J Intell Manuf (2017) 28:1269–1283
DOI 10.1007/s10845-015-1049-1

A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem

Antonio Costa · Fulvio Antonio Cappadonna · Sergio Fichera

Received: 14 November 2013 / Accepted: 30 January 2015 / Published online: 14 February 2015

Abstract In this paper, the flow-shop sequence-dependent group scheduling (FSDGS) problem is addressed with reference to the makespan minimization objective. In order to effectively cope with the issue at hand, a hybrid metaheuristic procedure integrating features from genetic algorithms and random sampling search methods has been developed. The proposed technique makes use of a matrix encoding able to simultaneously manage the sequence of jobs within each group and the sequence of groups to be processed along the flow-shop manufacturing system. A well-known problem benchmark arisen from literature, made by two, three and six-machine instances has been taken as reference for both tuning the relevant parameters of the proposed procedure and assessing performances of such approach against the two most recent algorithms presented in the body of literature addressing the FSDGS issue. The obtained results, also supported by a properly developed ANOVA analysis, demonstrate the superiority of the proposed hybrid metaheuristic in tackling the FSDGS problem under investigation.

Keywords Group scheduling · Genetic algorithm · Flow shop · Setup dependent · Cellular manufacturing

A. Costa (✉) · S. Fichera
Dipartimento di Ingegneria Industriale (DII),
University of Catania, Viale Andrea Doria 6, 95125 Catania, Italy
e-mail: antonio.costa@dii.unict.it

F. A. Cappadonna
Dipartimento di Ingegneria Elettrica, Elettronica e Informatica (DIEEI),
University of Catania, Viale Andrea Doria 6,
95125 Catania, Italy

Introduction

In the last decades several manufacturing companies have benefited from implementing the production philosophy based on cellular manufacturing systems (CMS). Cellular manufacturing (CM) entails a production system wherein parts needing similar technological processes are grouped in distinct working cells. Benefits a firm may gather by implementing a CM system should include reduction in setup time, throughput time, tooling needs, and work-in-process inventories, simplified flow of parts, and improved human relations (Hyer and Wemmerloev 1989; Paredes et al. 1998; Shankar and Vrat 1999). A manufacturing framework based on CMS also facilitates successful implementation of modern manufacturing technologies, such as computer integrated manufacturing, JIT production and flexible manufacturing systems (Gallagher and Knight 1986). Though implementation of a CMS may generate several advantages, it is undeniable that designing and planning such a kind of manufacturing systems constitutes a demanding challenge. Indeed, configuring a CMS entails a series of designing and planning phases, such as cell formation problem, layout of CMS, production planning in CMS, scheduling in CMS, etc.

Though cell formation problem has been gaining most of the scientific attention (Mahmoodi and Dooley 1991; Logendran et al. 1995; Soleymanpour et al. 2002; Baykasoglu 2004), benefits arising from the effective implementation of scheduling in CMS have extensively been demonstrated since Seventies (Hitomi and Ham 1976). In the design of CMS, whether a perfect group formation is performed, dissimilar machines that belong to the same cell have to process a set of parts belonging to a specific part family (Logendran and Sirikrai 2000). Therefore, a part family can be divided into several groups so that each group needs similar setup requirements (Schaller 2001). Hence, a group is a subset of a

part family and each group includes a number of jobs. Since jobs in the same group are similar, the setup time required to change from one job to another is assumed to be negligible compared to the run time. Whereas, a measurable setup time would be required to change from one group to another because of the significant difference between the technological requirements of two distinct groups.

The body of literature has frequently investigated the Flow-Shop Group Scheduling (FSGS) problem, basically due to the affinity between such theoretical model and many real-world manufacturing situations. Whether setup times of groups are sequence dependent, the problem may be denoted as flow shop sequence dependent group scheduling (FSDGS) problem (Salmasi et al. 2011). Therefore, there exists a clear advantage in processing together jobs belonging to the same group, thus arranging the whole production schedule through subsequent part families. In words, the decision-making problem to be tackled consists in finding the optimal sequence of groups and the optimal sequence of jobs within each group, with reference to a certain performance measure. However, since each feasible solution for a FSGS problem may be described as a regular sequence of jobs to be processed by each machine of the shop floor, such a problem still remains a permutation scheduling issue, like in the traditional flow-shop model.

One of the first studies concerning the FSGS problem is ascribable to Ham et al. (1985), who presented an optimizing algorithm for minimizing the total completion time in a two-machine group scheduling problem. Their work was further developed by Logendran and Sriskandarajah (1993), who demonstrated how a two-machine group scheduling problem with no buffer and anticipatory setup is strongly NP-hard. Two years later, Logendran et al. (1995) addressed the more general case of a M-machine ($M > 2$) FSGS problem under the total completion time viewpoint, proposing an efficient heuristic solution algorithm.

Recent researches dealing with group technology applications in flow-shop manufacturing environments mainly focus on the flow-shop sequence-dependent group scheduling (FSDGS) problem, i.e. a particular case of the FSGS issue in which the setup time required by a certain group of jobs depends on the technological features of the previously processed group. The growing interest towards such a variant of the classical flow shop group scheduling problem is essentially due to its undeniable industrial implications. Examples of FSDGS problems arising from the real industrial practice regarded: Printed Circuit Board (PCBs) manufacturing (Schaller et al. 2000; Pinedo 2012), label sticker manufacturing (Lin and Liao 2003) and automotive production (Salmasi et al. 2010). Recently, results of a case study revealed that the group-based method can reduce the manufacturing lead time because of the reducing of processing set-up efforts (Yu et al. 2013).

Three comprehensive reviews involving the FSDGS problem have been proposed by Allahverdi et al. (1999), Cheng et al. (2000), and Zhu and Wilhelm (2006), respectively. Schaller et al. (2000) approached the problem of scheduling part families with sequence dependent setup times and jobs within each part family in a flow line manufacturing cell with reference to the makespan minimization objective. To this aim, authors tested several heuristics and compared their performance against a set of lower bounds obtained through a generalization of the machine-based bounding method, commonly used for the regular flow-shop issue. França et al. (2005) developed two evolutionary algorithms, namely a Genetic Algorithm (GA) and a Memetic Algorithm (MA) with a local search, for minimizing makespan in a flow-shop manufacturing cell with sequence dependent setups among families. After an extensive experimental analysis, authors put in evidence the superiority of the proposed procedures with respect to a set of well-known heuristic algorithms coming from literature. Notably, the memetic procedure slightly outperformed the GA-based technique. Logendran et al. (2006) proposed three search algorithms based on Tabu Search (TS) for solving industry-size two-machine group scheduling problems with sequence dependent setups times. They evaluated the quality of obtained solutions through a properly developed lower bound method. Hendizadeh et al. (2008) and Salmasi and Logendran (2008) also investigated the use of TS-based algorithms for minimizing makespan in the more general case of the M-machine FSDGS problem. Celano et al. (2010) used the FSDGS model with limited inter-operational buffer capacity for addressing a scheduling problem truly observed in the inspection department of a semiconductor manufacturing company. In order to minimize the makespan, the authors developed a matrix-encoding GA, whose effectiveness has been proven against a TS and the heuristic proposed by Nawaz et al. (1983) for the classical flow shop problem. Salmasi et al. (2010) presented a Mixed Integer Linear Programming (MILP) formulation and two metaheuristic methods, namely a TS and a Hybrid Ant Colony Optimization (HACO) algorithm, for minimizing the total flow time in a FSDGS problem. Both metaheuristics were tested over a wide range of test cases, from which the superiority of the HACO procedure clearly emerged. One year later Salmasi et al. (2011) investigated the use of the HACO algorithm for minimizing makespan in a FSDGS problem. A similar issue was addressed by Hajinejad et al. (2011), who succeeded in outperforming the HACO approach by means of a Hybrid Particle Swarm Optimization (HPSO) algorithm. Finally, Naderi and Salmasi (2012) proposed two different MILP formulations, along with a hybrid metaheuristic technique named GSA composed by both genetic and simulated annealing algorithm, to cope with the FSDGS issue in terms of total completion time minimization.

