_3	1.89	0.88	0.51	0.12
G_4	2.65	0.67	0.25	-

M_{11}: G_1 (J_{11}- J_{12}) – G_3 (J_{33}- J_{32}-J_{31})
M_{12}: G_2 (J_{22}- J_{21}- J_{23}) – G_4 (J_{42}-J_{41})

2 Literature review

This literature review will have one component regarding group scheduling and a second regarding parameter setting in metaheuristic algorithms.

Liaee and Emmons [13] reviewed the literature on scheduling groups of jobs on single or parallel machines with the presence of setup times. They considered makespan and number of tardy job objectives for the scheduling with and without the group technology assumption. Regular flow shop scheduling problem was considered by some authors and they proposed heuristic methods for minimizing the makespan in level-1 problem [2, 14, 26]. Nakamura et al. [22] and Ozden et al. [24] examined the single-machine scheduling problem with the objective of minimizing the total tardiness. For minimizing the mean completion time, a dynamic programming in order to identify an optimal sequence of parts was proposed by Baker [3]. The group scheduling problem to minimize maximum lateness and the total weighted flow time was reviewed by Webster and Baker [29]. They considered item availability, batch availability, and batch processing and proved some properties to avoid searching the whole feasible schedules.

To minimize the makespan in a two-machine group scheduling problem with sequence-independent setups, a polynomial-time algorithm is proposed by Ham et al. [8].

Allison [1] combined a single-pass heuristic by Petrov (PT) [25] and a multiple-pass heuristic by Campbell et al. (CDS) [4]. He examined the performance of it to minimize the makespan in group scheduling problem.

Longendran and Nudtasomboon [14] proposed a new algorithm (LN) for solving the level-1 problem. Based on this new algorithm, the *m* machine group scheduling problem is considered by Logendran et al. [15]. They compared the performance of LN with CDS when each is combined with PT to minimize the makespan of the problem.

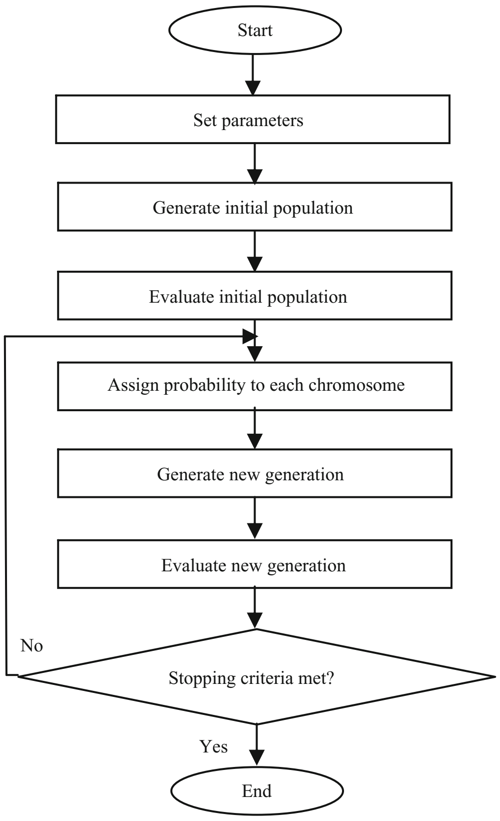

Fig. 6 Flow chart of the GA-based algorithm

Schaller et al. [28] developed some heuristics and a branch and bound approach to solve the sequence-dependent flow shop group scheduling. The sequence-dependent group scheduling problem is examined by Reddy and Narendran [27]. They used simulation methods for solving problem under some dynamic conditions like nonavailability of all jobs at the beginning of the planning horizon.

Kurz and Askin [11, 12] compared scheduling rules for flexible flow lines to minimize makespan with sequence-

	M_1	J_1	J_2	J_3
G_1	1.35	0.68	0.59	0.33
G_2	2.53	0.15	0.42	0.88
G_3	1.86	0.94	0.32	0.57

Parent 1

	M_1	J_1	J_2	J_3
G_1	2.64	0.12	0.68	0.94
G_2	2.39	0.38	0.41	0.19
G_3	1.25	0.29	0.85	0.71

Parent 2

	M_1	J_1	J_2	J_3
G_1	0.26	0.85	0.92	0.68
G_2	0.18	0.52	0.91	0.76
G_3	0.83	0.29	0.67	0.46

Mask

	M_1	J_1	J_2	J_3
G_1	1.35	0.12	0.68	0.33
G_2	2.53	0.15	0.41	0.19
G_3	1.25	0.94	0.32	0.57

Offspring

Fig. 5 Uniform crossover example

Table 1 Factor levels

Factor	Levels
Number of stages	2–3 (s), 5–6 (m), 8–9 (l)
Number of groups	4–5 (s), 7–9 (m), 11–12 (l)
Number of jobs	3–5 (s), 7–9 (m), 10–12 (l)
Processing times	Uniform (5–75)
Setup times	Uniform (5–25)
Skipping probability	0.2
Flexibility	1/3, 2/3, 3/3

