Contents lists available at ScienceDirect

Applied Mathematical Modelling

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

Hybrid flowshop scheduling with machine and resource-dependent processing times

J. Behnamian, S.M.T. Fatemi Ghomi *

Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Avenue, 1591634311 Tehran, Iran

ARTICLE INFO

Article history:
Received 26 December 2009
Received in revised form 21 July 2010
Accepted 29 July 2010
Available online 4 August 2010

Keywords:
Hybrid flowshop scheduling
Hybrid metaheuristic
Multi-objective optimization
Resource allocation
Machines with different speeds
Sequence-dependent setup times

ABSTRACT

Most of research in production scheduling is concerned with the optimization of a single criterion. However the analysis of the performance of a schedule often involves more than one aspect and therefore requires a multi-objective treatment. In this research, with combination of two multiple objective decision-making methods, min–max and weighted techniques, a new solution presentation method and a robust hybrid metaheuristic, we solved sequence-dependent setup time hybrid flowshop scheduling problems. In this paper for reflecting real-world situation adequately, we assume the processing time of each job depends on the speed of machine and amount of resource allocated to each machine at the stage which is processed on it. In formulation of min–max type, the decision-maker can have the flexibility of mixed use of weights and distance parameter in expressing desired improvement on produced Pareto optimal solutions. To minimize makespan and total resource allocation costs, the proposed hybrid approach is robust, fast, and simply structured, and comprises two components: genetic algorithm and a variable neighborhood search. The comparison shows the proposal to be very efficient for different structure instances.

1. Introduction

A hybrid flowshop scheduling problem (HFSP), as described by Linn and Zhang [1], consists of series of production stages, each of which has multiple machines operating in parallel and at least one stage must have more than one machine to differ from traditional flow shop environment. The HFSP is an adequate model for several industrial settings such as semiconductors, electronics manufacturing, airplane engine production, and petrochemical production [2].

Many real-world problems involve simultaneous optimization of several objective functions. In general, these functions often compete and conflict with themselves [3]. Due to the impossibility to achieve optimal values in all objectives simultaneously, multiple criteria decision making (MCDM) always involves a choice problem. The final solution represents a compromise between the different objectives depending on the preferences of the decision-maker. The scientific area concerned with modeling and analyzing preference structures to formalize the choice process from usually small, explicit list of alternatives is called multi attribute decision analysis [4].

Many decision problems contain a large, possibly infinite number of decision alternatives. In such cases, it is impossible to explicitly compare all alternatives, and therefore the choice problem is accompanied by a search problem to filter promising (optimal) from unpromising (non-optimal) alternatives. Problems of this type are treated in the area called multi-objective optimization (MOO). A typical classification of methods for multi-objective decision making is given by Hwang and Masud

* Corresponding author. Tel.: +98 2166413034; fax: +98 2166413025.
E-mail address: fatemi@aut.ac.ir (S.M.T. Fatemi Ghomi).

0307-904X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2010.07.057

[5], who distinguish four classes according to when the decision-maker's preferences enter the formal decision making process. These different possibilities are:

- No articulation of preference information (only search).
- Priori aggregation of preference information (choice before search).
- Progressive articulation of preference information (integration of search and choice).
- Posteriori articulation of preference information (search before choice).

Recently interest in multi-objective scheduling has been increasing but has been limited when compared to research in single criterion problems. In a real manufacturing environment, several objectives frequently need to be considered at the same time. In this study, the following criteria are to be minimized:

- f_1: makespan or maximal completion time of machines costs, and
- f_2: total resource allocation costs.

In many real-world cases, since older machines have high replacement costs, they may be used side by side with newer machines to perform the same operations. But because of older machines are less efficient, we expect they would generally require a longer operating time in comparison with new and modern ones [6].

Multi-objective problems are most commonly solved indirectly using conventional (single objective) optimization techniques. The solution process includes converting multi-objective to single objective formulations (i.e., scalarizing the vector optimization problems) by linear/convex combination, equality/inequality constraining, or using distance measure, etc. [7]. This paper focuses on direct selection methods that are based on the concept of least distance, i.e., approaching *as close as possible* certain desired objective values.

Several industries encounter setup times, which result in even more difficult classes of scheduling problems [8]. For instance this may occur in a painting operation, where different initial paint colors require different levels of cleaning when being followed by other paint colors. Due to great saving when setup times are explicitly included in scheduling decisions, we take into account the existence of sequence-dependent setup times (SDST) in our problem.

The underlying assumption in this paper is that, there are machines on hybrid flowshop problem with different speeds in each stage which processing time of each job depends on these speeds. We also assume the job processing times can be controlled by changing the allocation of resources to the jobs, which may result in further efficiencies. This means that the processing times of the jobs depend on the allocated resources to each machine and speed of machine on a particular stage and more speed or allocation more resource to work on the same job will decrease job completion time. Considering such real situation is common in many operations management, especially in breaking processing bottlenecks in the lean production [9].

To obtain an optimal solution for this type of complex problems using traditional approaches in reasonable computational time is extremely difficult. So in this paper, the hybrid metaheuristic (HMH) method composed of genetic algorithm (GA) and variable neighborhood search (VNS) metaheuristics is proposed to solve the problems. In proposed hybrid algorithm the VNS improves a solution using three neighborhood searches. In addition we used "no articulation of preference information given" technique as multi-objective method.

Our goal in this paper is to develop an efficient hybrid metaheuristic to bi-objective hybrid flowshop with sequence-dependent setup time and machine and resource-dependent processing times. The paper has the following structure. Section 2 gives the brief literature review of hybrid flowshop scheduling. Section 3 is the problem description, mathematical model and multi-objective technique. Section 4 introduces the proposed hybrid algorithm. Section 5 presents the experimental design which compares the results achieved by proposed hybrid algorithm with those achieved by past algorithm. Finally, Section 6 is devoted to conclusions and future works.

2. Literature review

2.1. Multi-objective metaheuristic

Various metaheuristic algorithms have ever been derived for bi-objective or multi-objective optimization problems. Vector evaluated genetic algorithm (VEGA) is the first method modifying the GA to solve multi-objective problems which is propsed by Schaffer [10] to solve the Pareto-optimal solution of multi-objective problem. Murata et al. [11] proposed multi-objective genetic algorithm (MOGA). One characteristic of MOGA is using the dynamic weighting to transform the multiple objectives into single objective, which randomly assigns different weight value to different objectives. Non-dominated sorting genetic Algorithm 2 (NSGA-II) was proposed by Deb et al. [12], where the elitism strategy was adopted. Besides, in order to keep the solution diversity, the algorithm also provided a crowding distance to measure the density of individuals in solution space. Zitzler et al. [13] modified strength Pareto evolutionary algorithm (SPEA) as SPEA II for multi-objective optimization. In addition, some sub-population like approaches also can be found in related literatures, such as segregative genetic algorithms [14], multi-population genetic algorithm [15], hierarchical fair competition model [16], and multi-objective particle swam optimization [17] and two phases sub-population genetic algorithm [18]. Recently the global archive sub-population genetic algorithm with adaptive strategy (GSPG) proposed by Chang et al. [19] for parallel machine scheduling.

2.2. Hybrid flowshop scheduling

Lee et al. [20] have applied genetic algorithms to the joint problem of determining lot sizes and sequence to minimize makespan in flexible flowshops. Though this research included sequence-dependent setup times, buffers between stages were limited and a permutation schedule was required. Combining genetic algorithms with simulated annealing was also considered. Rios-Mercado and Bard [21] also considered the sequence-dependent setup time flowshop and developed several valid inequalities for models based on the traveling salesman problem.

