J Intell Manuf (2011) 22:979–989
DOI 10.1007/s10845-009-0374-7

An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times

M. Zandieh · N. Karimi

Received: 6 September 2008 / Accepted: 28 December 2009 / Published online: 30 January 2010

Abstract In this paper we consider a multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times by minimizing total weighted tardiness and maximum completion time simultaneously. Whereas these kinds of problems are NP-hard, thus we proposed a multi-population genetic algorithm (MPGA) to search Pareto optimal solution for it. This algorithm comprises two stages. First stage applies combined objective of mentioned objectives and second stage uses previous stage's results as an initial solution. In the second stage sub-population will be generated by re-arrangement of solutions of first stage. To evaluate performance of the proposed MPGA, it is compared with two distinguished benchmarks, multi-objective genetic algorithm (MOGA) and non-dominated sorting genetic algorithm II (NSGA-II), in three sizes of test problems: small, medium and large. The computational results show that this algorithm performs better than them.

Keywords Multi-objective optimization · Genetic algorithms · Hybrid flexible flowshops · Sequence-dependent group scheduling

Introduction

Scheduling which is allocating the limited existing resources for task has a significant role in manufacturing systems and its goal is minimizing problems performance measures. Scheduling and producing variety of product may cause complexities in productions process. Since grouping various parts and products based on their similarities in design or process would highly affect problems measure of effectiveness, we use cellular manufacturing concept in our study. Concepts of group scheduling emerged at the beginning of 20th century that caused to reduce setup times by Mitrofanov (1996) and Burbidge (1975). In this approach parts become sorted with respect to their shape, size, material, operations or amount of production Kusiak (1987). This would lead to reduction in setup time, work in process inventories, tools requirement. Specified numbers of part families should be assigned to each cell which consists of group of machines. Liaee and Emmons (1997) reviewed literature of scheduling group of jobs on single or parallel machines. Mansini et al. (2004) studied a scheduling group of task with precedence constraints on three dedicated processors. In order to solve a group scheduling problem by minimizing makespan Baker (1988) presented a goal programming approach. Another heuristic was presented for solving group scheduling problem and a comparison was carried out to show its preference by Logendran et al. (1995). Liu and Yu (1999) proposed group technology approach to minimize number of late job. Allahverdi et al. (1999) and Cheng et al. (2000) presented a cell scheduling problem with sequence-dependent setup time. This technique combines flexibility of job shop with a flowshop production. Each group's jobs should be sequenced at first, and then sequence of groups should be determined. For changing part from one group to another group, set-up time must be considered. But for changing between parts in a group no set-up time should be considered because setup time in each group is negligible and this is because of similarities between group parts which are an important difference between group scheduling problem and flowshop scheduling problem.

M. Zandieh (✉)
Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G. C., Tehran, Iran
e-mail: m_zandieh@sbu.ac.ir; mostafazzz@yahoo.com

N. Karimi
Faculty of Industrial and Mechanical Engineering, Qazvin Islamic Azad University, Qazvin, Iran

The first method in optimizing a two-machine sequence-independent group scheduling problem was presented by Ham et al. (1985). A two-machine flowshop group scheduling was presented by Baker (1990) and Sekiguchi (1983), in which each group has a setup time. Yang and Chern (2000) generalized Baker and Sekiguchi's algorithm by another polynomial time algorithm. Kuo and Yang (2006) proposed two polynomial time algorithms for a time-dependent learning effect and introduce it into the single-machine group scheduling problems. Polynomial time algorithms based on solving linear programming problems are presented to find an optimal job sequence and resource values in single machine group scheduling with resource dependent setup and processing times by Janiak et al. (2005).

Schaller et al. (2000) and Reddy and Narendran (2003) presented heuristics for solving the sequence-dependent flowshop group scheduling problem by considering different situation like non-availability of all jobs at the beginning. Leu and Nazemetz (1995) analyzed different heuristic on group technology in flowshop manufacturing problems. A heuristic for the case of multiple setups per group is developed and compared to a single setup per group at each stage, and integrated into a genetic algorithm for scheduling non-similar groups on a flow line by Wilson et al. (2004).

Flowshop problems schedule number of jobs which should be processed on stages to make problems performance measure optimized. If some of these stages (at least one) consist of multiple identical machines, problem would change to hybrid flexible flowshop in which jobs should be processed on at most one machine in each stage. In these kinds of problem, jobs cannot come back to any of stages. One of the most important features of hybrid flexible flowshop is the jobs capability to skip from stages which means that job do not need to be processed at that stage (Naderi et al. 2008). Vickson and Alfredsson (1992) proposed Johnson's rule for two-machine flowshop group scheduling problem and Cetinkaya and Kayaligil (1992) continued and developed their method by considering setup time. Group scheduling in hybrid flexible flowshops was presented by Logendran et al. (2005). Group scheduling within the context of sequence-dependent setup times in hybrid flexible flowshops is considered in Logendran et al. (2006b) paper. They presented a heuristic to solve it.

Two evolutionary algorithms, a genetic algorithm and a memetic algorithm with local search, are proposed and empirically evaluated for scheduling a flowshop manufacturing cell with sequence-dependent family setups by Franca et al. (2005). Logendran et al. (2006b) focused on two-machine group scheduling problems with sequence-dependent setups by using tabu search algorithm. Zandieh et al. (2008) proposed two new meta-heuristics based on GA and SA to minimize makespan in group scheduling problems with sequence-dependent setup times in hybrid flexible flow shops. They showed these new meta-heuristics outperformed the TS of Logendran et al. (2006b). Behnamian et al. (2009) suggested a hybrid meta-heuristic algorithm which combines particle swarm optimization (PSO), SA, and variable neighborhood search (VNS) in a population-based context. The hybrid method outperformed the GA and SA of Zandieh et al. (2008).