s Small problem, *m* medium problem, *l* large problem

Table 2 SA-based metaheuristic factor levels

Problem size	Factor			
	N		Nlimit	
	Lower limit	Upper limit	Lower limit	Upper limit
Small	20	50	20	50
Medium	30	60	40	90
Large	40	100	90	140

dependent setup times. Zandieh et al. [31] considered hybrid flow shop and described an immune algorithm approach to the scheduling of a sequence-dependent setup time hybrid flow shop. Group scheduling problem in hybrid flexible flow shops with sequence-independent setups to minimize makespan was investigated by Logendran et al. [16]. To the best of the authors' knowledge, the only research that is available in hybrid flexible flow shop group scheduling problem with sequence-dependent setup times has been done by Logendran et al. [17]. They developed three tabu search-based algorithms to minimize the makespan required to process jobs in all groups released on the shop floor.

The issue of setting the values of various parameters of metaheuristic algorithm is crucial for good performance. Finding the appropriate setup for metaheuristic algorithm is a long-standing grand challenge of the field. The values of these parameters greatly determine whether the algorithm will find an optimal or near-optimal solution and whether it will find such a solution efficiently. Generally, there are two major forms of setting parameter values: *parameter tuning* and *parameter control*. By parameter tuning, we mean the commonly practiced approach that amounts to finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. Parameter control forms an alternative, as it amounts to starting a run with initial parameter values that are changed during the run [5, 7, 19, 20].

Table 3 GA-based metaheuristic factor levels

Problem size	Factor					
	Gen size		Pop size		Pc	
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit (%)	Upper limit (%)
Small	10	50	25	50	75	80
Medium	25	75	50	100	75	80
Large	30	100	75	100	75	80

Table 4 Significant factors for makespan in SA

Problem size	Factor		
	N	Nlimit	N×Nlimit
Small	●	●	
Medium	●	●	
Large	●		●

Methods for changing the value of a parameter can be classified into one of three categories. *Deterministic parameter control* takes place when the value of a strategy parameter is altered by some deterministic rule. This rule modifies the strategy parameter in a fixed, predetermined way without using any feedback from the search. *Adaptive parameter control* takes place when there is some form of feedback from the search that serves as inputs to a mechanism used to determine the direction or magnitude of the change to the strategy parameter. And finally, the idea of the evolution can be used to implement the *self-adaptation of parameters*. The forms of setting parameters are provided in Fig 1.

3 Proposed metaheuristics

3.1 Simulated annealing algorithm

Simulated annealing (SA) is motivated by an analogy to annealing in solids. The idea of SA comes from a paper published by Metropolis et al. in 1953 [18]. Metropolis algorithm simulated the material as a system of particles. The algorithm simulates the cooling process by gradually lowering the temperature of the system until it converges to a steady, frozen state. Kirkpatrick et al. [10] took the idea of the Metropolis algorithm and applied it to combinatorial and other optimization problems.

The major advantage of SA over other methods is an ability to avoid becoming trapped at local minimum. The algorithm employs a random search, which not only accepts changes that improve objective function but also some changes that do not improve it.

SA is a variation of hill climbing in which, during search process, some nonimproving moves may be made. SA first

Table 5 Significant factors for elapsed time in SA

Problem size	Factor		
	N	Nlimit	N×Nlimit
Small	●	●	
Medium	●	●	●
Large	●	●	

Table 6 Significant factors for makespan in GA

Problem size	Factor						
	Gen size (a)	Pop size (b)	Pc (c)	(a)×(b)	(a)×(c)	(b)×(c)	(a)×(b)×(c)
Small	●	●					
Medium	●	●			●		
Large	●	●		●	●		

generates neighborhood solutions from a current solution and move to one of them until stopping criteria is met.

The performance of SA is influenced by some factors such as initial temperature (T_0), final temperature (T_f), number of temperature decline (N), and number of iterations at each temperature (*Nlimit*).

SA begins with an initial solution (IS) and, after evaluating it by means of a cost function, a new neighborhood is generated. This new neighborhood solution is accepted when it improves the value of cost function. In a condition where the value of cost function of new neighborhood solution is worse, it also can be accepted if it satisfies the following criteria:

$$r > \exp\left(-\frac{\Delta}{T}\right) \quad (1)$$

where r is generated using uniform distribution between 0 and 1; Δ is the difference between the value of the cost function of the current solution and the new neighborhood solution, and T is the current temperature.

3.1.1 Initial solution

Initial solution is generated according to the last initial solution-finding mechanism (IS3) in Logendran et al. [17]. To do this, a key machine (stage) is identified using the longest cumulative processing time (LCPT) for all groups. Because setup times are sequence dependent, the minimum setup time is used to calculate LCPT. Cumulative processing time for a machine is evaluated as the sum of the minimum setup time for each group on that machine and the run times for jobs in all groups. The group sequence is identified based upon the longest cumulative run time on the key machine for each group and the job sequence within each group is identified based upon the longest run time. Should there be a tie in the determination of LCPT, it is broken in favor of the smallest machine number.