Botta-Genoulaz [22] proposed several heuristics for a flowshop with multiple identical machines per stage, positive time lags and out-tree precedence constraints as well as sequence-independent setup and removal times. Azizoglu et al. [23] considered the total flow time measure in a multistage hybrid flowshop and suggested a branch and bound algorithm that gives the optimal solutions for moderate-size problems. Gupta et al. [24] proposed some iterative algorithms to solve a hybrid flow shop problem with controllable process time and assignable due dates. Harjunkoski and Grossmann [25] included setup times in their work but are only dependent on the machine and not on the job. Kurz and Askin [26] compared several methods for a makespan minimization problem with sequence-dependent setup times. Jobs are allowed to skip stages. In subsequent study, by same authors the new research is done in which an integer model, some heuristics and a random keys genetic algorithm is developed for SDST flexible flowshop in 2004. Wardono and Fathi [27] developed a tabu search for multistage parallel machine problem with limited buffers. Another research which addressed the real-world industries problems is studied by Andrés et al. [28]. They considered the problem of products grouping in a tile industry. They proposed some heuristic and metaheuristic methods for a three-stage HFSP with sequence-dependent setup times. For HFSP Tang et al. [29] proposed a new Lagrangian relaxation algorithm based on stage decomposition to minimize the total weighted completion time. For SDST hybrid flowshop an immune algorithm (IA) is proposed by Zandieh et al. [30]. Jina et al. [31] considered the multistage hybrid flowshop scheduling problem. For minimizing the makespan based on simulated annealing and the variable-depth search they proposed the optimization procedure. This study revealed that the proposed metaheuristic in comparison with Johnson [32] and SPT rules and tabu search is an efficient algorithm. Janiak et al. [33] proposed some approximation algorithms for the HFSP with cost-related criterion. In this paper the scheduling criterion consists of three parts: the total weighted earliness, the total weighted tardiness and the total weighted waiting time. An improved ant colony optimization for hybrid flowshop scheduling is considered by Alaykiran et al. [34] to minimize C_{max} criterion. In order to achieve better results, they conducted a parameter optimization study. Choi et al. [35] consider the flow shop scheduling problem with one machine of different speed. They proposed two heuristics and some optimality conditions in order to minimize makespan. Ying [36] proposed an iterated greedy heuristic to minimize makespan in a multistage hybrid flowshop with multi-processor tasks. In this research for validation and verification of the proposed heuristic, computational experiments have been performed on two benchmark problem sets. Kim et al. [37] focused on the scheduling problem of minimizing makespan for a given set of jobs in a two-stage hybrid flowshop subject to a product-mix ratio constraint. Gholami et al. [38] showed how they can incorporate simulation into genetic algorithm approach to the scheduling of a sequence-dependent setup time hybrid flowshop with machines that suffer stochastic breakdown to optimize objectives based on expected makespan. Wang and Tang [39] investigated the hybrid flowshop scheduling with finite intermediate buffers. In this research, they proposed tabu search combined with a scatter search mechanism to minimize the sum of weighted completion time of all jobs. Naderi et al. [40] addressed the problem of scheduling hybrid flowshops where the setup times are sequence-dependent to minimize total completion time and total tardiness. They hybridized the simulated annealing with a simple local search to solve this problem. Su and Lien [41] proposed a heuristic for the parallel machine scheduling when the processing time of each job depends on the amount of resource consumed. They proved such scheduling problem is NP-hard even for the fixed job processing times. Luo et al. [42] considered a two-stage hybrid flowshop scheduling problem in a metal-working company. In this research for minimizing the makespan a genetic algorithm is used to obtain a near-optimal solution. Behnamian et al. [43] considered the problem of sequence-dependent setup time hybrid flowshop scheduling with the objectives of minimizing the makespan and sum of the earliness and tardiness of jobs, and presented a three-phase multi-objective method.

Jungwattanakit et al. [6] considered the flexible flowshop scheduling with sequence-dependent setup times, unrelated parallel machines and the objectives of minimizing weighted sum of the makespan and number of tardy jobs. They proposed genetic algorithm and simulated annealing for the problems. Khalouli et al. [44] propose an ant colony optimization method to deal with a hybrid flow shop scheduling problem considering the minimization of the sum of the total earliness and tardiness penalties. Recently, Kahraman et al. [45] proposed a parallel greedy algorithm to solve multi-processor task scheduling in multistage hybrid flowshops problem. In that study parallel greedy algorithm is applied by two phases iteratively, called destruction and construction. The authors also proposed four constructive heuristic methods.

3. Mathematical model

This section applies one of the well-known multiple criteria decision making (MCDM) methods, the min–max formulation, to solve the addressed scheduling problem with simultaneous consideration of two performance measures; makespan and total resource allocation costs.

3.1. Notations

Following defines the notations used to develop a mathematical model of the addressed problem:

g: the number of serial stages, $t \in \{1, 2, \ldots, g\}$
n: the number of jobs to be processed, $i \in \{1, 2, \ldots, n\}$
m^t: the number of machines in parallel at each stage t, $j \in \{1, 2, \ldots, m^t\}$
C_i^t: completion time of job i at stage t,
C_i: completion time of job i,
C_{max}: max $\{C_l, \ldots, C_n\}$,
p_i^t : standard processing time for job*i* at stage t
s_{ik}^t : sequence-dependent setup time from job i to job k at stage t
v_{ij}^t : the relative speed of machine j at stage t for job i
$\widetilde{p_{ij}^t}$: modified processing time for job i on machine j at stage t
$x_{ij}^t = \begin{cases} 1 & \text{if job } i \text{ is allocated to machine } j \text{ at stage } t, \\ 0 & \text{otherwise.} \end{cases}$

3.2. Assumptions

The HFSP considered in this paper is characterized as follows:

1. All data in all problems used in this paper are known deterministically when scheduling is undertaken.
2. Each stage has at least one machine, and at least one stage must have more than one machine.
3. A job once started on the machine must be completed on it without interruption.
4. The jobs can wait between stages and the intermediate storage is unlimited.
5. A job can be processed by any of the machines and will be processed by a single machine in each stage which can have different speeds.
6. A machine can process only one job at a time.
7. There is no travel time between stages.
8. Processors are available at all times if are not busy, with no breakdowns or scheduled or unscheduled maintenance.
9. Jobs are available for processing at a stage immediately after releasing from the previous stage.
10. Setup times depend on sequencing of jobs which means the setup times are sequence dependent and length of time required to do the setup depends on both the prior and the current jobs to be processed.
11. The job processing times can be controlled by changing the allocation of resources to the jobs.
    This paper assumes that the processing times of jobs are dependent to the resources allocated to the stages. In other word, during the processing of job i, stage t may allocate an amount $u^t \in \{1, \ldots, U^t\}$ of resources to each machine, that may speed up its processing. U^t is the maximum available units of the resource which is ependent on resource management policy. It means that each machine at stage at least must have one unit of resource and maximum amount of resource that we can assign to all machines in particular stage can not exceed m^tU^t.
    Based on Gupta et al. [24], if u^t resources are allocated to each machine at stage t, the processing time of job i at this stage is given by Eq. (1):

$$\widetilde{p_i^t} = p_i^t - a^t u^{\ t} > 0, \tag{1}$$

where p_i^t is the 'standard' processing time for job i at stage t that can be reduced by an amount of a^tu^t and coefficient $a^t \geqslant 0$ is the compressed unit processing time for stage t which is given by technological constraints.
The only assumption on the processing times, regarding their dependence on the amount of allocated resources to the machine, is monotonicity. In other word, we assume that

$$\widetilde{p_i^t}(u^t = 1) > \widetilde{p_i{}^t}(u^t = 2) > \cdots > \widetilde{p_i^t}(u^t = U^t). \tag{2}$$

So the lower and upper bounds of process time are $p_i^t - a^tU^t$ and p_i^t.