The majority of papers on these problems have concentrated on single objective or criterion problems, while consideration of multiple objectives or criteria is more realistic. Danneberg et al. (1999) proposed a permutation flowshop scheduling problem with setup times by grouping similar jobs together, but there is a limitation for number of jobs in each group. They considered makespan and weighted sum of completion time as objectives.

In this paper, we deal with a multi-objective hybrid flexible flowshop group scheduling with sequence-dependent setup times problem. We also present a mathematical model for this problem. The makespan and total weighted tardiness are to be optimized simultaneously. Because the simpler version of this study (for example, researches of Gupta and Darrow (1985, 1986), the two machine sequence-dependent makespan minimization job scheduling problem) are proved to be NP-hard, so our problem is NP-hard too and we have to use meta-heuristic to solve it. To solve this problem we apply a two stage multi-population genetic algorithm (MPGA) for searching Pareto optimal solutions and also develop and adapt two different meta-heuristics and compare them. The proposed algorithm was presented by Cochran et al. (2003) for parallel machine but we here investigate it for group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times. The remainder of the paper is organized as follows: "Problem definitions" gives the problem definitions. Definition of multi-objective optimization problems is presented in "Multi-objective optimization". "Adaptive genetic algorithm" presents adaptive multi-population genetic algorithm. "Experimental design" is developed to our numerical results. Finally in the last section some conclusions are given.

Problem definitions

In this paper, we consider a hybrid flexible flowshop group scheduling problem. This problem investigates the sequence of jobs in their group and sequence of these groups themselves. Skipping is a feature that gives a flexibility characteristic to our problem. Skipping is a probability of not being processed at some stages. Because of problems flexibility characteristic there would be more than one machine in some stages. So jobs of different groups may be processed in same stage at the same time. For groups of jobs only a setup time is required and this setup times are sequence-dependent. Following assumptions are considered for this problem:

- Jobs are available at zero time
- Job processing cannot be interrupted
- Machines are always available, with no breakdowns or scheduled or unscheduled maintenance
- Infinite buffer exist between stages and before the first and after the last stage
- Jobs are available for processing at a stage immediately after processing completion at the previous stage
- Machines in parallel are identical in capability and processing rate
- Each machine can process only one job at the same time
- The ready time for each job is larger than 0 and the time it completes processing on the previous stage
- A job cannot process on more than one machine at the same time.

Multi-objective optimization

In single objective optimization problem, algorithm's termination depends on optimizing the objective function. But in multi-objective problem it is a complex task to optimize some objectives simultaneously. Thus we have to find a set of solutions depending on non-dominance criterion.

A general multi-objective optimization (minimization) problem can be defined as follows:

Find a vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$minimizes the vector function $f(x) = [f_1(x), f_2(x), \ldots, f_k(x)]$, where $x = (x_1, x_2, \ldots, x_n)$ is the vector of decision variables.

Also a general definition of dominance solution is as follows:

A vector $u = (u_1, u_2, \ldots, u_k)$ is said to dominate $v = (v_1, v_2, \ldots, v_k)$ if and only if u is partially less than v, i.e.,

$$\forall i \in \{1, 2, \ldots, k\}, u_i \le v_i \wedge \exists i \in \{1, 2, \ldots, k\}, u_i < v_i$$

Different approaches have been used to solve multi-objective problems which can be classified as below:

- Weighting approach
- Hierarchical approach
- Goal programming
- Pareto approach
- Interactive approach
- Fuzzy method

We use the concept of Pareto approach via a meta-heuristic method. Different methods to solve a multi-objective problem through a meta-heuristic procedure have been carried out, for example a parallel search model based on the Pareto archived evolution strategy (PAES) which is presented by Knowles and Corne (2000), MOGA that is proposed by Fonseca and Fleming (1993) and an integrated multi-objective immune algorithm (MOIA) is offered by Hou et al. (2008). Murata et al. (1996), Zitzler et al. (2001) offered the strength Pareto evolutionary algorithm (SPEA2) and multi-objective scatter search (MOSS) was used by Beausoleil (2006) and Deb et al. (2002) presented a non-dominated sorting genetic algorithm II (NSGA-II).

Objectives functions

Minimizing makespan is the first objective that we consider for this problem and is as:

$C_{\max} = \max\{C_j\}$, where C_j is the completion time of job j.

The second objective is to minimize total weighted tardiness as below:

$$\mathrm{TWT} = \sum_{j=1}^{P} w_j T_j$$

where $T_j = \max\{0, C_j - d_j\}$, d_j is the due dates of job j, w_j is a weight related to job j.

Mathematical model

In this section we present a formulation of our problem using Kurz and Askin (2004) concepts.

If we assume n as the number of jobs, at each stage we should consider set up time for job 0 to job $n+1$. Definitions of required variable are as below:

n:	number of true jobs to be scheduled
g:	number of serial stages
n_G:	number of groups
g_j:	last stage visited by job j
m^t:	number of machines at stage t
p_{ij}^t:	processing time for job i in group j at stage t (assumed to be integral)
P_j^t:	processing time for jobs in group j at stage t
d_i:	due date of job i
s_{pj}^t:	setup time for group j on stage t, if group j is processed immediately after group p
S_i:	set of stages visited by group i
e_i:	set of stages visited by job i
S^t:	set of group that visit stage $t = \left\{i : p_i^t > 0\right\}$
c_i^t:	completion time for job i at stage t
CG_i^t:	completion time for group i at stage t
W_j:	weight of job j
x_{ij}^t:	1 if job i is scheduled immediately before job j at stage t and 0 otherwise
y_{ij}^t:	1 if group i is scheduled immediately before group j at stage t and 0 otherwise
U_j:	A Boolean variable; 1 if job j is tardy, and 0 otherwise

The time for moving from point to point is named as setup time. For jobs 0 and $n+1$ processing times is consider as zero. Processing of job 0 would be terminated at each stage as soon as the beginning of set up at that stage.