These last two research contributions represent two milestones in the field of the meta-heuristic optimization of the FSDGS problem. Nevertheless, though the authors have extensively demonstrated the effectiveness of their algorithms, both methods may suffer from some limitations. With reference to the PSO technique, it is well-known that the basic equations governing such method are suitable for continuous optimization problems, and their use in discrete domains, like permutation scheduling problems, could cause loss of information, leading to a general weakening of the search strategy (Marinakis and Marinaki 2013). In addition, whenever a PSO is employed for addressing a permutation problem, it requires a specific sorting procedure to transform a real coded solution into a permutation one, thus slowing down the overall optimization procedure due to a higher computational burden.

Every search algorithm needs to address the exploration and exploitation of a search space. Exploration is the process of visiting entirely new regions of a search space, whilst exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points. In order to be successful, evolutionary algorithms like GAs need to establish a good ratio between exploration and exploitation (Crepinsek et al. 2013); thus, integrating a simulated annealing with a GA constitutes a challenging task that may lead to unsatisfying results, especially when parameters of both algorithms are not selected through an extensive tuning analysis. Furthermore, a combined use of two metaheuristic procedures could require a larger computational burden to converge, thus affecting the global efficiency of the search method.

In light of the aforementioned remarks, this paper aims to present a novel metaheuristic procedure able to outperform the latest optimization algorithms proposed in the field of FSDGS problem. Since Zandieh and Karimi (2011) and Luo et al. (2013) recently revealed the effectiveness of Genetic Algorithms in solving group scheduling problems, and Gao et al. (2006) and Meeran and Morshed (2012) demonstrated as local search methods may strongly enhance the search ability of genetic algorithms, a proper GA powered by a specific random sampling technique (RS) has been developed.

In order to outperform the two mentioned competitors in addressing the FSDGS issue, namely HPSO and GSA, the proposed metaheuristic algorithm makes full use of two distinct crossover operators, two mutation operators, as well as elitism and diversity operators that work through a permutation encoding to avoid any premature convergence. In addition, a Biased Random Sampling (BRS) technique has been embedded within the genetic framework as to enhance the performance of the proposed method under both the quality of solution and computational burden viewpoint.

The remainder of the paper is organized as follows: section "Problem description" deals with the description of the FSDGS problem, section "The proposed hybrid genetic algorithm" presents the structure and the operators of the proposed metaheuristic procedure. Section "Experimental calibration" reports the results of the calibration campaign performed in order to properly set all the relevant parameters of the algorithm. In section "Computational experiments and results" an extensive comparison among the proposed optimization procedure and the two latest algorithms presented in the field of FSDGS problems is reported. Finally, section "Conclusions" concludes the paper.

Problem description

In a FSDGS problem a set of G groups of jobs ($k = 1, \ldots, G$) has to be processed along a serial manufacturing system characterized by M ($m = 1, \ldots, M$) workstations. Each group k holds n_k jobs ($j = 1, \ldots, n_k$) and the total amount of jobs to be processed along the system is equal to $\sum_{k=1}^{G} n_k = N$. According to the group technology theory, jobs taking part to a specific group are characterized by the same technological requirements and, as a consequence, setup times between them can be ignored or, in alternative, included into processing times. On the other hand, jobs that belong to different groups require different technological processes and, for such a reason, setup times between groups themselves $s_{l,k}$ ($l, k \in 1, \ldots, G | l \neq k$) cannot be ignored.

Similarly being done by previous researches addressing the FSDGS issue, group setup operations are here assumed to be anticipatory, i.e. they can be performed even if the first job belonging to a group to be processed is still unavailable. Furthermore, neither precedence relationships exist among groups nor among jobs within the same group. Pre-emption is not allowed, i.e. when a job starts to be processed, it must be completed before leaving the workstation. Both group and job passing is not allowed, i.e. they must visit in succession all the workstations of the manufacturing system. Both groups and jobs must visit machines according to the same order (sequence). All jobs are ready to be worked at the beginning of the planning period, i.e. job release times are equal to zero. Machines are continuously available during the whole production session. Buffers between workstations have unlimited capacity. Jobs descriptors like processing times and setup times are a-priori known. The objective function to be minimized coincides with the completion time of the last job pertaining to the last group in the last workstation.

The proposed hybrid genetic algorithm

As demonstrated by Schaller et al. (2000), minimizing makespan in a FSDGS problem is NP-hard. Whenever a

NP-hard combinatorial problem needs to be solved, metaheuristic algorithms may represent an effective and time-saving alternative to other exhaustive approaches. Since Celano et al. (2010) recently proved both the efficacy and the efficiency of the genetic algorithm approach for solving the FSDGS problem, a GA based optimization procedure embedding a random sampling search technique has been proposed for tackling the problem under investigation.

Generally, a GA works with a set of problem solutions called *population*. At every iteration, a new population is generated from the previous one by means of two operators, *crossover* and *mutation*, applied to solutions (*chromosomes*) selected on the basis of their *fitness*, i.e. the objective function value; thus, best solutions have greater chances of being selected. Crossover operator generates new solutions (*offspring*) by coalescing the structures of a couple of existing ones (*parents*), while mutation operator brings a change into the scheme of selected chromosomes, with the aim to avoid any premature convergence into local optima. The algorithm proceeds by evolving the population through successive *generations,* until a given stopping criterion is reached.

Whenever a real problem should be addressed through an evolutionary algorithm, the choice of a proper *encoding* scheme (i.e. the way a solution is represented by a string of *genes*) plays a key role under both the quality of solutions and the computational burden viewpoints (Costa et al. 2013). In addition, a valid *decoding* procedure able to transform a given string into a feasible solution should be provided.

The following subsections deal with a detailed description of the proposed GA-based optimization procedure, named HGA, illustrating encoding/decoding strategies and genetic operators adopted, as well as the random sampling local search technique embedded in the algorithm for enhancing search performances.