12. The processing time of each job depends on speeds of machine which processed on it. In other word machine j at stage t can process job i at the relative speed v_{ij}^t. The processing time p_{ij}^t of job i on machine j at stage t is equal to p_i^t/v_{ij}^t.

$$p_{ij}^t = p_i^t/v_{ij}^t. \tag{3}$$

13. The objective of the problem is minimizing costs of makespan and resource allocation as given by the following equations:

$$\begin{cases} f_1 = \mu C_{\max} \\ f_2 = \sum\limits_{t=1}^{g} m^t \cdot u^t \cdot \lambda^t, \end{cases} \tag{4}$$

where $C_{\max} = \max_{i=1,\ldots,n} C_i$ and C_i is the completion time of job *i*. μ is nonnegative parameter representing the cost of one unit of operation time. Furthermore in this paper it is possible to reduce the job processing time by allocation additional resource which consequently leads to additional cost, so λ^t is nonnegative parameter representing the cost of one unit of resource at stage *t*.

3.3. Multi-objective technique: no preference articulation

Let us consider a problem where we want to minimize two conflicting objective functions $f_1(x)$ and $f_2(x)$ simultaneously subject to a general constraint $x \in S$. The vector of objective functions, called *objective vector*, is denoted by $F(x) = (f_1(x), f_2(x))^T$, and the vector $x = (x_1, x_2, \ldots, x_n)^T$ is called a *decision vector*. f_1^* and f_2^* will be used to denote the individual minima of each respective objective function, and the utopian solution is defined as $\mathbf{F}^* = (f_1^*, f_2^*)^T$. As $\mathbf{F}^*$ simultaneously minimizes all objectives, it is an ideal solution subject to the constraint $x \in S$. Note that, with conflicting objectives, the ideal objective vector is infeasible [46].

These types of methods do not use any preference information. Examples are the min–max formulation and global criterion method Hwang et al. [47], Osyczka [48], and Steuer [49]. The min–max formulation is based on minimization of the relative distance from a candidate solution to the utopian solution $\mathbf{F}^*$, see Fig. 1 [50].

Multi-objective optimization problems are usually solved by scalarization [46]. *Scalarization* means converting the problem with multiple objectives into a single objective optimization problem or a family of single objective optimization problems. When a multi-objective optimization problem has been scalarized, methods developed for single objective optimization can be used to solving the problem. The objective function of the single objective problem is called a *scalarizing function* [41].

The following achievement scalarizing function is minimized subject to the constraint of the problem:

$$z_p(x) = \min \left[\sum_{i=1}^{i=2} \left(\frac{f_i(x) - f_i^*}{f_i^*} \right)^p \right]^{\frac{1}{p}} \quad \text{s.t.} \quad X \in S, \quad 1 \leqslant p \leqslant \infty. \tag{5}$$

The exponent p gives different ways of calculating the distance. When p increases, distance $d_p(x)$ decreases, i.e., $d_\infty(x) \leqslant \cdots \leqslant d_2(x) \leqslant d_1(x)$ and a greater emphasis is given to the largest deviation in forming the total. The most frequently used values for p are 1 for the simplest formulation, 2 for the Euclidean distance, and ∞ for Tchebycheff norm. The difficulty with this approach is finding the value for p that will maximize the satisfaction of the decision-maker (DM).

In this method the output is just one point on the Pareto front, and the DM has to accept it as the final solution. By changing the exponent in the distance formulation and by giving the single objectives different weights, different points on the Pareto front could be found. Then, however preference information from the decision-maker is needed [50]. In this paper, we will set $p = 1$, $p = 2$ and $p = 100$ (as large number instead of ∞).

The optimum, in the min–max sense, gives a solution that treats all the objectives on terms of equal importance, and presents the advantage of being very efficient and easy to implement. Furthermore, when the min–max approach is combined with the weighting method, we can generate the set of Pareto (non-dominant) solutions for both convex and non-convex problems. This technique was tested with multi-objective engineering design problems found in the literature [51]. For this reason, we combined min–max method with the weighting method to generate various Pareto solutions. Thus, the objective function value is defined by:

$$z_{p,w}(x) = \left[w \left(\frac{f_1(x) - f_1^*}{f_1^*} \right)^p + (1-w) \left(\frac{f_2(x) - f_2^*}{f_2^*} \right)^p \right]^{\frac{1}{p}}, \tag{6}$$

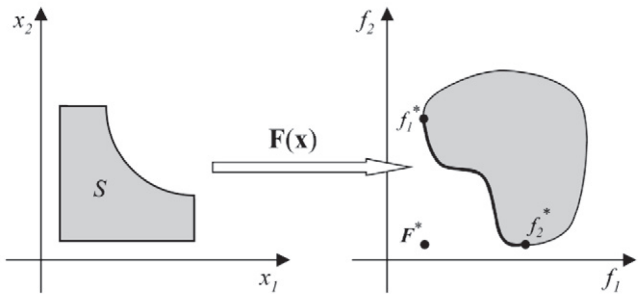

Fig. 1. Solution and objective space for a two dimensional problem with two objectives [3].

where $0 \leqslant w \leqslant 1.w$ denotes the weight (or relative importance) given to makespan costs and $(1 - w)$ denotes the weight given to the resource allocation costs.

All instances are solved using 10 different seeds for each algorithm considering one of both objective and the minimum solution in all runs is used as f_1^* and f_2^* for each objective. Then they are replaced in Eqs. (5) and (6).

3.4. Mixed integer linear programming

The mathematical modeling concerned in this paper is as follows:

$$\text{Min } Z = \left(\mu C_{\max}, \sum_{t=1}^{g} m^t \cdot u^t \cdot \lambda^t\right) \tag{7}$$

$$\text{s.t. } \sum_{j=1}^{m^t} x_{oj}^t = 1, \quad t = 1,2,\ldots,g, \tag{8}$$

$$\sum_{j=1}^{m^t} x_{ij}^t = 1, \quad i = 1,2,\ldots,n, \quad t = 1,2,\ldots,g, \tag{9}$$

$$\sum_{i=1}^{n} x_{ij}^t = 1, \quad j = 1,2,\ldots,m^t, \quad t = 1,2,\ldots,g, \tag{10}$$

$$C_i^{t+1} - C_i^t + L(1 - x_{ij}^t) \geqslant \widetilde{p_{ij}^t} + s_{ik}^t, \quad i,k = 1,2,\ldots,n, \quad j = 1,2,\ldots,m^t, t = 1,2,\ldots,g, \tag{11}$$

$$C_i \geqslant \sum_{t=1}^{g} \left(\widetilde{p_{ij}^t} + s_{ik}^t\right), \quad i,k = 1,2,\ldots,n, \tag{12}$$

$$\begin{cases} C_i^1 - C_0^1 \leqslant M^t \widetilde{p_{ij}^t}, \\ C_i^t - C_i^{t-1} \leqslant M^t \widetilde{p_{ij}^t}, \end{cases} \quad i = 1,2,\ldots,n, \quad t = 1,2,\ldots,g, \tag{13}$$

$$C_i^t \geqslant C_0^t, \quad i = 1,2,\ldots,n, \quad t = 1,2,\ldots,g, \tag{14}$$

$$\begin{cases} x_{ij}^t \leqslant \widetilde{p_{ij}^t}, \\ x_{ji}^t \leqslant \widetilde{p_{ij}^t}, \end{cases} i,j = 0,1,2,\ldots,n,n+1, \quad t = 1,2,\ldots,g, \tag{15}$$

$$f_1 \geqslant C_i^{g_i}, \quad i = 1,2,\ldots,n, \tag{16}$$

$$x_{ij}^t \in \{0,1\}, \quad i = 1,2,\ldots,n, \quad j = 1,2,\ldots,m^t, \quad t = 1,2,\ldots,g, \tag{17}$$

$$x_{ij}^t = 0, \quad i = j, \quad t = 1,2,\ldots,g, \tag{18}$$

$$C_i^t \geqslant 0, \quad i = 1,2,\ldots,n, \quad t = 1,2,\ldots,g, \tag{19}$$

$$\widetilde{p_{ij}^t} = (p_i^t - a^t u^t)/v_{ij}^t, \tag{20}$$

Eq. (7) defines the objective function which is to minimize makespan and total resource allocation costs. Constraint set (8) ensures that m^t machines are scheduled in each stage. Relation (9) indicates that job i in stage t requires only one machine. Relation (10) guarantees that a machine can process at most one job at a time. Relations (11) and (12) assure that completion time of each job that immediately precedes another job is greater than or equal to the sum of processing and setup times of that job on all machines. In Relation (11) L denotes large positive number. Constraint sets (13) and (14) ensure that the completion time of a job at stage t that does not visit stage t, is set to the job's completion time at stage $t - 1$. The value M^t is an upper bound on the time stage t completes processing, similar to the upper bound introduced in Rios-Mercado and Bard [21].

$$M^t = \sum_{i=1}^{n} \left(\widetilde{p_{ij}^t} + \max_{k \in \{0,1,\ldots,n\}} s_{ki}^1\right) \quad \text{and} \quad M^t = M^{t-1} + \sum_{i=1}^{n} \left(\widetilde{p_{ij}^t} + \max_{k \in \{0,1,\ldots,n\}} s_{ki}^t\right).$$

Constraint set (15) ensures that jobs that do not visit a stage are not assigned to that stage. Constraint set (16) links the decision variables $C_i^{g_i}$ and f_1. Relations (17)–(19) represent the state of the variables.