This inequality $\left|S^t\right| \geq m^t, t = 1, 2, \ldots g$, so $n \geq \max_t\{m^t\}$ show the limitation on number of jobs in each stage with respect to the number of machine in it.

$$P: \min\ Z_1, Z_2 \quad s.t. \tag{1}$$

$$\sum_{j=1}^{n} x_{oj}^t = m^t \quad t = 1, \ldots, g, \tag{2}$$

$$\sum_{j\in\{s^t, n+1\}} y_{ij}^t = 1, \quad i = 1, \ldots, n_G,\ t \in S_i \tag{3}$$

$$\sum_{j\in\{0, s^t\}} y_{ij}^t = 1, \quad j = 1, \ldots, n_G,\ t \in S_i \tag{4}$$

$$c_j^t - c_i^t + M^t(1 - x_{ij}^t) \geq p_{ij}^t \quad i = o, \ldots, n, j = 1, \ldots, n, t \in e_i \tag{5}$$

$$c_j^t - c_i^{t-1} + M_j^t(1 - x_{ij}^t) \geq p_{ij}^t \quad i = o, \ldots, n, j = 1, \ldots, n, t \in e_i - \{1\} \tag{6}$$

$$CG_j^t - CG_i^t + M^t(1 - y_{ij}^t) \geq s_{ij}^t + P_j^t \quad i = 0, \ldots, n_G, j = 1, \ldots, n_G, t \in S_i \tag{7}$$

$$CG_j^t - CG_j^{t-1} + M_{Gj}^t(1 - y_{ij}^t) \geq s_{ij}^t + P_j^t \quad i = 0, \ldots, n_G, j = 1, \ldots, n_G, t \in S_i - \{1\} \tag{8}$$

$$\begin{cases} x_{ij}^t \leq p_{ij}^t \\ x_{ji}^t \leq p_{ij}^t \end{cases} \quad i, j \in \{0, 1, \ldots, n, n+1\}, t = 1, \ldots, g, \tag{9}$$

$$c_j^t \geq c_o^t \quad j = 1, \ldots, n, t = 1, \ldots, g \tag{10}$$

$$Z_1 \geq c_j^{gj} \quad j = 1, \ldots, n, \tag{11}$$

$$T_j \geq c_j^{gj} - d_j \quad j = 1, \ldots, n, \tag{12}$$

$$T_j \geq 0 \quad j = 1, \ldots, n, \tag{13}$$

$$MU_j \geq T_j \quad j = 1, \ldots n, \tag{14}$$

$$Z_2 = \sum_{j=1}^{n} W_j T_j \tag{15}$$

$$U_i \in \{0, 1\}, \quad i = 1, \ldots, n, \tag{16}$$

$$x_{ij}^t \in \{0, 1\}, \quad i, j \in \{0, 1, \ldots, n, n+1\}, \quad t = 1, \ldots, g, \qquad x_{ij}^t = 0, \quad i = j, t = 1, \ldots, g, \tag{17}$$

$$y_{ij}^t \in \{0, 1\}, \quad i, j \in \{0, 1, \ldots, n_G, n_G+1\}, \quad t = 1, \ldots, g, \qquad y_{ij}^t = 0, \quad i = j, \quad t = 1, \ldots, g, \tag{18}$$

$$c_j^t \geq 0, \quad j = 0, \ldots, n, \quad t = 1, \ldots, g, \tag{19}$$

Objective functions, makespan (Z_1) and total weighted tardiness (Z_2), are presented in (1). Constraint set (2) shows that m^t machines are scheduled in each stage. We ensure the scheduling of each group on one and only one machine in each stage by constraint sets (3) and (4). By constraint (5) job j follow job i by at least i's processing time. An upper bound for processing completion time at stage t is presented by M^t. Constraint set (6) is required to show the processing completion of job j at stage t after completion of its process at stage $t-1$, plus its processing time at stage t. M_j^t is calculated through following equality $M_j^t = P_j^t$. Constraint sets (5) and (6) together prevent from beginning setup for a job until its availability (being done at the previous stage) and previous job's completion at the current stage.

Constraint set (7) ensures that group j come after group i by at least i's total group processing time plus the setup time from i to j if i is immediately before j. Group j at stage t should be completed after its completion at stage $t-1$ plus its processing time at stage t, plus the setup time from its predecessor to j because of constraint set (8).The value M_{Gj}^t is equal to $p_j^t + \max_i\{s_{ij}^t\}$. The same as Constraints (5) and (6) for jobs, constraints (7) and (8) together ensure that a group cannot begin setup until it is available (done at the previous stage) and the previous group at the current stage is complete. Constraint sets (5), (6), (7) and (8) also used for prohibition of creating any sub-tour.

Constraint set (9) prevents jobs from visiting a stage that they are not assigned to that stage. Constraint sets (10) and (11) state that a job at stage does not visit stage t, its completion time at this stage would set to its completion time at stage $t-1$. Constraint (11) is for creating the decision variables c_j^{gj} and Z_1.