Problem encoding

Problem encoding is the way a given problem to be optimized through a metaheuristic procedure can be represented by means of a numerical chromosome. With reference to the proposed HGA, a matrix-based encoding scheme has been employed. Following the same notation adopted in the introduction Section, each solution is described by a $(G+1)\times n_{\max}$ matrix, being $n_{\max} = \max_{k=1}^{G}\{n_k\}$. The first G rows consist of the permutation vectors π^k indicating the sequence of jobs within each group k, while the last row is the permutation vector Ω representing the sequence of groups to be processed:

$$\begin{bmatrix} \pi_1^1, \ldots, \pi_{n_1}^1 \\ \ldots \\ \pi_1^k, \ldots, \pi_{n_k}^k \\ \ldots \\ \frac{\pi_1^G, \ldots, \pi_{n_G}^G}{\Omega_1, \ldots, \Omega_G} \end{bmatrix} \quad (1)$$

Hereinafter, row r ($r = 1, 2, \ldots, G+1$) of the partitioned matrix will be denoted as a *sub-chromosome*. Hence, a certain sub-chromosome r ($r = 1, 2, \ldots, G$) corresponds to the sequence of jobs scheduled within group r; sub-chromosome $r = G+1$ identifies the sequence Ω of groups. For sake of clarity, a feasible solution for a problem in which $G = 5$ and $n_{\max} = 5$ could be represented by the following [C$_1$] chromosome:

$$[C_1] = \left[\begin{array}{ccccc} 3 & 1 & 2 & 0 & 0 \\ 2 & 5 & 1 & 4 & 3 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 4 & 1 & 2 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ \hline 5 & 1 & 3 & 2 & 4 \end{array}\right] \quad (2)$$

Sub-chromosomes from 1 to 5 hold the schedules of jobs within each group (i.e., schedule 3-1-2 for group 1, schedule 2-5-1-4-3 for group 2, schedule 2-1 for group 3, schedule 3-4-1-2 for group 4, schedule 1-2-3 for group 5); sub-chromosome $r = G+1$ fixes the sequence of groups Ω = 5-1-3-2-4. All the digits equal to zero do not take part either to the solution decoding or to the genetic evolutionary process. Once the problem encoding is defined, the fitness function composed by N_{pop} individuals pertaining to the genetic population may be computed.

Crossover operator

Crossover operator allows the genetic material of two properly selected parents to be recombined to generate offspring. The selection mechanism employed by the proposed HGA is the well-known roulette wheel scheme (Michalewicz 1994), which assigns to each solution a probability of being selected inversely proportional to the makespan value. Once two parent chromosomes have been selected, each couple of sub-chromosomes belonging to the parent solutions undergoes crossover according to an a-priori fixed probability, hereinafter called p_{cr}. Two crossover operators have been adopted to recombine alleles within each couple of sub-chromosomes: they are denoted by *Position Based Crossover* (PBC) and *Two Point Crossover* (TPC), respectively. Both of these two operators have been largely adopted by literature within GAs applied to combinatorial problems (Gen and Cheng 2000), (Celano et al. 2010), (Kim et al. 2003). PBC generates offspring by considering the relative order in which some alleles are positioned within the parents. Indeed, it works on a couple of sub-chromosomes (P1) and (P2) as

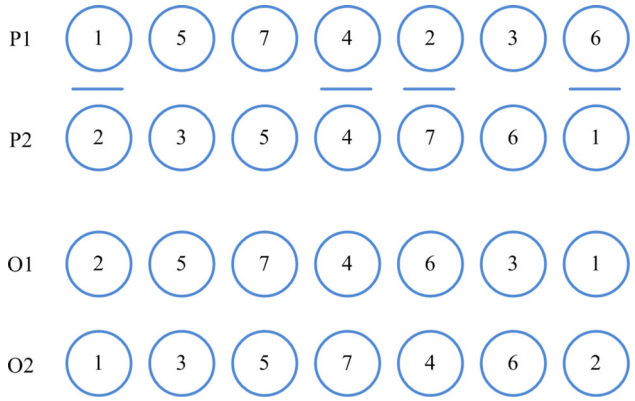

Fig. 1 Position based crossover (PBC)

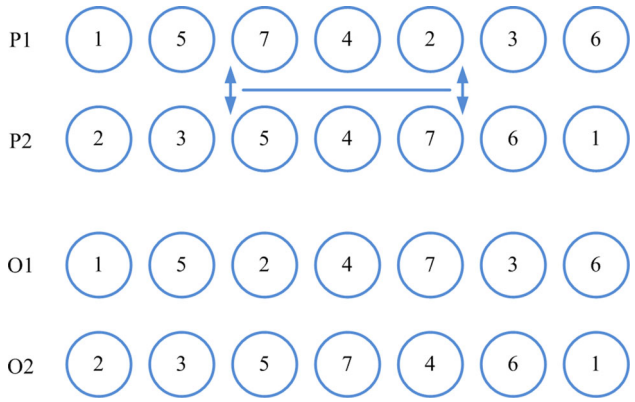

Fig. 2 Two point crossover (TPC)

follows: 1) one or more alleles are randomly selected; 2) selected alleles of *parent 1* (P1) are reordered in *offspring 1* (O1) as they appear within *parent 2* (P2); 3) remaining elements are positioned in the sequence by copying directly from *parent 1* the unselected alleles. The same procedure is applied to the second parent, namely *parent 2*, to obtain offspring (O2). Figure 1 shows application of PBC to a couple of parents where alleles in positions {1}, {4}, {5}, and {7} have been selected. As far as the TPC method is concerned, two positions are randomly selected and each sub-chromosome parent is divided into three blocks of alleles: both first and third block are copied directly in the corresponding offspring, while the alleles belonging to the middle block are reordered within the offspring in the same order as they appear in the other parents (see Fig. 2). A "fair coin toss" probability equal to 0.5 has been chosen for selecting either PBC or TPC crossover.

Mutation operator

After a new population has been generated by means of crossover, mutation operator is applied according to an a-priori fixed probability p_m. Whether mutation occurs, a chromosome is randomly chosen from the population; within such chromosome, a sub-chromosome is randomly selected for mutation. Two kind of operators have been adopted in the present research: an *Allele Swapping Operator* (ASO), which performs an exchange of two randomly selected alleles of the sub-chromosome; and a *Block Swapping Operator* (BSO), which performs a block exchange (see Fig. 3). A "fair coin toss" probability equal to 0.5 has been chosen for selecting either ASO or BSO mutation operator.

Elitism and diversity operators

To avoid any loss of the current best genetic information, the survival of the two fittest individual within the population is ensured by an elitist strategy, generation by generation. In addition, a population diversity control technique has been embedded within the proposed optimization procedure, in order to mutate those identical chromosomes exceeding a pre-selected value D_{max}. In the present research, a D_{max} value equal to 2 has been selected, thus avoiding to have more than two identical solutions within the current population.

Local search and termination rule

In order to improve the performances of the proposed meta-heuristic procedure, a Biased Random Sampling (BRS) search scheme (Baker and Trietsch 2009) has been embedded within the evolutionary optimization strategy of the proposed HGA. Such procedure operates only on a sub-population, whose size is equal to $N_{best} < N_{POP}$, which includes just the best individuals obtained after each generation. For each generic chromosome C_s ($s \in 1, 2, \ldots, N_{best}$), a sample of N_{BRS} neighbour solutions is generated by modifying the sequence Ω of groups related to C_s. In the present research N_{BRS} has been set equal to 4. Each neighbour chromosome NC_s^i ($i = 1, 2, \ldots, N_{BRS}$) of C_s, holds a sequence of groups (hereinafter denoted as $\overline{\Omega}^i$) obtained as follows. The first group $\overline{\Omega}_1^i$ is drawn by the genes of Ω according to the following distribution of probability:

$$p_{1,k} = \alpha^k \cdot \frac{1}{\sum_k \alpha^k} \qquad k = 1, 2, \ldots, G \tag{3}$$

where, $p_{1,k}$ is the probability to select Ω_k, i.e. the k-th group of sequence Ω, as first element of $\overline{\Omega}^i$. Such probability is defined as "biased" since it favours the first group of Ω to the second, the second to the third, and so on. The parameter α is used to control how the probability decreases when moving from one group of Ω to another. For the proposed HGA, a value of $\alpha = 0.8$ has been selected. Thus, supposing to have $G = 5$, the first element of the new sub-chromosome $\overline{\Omega}^i$ will be drawn from Ω according to the probabilities reported in Table 1.