3.1.2 Neighborhood-solution-generation mechanism

In order to start the optimization process in SA, we need a neighborhood solution-generation mechanism to generate neighborhoods from current solution. To do this, we consider the current sequence of groups and jobs within each group (current solution) and we select two groups and two jobs within each group randomly and swap them. That is, we change the positions of two groups and then two jobs within each group swapped. For example, consider a current solution where there are three groups and three jobs within each group. Suppose the selected groups are G_2 and G_3 and the jobs that must be swapped are J_{21}, J_{22} in G_2, and J_{31}, J_{33} in G_3. Figure 2 shows the current solution and the new neighborhood solution.

The steps of SA-based metaheuristic are provided in Fig. 3. Further explanations about algorithm parameters and their values are discussed in “Section 5”.

3.2 Genetic algorithm

Genetic algorithm (GA) was introduced by Holland, DeJong, and Goldberg in 1970. A genetic algorithm is a directed random search technique that evolves an initial population of solutions using genetic operators to create generations that are successively better with respect to the objective of interest and is especially suitable when the search space is unknown. In GA, every individual or chromosome is encoded into a structure that represents its properties. The set of chromosomes forms the population. The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated using some measures of fitness. A new

Table 7 Significant factors for elapsed time in GA

Problem size	Factor						
	Gen size (a)	Pop size (b)	Pc (c)	(a)×(b)	(a)×(c)	(b)×(c)	(a)×(b)×(c)
Small	●	●	●				
Medium	●	●	●	●	●		
Large	●	●	●	●	●		

$$Max \quad \sum w_i \alpha_i$$

$$s.t:$$

$$Z_i \le U_i - \alpha_i (U_i - L_i)$$
$$-1 \le x_j \le 1$$
$$0 \le \alpha_i \le 1$$

Z_i : i th objective function (i th regression model)

α_i : satisfactory level of Z_i

w_i : weight of Z_i

U_i : maximum value in ith column in payoff table

L_i : minimum value in ith column in payoff table

x_j : j th coded variable (based on identified significant factors)

i=1, ..., 8 (for small and large problems)
i=1, ..., 12 (for medium problems)
j=1, 2 (for SA)
j=1, 2, 3 (for GA)

Fig. 7 General MODM model for solving problem

generation is obtained as follows: A selection mechanism picks chromosomes and the selected chromosomes mate and generate new chromosomes, called the offsprings. This is done by crossover operator and sometimes another operator called mutation is used. So a new generation is formed according to the fitness values of chromosomes by genetic operators. Then, a new generation is evaluated and this process is repeated until stopping criteria is met. The effectiveness of GA greatly depends on the correct choice of solution representation, selection mechanism, stopping criteria, genetic operators, mechanism used to create initial population, and parameter values.

In this paper, in order to develop a metaheuristic based on genetic algorithm for solving the problem, we use the idea of a random key genetic algorithm (RKGA). RKGA differs from traditional genetic algorithms most notably in solution representation. In RKGA, random numbers are used in coding the solutions. For example, Norman and Bean [23] suggested the following solution representation for an identical multiple-machine problem. Each job is assigned a real number whose integer part is the machine number to which the job is assigned and whose fractional part is used to sort the jobs assigned to each machine. In this research, this idea is used to assign and sort the groups and jobs within each group at the first stage. Because we assume that only one setup is performed for each group at each stage, group assignment determines job assignment. To assign the groups to the first stage, machines' number of groups (*ng*) random real numbers in the interval of one and number of machines in the first stage plus one are generated. The integer part is the machine number to which the group is assigned and the fractional part is used to sort the groups assigned to each machine. In order to determine the sequence of jobs within each group for each job, a random number using uniform distribution between 0 and 1 is generated and the job with the smallest number is processed first. In subsequent stages, the group and job assignment is done according to completion time in the previous stages. An example for representation and its related schedule is shown in Fig. 4. In this example, there are four groups and number of jobs within each group is 2, 3, 3, and 2, respectively. We assume that the number of identical parallel machines at the first stage is two (M_{11}, M_{12}).

The initial population is generated randomly. To evaluate each solution, we use makespan as fitness function. Triple genetic operators (reproduction, crossover, and mutation) are applied to form a new generation. An elitist strategy is used for reproduction. Each chromosome from the current generation is decoded and its makespan is evaluated. Twenty percent of chromosomes with the best value of makespan are copied to the next generation. Uniform crossover is used to generate the most proportion of the next generation. This percentage varies for different problem sizes. Two chromosomes from the current generation are selected according to the roulette wheel selection strategy and the better chromosome is known as parent 1 and the other is called parent 2. Then, for each gen, a random number between 0 and 1 is generated. If this value is less than 0.7, the value from the parent 1 is copied to the new chromosome; otherwise, the value from parent 2 is selected. Mutation operator forms the remaining of the new generation. To do this, related values of the two groups in a randomly selected chromosome based on roulette wheel are swapped and for each group the values of two jobs are swapped randomly. The above procedures are repeated until stopping