The lateness (L_j) value can be calculated by constraint set (12). Constraint set (13) specifies only the positive lateness as the tardiness $T_j = \max\left\{0, c_j^{gj} - d_j\right\}$. Constraint sets (14)–(16) link the decision variable of the number of tardy jobs; that is, if the tardiness is larger than zero, then the job is tardy; otherwise, this job is not tardy. The value of M is set to a very large constant, i.e., greater than the sum of all job processing times and setup times. Constraints (17), (18) and (19) provide limits on the decision variables.

Adaptive genetic algorithm

Genetic algorithm in general

Genetic algorithm is one of the widely used stochastic search method in order to solve an optimization problem, which is based on Darwinian Theory. This algorithm simulates the natural mechanism of survival of the fittest tenet and offspring creation. Each solution (chromosomes) is evaluated with a fitness function that is appropriate for the problem at hand.

Some papers have applied the genetic algorithm heuristic to multiple objective problems. Schaffer (1985) developed the vector evaluated genetic algorithm (VEGA) method for finding Pareto optimal solutions of multiple objective optimization problems. In his approach, population would be divided into disjoint sub-populations, end each sub-population find its objective's optimal solution.

Cavalieri and Gaiardelli (1988) also presented hybrid genetic algorithm for a multiple-objective scheduling problem. Murata et al. (1996) proposed a MOGA and applied it to flowshop scheduling. Hyun et al. (1998) developed a new selection scheme in GA, and showed its superiority for multi-objective scheduling problems in assembly lines.

Adaptive multi-population genetic algorithm

In this paper we use the approach of a two-stage multi-population genetic algorithm which was presented by Cochran et al. (2003) First stage of the algorithm work by minimizing combining of objective functions. After rearrangement of solution of this stage, second stage start by using these solution as its initial solutions in different population separately.

First stage

Step 1. *Initialization*: in order to start the algorithm some of the algorithm's parameters like initial population, size of each sub-populations, and number of iteration in each stage and rate of crossover should be set.
Then initial population should be generated. Each job is assigned a random real number whose integer part is the machine number to which the job is allocated and whose fractional part is used to sort the jobs assigned to each machine. So we use this concept in order to allocate group to machines. Because of considering one setup time for each group, jobs assignment is the same as group assignment. For the first stage, we should produce real number for each group between 1 and number of machines plus 1. To define sequence of jobs in each group random numbers should be produced and the job with smaller number would be processed earlier. In other stages groups sequence would be defined based on their completion time.
We choose the matrix representation form to show our solution, in which rows represent groups and each digit in each row is that group's job.

Step 2. *Evaluation*: In this phase solution should be evaluated through evaluation metrics. The most common evaluation metric to assess and compare solutions is the relative percentage deviation (RPD) which is as below (Kim 1993; Kim et al. 1996):

$$\text{RPD} = \frac{A\,\text{lg}_{\text{Sol}} - \text{Best}_{\text{Sol}}}{\text{Best}_{\text{Sol}}} \times 100$$

where $A\,\text{lg}_{\text{Sol}}$ is the solution obtained by a given method and Best_{Sol} is the best solution obtained among all the methods or the best known solution, possibly optimal. If the objective function is tardiness, the optimal solution may become zero and this equation would become divided by zero. Thus we have to another measure for calculating performance of algorithms. Relative deviation index (RDI) is proposed for this reason (Kim 1993; Kim et al. 1996):

$$\text{RDI} = \frac{A\,\text{lg}_{\text{Sol}} - \text{Best}_{\text{Sol}}}{\text{Worst}_{\text{Sol}} - \text{Best}_{\text{Sol}}} \times 100$$

Best_{Sol} and $\text{Worst}_{\text{Sol}}$ are the best and worst evaluated solution.
We present and examine two approaches of combining evaluation metrics in this study.
If we assume that N objectives are to be optimized, combined objective values can be formulated as:
The first method is summation of measures with random weights:

$$f_{t,j}(x) = \sum_{i=1}^{N} w_{t,i} f_{t,i,j}(x); \quad i = 1, \ldots, N, \; j = 1, 2, \ldots, P$$

Second way of combining evaluation metrics that is used here is multiplication of them:

$$f_{t,j}(x) = \prod_{i=1}^{N} f_{t,i,j}(x); \quad i = 1, \ldots, N, \; j = 1, 2, \ldots, P$$

where P is the population size, $f_{t,j}(x)$ is the combined function at generation t of string j, $w_{t,i}$ is the weight of evaluation metrics i at generation t, and

$f_{t,i,j}(x)$ is the ith evaluation metrics of string j at generation t. Note that $w_{t,i}$ is a predefined measure.

Step 3. *Elitist strategy*: The best solution of each objective and the best one of the combined objective function are stored in each generation. At the end of each generation, if the selected solution is worse than the preserved one, a randomly selected string is replaced with the preserved solution.

Step 4. *Selection*: Selection is an operation that selects two chromosomes as parents for crossover and mutation operators. We apply roulette wheel strategy that uses each chromosome's probability for selection. Each has a better fitness value would have greater probability to be selected. The following formula is used to calculate probability:

$$p_{t,j}(x) = \frac{S - f_{t,j}(x)}{\sum_{j=1}^{P}(S - f_{t,j}(x))}$$

where $f_{t,j}(x)$ is the value of objective function for jth string in tth replication and S is the summation of $f_{t,j}(x)$ through population size.