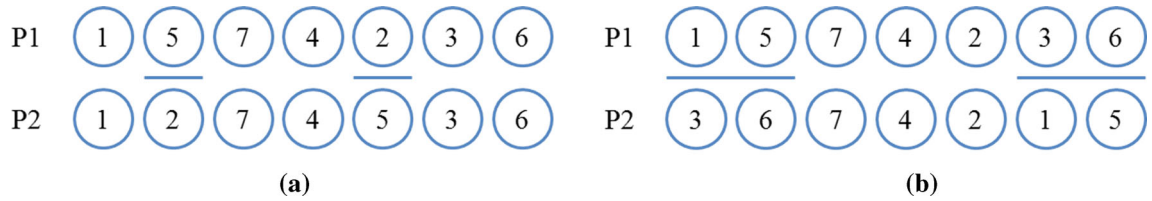

Fig. 3 **a** Allele swapping (ASO) and **b** block swapping (BSO) mutation operators

Table 1 RS probability distribution for $G = 5$, $\alpha = 0.8$

k	1	2	3	4	5
$p_{1,k}$	0.297	0.238	0.190	0.152	0.122

Once $\overline{\Omega}_1^i$ has been drawn, the second group, i.e. $\overline{\Omega}_2^i$, will be extracted from the remaining ones of Ω in a similar fashion. More in general, the j-th group of $\overline{\Omega}^i$ will be drawn from the remaining elements of Ω on the basis of the following distribution of probability:

$$p_{j,k} = \alpha^k \cdot \frac{1}{\sum_k \alpha^k} \qquad k = 1, 2, ..., G + 1 - j \tag{4}$$

With this structure, the first job on Ω has the highest probability of being selected, the second job has the second highest probability, and so on. In addition, the probabilities decrease in a geometric manner, but the nature of the decrease can be controlled by selecting the parameter α.

Hence, a new candidate solution generated through BRS method may generally differ from the seed chromosome C_s for more than a couple of genes. Therefore, the proposed BRS allows performing both an exploration and an exploitation phase on the seed sequence, thus reducing the risk for the GA of being trapped into local optima.

After a total of N_{BRS} solutions are originated from chromosome C_s, the best one is used for replacing C_s in the current population, whether it leads to a better makespan value. Such procedure is executed for all the N_{best} individuals originally selected. Once the local search mechanism is completed, the new obtained population drive the next generation cycle.

The termination rule of the proposed HGA consists in $N \cdot M$ seconds of CPU time, similarly being done by Naderi and Salmasi (2012). During the experimental calibration phase, properly discussed in the following section, it has been observed that such time limit guarantees a satisfactory convergence path towards the local optimum, regardless of the adopted parameters.

Pseudo-code of HGA optimization strategy

To sum up, the whole optimization strategy followed by the proposed HGA can be illustrated through the following steps:

Step 1: Initialization of parameters N_{pop}, p_{cr}, p_m, D_{max}, N_{best}, N_{BRS}, α;
Step 2: Generation: N_{pop} chromosomes composing the initial population are randomly generated;
Step 3: Selection and Crossover: two individuals are selected according to the roulette-wheel selection criterion. If probability of crossover p_{cr} is satisfied then Position Based or Two Point Crossover is applied to generate offspring, else the two individuals are copied into the new population. If crossover is applied, the two best chromosomes individuated between parents and offspring are placed into the new population;
Step 4: Mutation: each individual of the population gets mutated if probability of mutation p_m is satisfied. Mutation operator may be randomly chosen between: Allele Swapping and Block Swapping;
Step 5: Population control: a mutation operator is applied to those duplicates of a given chromosome exceeding D_{max};
Step 6: Local search: application of BRS procedure to the N_{best} best individuals of the population;
Step 7: Replacement: Updating of the current population due to BRS;
Step 8: Exit criterion: if exit criterion is encountered then Stop algorithm, else go to Step 3.

Experimental calibration

A proper calibration phase has been carried out, with the aim of selecting a suitable set of parameters for the proposed HGA. To this end, the same test problem specifications proposed by Salmasi et al. (2011) have been taken into account. Basically, three distinct benchmarks of problems characterized by different numbers of machines, i.e. 2, 3, and 6, respectively, have been generated. For each benchmark, three distinct factors, namely number of groups, number of jobs within each group and setup times of groups on each machine have been combined as to obtain a full factorial experimental plan as shown in Tables 2, 3 and 4, where symbol U[a, b] denotes a value extracted by a uniform distribution between a and b.

It is worth noting that a total of $27 + 81 + 27 = 135$ separate instances describing a consistent data set for

Table 2 Benchmark of instances for the FDSGS problem with 2 machines

Factor	Level	Value	
Number of groups	1	U [1,5]	
	2	U [6,10]	
	3	U [11,16]	
Number of jobs in a group	1	U [2,4]	
	2	U [5,7]	
	3	U [8,10]	
Setup times of machine M_i	1	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [17,67]
	2	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [1,50]
	3	$M_1 \rightarrow$ U [17,67]	$M_2 \rightarrow$ U [1,50]

calibrating the HGA procedure have been generated. All instances have been created by extracting job processing times according to a uniform distribution in the range [1,20].

The calibration phase has been executed in order to properly tune the following four parameters characterizing the developed HGA, i.e.: population size (N_{pop}), crossover and mutation probabilities (p_{cr}, p_m), size of the sub-population to be modified through BRS procedure (N_{best}). For each parameter, three different levels have been taken into account, as illustrated in Table 5, thus generating a total of $3^4 = 81$ different configurations of the proposed metaheuristic algorithm. For each instance, all the provided algorithm configurations have been tested. Therefore, a total of $135 \cdot 81 = 10,935$ runs have been considered.

In order to identify the best combination of values for the aforementioned parameters, an ANOVA analysis (Montgomery 2008) has been performed by means of Design Expert® 7.0.0 version commercial tool. The response variable studied was the Relative Percentage Deviation (*RPD*), calculated according to the following formula:

$$RPD = 100 \cdot \frac{HGA_{sol} - BEST_{sol}}{BEST_{sol}} \quad (5)$$

where HGA_{sol} is the makespan found by the HGA procedure running a specific combination of parameters, and $BEST_{sol}$ is the best solution over the whole set of results concerning the same instance. The proposed GA has been coded in MATLAB® language and executed on a 2GB RAM virtual machine embedded on a workstation powered by two quad-core 2,39 GHz processors.

Figures 4, 5, 6 and 7 report the means plots with LSD intervals at 95 % confidence level obtained for each one of the tuned parameters.

Figure 4 shows that the best value of population size (N_{pop}) among those tested is 70. By Fig. 5 it can be infered that crossover probability (p_{cr}) should be set to 0.9. Though it is not statistically significant, as shown in Fig. 6, mutation probability was set to 0.1. Finally, Fig. 7 suggests that only the best 30 % individuals of each population should be subjected to BRS procedure.

The lack of a statistically significant difference among levels concerning p_m may be easily explained if the matrix encoding scheme is taken into account. In fact, each chromosome is mainly composed by job sub-chromosomes while just one sub-chromosome holds the sequence of groups. Therefore, there is a higher probability of applying the mutation operator to a job sub-chromosome instead of the Ω vector; thus, being jobs within each group quite similar, different p_m levels should not influence the response variable.