Table 8 SA-based metaheuristic parameter values

Problem size	Factor	
	N	Nlimit
Small	50	46
Medium	60	74
Large	100	94

Table 9 GA-based metaheuristic parameter values

Problem size	Factor		
	Gen size	Pop size	Pc (%)
Small	50	50	80
Medium	75	100	75
Large	100	100	75

Table 10 Average RPD for small problems

Algorithm	TS	SA	GA
Average RPD	2.5086	0.1666	0.3833

Table 11 ANOVA results for small problems

Source	*df*	SS	MS	*F*	*P* value
Algorithm	2	0.0171	0.0085	52.6	0.000
Error	465	0.0755	0.0002		
Total	467	0.0926			

criteria is met. Here, a number of generations is used as stopping criteria, which is different for each problem size.

An example of the uniform crossover is provided in Fig. 5. In this example, there are three groups and three jobs within each group. The number of identical parallel machines at the first stage is two.

Complementary explanations about generation size, population size, and percentage of crossover operator are presented in "Section 5." For further explanation, the flow chart of the algorithm is presented in Fig. 6.

4 Data generation

In order to evaluate and compare the performance of the existing algorithm based on tabu search and the two new metaheuristics developed in this research for solving the group scheduling problem, a plan to generate test data is considered. Required data to define a problem are problem size (*ns*, *ng*, *nj*), jobs' processing time, setup times, number of machines at each stage, and skipping probability. In a flow shop group scheduling, problem size is defined by three parameters: number of machines, number of groups (*ng*), and number of jobs (*nj*) [15]. In hybrid flexible flow shop, the representative parameter for number of machines is number of stages (*ns*), so some stages can have parallel machines [15]. Based on Logendran et al. [15], three different problem sizes, namely small, medium, and large, are considered. The ranges of each parameter to determine the problem size are as follows. Number of stages varies from 2 to 3, 5 to 6, and 8 to 9 for small, medium, and large problems, respectively. These values vary from 4 to 5, 7 to 9, and 11 to 12 for number of groups, while the number of jobs within each group varies from 3 to 5, 7 to 9, and 10 to 12 for three instances, respectively. In order to determine the number of machines at each stage, another parameter called flexibility is considered. The flexibility in a hybrid flexible flow shop is introduced by the number of stages having identical parallel machines. According to Logendran et al. [17], three levels of flexibility is considered: low, medium, and high. In low flexibility, one third of stages have parallel machines; for medium and high cases, two thirds and three thirds of stages have parallel machines. To determine which stages have parallel machines, we generate a random permutation of the value number of stages (*ns*) and then multiply flexibility by the number of stages ($d = flex \times ns$). If the answer is not an integer, we round it to the next integer number. Then, the first *d* numbers are assumed to have parallel machines. For parallel stages, number of machines is 2 or 3 with equal probability of 0.5. Other data are generated exactly in the same way as in Logendran et al. [17]. Factors and their levels are show in Table 1. Thus, there are 36, 54, and 36 test problems as small, medium, and large, respectively.

Table 12 Fisher's least-significant difference method results for small problems

Algorithms	Difference of means	LSD	Significant difference at 95% level
TS and SA	0.0234	0.0045	Yes
TS and GA	0.0213	0.0045	Yes
SA and GA	0.0022	0.0024	No

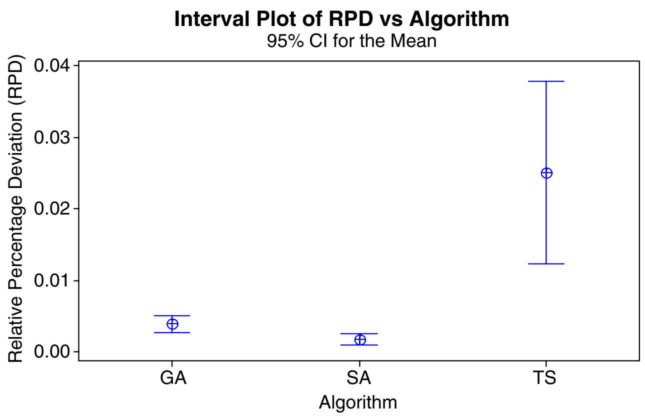

Fig. 8 Means and interval plot for small problems

5 Parameter setting

The performance of an algorithm is affected by some various factors such as the assigned values to parameters. If these values do not select correctly, we do not obtain good solutions. To select the values that result in solutions with high quality, we consider problems in three different sizes that were described before and try to set parameters for each size separately. Some problems are selected as a sample at each size. Sample sizes are 4, 6, and 4 for small, medium, and large problems, respectively.