Step 5. *Crossover*: Crossover is an operator which consists of obtaining new individuals from two parent chromosomes with the aim of producing better chromosome. These new individuals which called offspring have features of both parents and can be either better or worse than them. For this study we perform uniform crossover. In order to do that two chromosomes in the current generation are selected at random as a group and one job is chosen at random in each of them as a job. For each job, a random number is generated. If the value is <0.7, the value from the "first" chromosome is copied to the new chromosome, otherwise the value from the "second" chromosome is selected (Bean 1994) (Fig. 1).

Step 6. *Mutation*: Mutation operator produces random changes in a chromosome and introduce new chromosome and is generally increase diversification of population. Here random mutation operator is used. A digit is selected according to pre-defined mutation probabilities and replaced with a different number.

Step 7. *Turning criterion*: After a predefined number of generations, the algorithm switches to the next stage.

(a)

Parent 1				
	2.23	*3.1*	*2.5*	*1.6*
	1.23	***2.03***	***1.45***	***2.89***
	1.7	*2.9*	*1.78*	*2.23*
	3.4	*1.44*	*1.83*	*2.44*

Parent 2				
	2.36	*3.1*	*1.48*	*2.81*
	3.79	*1.54*	*2.31*	*1.19*
	1.28	*2.36*	*3.14*	*2.58*
	2.15	***2.45***	***1.85***	***1.03***

(b)

Random numbers				
	0.45	*0.83*	*0.68*	*0.75*

(c)

Offspring				
	2.23	*3.1*	*2.5*	*1.6*
	1.23	***2.45***	***1.45***	***1.03***
	1.7	*2.9*	*1.78*	*2.23*
	3.4	*1.44*	*1.83*	*2.44*

Fig. 1 Uniform crossover example

Second stage

Step 1. *Initialization*: Rearrangement should be carried out on solutions of previous stage due to their performance. If N objectives should be optimized, $N + 1$ sub-populations will be created after rearrangement to evolve separately using N mentioned objective functions and combined objective.

Step 2. *Selection, crossover, and mutation*: The same selection, crossover, and mutation procedures used in stage 1 are applied in each sub-population.

Step 3. *Elitist strategy*: The elitist strategy should select the best solution of each objective and combined objective across all sub-populations. Selected chromosomes should be stored and replace the worst of each objective and the combined objective in the next generation.

Step 4. *Stopping criterion*: After a certain number of generations, the algorithm terminates.

Adaptive Pareto archive set

Pareto optimal solution can be the solution to multi-objective problems instead of a single solution and provide decision maker sufficient insight into the problem to make the final decision. A Pareto optimal solution is such a solution that dominates other ones, in the other words they are not worse with respect to every objective, and better for at least one objective. The archive is used to store and maintain some of the non-dominant solution produced in multi-objective evolutionary algorithm. Generally it has a predetermined size. In this study, we use an adaptive Pareto archive set updating method to prevent losing new non-dominated solutions found when Pareto archive size has reached its maximum size (Tavakkoli-Moghaddam et al. 2008). Pareto archive is updated by one of following strategy at the end of each iteration when new solution is found:

– If the archive size is less than its maximum value, the solution will be added to the archive.

– If the archive size is less than its maximum value, the new solution will be added to archive if its distance to the nearest non-dominant solution of archive is greater than “Duplication area”. This area is defined as a bowl of center of the solution with the specified radius of λ. This area is used to maintain the diversity of solutions. The mentioned distance value would be calculated in the Euclidean distance form.

When the new solution is added to the archive it would be checked that if the new solution dominate some members of the archive, so those members should be deleted from the archive.

Experimental design

The comparison between performance of the proposed multi-population genetic algorithm in solving multi-objective problem and MOGA which has a very good performance for solving flowshop scheduling problems and NSGA-II which is professional in solving multi objective problems is presented in this section. These algorithms have been coded in the Borland C++ 5.2.

Data generation

The experiments are implemented on three size of problem: small, medium and large. For all of them we consider following assumptions:

Processing times are from a uniform distribution of $U(5, 75)$; setup times are uniformly generated in the interval $(5, 25)$. The due dates are uniformly distributed over the interval $[\bar{d} - R\bar{d}, \bar{d}]$ with probability τ and over the interval $[\bar{d}, \bar{d} + (C_{\max} - \bar{d})R]$ with probability $(1 - \tau)$, and $\tau = 1 - \frac{\bar{d}}{C_{\max}}$. The values of τ and R are set to 0.6 and 0.4 respectively, jobs weights are uniformly distributed integers in the interval $[1, 4]$ and each experiment is repeated 10 times. As mentioned above, skipping probability is considered in this problem. Uniformly distributed random number is generated between 0 and 1 for each job. If this number is less than one minus probability value, job has positive processing times, otherwise job’s processing times is zero.

Because of this problem flexible characteristic, we should define the measure of flexibility for it. These values are presented in Table 1. Number of stages and groups are also shown in Table 1. By multiplying the number of stages and measure of flexibility we can calculate number of stages which has parallel machines if this is a real number we should round it to greater number (n). To know which stages have more than one machine, we produce random permutation of number of stage, then first n numbers are stages that have parallel machine. These stages have 2 or 3 machines by probability value equal to 0.5. Numbers of jobs for each group are: in small problems between 3 and 5, in medium between 7 and 9 and in large between 10 and 12.

Table 1 Characteristics of test problem

Factors	Level	Description
Number of stages	Small	2–3
	Medium	5–6
	Large	7–8
Number of groups	Small	4–5
	Medium	7–9
	Large	11–12
Flexibility	Small	1/3
	Medium	2/3
	Large	1
Skipping probability		0.2

We also carry out tests for both of defined combined objectives and for this reason examine 3 different weights for first combining objective. These weights are: (0.2, 0.8), (0.5, 0.5) and (0.8, 0.2).