Table 3 Benchmark of instances for the FDSGS problem with 3 machines

Factor	Level	Value		
Number of groups	1	U [1,5]		
	2	U [6,10]		
	3	U [11,16]		
Number of jobs in a group	1	U [2,4]		
	2	U [5,7]		
	3	U [8,10]		
Setup times of machine M_i	1	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [45,95]
	2	$M_1 \rightarrow$ U [17,67]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [17,67]
	3	$M_1 \rightarrow$ U [45,95]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [1,50]
	4	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [17,67]
	5	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [1,50]
	6	$M_1 \rightarrow$ U [17,67]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [45,95]
	7	$M_1 \rightarrow$ U [17,67]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [1,50]
	8	$M_1 \rightarrow$ U [45,95]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [45,95]
	9	$M_1 \rightarrow$ U [45,95]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [17,67]

Table 4 Benchmark of instances for the FDSGS problem with 6 machines

Factor	Level	Value					
Number of groups	1	U [1,5]					
	2	U [6,10]					
	3	U [11,16]					
Number of jobs in a group	1	U [2,4]					
	2	U [5,7]					
	3	U [8,10]					
Setup times of machine M_i	1	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [17,67]	$M_3 \rightarrow$ U [45,95]	$M_4 \rightarrow$ U [92,142]	$M_5 \rightarrow$ U [170,220]	$M_6 \rightarrow$ U [300,350]
	2	$M_1 \rightarrow$ U [1,50]	$M_2 \rightarrow$ U [1,50]	$M_3 \rightarrow$ U [1,50]	$M_4 \rightarrow$ U [1,50]	$M_5 \rightarrow$ U [1,50]	$M_6 \rightarrow$ U [1,50]
	3	$M_1 \rightarrow$ U [300,350]	$M_2 \rightarrow$ U [170,220]	$M_3 \rightarrow$ U [92,142]	$M_4 \rightarrow$ U [45,95]	$M_5 \rightarrow$ U [17,67]	$M_6 \rightarrow$ U [1,50]

Table 5 Experimental calibration of proposed HGA

Parameter	Notation	No. of levels	Levels
Population size	N_{pop}	3	(30, 50, 70)
Crossover probability	p_{cr}	3	(0.5, 0.7, 0.9)
Mutation probability	p_m	3	(0.05, 0.1, 0.2)
Number of individuals subjected to BRS procedure	N_{best}	3	$(0.3 \cdot N_{pop}, 0.5 \cdot N_{pop}, N_{pop})$

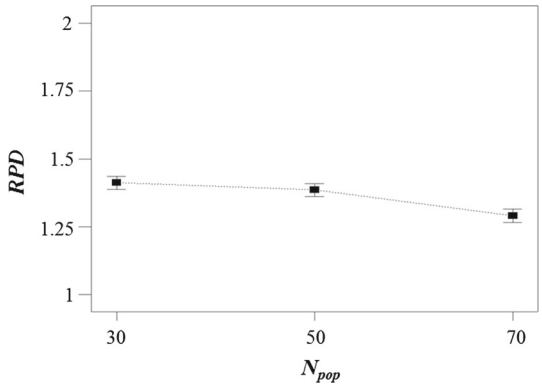

Fig. 4 Means plot with 95 % LSD intervals obtained for N_{pop} parameter

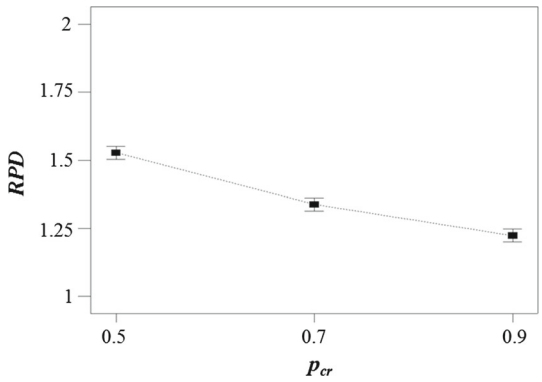

Fig. 5 Means plot with 95 % LSD intervals obtained for p_{cr} parameter

Computational experiments and results

After the best combination of parameters has been selected for the proposed HGA, an extensive comparative campaign has been performed with the aim of assessing the developed metaheuristic procedure against the two most recent methods arisen from the relevant literature in the field of FSDGS scheduling. In the following paragraphs, a brief description of such algorithms is reported.

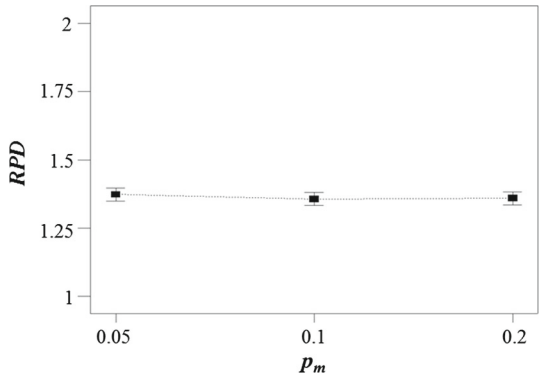

Fig. 6 Means plot with 95 % LSD intervals obtained for p_m parameter

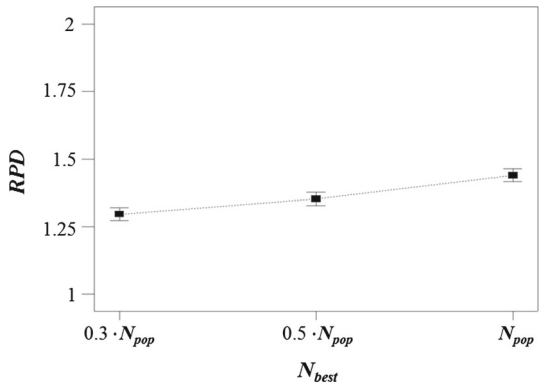

Fig. 7 Means plot with 95 % LSD intervals obtained for N_{best} parameter

– The Hybrid Particle Swarm Optimization algorithm (HPSO) devised by Salmasi et al. (2011). Such method employs a real number matrix-encoding scheme and makes full use of a properly-developed transformation procedure able to convert the components of each solution to integer numbers, so to obtain the sequence of groups and jobs within groups to be scheduled. Furthermore, the authors equipped the algorithm with a neighborhood search strategy, called *individual enhancement*, aimed to improve the search mechanism by balancing exploration and exploitation phases.
– The metaheuristic procedure hybridizing Genetic and Simulated Annealing algorithms (hereinafter coded as GSA) proposed by Naderi and Salmasi (2012). Similarly to the proposed HGA, such algorithm works by a matrix-encoding based on integer numbers. It employs a twofold optimization strategy: a genetic algorithm is used to find the sequence of groups, while a simulated annealing-based local search engine drives the search towards better job sequences.

The comparative campaign has been performed on the basis of the same test problem specifications employed in the calibration phase. This time, however, two distinct replicates have been randomly generated for each problem of the proposed benchmark. Therefore, a total of $54 + 162 + 54 = 270$ separate instances have been created. The overall set of instances has been solved by means of the three mentioned optimization procedures, namely HGA, HPSO and GSA. Thus, a total of $270 \cdot 3 = 810$ runs have been taken into account. Stopping criterion was set to $N \cdot M$ seconds of CPU time for all algorithms tested. The key performance indicator used to compare the alternative metaheuristics is the Relative Percentage Deviation (*RPD*), calculated as follows:

$$RPD = 100 \cdot \frac{ALG_{sol} - BEST_{sol}}{BEST_{sol}} \tag{6}$$

where ALG_{sol} is the solution provided by a given algorithm with reference to a certain instance and $BEST_{sol}$ is the lowest makespan value among those obtained by the executed optimization procedures.