Relation (20) modified processing time for job i on machine j at stage t according to the relative allocated resources and speeds of machine.

4. Proposed hybrid metaheuristic

The construction of hybrid metaheuristics is motivated by the need to achieve a good tradeoff between the global exploration and the local exploitation during the search. In this paper, the proposed hybrid metaheuristic integrates features from genetic algorithm and variable neighborhood search. Let us now discuss which aspects have been borrowed from these algorithms and how are integrated as one solving method.

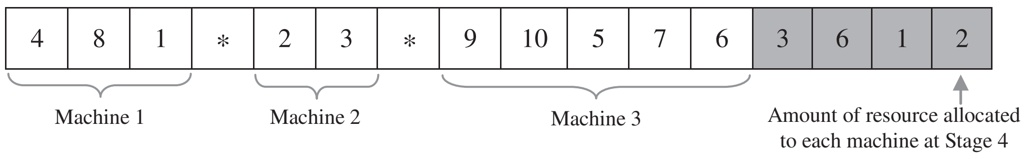

Fig. 2. Solution representation.

4.1. Encoding scheme

The proposed representation for the HMH to solve scheduling and resource allocation problems is based on coding all jobs and the amount of resource allocated to each machine at stage as genes in a 1-by-$(n + m^1 + g - 1)$ string which is composed of two parts: scheduling and resource allocation. In this type of representation, $(m^1 - 1)$ genes consisting of "*"s are used to differentiate from one machine to the other one. The last g blocks of string are used for presenting the resource allocation.

Fig. 2 shows an example for representation. In this example, there are ten jobs and the number of parallel machines at the first stage is 3. As shown in Fig. 2, jobs 1, 4 and 8 with order $4 \rightarrow 8 \rightarrow 1$ are assigned to machine 1; jobs 2 and 3 are assigned to machine 2. Jobs 9, 10, 5, 7 and 6 are assigned to the last machine. It is important to note that, the sequence of jobs is represented by the number of genes from the left side to the right side. This example assumed that there are 4 stages. The resource allocation states that three units of resource are assigned to stage 1, six units of resource are assigned to stage 2, one unit of resource is assigned to stage 3, and two units of resource are assigned to stage 4.

4.2. Initial solution

Any available method would be sufficient to generate a feasible solution for the first part of our algorithm. Random initial solutions for constructive metaheuristics such as GAs are a common and popular procedure. Randomness characteristic of this technique causes varying results in different runs. To generate random solutions for the initial population, the procedure is as follows.

Generate $(m^1 - 1)$ asterisks "*" and randomly assign to genes of the chromosome in which none of them are assigned to the first and the last genes, and between the asterisks there must be at least one unfilled gene. Then, assign the numbers from 1 to n to the rest of the unfilled blocks in the $(n + m^1 - 1)$ first genes.

4.3. Genetic algorithm

Genetic algorithm proposed by Holland [52] is classified as a stochastic optimization algorithm. During the last decades, GA has become one of the most well-known metaheuristics and is widely used in many combinatorial optimization problems, including machine scheduling [53]. Genetic operators such as crossover and mutation are used to generate new individuals and to keep away from local optimum or premature convergence.

4.3.1. Crossover operator

Since the solution string has two parts, we apply two different crossover operators. In the first part (first $(n + m^1 - 1)$ genes) a kind of uniform crossover namely position-based operator [54] is used. For resource allocation at stages (last g genes), 0.5-uniform crossover is implemented [55]. The algorithmic structure of this crossover is as follows:

1. Choose two parents, parent 1 (P1) and parent 2 (P2), from the population.

P1	4	8	1	*	2	3	*	9	10	5	7	6	3	6	1	2
P2	2	7	8	1	*	3	10	4	6	5	*	9	2	4	3	9

2. Create binary template (BT) and assign a randomly generated binary (0–1) to each cell.

BT	1	0	0	1	0	1	1	0	1	0	0	1	1	0	0	1

3. Copy the genes from Parent 1 corresponding to the locations of the "1"s in the binary string to the same positions in the offspring (OS).

OS	4			*		3	*		10			6	3			2

4. In the first part (first ($n + m^1 - 1$ block)) the genes that have already been selected from the first parent are deleted from the second one, so that the repetition of a gene in the new offspring is avoided.

P2	2	7	8	1	*	–	–	–	–	5	*	9	2	4	3	9

5. Complete the remaining empty gene locations with the undeleted genes that remain in the parent 2 by preserving their gene sequence.

O**S**	4	2	7	*	8	3	*	1	10	5	9	6	3	4	3	2

Note that this type of representation is used for first stage and other stages are scheduled using an ordered list which is obtained by referring to completion times of jobs at the previous stage.

4.3.2. Mutation operator

Through the mutation process, we make a slight change on the sequence, i.e. generating a new but similar sequence. The purpose is to avoid convergence to a local optimum and diversifying the population. Since the solution string has two parts, in this section we apply mutation operator in each section separately. In this paper, swap mutation operator is used. In swap mutation, two positions are randomly chosen on the chromosome and the contents are swapped. Fig. 3 shows the swap mutation.

Note that, in all operators only those solutions are kept in the population which the maximum amount of resource allocation to each machine is less than U^t and asterisks "*" are not assigned to the first and the last genes, and also between the asterisks there is at least one unfilled gene.

4.3.3. Reproduction operator

In this paper an elitist strategy is used as reproduction operator. In this operator the chromosomes with the higher fitness values are automatically copied to the next generation.

4.3.4. Selection

In this paper the *ranking* and *tournament* selections are used for crossover operation (the choice is made in calibration section), the random selection strategy is used as mutation operation and the elitism strategy is used as reproduction operation.

4.4. Variable neighborhood search for HFSP

Variable neighborhood algorithm is different from the most local search heuristics in that it uses two or more neighborhoods, instead of one, in its structure. In particular, it is based on the principle of systematic change of neighborhood during the search. The motivations behind the concept of VNSs are: (i) A local minimum in one neighborhood structure is not necessarily the local minimal with respect to another neighborhood structure. (ii) A global optimum is the locally optimal with respect to all neighborhood structures.

In our algorithm, to avoid costing too much computational time, the number of neighborhoods is fixed to three neighborhood structures (NS), so index l is defined to show local search type. The basic VNS structure is illustrated by Algorithm 1 [43].