To determine the values of parameters, we use response surface methodology (RSM). We try to set parameters by employing RSM because compared with factorial design it

Table 13 Average RPD for medium problems

Algorithm	TS	SA	GA
Average RPD	4.7613	1.7637	0.8984

Table 14 ANOVA results for medium problems

Source	*df*	SS	MS	*F*	*P* value
Algorithm	2	0.0708	0.0354	183.06	0.000
Error	699	0.1351	0.0002		
Total	701	0.2059			

can result in continuous parameter setting. First, we try to recognize factors that are statistically significant for each algorithm in the aspects of makespan and elapsed time. To identify significant factors, a two-level factorial design for each algorithm is considered. Thus, for SA-based metaheuristic, two factors and for GA-based metaheuristic three factors are considered. In the case of SA, independent variables are number of temperature decline (*N*) and number of iterations at each temperature (Nlimit). In GA, they are generation size (gen size), population size (pop size), and percentage of crossover (pc). Each factor is measured at two levels, which can be coded as value −1 when the factor is at its low level and +1 when the factor is at its high level. Coded variable can be defined as follows:

$$x_i = \frac{r_i - \left(\frac{h+l}{2}\right)}{\left(\frac{h-l}{2}\right)} \qquad (2)$$

where x_i and r_i are coded variable and natural variable, respectively. *h* and *l* represent high level and low level of factor. Factors and their levels are shown in Tables 2 and 3.

Each algorithm is run using different combinations of factors in Tables 2 and 3 and there are three replicates for each combination (R=3). For each run, the best makespan and the related elapsed time are recorded. The response variables of the experiment are then calculated with the following expression:

$$\text{Relative percentage deviation (RPD)} = \frac{Alg_{\text{sol}} - \text{Min}_{\text{sol}}}{\text{Min}_{\text{sol}}} \qquad (3)$$

where Alg_{sol} is the solution obtained by a given algorithm alternative on a given instance and Min_{sol} is the minimum solution for given instances. Significant factors and interactions are shown in Tables 4, 5, 6, and 7.

After identifying significant factors and interactions, we develop regression models for each problem size separately, with respect to significant factors that were identified before. For example, for small problems, because we have the sample size of four and on the other hand two objective functions (i.e., makespan and elapsed time), we develop eight regression models (i.e., four models related to make-span and four models for elapsed time). In these models, two mentioned objective functions are response variables and independent variables are those given in Tables 4, 5, 6, and 7. These regression models are developed for each response variable and for each sample. Consequently, there are eight, 12, and eight regression models for small, medium, and large problems, respectively.

General models for SA and GA can be represented in Eqs. 4 and 5:

$$Y = b_0 + \sum_{i=1}^{2} b_i x_i + b_{ij} x_i x_j \quad i > j \qquad (4)$$

$$Y = b_0 + \sum_{i=1}^{3} b_i x_i + \sum_{i=1}^{3} \sum_{j=1}^{3} b_{ij} x_i x_j + b_{ijk} x_i x_j x_k \quad i > j > k \qquad (5)$$

Independent variables are represented as x_i. Then, using multiple-objective decision making (MODM), the model is solved and the values of factors for each metaheuristic and each problem size is determined. For solving MODM problem, we use fuzzy logic. General model is provided in Fig. 7. We assume that the weight of makespan objective functions is seven times more than the weight of elapsed time objective functions. The obtained values for each factor are provided in Tables 8 and 9.

For SA-based metaheuristic, we consider 20 and 0.0001 for initial and final temperature, respectively, and assume that temperature is decreased in a linear fashion.

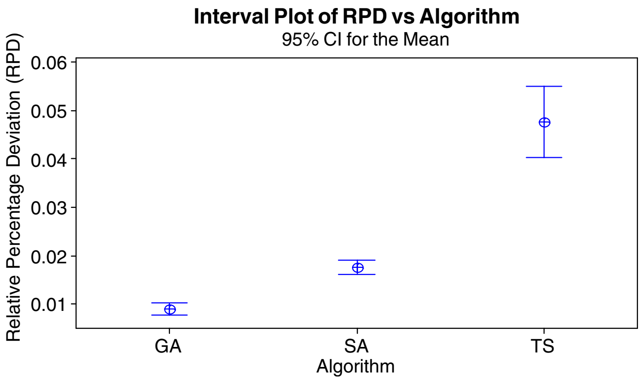

Fig. 9 Means and interval plot for medium problems

Table 15 Fisher's least-significant difference method results for medium problems

Algorithms	Difference of means	LSD	Significant difference at 95% level
TS and SA	0.03	0.004	Yes
TS and GA	0.039	0.004	Yes
SA and GA	0.0087	0.0021	Yes

Table 16 Average RPD for large problems

Algorithm	TS	SA	GA
Average RPD	4.9145	1.6657	0.8899

Table 17 ANOVA results for large problems

Source	*df*	SS	MS	*F*	*P* value
Algorithm	2	0.0505	0.0252	285.91	0.000
Error	465	0.0410	0.0001		
Total	467	0.0915			

6 Computational results

In this section, we are going to compare two developed metaheuristics based on GA and SA and existing meta-heuristic based on TS. In order to compare the algorithms, we consider makespan and elapsed time as measures of effectiveness and efficiency and examine algorithms on small, medium, and large problems separately.