In the following sub-sections, obtained results with reference to two-, three- and six-machine problems, are reported, respectively.

Comparison for two-machine problems

As far as two-machine problems are concerned, three levels for each experimental factor (i.e., number of groups, number of jobs within groups and setup times) have been combined as reported in Table 2. For each problem configuration, two separate instances have been randomly generated. Thus, a total of 54 test problems have been solved by each one of the tested metaheuristic procedures.

Table 6 reports the average *RPD*s obtained by the algorithms, along with a performance index, hereinafter coded as N_{opt}, denoting the number of times (out of two) each optimization procedure achieves the best solution among those provided by the three metaheuristics, for a specific problem configuration.

The obtained results highlight the effectiveness of both HGA and HPSO algorithms in solving the instances embedded in the proposed benchmark. A slight outperformance of the proposed hybrid genetic algorithm can be noticed. HGA ensures the lowest average *RPD* and reaches the best solution 42 times out of 54, thus overcoming both HPSO, whose N_{opt} is equal to 41, and GSA, that finds the minimum makespan value in just 24 test problems out of 54.

In order to infer some statistical conclusion over the difference observed among the tested algorithms, an ANOVA analysis has been performed through Design Expert® 7.0.0 version commercial tool, calculating LSD intervals at 95 % confidence level for the *RPD*s connected to each optimization procedure. The corresponding chart is reported in Fig. 8.

The chart clearly shows the superiority of HGA and HPSO algorithms compared to GSA. Even though the average *RPD* value reported by the proposed hybrid genetic algorithm is

Table 6 Average *RPD* and N_{opt} values for two-machine problems

Level of factors			Average *RPD*			N_{opt}		
Number of groups	Number of jobs in a group	Setup times of machine M_i	HGA	HPSO	GSA	HGA	HPSO	GSA
1	1	1	0.000	0.000	0.000	2	2	2
1	1	2	0.000	0.000	0.000	2	2	2
1	1	3	0.000	0.000	0.000	2	2	2
1	2	1	0.000	0.000	0.000	2	2	2
1	2	2	0.000	0.000	0.000	2	2	2
1	2	3	0.000	0.000	0.000	2	2	2
1	3	1	0.000	0.000	0.000	2	2	2
1	3	2	0.000	0.000	0.000	2	2	2
1	3	3	0.000	0.000	0.000	2	2	2
2	1	1	0.000	0.000	0.681	2	2	1
2	1	2	0.000	0.565	0.706	2	1	1
2	1	3	0.220	0.107	3.294	1	1	1
2	2	1	0.309	0.000	2.222	1	2	1
2	2	2	0.135	0.000	2.652	1	2	0
2	2	3	0.105	0.000	3.809	1	2	0
2	3	1	0.473	0.000	2.062	1	2	1
2	3	2	0.000	0.000	0.227	2	2	1
2	3	3	0.194	0.000	1.507	1	2	0
3	1	1	0.731	3.063	11.406	1	1	0
3	1	2	0.985	0.268	9.803	1	1	0
3	1	3	0.000	1.663	8.195	2	0	0
3	2	1	0.000	2.168	6.922	2	1	0
3	2	2	0.774	0.935	9.397	1	1	0
3	2	3	1.244	0.000	4.583	0	2	0
3	3	1	0.545	0.331	3.328	1	1	0
3	3	2	0.000	1.718	4.323	2	0	0
3	3	3	0.000	1.997	5.729	2	0	0
Average/Total			0.212	0.475	2.994	42	41	24

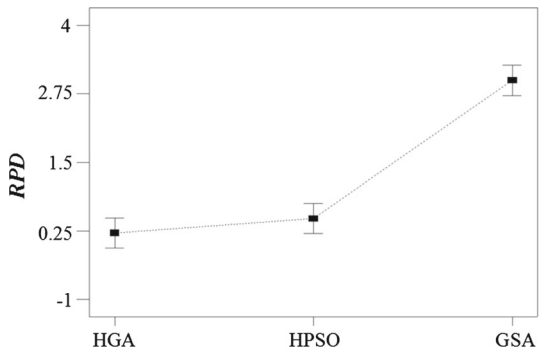

Fig. 8 Means plot with 95 % LSD intervals obtained for 2-machine problems

lower than that obtained by HPSO procedure, such a difference cannot be considered statistically significant, as the LSD intervals of the two algorithms are partially overlapped. On the other hand, the narrow difference of performance between the two algorithms should depend on the poor computational complexity of the instances handled in the proposed benchmark of problems.

Comparison for three-machine problems

As concerns the three-machine problems, a total of 162 separate test cases have been solved by each metaheuristic procedure. Such instances have been generated by combining three levels for the number of groups, three levels for the number of jobs within each group, and nine levels for setup times as reported in Table 3. For each combination, two random replicates have been created.

The average *RPD*s as well as the values of N_{opt} obtained by the three algorithms tested are reported in Table 7.

Table 7 Average *RPD* and N_{opt} values for three-machine problems

Level of factors			Average *RPD*			N_{opt}		
Number of groups	Number of jobs in a group	Setup times of machine M_i	HGA	HPSO	GSA	HGA	HPSO	GSA
1	1	1	0.000	0.000	0.000	2	2	2
1	1	2	0.000	0.000	0.000	2	2	2
1	1	3	0.000	0.000	0.000	2	2	2
1	1	4	0.000	0.000	0.000	2	2	2
1	1	5	0.000	0.000	0.000	2	2	2
1	1	6	0.000	0.000	0.000	2	2	2
1	1	7	0.000	0.000	0.000	2	2	2
1	1	8	0.000	0.000	0.000	2	2	2
1	1	9	0.000	0.000	0.000	2	2	2
1	2	1	0.000	0.000	0.000	2	2	2
1	2	2	0.000	0.000	0.000	2	2	2
1	2	3	0.000	0.766	0.000	2	1	2
1	2	4	0.000	0.000	0.000	2	2	2
1	2	5	0.000	0.000	0.000	2	2	2
1	2	6	0.000	0.000	0.000	2	2	2
1	2	7	0.000	0.095	0.284	2	1	1
1	2	8	0.000	0.000	0.000	2	2	2
1	2	9	0.000	0.000	0.000	2	2	2
1	3	1	0.000	0.195	0.000	2	1	2
1	3	2	0.000	0.209	0.000	2	1	2
1	3	3	0.110	0.000	0.881	1	2	1
1	3	4	0.000	0.000	0.000	2	2	2
1	3	5	0.000	0.422	0.422	2	1	1
1	3	6	0.000	0.000	0.000	2	2	2
1	3	7	0.000	0.115	0.000	2	1	2
1	3	8	0.000	0.000	0.000	2	2	2
1	3	9	0.000	0.405	0.202	2	1	1
2	1	1	0.383	0.000	2.423	1	2	1
2	1	2	0.000	0.000	2.642	2	2	0
2	1	3	0.123	0.000	2.176	1	2	0
2	1	4	0.813	0.000	1.951	1	2	1
2	1	5	0.000	0.000	3.476	2	2	0
2	1	6	0.000	0.000	0.097	2	2	1
2	1	7	0.763	0.000	0.000	1	2	2
2	1	8	0.000	0.000	3.502	2	2	1
2	1	9	0.000	0.000	0.000	2	2	2
2	2	1	0.000	0.000	0.638	2	2	1
2	2	2	0.350	0.133	0.117	1	1	1
2	2	3	0.000	0.321	1.716	2	1	0
2	2	4	0.000	0.126	1.667	2	1	0
2	2	5	0.000	0.000	2.964	2	2	0
2	2	6	0.000	0.000	0.454	2	2	1
2	2	7	0.000	0.224	1.683	2	1	0
2	2	8	0.000	0.624	0.841	2	0	0
2	2	9	0.000	0.362	1.027	2	1	1