Archive size is set to 30 and the value of λ for Pareto duplication area is considered 1,000.

Performance measure

In order to calculate the performance of algorithm and do comparisons we use performance measures presented by Hyun et al. (1998). These measures are qualitative and quantitative measures. If N_1 and N_2 are Pareto optimal solutions of algorithms A_1 and A_2 respectively, combined Pareto front have N Pareto optimal solutions which is greater than N_1 and N_2 and less that $N_1 + N_2$.

Quantitative measure of each algorithm is its number of Pareto optimal solutions (N_1 and N_2 respectively) and qualitative measures of algorithms A_1 and A_2 are represent as N_1/N and N_2/N respectively.

MPGA parameters tuning

There are several different factors which affect the algorithm in reaching to a suitable and desirable solution and parameters value is one of them. Whereas different parameters combinations would lead to different solutions, we try to find the most appropriate set of parameters for MPGA.

To achieve this goal firstly we carry out extensive experiments in order to determine effective parameters using design of experiments (DOE). Then an empirical study of various possible combinations of parameters will be done to define the best levels of those parameters (Table 2).

Table 2 Level of genetic algorithm parameters

Problem size	Popsize		Iteration 1		Iteration 2		Crossover	
	Lower level	Upper level	Lower level	Upper level	Lower level	Upper level	Lower level	Upper level
Small	25	80	10	30	20	50	0.60	0.85
Medium	50	200	30	100	30	100	0.60	0.85
Large	100	400	100	500	50	300	0.60	0.85

Using this approach we recognize population size (popsize), number of iteration of both stage (iteration 1 and iteration 2) and crossover rate (crossover) as effective factors. During experiments it is achieved that the suitable size of each sub-population) is about 0.4% of popsize value. Ranges of these parameters are considered as follows:

Now here we perform response surface methodology (RSM) approach to find the suitable value of each parameter in each size of problem. RSM is a technique for determining and representing the cause-and-effect relationship between true mean responses and input control variables influencing the responses as a two- or three-dimensional hyper surface (Gunaraj and Murugan 1999).

In order to simplify the procedure we use SAS software for finding the relation between outputs (objective functions) and effective factors on outputs (popsize, iteration 1, iteration 2 and crossover) and apply a standard model (fractional central composite design) in it. RSM give us models as follow respectively for each size of problem, moreover, it adjust the levels of the input variables to the point that the set of outputs optimize using these models. Results are shown in Table 3.

$$\begin{aligned}
Y1 = {} & 0.035386 + 0.0064 \times X1 - 0.000476 \times X2 \\
& - 0.004964 \times X3 - 0.00184 \times X4 \\
& + 0.007844 \times X1 \times X2 \times X3 \\
& + 0.00912 \times X1 \times X2 \times X4 \\
Y2 = {} & 0.02037 - 0.00768 \times X1 \\
& + 0.00212 \times X2 - 0.00172 \times X3 \\
& - 0.00384 \times X4 + 0.00279 \times X1 \times X4 \\
& - 0.003227 \times X2 \times X3 \\
& - 0.00213 \times X2 \times X4 \\
& + 0.002164 \times X1 \times X2 \times X3 \\
Y3 = {} & 29.7986 - 3.5347 \times X1 \\
& + 0.854167 \times X2 - 0.78472 \times X3 \\
& + 5.4236 \times X4 + 1.7986 \times X1 \times X2 \\
& + 4.479167 \times X1 \times X4 + 5.6736 \times X3 \times X4
\end{aligned}$$

Table 3 Parameter setting results

Parameter	Small	Medium	Large
Population size (popsize)	65	185	300
Iteration for first stage (iteration 1)	15	40	400
Iteration for second stage (iteration 2)	45	50	70
Crossover rate (crossover)	0.75	0.7	0.8

Experimental results

Several instances of different combination of data which are presented before were solved for doing comparison between this MPGA and benchmark MOGA and NSGA-II. Each instance was run 10 times. These experiments contain tests with both combined objectives, and all mentioned weights for first objective. These experimental results show that first combined objective by all weights value is better than second one, and set of (0.5, 0.5) is the best weight among others.

For each problem size Table 4 shows results which are achieved via our considered measures. Each digit is average of 10 run of each problem with its respective method. Win column show the number of preference of MPGA than MOGA and NSGA-II in iterations for each test problem.

As shown in Table 4 in most tests MPGA outperforms MOGA and NSGA-II qualitatively and quantitatively and it works better as the size of problem get larger.

Better performance of MPGA than MOGA and NSGA-II both in number of solutions in Pareto front and value of solutions can be illustrated by figures for different sizes of problem. Figures 2, 3 and 4 show the improvement of solutions through iterations in small, medium and large problem respectively.

Conclusion

This paper has presented an adaptive multi-population genetic algorithm (MPGA) for solving a hybrid flexible flowshop group scheduling with sequence-dependent setup times problem with respect to the total weighted tardiness and maximum completion time simultaneously.