Algorithms are coded in MATLAB 7.1 and we carry out six independent runs for each algorithm (with the exception of TS) and at each run the best makespan and related elapsed CPU time are recorded. Algorithms run on a PC with a Pentium 4 3-GHz processor with 512 MB of RAM and Windows XP professional operating system. For evaluating the different methods, we use the same performance measure given in Eq. 3. This means each test problem is run six times and the minimum value is considered as $\mathrm{Min}_{\mathrm{sol}}$ (this is done for both objective functions separately) and then related percentage deviation (RPD) for that test problem is calculated using this minimum value. Average related percentage deviation is computed using average of all RPDs.

6.1 Comparing effectiveness of algorithms

We use makespan as a measure that shows the effectiveness of algorithms and at each run the related makespan for each algorithm is recorded.

6.1.1 Small problems

Average related percentage deviation of algorithms on small problems is shown on Table 10. In order to determine if there is a significant difference among the performance of algorithms, a single-factor analysis of variance (ANOVA) was performed and results are shown in Table 11. These results indicate that there is at least one algorithm that is different in mean response. This motivated the use of Fisher's least-significant difference (LSD) method, with the results summarized in Table 12. These values indicate that SA- and GA-based algorithms are preferred with 95% confidence and there is no significant difference between them, although the former algorithm has smaller mean response. The means and interval plot for small problems can be observed in Fig. 8.

6.1.2 Medium problems

Average RPD of algorithms on medium problems is shown on Table 13 and the results of single-factor ANOVA are provided in Table 14. These results indicate that at least one algorithm is different in mean response. To further analysis, we performed LSD method and results are shown in Table 15. These results demonstrate that GA-based algorithm is preferred with 95% confidence. Figure 9 shows the corresponding interval plot.

6.1.3 Large problems

For large problems, obtained results for average RPD, ANOVA, and LSD methods are shown in Table 16, 17, and 18, respectively. Based on these results, it is clear that GA-based algorithm outperforms SA and TS and SA is preferred to TS with 95% confidence. Figure 10 shows interval plot for large problems.

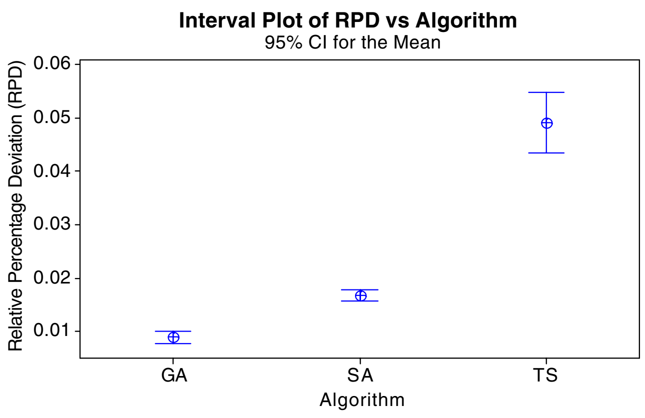

Fig. 10 Means and interval plot for large problems

Table 18 Fisher's least-significant difference method results for large problems

Algorithms	Difference of means	LSD	Significant difference at 95% level
TS and SA	0.0325	0.0033	Yes
TS and GA	0.0402	0.0033	Yes
SA and GA	0.0078	0.001	Yes

Table 19 Average RPD for small problems

Algorithm	TS	SA	GA
Average RPD	1,102.969	486.811	630.318

Table 20 ANOVA results for small problems

Source	*df*	SS	MS	*F*	*P* value
Algorithm	2	1,207.4	603.7	9.48	0.000
Error	465	29,615.5	63.7		
Total	467	30,822.9			

Table 21 Fisher's least-significant difference method results for small problems

Algorithms	Difference of means	LSD	Significant difference at 95% level
TS and SA	6.162	2.82	Yes
TS and GA	4.727	2.82	Yes
SA and GA	1.435	1.51	No

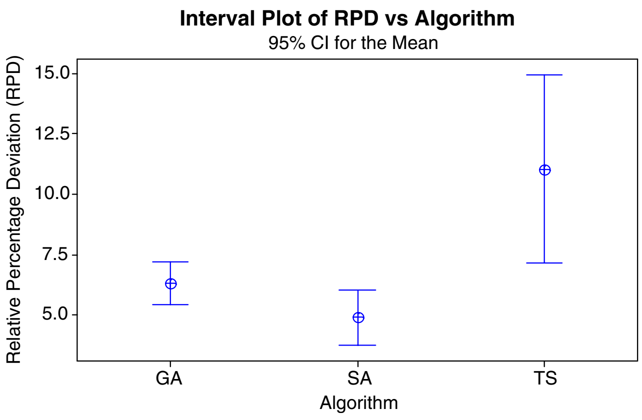

Fig. 11 Means and interval plot for small problems

Table 22 Average RPD for medium problems

Algorithm	TS	SA	GA
Average RPD	1,301.13	284.46	592.7498

Table 23 ANOVA results for medium problems

Source	*df*	SS	MS	*F*	*P* value
Algorithm	2	5,248	2,624	19.37	0.000
Error	699	94,684	135		
Total	701	99,932			