Algorithm 1. Basic VNS structure

```
Begin
Find an initial solution $S^*$;
  $l \leftarrow 1$;
  for iterations ← 1 to a maximum number of iterations do
    $S \leftarrow S^*$;
    Shake procedure: find a random solution $S' \in N_l(S)$;
    Perform a local search on $N_l(S')$ to find a solution $S''$;
    if $f(S'') \leqslant f(S^*)$ then
      $S^* \leftarrow S''$;
      $l \leftarrow 1$;
    end if
    $l \leftarrow l + 1$;
  end for
End
```

4.4.1. Local searches

Every time a neighborhood is selected, a random procedure is called. This procedure selects a random solution from the selected neighborhood structure. Therefore, the neighborhoods $N_1(S)$, $N_2(S)$ and $N_3(S)$ are created in the following manner [56], one for each l respectively:

1. For $N_1(S)$:
   - Choose randomly a machine j.
   - Choose randomly two different jobs i_1 and i_2 from machine j.
   - Swap jobs i_1 and i_2.
2. For $N_2(S)$:
   - Choose randomly two different machines j_1 and j_2.
   - Choose randomly a job i_1 for j_1 and a job i_2 for j_2.
   - Swap jobs i_1 and i_2.
3. For $N_3(S)$:
   - Choose randomly one job i_1 and one machine j_2, where i_1 does not belong to j_2.
   - Choose randomly a valid position '*pos*' in j_2.
   - Transfer job i_1 to j_2 at the position *pos*.

Note that the size of neighborhood for $N_1(S)$ is $O(m \cdot n^2)$, for neighborhood $N_2(S)$ is $O(\mathrm{m}^2 \cdot n^2)$ and for neighborhood $N_3(S)$ is $O(n^2)$.

The algorithm always tries to use the fastest local search available first. If after an iteration no improvement is made, then another neighborhood is used (l is incremented), and every time a new solution is found, the first and fastest local search is used ($l = 1$) [43].

4.5. A hybrid method

The hybrid algorithm proposed here combines two methods previously presented. The properties of the HMH algorithm are as follows:

Trait 1: *Generating the initial solution via a random initial solution*: The method starts with generating initial chromosomes randomly.

Trait 2: *Intensification phase using VNS local search*. The concept borrowed by the HMH from the VNS is that varying the neighborhood structure during the search process could facilitate the avoidance of traps and enlarges the search scope. In this phase first, the *objective value* for each neighborhood solution is calculated. VNS works by performing movements that upgrade the solutions. So, the solution that has the minimal *objective value* is selected as a move.

Trait 3: *Diversification phase via a GA and shaking function in VNS*. In HMH, the main task of the mutation operator is to maintain the diversity of the population in the successive generations and to exploit the solution space. Also choosing a random chromosome in unexplored regions is a part of the diversification strategy in GA. However, the shaking

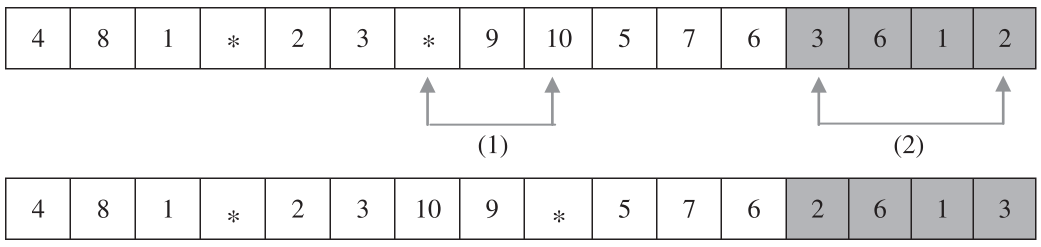

Fig. 3. Mutation operator.

Table 1
Factor levels.

Factor	Levels		
Number of jobs	6	30	100
Machine distribution	Constant: 1	2	10
	Variable: Uniform (1, 4)	Uniform (1, 10)	
Number of stages	2	4	8
Processing times	Uniform (50, 70)	Uniform (20, 100)	

procedure in the basic VNS approach is a diversification factor whilst the local search will intensify the search to lead it converges to a local optimum.

The basic proposed hybrid algorithm structure is designed as shown in Algorithm 2.

Algorithm 2. Main body of hybrid algorithm

```
Begin
Step 1. Initialization
  Input data set;
  Initialize parameters: Number of initial population (popsize); Crossover rate ($C_r$), Mutation rate ($M_r$),
  Reproduction rate ($R_r$) and Selection mechanism in crossover; $\mu$ and $\lambda^t$; run time ($R_{time}$), $l \leftarrow 1$;
  Initial population generation: Randomly generate an initial population of popsize chromosomes;
  Objective function evaluation;
Step 2. Scheduling and Assignment
  while run time < $R_{time}$ do
    Crossover $C_r$%;
    Mutation $M_r$ %;
    Solution improvement:
    {
        $S^* \leftarrow$ GA solution ()
        for iterations $\leftarrow$ 1 to a maximum number of iterations do
          $S \leftarrow S^*$;
          Shake procedure: find a random solution $S' \in N_l(S)$;
          Perform a local search on $N_l(S')$ to find a solution $S''$;
          if $f(S'') \leqslant f(S^*)$ then
            $S^* \leftarrow S''$;
            $l \leftarrow 1$;
          else
            $l \leftarrow l + 1$;
        end for
    }
    Objective function evaluation;
    Reproduction operation $R_r$ %;
    Generate next generation randomly (popsize - $R_r$) %;
  end while
Step 3. Report results
End
```

Note that, VNS is only applied for scheduling improvement and resource allocation is done by GAs operators. In other word, in HMH, when VNS searched for better solutions, the amount of resource allocated to each machine at stages are fixed.

4.6. Stopping criterion

Recently, many researchers use a limited computational time as a stopping criterion because of its advantages. In this paper, the stopping criterion used when testing all instances with the algorithms is set to a CPU time limit fixed to $(n^2 \times \sum_{t=1}^{g} m^t / g) \times 5$ ms. This stopping criterion not only is responsive to the number of stages, but also is sensitive toward rise in the number of jobs and the number of machines at each stage [43].

5. Experimental design

5.1. Data generation and settings

The problem data can be characterized by four factors, and each of these factors can have at least two levels. These levels are shown in Table 1.

The setup times are uniformly distributed from 12 to 24 which are 20% to 40% of the mean of the processing time [21]. The setup time matrices are asymmetric and satisfy the triangle inequality. Probability of skipping a stage is the important character in the hybrid flowshop problem which in this paper is set at 0, 0.05, or 0.40 [57].

There are 252 test scenarios and six data sets are generated for each one. Note that some further restrictions are introduced. The largest number of machines in a stage must be less than the number of jobs. Thus, the combination with

10 machines at each stage and 6 jobs will be skipped and the combination of 1–10 machines per stage with 6 jobs will be changed to 1–6 machines per stage with 6 jobs. Also, the variable machine distribution factor requires that at least one stage has a different number of machines than the others [43].

5.2. Multi-objective hybrid algorithm parameters tuning

The efficiency of a hybrid algorithm depends on the choice of the best parameters in order to prevent premature convergence, to ensure diversity in the search space, to intensify the search around interesting regions, etc. This section describes an empirical testing approach to find the best tuning parameters of our hybrid algorithm. We have applied parameters tuning for the crossover rate (C_r), mutation rate (M_r), population size (*popsize*) and selection mechanism in crossover operator, considering the following ranges:

- C_r: three levels (0.70, 0.80 and 0.90),
- M_r: three levels (0, 0.01 and 0.05),
- *popsize*: three levels (100, 200 and 300),
- *Selection mechanism*: two levels (ranking and tournament)

54 different combinations are obtained by these levels. We generate 6 instances, 2 small, 2 medium and 2 large, for each combination of n, m and SDST resulting a total of 252 instances. All these instances are solved by 54 different combination algorithms. It is worth to notice that in order to avoid over-fitting in the results we use the different instances for the tuning of parameters and for evaluating the algorithm.

The results are analyzed by the means of multi-factor Analysis of Variance (ANOVA) technique. It is necessary to notice that to use ANOVA, three main hypotheses, normality, homogeneity of variance and independence of residuals, must be checked. We did that and found no bias for questioning the validity of the experiment. Table 2 shows the results for different sizes of problems; small, medium and large.

It is worth to notice that in order to reduce the number of combinations, some parameters of the HMH are fixed a priori based on the claims suggested in the literature.