Table 4 Comparison of MPGA versus MOGA and NSGA-II

Problem	Size	Stage	Group	Quantitative				Qualitative			
				MPGA	MOGA	NSGA-II	WIN	MPGA	MOGA	NSGA-II	WIN
1	SMALL	N	N	6.00	4.20	5.3	5	0.52	0.18	0.3	8
2	SMALL	N	H	5.10	2.80	3.2	10	0.41	0.21	0.38	10
3	SMALL	H	N	7.10	3.20	4	3	0.4	0.27	0.33	9
4	SMALL	H	H	6.00	2.60	4.1	10	0.39	0.27	0.34	10
5	MEDIUM	N	N	13.60	3.00	2.5	10	0.64	0.17	0.19	10
6	MEDIUM	N	M	10.33	5.00	7.4	8.0	0.62	0.18	0.2	9.5
7	MEDIUM	N	H	18.00	2.10	6.8	10	0.58	0.17	0.25	10
8	MEDIUM	H	N	11.70	4.10	4.5	10	0.55	0.15	0.3	10
9	MEDIUM	H	M	12.56	5.30	8.9	9.0	0.7	0.11	0.19	9
10	MEDIUM	H	H	14.20	3.30	6.8	10	0.67	0.12	0.21	8.5
11	LARGE	N	N	14.40	3.10	5.5	10	0.59	0.17	0.24	9
12	LARGE	N	H	4.40	2.40	3.9	9.0	0.47	0.22	0.31	9
13	LARGE	H	N	12.10	3.10	4.6	10	0.39	0.29	0.32	9.5
14	LARGE	H	H	4.60	2.30	3.7	8.5	0.51	0.22	0.27	10.0

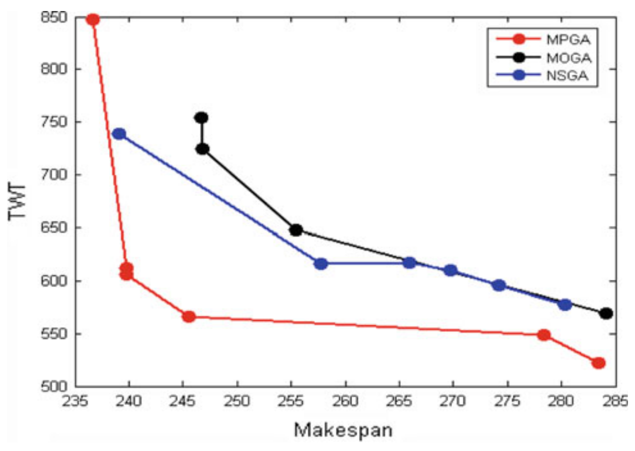

Fig. 2 Pareto front solutions for small problem

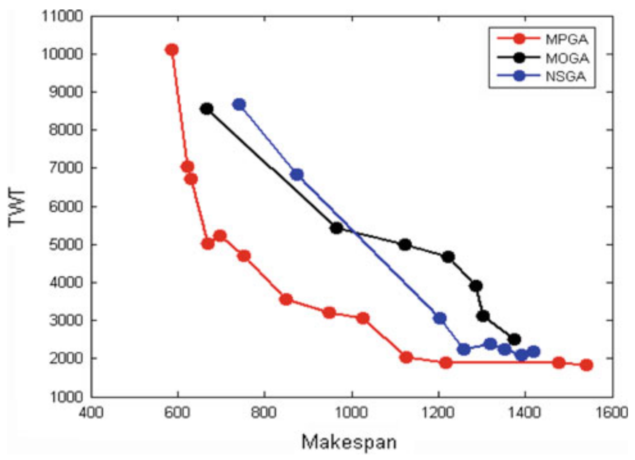

Fig. 3 Pareto front solutions for medium problem

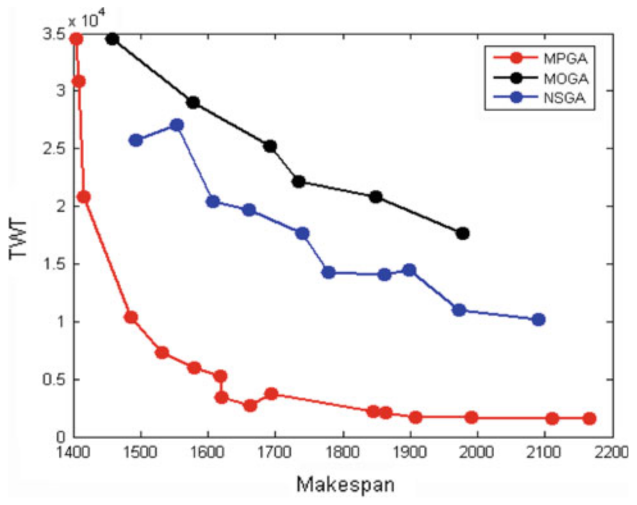

Fig. 4 Pareto front solutions for large problem

In order to evaluate performance of proposed algorithm we used extensive test problems and carried out comparison with two benchmark problems MOGA and NSGA-II. Comparisons were based on two performance metrics, qualitative and quantitative measures. This combination of two heuristic (VEGA and MOGA) has the better performance of both of them in each three sizes, especially in large problems according to above figures.

Maintaining the best solution in each iteration is one of our aims and we use Pareto optimal solution for this end. Using Pareto archive we can avoid losing best solution through iterations and early convergence of algorithm, in addition we can maintain diversity of solution space.

References

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

Baker, K. R. (1988). Scheduling the production of components at a common facility. *IIE Transactions, 20*, 32–35.

Baker, K. R. (1990). Scheduling groups of jobs in the two-machine flow shop. *Mathematical and Computer Modeling, 13*, 29–36.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing, 6*, 154–160.

Beausoleil, R. P. (2006). "MOSS" multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research, 169*, 426–449.

Behnamian, J., Zandieh, M., & Fatemi Ghomi, S. M. T. (2009). Due windows group scheduling using an effective hybrid optimization approach. *International Journal of Advanced Manufacturing Technology, 44*(7), 795–808.