Table 7 continued

Level of factors			Average *RPD*			N_{opt}		
Number of groups	Number of jobs in a group	Setup times of machine M_i	HGA	HPSO	GSA	HGA	HPSO	GSA
2	3	1	0.067	0.000	2.787	1	2	0
2	3	2	0.000	0.525	0.700	2	0	0
2	3	3	0.138	0.138	0.000	1	1	2
2	3	4	0.000	0.389	1.495	2	0	0
2	3	5	0.000	0.336	1.585	2	0	0
2	3	6	0.000	0.000	0.810	2	2	0
2	3	7	0.000	0.342	1.534	2	1	0
2	3	8	0.000	0.000	0.000	2	2	2
2	3	9	0.000	0.000	0.047	2	2	1
3	1	1	0.630	0.382	4.758	0	1	1
3	1	2	0.000	2.019	6.762	2	0	0
3	1	3	0.000	1.605	7.289	2	1	0
3	1	4	0.000	0.514	5.272	2	0	0
3	1	5	0.000	2.233	5.369	2	0	0
3	1	6	0.000	1.619	9.086	2	0	0
3	1	7	0.600	0.712	6.417	1	1	0
3	1	8	0.000	1.709	4.923	2	0	0
3	1	9	0.000	1.023	6.037	2	0	0
3	2	1	0.000	0.947	6.541	2	0	0
3	2	2	0.000	0.933	6.214	2	0	0
3	2	3	0.000	2.121	6.002	2	0	0
3	2	4	0.000	2.318	5.979	2	0	0
3	2	5	0.000	2.205	5.559	2	0	0
3	2	6	0.000	2.390	6.656	2	0	0
3	2	7	0.000	1.752	6.213	2	0	0
3	2	8	0.000	0.547	4.249	2	0	0
3	2	9	0.359	1.250	5.497	1	1	0
3	3	1	0.120	0.470	3.822	1	1	0
3	3	2	0.187	0.135	2.046	1	1	0
3	3	3	0.260	0.577	4.772	1	1	0
3	3	4	0.000	0.987	3.229	2	0	0
3	3	5	1.132	0.533	4.220	1	1	0
3	3	6	0.000	0.871	2.069	2	0	0
3	3	7	0.209	0.881	3.660	1	1	0
3	3	8	0.000	0.923	3.407	2	0	0
3	3	9	0.000	1.050	4.025	2	0	0
Average/Total			0.077	0.474	2.200	145	95	68

This time HGA procedure clearly outperforms both of the metaheuristic procedures taken as competitors. The proposed hybrid genetic algorithm by far assures the lowest average *RPD*, also gathering the best makespan value 145 out of 162 times, equal to 89.5 % of the tested instances. The N_{opt} value achieved by HPSO is equal to 95, while GSA finds the best solution only 68 times. Figure 9 reports the means plot with 95 % confidence level LSD intervals, confirming the superiority of the HGA-based approach under a statistical viewpoint.

Comparison for six-machine problems

With regards to six-machine problems, 54 different instances have been generated by combining three levels for each experimental factor, as reported in Table 4, and by testing two

replicates for each problem configuration. Average *RPD*s and N_{opt} values are reported in Table 8.

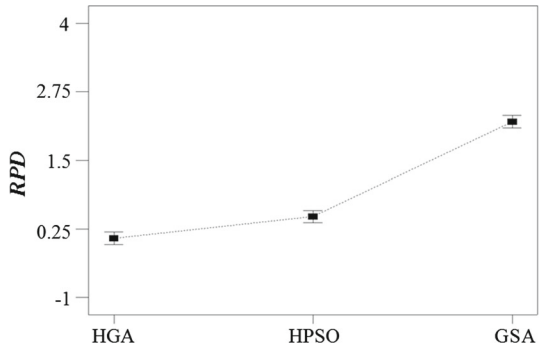

Fig. 9 Means plot with 95 % LSD intervals obtained for 3-machine problems

Again, obtained results emphasize the effectiveness of HGA in tackling six-machine FSDGS problems as well. The developed metaheuristic achieves the best performance both in terms of average *RPD* and total amount of N_{opt}, finding the best solution in 43 out of 54 instances. It outperforms both HPSO and GSA, whose values of N_{opt} are 34 and 20, respectively.

The superiority of performance of the proposed HGA is statistically significant, as shown by the means plot with 95 % LSD interval illustrated in Fig. 10.

Conclusions

In this paper, a properly developed hybrid genetic algorithm integrating features from random sampling search methods (HGA) has been employed for the makespan minimization

Table 8 Average *RPD* and N_{opt} values for six-machine problems

Level of factors			Average *RPD*			N_{opt}		
Number of groups	Number of jobs in a group	Setup times of machine M_i	HGA	HPSO	GSA	HGA	HPSO	GSA
1	1	1	0.000	0.000	0.000	2	2	2
1	1	2	0.000	0.000	0.000	2	2	2
1	1	3	0.000	0.000	0.000	2	2	2
1	2	1	0.000	0.000	0.000	2	2	2
1	2	2	0.000	0.000	0.000	2	2	2
1	2	3	0.000	0.000	0.000	2	2	2
1	3	1	0.000	0.000	0.000	2	2	2
1	3	2	0.000	0.234	1.245	2	1	0
1	3	3	0.039	0.000	0.000	1	2	2
2	1	1	0.000	0.000	0.000	2	2	2
2	1	2	0.431	0.000	0.323	1	2	1
2	1	3	0.103	0.000	0.686	0	2	0
2	2	1	0.000	0.000	0.094	2	2	1
2	2	2	0.000	1.291	3.038	2	0	0
2	2	3	0.000	0.109	0.544	2	0	0
2	3	1	0.000	0.000	0.531	2	2	0
2	3	2	0.000	0.507	0.984	2	0	0
2	3	3	0.000	0.296	0.808	2	0	0
3	1	1	0.135	0.317	1.621	1	1	0
3	1	2	0.312	1.326	8.213	1	1	0
3	1	3	0.000	0.541	1.123	2	0	0
3	2	1	0.050	0.201	1.537	1	1	0
3	2	2	0.608	1.481	4.771	1	1	0
3	2	3	0.085	0.186	1.195	1	1	0
3	3	1	0.058	0.000	1.478	0	2	0
3	3	2	0.000	1.689	1.855	2	0	0
3	3	3	0.000	1.072	1.531	2	0	0
Average/Total			0.067	0.343	1.169	43	34	20

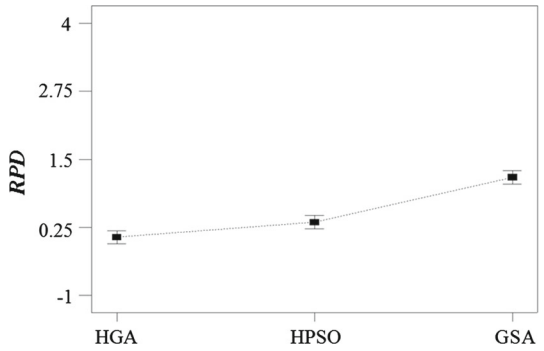

Fig. 10 Means plot with 95 % LSD intervals obtained for 6-machine problems

in a flow shop sequence-dependent group scheduling problem. The proposed technique makes use of a matrix-encoding scheme able to simultaneously define the sequence of groups as well as the sequence of jobs within each group to be processed along the manufacturing system. It combines two distinct crossover operators as well as two mutation techniques and employs a diversity control operator aiming to limit the number of duplicates within each population. Furthermore, the optimization strategy of the proposed procedure is enhanced by a biased-random sampling search scheme, which investigates the neighborhood of the most promising solutions included into each population.