Table 24 Fisher's least-significant difference method results for medium problems

Algorithms	Difference of means	LSD	Significant difference at 95% level
TS and SA	10.17	3.35	Yes
TS and GA	7.08	3.35	Yes
SA and GA	3.09	1.79	Yes

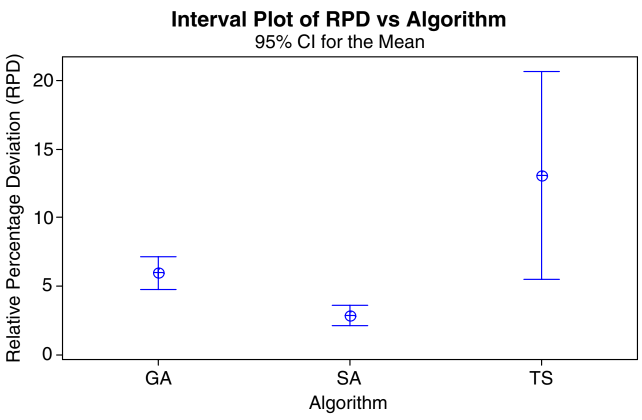

Fig. 12 Means and interval plot for medium problems

Table 25 Average RPD for large problems

Algorithm	TS	SA	GA
Average RPD	1,751.908	196.9846	267.0161

Table 26 ANOVA results for large problems

Source	*df*	SS	MS	*F*	*P* value
Algorithm	2	7,730	3,865	33.57	0.000
Error	465	53,542	115		
Total	467	61,271			

6.2 Comparing efficiency of algorithms

We compare different algorithms in another aspect that is efficiency and use the elapsed CPU time to obtain the best makespan as efficiency measure.

6.2.1 Small problems

Average RPD of algorithms on small problems is shown in Table 19. Based on ANOVA results that are provided in Table 20, there is a significant difference among algorithms' performance. LSD method results are shown in Table 21 and indicate that SA and GA are preferred with 95% confidence. SA has better results than GA but it is not statistically significant. Interval plot for small problems is provided in Fig. 11.

6.2.2 Medium problems

Average RPD of algorithms on medium problems is shown in Table 22 and the results of single-factor ANOVA are provided in Table 23. These results indicate that at least one algorithm is different in mean response. To further analysis, we performed LSD method and results are shown in Table 24. These results demonstrate that SA is preferred with 95% confidence. Interval plot for medium problems can be observed in Fig. 12.

6.2.3 Large problems

For large problems, obtained results for average RPD, ANOVA, and LSD methods are shown in Table 25, 26, and 27, respectively. Based on these results, it is clear that SA and GA are preferred with 95% confidence and there is no significant difference between them, although SA results in smaller mean response. Figure 13 shows the corresponding interval plot.

Table 27 Fisher's least-significant difference method results for large problems

Algorithms	Difference of means	LSD	Significant difference at 95% level
TS and SA	15.55	3.78	Yes
TS and GA	14.85	3.78	Yes
SA and GA	0.7	2.02	No

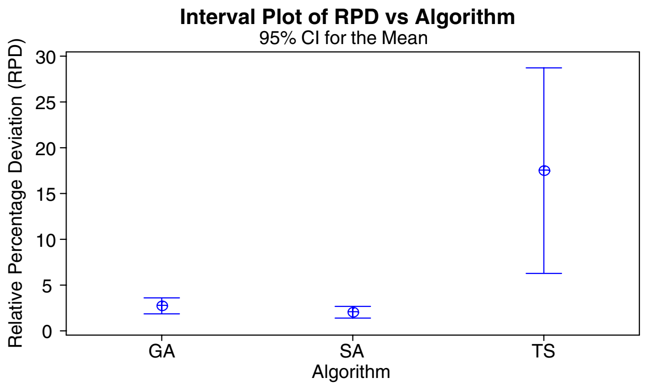

Fig. 13 Means and interval plot for large problems

7 Conclusions and future work

This paper examined three methods to find schedules minimizing makespan in group scheduling problems with sequence-dependent setup times in hybrid flexible flow shops. These methods are based on TS, SA, and GA. TS-based algorithm was developed by Logendran et al [17] and two new metaheuristics based on SA and GA were developed in this research. To evaluate the different methods, 126 test problems were generated including small, medium, and large problems. The data characteristics investigated were designed to reflect characteristics used by other researchers. The metaheuristics are compared on small, medium, and large problems separately in two different aspects: first, best makespan and, second, elapsed CPU time to obtain best makespan. Generally, GA and SA performed best on makespan and CPU time measures, respectively, on the problems examined here.

There are potentially unlimited opportunities for research in scheduling to minimize the makespan in group scheduling in hybrid flexible flow shops with sequence setup times. In this paper, we have addressed only a few areas. In this research, we assume that parallel machines are identical and we can consider a new problem in group scheduling with uniform or unrelated parallel machines. Considering major and minor setup times can be used to define new problems, in this context, when a machine switches from one group to another, a major setup is required and a minor setup occurred between jobs within each group and these setups can be sequence independent or sequence dependent.