5.3. Experimental results

In this section we are going to compare the proposed hybrid algorithm with the random initial population simulated annealing (RNDSA) of Jungwattanakit et al. [6] which was originally proposed for the flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria.

The proposed algorithm and RNDSA are coded in C++ and run with an Intel Pentium IV dual core 2.5 GHz PC at 896 MB RAM under a Microsoft Windows XP environment.

5.3.1. Evaluation metrics

After computation of objective value of each algorithm for its instances, the best solution obtained for each instance (which is named Min_{sol}) by any of the two algorithms is calculated. Relative percentage deviation (RPD) is obtained by the following formula:

$$\mathrm{RPD} = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}}, \tag{21}$$

where Alg_{sol} is the objective value obtained for a given algorithm and instance. RPD of 4% for a given algorithm means that this algorithm is 4% over the best obtained solution on average. Clearly, lower values of RPD are preferred.

5.3.2. Analysis of objective value on RPD

For all problem sizes, we tested instances with $p \in \{1, 2, \text{and } \infty\}$ and $w \in \{0, 0.2, 0.5, 0.8, \text{and } 1\}$ in the objective function. Table 3 shows the results of the experiments for two subsets, averaged for each one of the n, g and p configurations.

Table 2
Parameters tuning.

Parameters	Problems		
	Small	Medium	Large
C_r	0.70	0.80	0.90
M_r	0.01	0.05	0.05
Popsize	100	200	200
Selection mechanism	Tournament	Tournament	Tournament

Table 3
Average relative percentage deviation ($\overline{RPD}$) for algorithms grouped by n, g, p and p with w=0.5.

Problem size	Algorithm					
	p = 1		p = 2		p = 100	
	HMH	RNDSA	HMH	RNDSA	HMH	RNDSA
6 × 2	0.043	0.323	0.067	0.251	0.079	0.401
6 × 4	0.001	1.902	0.000	1.576	0.001	1.729
6 × 8	0.000	3.061	0.000	2.682	0.000	3.901
6 Job	0.015	1.762	0.022	1.503	0.026	2.010
30 × 2	0.028	1.070	0.024	1.878	0.026	1.975
30 × 4	0.049	0.762	0.047	0.866	0.049	1.341
30 × 8	0.042	0.646	0.036	1.483	0.030	2.079
30 Job	0.040	0.826	0.036	1.409	0.035	1.798
100 × 2	0.031	0.358	0.038	0.345	0.034	0.421
100 × 4	0.028	0.504	0.043	0.792	0.048	0.630
100 × 8	0.031	0.518	0.021	0.802	0.033	0.856
100 Job	0.030	0.460	0.034	0.646	0.039	0.636
Average	0.028	1.016	0.031	1.186	0.033	1.482

As it can be seen, the hybrid algorithm provides better results than RNDSA. In order to verify the statistical validity of the results shown in Table 3 and to confirm which the best algorithm is, we have performed a design of experiments and an analysis of variance (ANOVA) where we consider the different algorithms as a factor and the response variable RPD.

We also for p = 1, we tested instances with $w \in \{0, 0.2, 0.5, 0.8, 1\}$ in the objective function. The idea behind the w values is to balance both objectives. Table 4 shows the results of the experiments for two subsets, averaged for each one of the n and g configurations. Note that, in Tables 3 and 4 each instance is solved using 6 different seeds and the average solution is considered.

The results demonstrate that there is a clear statistically significant difference between performances of the algorithms for $w \in \{0, 0.2, 0.5, 0.8, \text{and } 1\}$ with p = 1. Fig. 4 shows the means plot and LSD intervals (at the 95% confidence level) for two algorithms.

5.3.3. Analysis of controlled factors

5.3.3.1. Analysis of problem size factor (number of jobs). In order to see the effect of number of jobs on two algorithms, a two ways ANOVA is applied. Fig. 5 shows the means plot and LSD intervals (at the 95% confidence level) for the interaction among the factors of type of algorithm, number of jobs, w and p, separately.

As we can see, in the all cases of n = 6, n = 30 and n = 100 the HMH works better than RNDSA.

5.3.3.2. Analysis of g factor (number of stages). Another two ways ANOVA and LSD test are applied to see the effect of magnitude of stages, w and p on quality of the algorithms. Fig. 6 shows the results.

As we can see, in all cases the hybrid algorithm works better than RNDSA.

Table 4
Average relative percentage deviation ($\overline{RPD}$) for algorithms grouped by n, g and w with p = 1.

Problem size	Algorithm									
	w = 0		w = 0.2		w = 0.5		w = 0.8		w = 1	
	HMH	RNDSA	HMH	RNDSA	MH	RNDSA	HMH	RNDSA	HMH	RNDSA
6 × 2	0.032	0.246	0.084	0.285	0.043	0.323	0.005	0.435	0.067	0.065
6 × 4	0.046	0.232	0.000	1.027	0.001	1.902	0.002	1.071	0.037	0.064
6 × 8	0.036	0.229	0.001	3.609	0.000	3.061	0.000	1.856	0.027	0.089
6 Job	0.038	0.236	0.028	1.640	0.015	1.762	0.003	1.121	0.044	0.073
30 × 2	0.050	0.190	0.026	2.339	0.028	1.070	0.004	0.155	0.016	0.090
30 × 4	0.102	0.414	0.051	3.352	0.049	0.762	0.015	0.217	0.017	0.068
30 × 8	0.051	0.314	0.035	2.366	0.042	0.646	0.008	0.270	0.018	0.048
30 Job	0.068	0.306	0.037	2.686	0.040	0.826	0.009	0.214	0.017	0.069
100 × 2	0.039	0.216	0.057	1.550	0.031	0.358	0.038	0.197	0.013	0.046
100 × 4	0.072	0.285	0.077	1.105	0.028	0.504	0.057	0.554	0.019	0.052
100 × 8	0.035	0.320	0.050	1.094	0.031	0.518	0.041	0.339	0.018	0.037
100 Job	0.049	0.274	0.061	1.250	0.030	0.460	0.045	0.363	0.017	0.045
Average	0.051	0.272	0.042	1.859	0.028	1.016	0.019	0.566	0.026	0.062

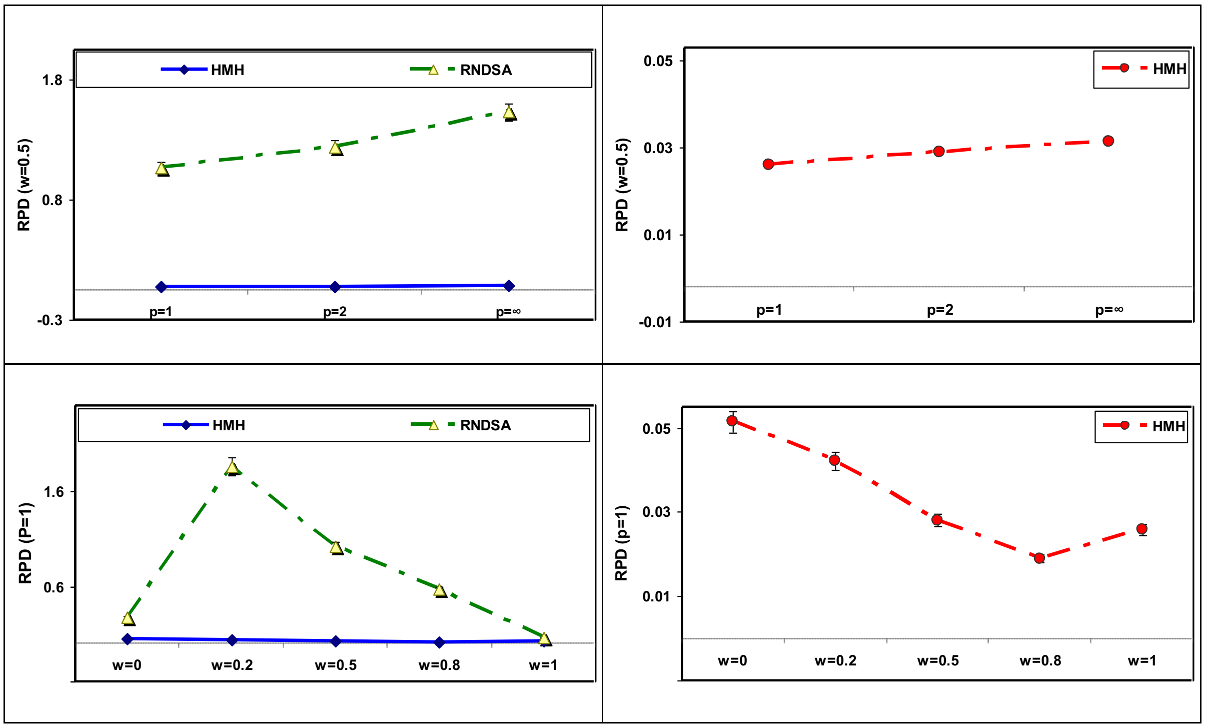

Fig. 4. Plots of $\overline{RPD}$ for the type of algorithm factor, *w*, and *p* value.