After an extensive calibration phase, the best combination of parameters for the proposed algorithm has been selected. Then, a comparison campaign based on three separate benchmarks arisen from literature involving two-, three- and six-machine problems has been fulfilled in order to test the performance of HGA with respect to the two most recent metaheuristic procedures presented by literature in the field of FSDGS scheduling problems. To this aim, an ANOVA analysis focusing on a statistical validation of the obtained outcomes has been performed. Numerical results highlighted the effectiveness of HGA in approaching the proposed scheduling problem, thus outperforming the two competitors for each benchmark of problems.

Future research should involve the application of the developed HGA to other variants of the flow-shop group scheduling issue, or to hybrid flow-shops manufacturing systems, as well. Another application of the proposed metaheuristics could encompass the FSDGS problems with blocking constraints. Finally, the implementation of alternative local search schemes to be integrated with a metaheuristic algorithm may be tackled.

References

Allahverdi, A., Gupta, J. N. D., & Aldowaisian, T. (1999). A review of scheduling research involving setup considerations. *OMEGA, The International Journal of Management and Science*, *27*(2), 219–239.

Baker, K. R., & Trietsch, D. (2009). *Principles of sequencing and scheduling*. Hoboken, NJ: Wiley.

Baykasoglu, A. (2004). A metaheuristic algorithm to solve quadratic assignment formulations of cell formation problems without presetting numbers of cells. *Journal of Intelligent Manufacturing*, *15*, 753–759.

Celano, G., Costa, A., & Fichera, S. (2010). Constrained scheduling of the inspection activities on semiconductor wafers grouped in families with sequence-dependent set-up times. *The International Journal of Advanced Manufacturing Technology*, *46*(5–8), 695–705.

Cheng, T. C. E., Gupta, J. N. D., & Wang, G. (2000). A review of flowshop scheduling research with setup times. *Production and Operations Management*, *9*(3), 262–282.

Costa, A., Cappadonna, F. A., & Fichera, S. (2013). A dual encoding-based meta-heuristic algorithm for solving a constrained hybrid flow shop scheduling problem. *Computers & Industrial Engineering*, *64*(4), 937–958.

Crepinsek, M., Liu, S. H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, *45*(3), art. 35.

França, P. M., Gupta, J. N. D., 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.

Gallagher, C. C., & Knight, W. A. (1986). *Group technology production methods in manufacturing*. England: Ellis Horwood Limited.

Gao, J., Gen, M., & Sun, L. (2006). Scheduling jobs and maintenance in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, *17*(4), 493–507.

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization (engineering design and automation)*. NY: Wiley.

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.

Ham, I., Hitomi, K., & Yoshida, T. (1985). *Group technology: Applications to production management*. Hingham, MA: Kluwer.

Hendizadeh, H., Faramarzi, H., Mansouri, S. A., Gupta, J. N. D., & ElMekkawy, T. Y. (2008). Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. *International Journal of Production Economics*, *111*(2), 593–605.

Hitomi, K., & Ham, I. (1976). Operations scheduling for group technology applications. *Annals of CIRP*, *25*, 419–422.

Hyer, N. L., & Wemmerloev, U. (1989). Group technology in the US manufacturing industry: A survey of current practices. *International Journal of Production Research*, *27*, 1287–1304.

Kim, K. W., Gen, M., & Yamazaki, G. (2003). Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling. *Applied Soft Computing*, *2*(3), 174–188.

Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, *86*(2), 133–143.

Logendran, R., & Sriskandarajah, C. (1993). Two-machine group scheduling problem with blocking and anticipatory setups. *European Journal of Operational Research, Special Issue on Cellular Manufacturing Systems*, *69*(3), 467–481.

Logendran, R., Mai, L., & Talkington, D. (1995). Combined heuristics for bi-level group scheduling problems. *International Journal of Production Economics*, *38*(2–3), 133–145.

Logendran, R., & Sirikrai, V. (2000). Machine duplication and part subcontracting in the presence of alternative cell locations in manufacturing cell design. *Journal of the Operational Research Society*, *51*, 609–624.

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

Luo, H., Zhang, A., & Huang, G. Q. (2013). Active scheduling for hybrid flowshop with family setup time and inconsistent family formation. *Journal of Intelligent Manufacturing* (in press). doi:10.1007/s10845-013-0771-9

Mahmoodi, F., & Dooley, K. J. (1991). A comparison of exhaustive and non-exhaustive group scheduling heuristics in a manufacturing cell. *International Journal of Production Research*, *29*, 1923–1939.

Marinakis, Y., & Marinaki, M. (2013). Combinatorial neighbourhood topology particle swarm optimization algorithm for the vehicle routing problem. *Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science*, *7832*, 133–144.

Meeran, S., & Morshed, M. S. (2012). A hybrid genetic tabu search algorithm for solving job shop scheduling problems: A case study. *Journal of Intelligent Manufacturing*, *23*(4), 1063–1078.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs* (2nd ed.). Berlin: Springer.

Montgomery, D. C. (2008). *Design and analysis of experiments* (7th ed.). Hoboken, NJ: Wiley.

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

Nawaz, M., Enscore, E. E, Jr, & Ham, I. (1983). A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. *OMEGA, The International Journal of Management and Science*, *11*(1), 91–95.

Paredes, F., Suresh, N. C., & Kay, J. M. (1998). *Group technology and cellular manufacturing: A state-of-the-art synthesis of research and practice*. Boston: Kluwer.

Pinedo, M. L. (2012). *Scheduling: Theory, algorithms and systems* (4th ed.). New York, NY: Springer.

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*, 699–710.

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

Schaller, J. (2001). A new lower bound for the flow shop group scheduling problem. *Computers & Industrial Engineering*, *41*, 151–161.

Shankar, R., & Vrat, P. (1999). Some design issues in cellular manufacturing using the fuzzy programming approach. *International Journal of Production Research*, *37*, 2545–2563.

Soleymanpour, M., Vrat, P., & Shankar, R. (2002). A transiently chaotic neural network approach to the design of cellular manufacturing. *International Journal of Production Research*, *40*, 2225–2244.

Yu, C., Ji, Y., Qi, G., Gu, X., & Tao, L. (2013). Group-based production scheduling for make-to-order production. doi:10.1007/s10845-013-0817-z.

Zandieh, M., & Karimi, N. (2011). An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *22*, 979–989.

Zhu, X., & Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setups: A literature review. *IIE Transactions*, *38*(11), 987–1007.