References

1. Allison JD (1990) Combining Petrov's heuristic and the CDS heuristic in group scheduling problems. In: Proceedings of the 12th Annual Conference on Computers and Industrial Engineering, Orlando, FL, pp 457–461
2. Al-Qattan I (1988) Designing GT cells enhanced by group scheduling. In: Proceedings of the IIE Integrated Systems Conference, pp 25–30
3. Baker KR (1988) Scheduling the production of components at a common facility. IIE Trans 20:32–35 doi:10.1080/07408178808966147
4. Campbell HG, Dudek RA, Smith ML (1970) A heuristic algorithm for the *n* job, *m* machine sequencing problem. Manage Sci 16:B630–B637
5. Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141 doi:10.1109/4235.771166
6. Gholami M, Zandieh M, Alem-Tabriz A (2008) Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. Int J Adv Manuf Technol doi:10.1007/s00170-008-1577-3
7. Grefenstette JJ (1986) Optimisation of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 16(1):122–128 doi:10.1109/TSMC.1986.289288
8. Ham I, Hitomi K, Yoshida T (1985) Group technology: applications to production management. Kluwer Academic, Hingham
9. Heragu S (1997) Facilities design. PWS, Boston
10. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680 doi:10.1126/science.220.4598.671
11. Kurz ME, Askin RG (2003) Comparing scheduling rules for flexible flow lines. Int J Prod Econ 85:371–388 doi:10.1016/S0925-5273(03)00123-3
12. Kurz ME, Askin RG (2004) Scheduling flexible flow lines with sequence-dependent setup times. Eur J Oper Res 159(1):66–82 doi:10.1016/S0377-2217(03)00401-6
13. Liaee MM, Emmons H (1997) Scheduling families of jobs with setup times. Int J Prod Econ 51:165–176 doi:10.1016/S0925-5273(96)00105-3
14. Logendran R, Nudtasomboon N (1991) Minimizing the makespan of a group scheduling problem: a new heuristic. Int J Prod Econ 22:217–230 doi:10.1016/0925-5273(91)90098-E
15. Logendran R, Mai L, Talkington D (1995) Combined heuristics for bi-level group scheduling problems. Int J Prod Econ 38:133–145 doi:10.1016/0925-5273(94)00083-M
16. Logendran R, Carson S, Hanson E (2005) Group scheduling in flexible flow shops. Int J Prod Econ 96(2):143–155 doi:10.1016/j.ijpe.2004.03.011
17. Logendran R, deSzoeke P, Barnard F (2006) Sequence-dependent group scheduling problems in flexible flow shops. Int J Prod Econ 102:66–86 doi:10.1016/j.ijpe.2005.02.006
18. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculation by east computing machines. J Chem Phys 21:1087–1091 doi:10.1063/1.1699114
19. Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Berlin
20. Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimisation problems. Evol Comput 4(1):1–32 doi:10.1162/evco.1996.4.1.1
21. Naderi B, Zandieh M, Fatemi Ghomi SMT (2008) Scheduling job shop problems with sequence-dependent setup times. Int J Prod Res doi:10.1080/00207540802165817
22. Nakamura N, Yoshida T, Hitomi K (1978) Group production scheduling for minimum total tardiness: part I. AIIE Transactions 10:157–162
23. Norman BA, Bean JC (1999) A genetic algorithm methodology for complex scheduling problems. Nav Res Logistics 46:199–211 doi:10.1002/(SICI)1520-6750(199903)46:2<199::AID-NAV5>3.0.CO;2-L
24. Ozden M, Egbelu PJ, Iyer AV (1985) Job scheduling in a group technology environment for a single facility. J Comput Ind Eng 9:67–72 doi:10.1016/0360-8352(85)90037-3
25. Petrov VA (1966) Flow line group production planning. Business Publications, London
26. Radharamanan R (1986) A heuristic algorithm for group scheduling. In: Proceedings of the International Industrial Engineering Conference, pp 229–236
27. Reddy V, Narendran TT (2003) Heuristics for scheduling sequence dependent set-up jobs in flow line cells. Int J Prod Res 41(1):193–206 doi:10.1080/00207540210163973
28. Schaller JE, Gupta JND, Vakharia AJ (2000) Scheduling a flow line manufacturing cell with sequence dependent family setup times. Eur J Oper Res 125:324–339 doi:10.1016/S0377-2217(99)00387-2
29. Webster S, Baker KR (1995) Scheduling groups of jobs on a single machine. Oper Res 43:692–703
30. Zandieh M, Fatemi SMT (2003) A framework and a classification scheme for modeling production systems. In: Proceedings of the Second National Industrial Engineering Conference, Yazd University, Yazd, Iran, pp 308–315
31. Zandieh M, Fatemi Ghomi SMT, Moattar Husseini SM (2006) An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Appl Math Comput 180:111–127 doi:10.1016/j.amc.2005.11.136