Burbidge, J. L. (1975). *The introduction of group technology*. London: Heinemann.

Cavalieri, S., & Gaiardelli, P. (1998). Hybrid genetic algorithms for a multiple-objective scheduling problem. *Journal of Intelligent Manufacturing, 9*, 361–367.

Cetinkaya, F. C., & Kayaligil, M. S. (1992). Unit sized transfer batch scheduling with setup times. *Computers and Industrial Engineering, 22*, 177–183.

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*, 283–302.

Cochran, J. K., Horng, S. M., & Fowler, J. W. (2003). A multipopulation genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers and Operations Research, 30*, 1087–1102.

Danneberg, D., Tautenhahn, T., & Werner, F. (1999). A comparison of heuristic algorithms for flow shop scheduling problems with setup times and limited batch size. *Mathematical and Computer Modelling, 29*(9), 101–126.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest (Ed.), *Proceedings of the fifth international conference on genetic algorithms* (pp. 416–423). Morgan Kaufman Publishers San Mateo, California University of Illinois at Urbana-Champaign.

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

Gunaraj, V., & Murugan, N. (1999). Application of response surface methodology for predicting weld bead quality in submerged arc welding of pipes. *Journal of Materials Processing Technology, 88*, 266–275.

Gupta, J. N. D., & Darrow, W. P. (1985). Approximate schedules for the two-machine flowshop with sequence dependent setup times. *Indian Journal of Management and Systems, 1*, 6–11.

Gupta, J. N. D., & Darrow, W. P. (1986). The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research, 24*, 439–446.

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

Hou, T. H., Su, C. H., & Chang, H. Z. (2008). An integrated multiobjective immune algorithm for optimizing the wire bonding process of integrated circuits. *Journal of Intelligent Manufacturing, 19*, 361–374.

Hyun, C. J., Kim, Y., & Kin, Y. K. (1998). A genetic algorithm for multiple objective sequencing problems in mixedmodel assembly. *Computers & Operations Research, 25*, 675–690.

Janiak, A., Kovalyov, M. Y., & Portmann, M. C. (2005). Single machine group scheduling with resource dependent setup and processing times. *European Journal of Operational Research, 162*, 112–121.

Kim, Y. D. (1993). Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of Operational Research Society, 44*, 19–28.

Kim, Y. D., Lim, H. G., & Park, M. W. (1996). Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research, 91*, 124–143.

Knowles, J. D., & Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation, 8*(2), 149–172.

Kuo, W., & Yang, D. (2006). Single-machine group scheduling with a time-dependent learning effect. *Computers & Operations Research, 33*, 2099–2112.

Kurz, M. E., & Askin, R. G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research, 159*, 66–82.

Kusiak, A. (1987). The generalized group technology concept. *International Journal of Production Research, 25*, 561–569.

Leu, B. Y., & Nazemetz, J. W. (1995). Comparative analysis of group scheduling heuristics in a flow shop cellular system. *International Journal of Operations & Production Management, 15*((9), 143–157.

Liaee, M. M., & Emmons, H. (1997). Scheduling families of jobs with setup times. *International Journal of Production Economics, 51*, 165–176.

Liu, Z., & Yu, W. (1999). Minimizing the number of late jobs under the group technology assumption. *Journal of Combinatorial Optimization, 3*, 5–15.

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

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

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

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

Mansini, R., Grazia Speranza, M., & Tuza, Z. (2004). Scheduling groups of tasks with precedence constraints on three dedicated processors. *Discrete Applied Mathematics, 134*, 141–168.

Mitrofanov, S. P. (1966). *Scientific principles of group technology*. London: National Lending Library. (English Translation Boston Spa).

Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Multi-objective genetic algorithm and its application to flowshop scheduling. *Computers & Industrial Engineering, 30*, 957–968.

Naderi, B., Zandieh, M., & Fatemi Ghomi, S. M. (2008). A study on integrating sequence dependent setup time flexible flow lines and preventive maintenance scheduling. *Journal of intelligent manufacturing*. doi:10.1007/s10845-008-0157-6.

Reddy, V., & Narendran, T. T. (2003). Heuristics for scheduling sequence dependent set-up jobs in flow line cells. *International Journal of Production Research, 41*(1), 193–206.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first ICGA* (pp. 93–100).

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*, 324–339.

Sekiguchi, Y. (1983). Optimal schedule in a GT-type flow-shop under series-parallel precedence constraints. *Journal of the Operations Research Society of Japan, 26*, 226–251.

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. R., & Mirzaei, A. H. (2008). Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm. *International Journal of Advanced Manufacturing Technology, 36*(9-10), 969–981.

Vickson, R. G., & Alfredsson, B. E. (1992). Two- and three-machine flow shop scheduling problems with equal sized transfer batches. *International Journal of Production Research, 30*, 1551–1574.

Wilson, A. D., King, R. E., & Hodgson, T. J. (2004). Scheduling non-similar groups on a flow line: Multiple group setups. *Robotics and Computer-Integrated Manufacturing, 20*, 505–515.

Yang, D., & Chern, M. S. (2000). Two-machine flowshop group scheduling problem. *Computers and Operations Research, 27*, 975–985.

Zandieh, M., Dorri, B., & Khamseh, A. R. (2008). Robust metaheuristics for group scheduling with sequence-dependent setup times in hybrid flexible flow shops. *International Journal of Advanced Manufacturing Technology*. doi:10.1007/s00170-008-1740-x.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. Technical report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.