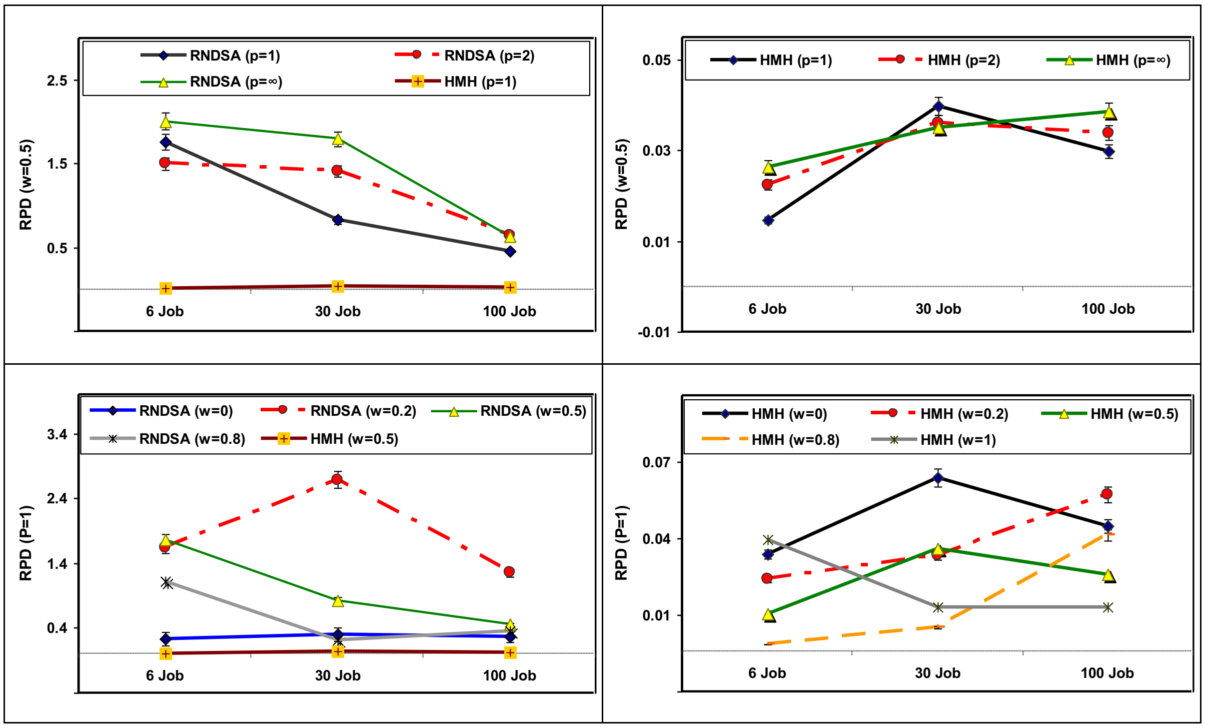

Fig. 5. Plots of $\overline{RPD}$ for the interaction among the type of algorithm, number of jobs, *w* and *p*.

5.4. Non-dominated solutions

As mentioned before, when the min–max approach is combined with the weighting method, we can generate the set of Pareto (non-dominant) solutions for the problems. In this section we are going to compare the proposed multi-objective

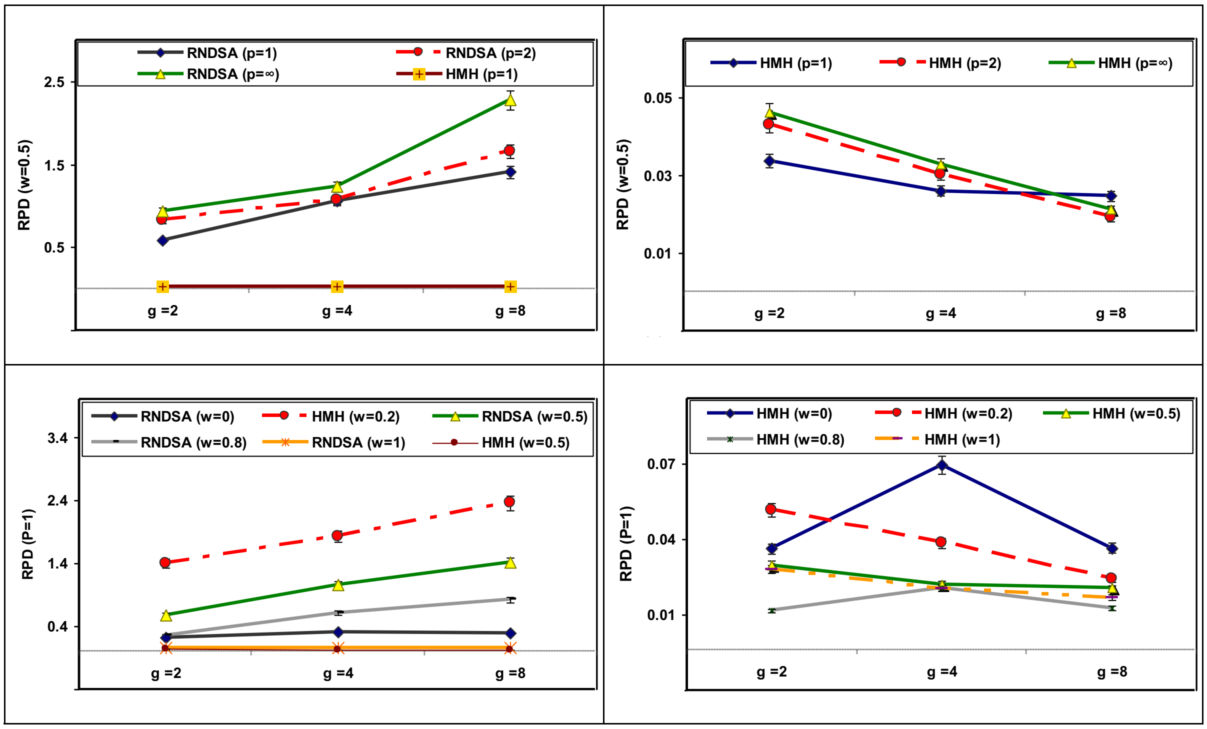

Fig. 6. Plots of $\overline{RPD}$ for the interaction among the type of algorithm, magnitude of stages, w and p.

hybrid algorithm (MOHM) with the NSGA-II (non-dominated sorting genetic algorithm-II) proposed by Deb et al. [12] and the random initial population simulated annealing (RNDSA) proposed by Jungwattanakit et al. [6]. In this paper NSGA-II and RNDSA were adapted to address problem.

5.4.1. Evaluation of non-dominated solution

In this section we need meter that provides a quantitative measure to compare different sets of Pareto solution. For this reason there are several approaches in multi-objective literature. In this paper we used four indices proposed in Behnamian et al. [43] as follows:

1. *MID* (mean ideal distance): The closeness between Pareto solution and ideal point (0, 0).
2. *SNS*: The spread of non-dominance solution.
3. *RAS*: The rate of achievement to two objectives simultaneously.
4. *ALC*: Area under linear regression curve that fits to Pareto solution data with Excel trend line function.

The equation of *MID* is defined as follows:

$$MID = \frac{\sum_{i=1}^{n} c_i}{n}, \tag{22}$$

where n is the number of non-dominated set and $c_i = \sqrt{f_{1i}^2 + f_{2i}^2}$. The lower value of *MID*, the better of solution quality we have. *SNS*, as diversity measure, can be expressed by the following equation:

$$SNS = \sqrt{\frac{\sum_{i=1}^{n} (MID - c_i)^2}{n-1}}. \tag{23}$$

The higher value of *SNS*, the better of solution quality we have (more diversity in obtained solution). Another evaluation metric which we proposed in this paper is *RAS*. The equation of *RAS* is represented as Eq. (24).

$$RAS = \frac{\sum_{i=1}^{n} \left(\frac{f_{1i} - F_i}{F_i}\right) + \left(\frac{f_{2i} - F_i}{F_i}\right)}{n}, \tag{24}$$

where $F_i = \min\{f_{1i}, f_{2i}\}$. The lower value of *RAS*, the better of solution quality we have. Also, all measures consider positive objective values. Therefore, solutions with negative objectives must be transformed into the positive part of the axis. Table 5 represents the values of the four metrics.

Table 5
Evaluation of non-dominated solution for algorithms.

Number of job	Algorithm and index											
	MID			SNS			RAS			ALC ($\times 10^3$)		
	MOHM	RNDSA	NSGA-II	MOHM	RNDSA	NSGA-II	MOHM	RNDSA	NSGA-II	MOHM	RNDSA	NSGA-II
30×4	638.34	628.28	594.51	21.726	17.649	16.461	4.5578	4.6006	4.5609	44.55	54.08	48.58
100×8	13,788	13,978	13,796	524.59	458.38	542.47	0.5705	0.5968	0.5708	48,840	49,809	50,284

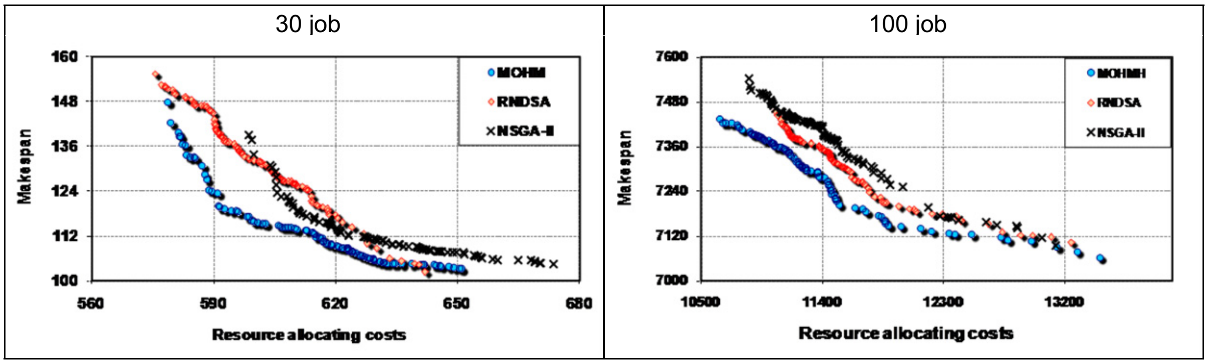

Fig. 7. A single run of the non-dominated solutions by MOHM, RNDSA and NSGA-II.

As it can be seen, the MOHM is superior to RNDSA and NSGA-II when indices *RAS* and *ALC* are concerned. Furthermore, with index *MID* in size 30×4, RNDSA provides better results than other algorithms and with index *SNS* in size 100×8, NSGA-II outperforms MOHM and NSGA-II.

5.4.2. Illustrations of the Pareto Front

Fig. 7 presents the non-dominated solutions of a single run by MOHM, NSGA-II and multi-objective RNDSA (for 2 hybrid flowshops of size (30×4), and (100×8)). The results show that the proposed algorithm works effectively in both size problems.

As we can see, the hybrid algorithm works effectively in both sizes.

6. Conclusions and future works

This paper considered the problem of sequence-dependent setup times hybrid flowshop scheduling with machine and resource-dependent processing times, and developed a multi-objective hybrid metaheuristic (MOHM). The MOHM seeks to obtain Pareto-optimal solutions through the mixed multi-objective approach. This approach combined min–max method with the weighting method that helps to generate many solutions on the non-dominated front. For minimizing makespan and total resource allocation costs, we develop a hybrid metaheuristic which combines GA and VNS in a population-based context. In the proposed metaheuristic, with population-based evolutionary searching ability of GA and the local improvement ability of VNS, the balance between the global exploration and the local exploitation was stressed.

The effectiveness of proposed algorithm was tested on 252 benchmark problems which contained up to 100 jobs. The computational results demonstrate that our method performs better than RNDSA and obtains superior results. In addition, the possibility of configuring the hybrid metaheuristic behavior provides a certain level of flexibility to adapt the algorithm to the specific characteristics of the instances to be solved. The future work is to change the objectives that we used in this paper and develop effective MOHM algorithm for other types of scheduling problems.

References

[1] R. Linn, W. Zhang, Hybrid flow shop scheduling: a survey, Comput. Ind. Eng. 37 (1) (1999) 57–61.
[2] C. Low, C.-J. Hsu, C.-T. Su, A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines, Comput. Oper. Res. 35 (3) (2008) 845–853.
[3] R. Tavakkoli-Moghaddam, A. Rahimi-Vahed, A. Hossein Mirzaei, A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness, Inform. Sci. 177 (2007) 5072–5090.
[4] G. Fandel, J. Spronk, Multiple Criteria Decision Methods and Applications, Springer Verlag, Berlin, 1985.
[5] C.-L. Hwang, A.S.M. Masud, Multiple Objectives Decision Making – Methods and Applications, Springer, Berlin, 1979.
[6] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, F.A. Werner, comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria, Comput. Oper. Res. 36 (2009) 358–378.

[7] J.G. Lin, On min–norm and min–max methods of multi-objective optimization, Math. Program. 103 (2005) 11–33.
[8] C. Zhou, P.J. Egbelu, Scheduling in a manufacturing shop with sequence-dependent setups, Robot. Comput.-Int. Manuf. 5 (1) (1989) 73–81.
[9] L-H. Su, C-Y. Lien, Scheduling parallel machines with resource-dependent processing times, Int. J. Prod. Econ. 117 (2) (2009) 256–266.
[10] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of First International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, Pittsburgh, PA, USA, 1985, pp. 93–100. July.
[11] T. Murata, H. Ishibuchi, H. Tanaka, Multi-objective genetic algorithm and its application to flowshop scheduling, Comput. Ind. Eng. 30 (1996) 957–968.
[12] K. Deb, S.A. Amrit Pratap, T. Meyarivan, A fast and elitist multi objective genetic algorithm-NSGA-II, in: Proceedings of the Parallel Problem Solving From Nature VI Conference, Paris, France, September 2000, pp. 849–858.
[13] E. Zitzler, M. Laumanns, S. Bleuler, A tutorial on evolutionary multiobjective optimization, in: Proceedings of the Workshop on Multiple Objective Metaheuristics (MOMH 2002), Springer-Verlag, Paris, France, 2004, pp. 3–38. July.
[14] M. Affenzeller, New generic hybrids based upon genetic algorithms, Lect. Notes Comput. Sci. 2527 (2002) 329–339.
[15] J.K. Cochran, S.M. Horng, J.W. Fowler, A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines, Comput. Oper. Res. 30 (2003) 1087–1102.
[16] J. Hu, E. Goodman, K. Seo, Z. Fan, R. Rosenberg, The hierarchical fair competition framework for sustainable evolutionary algorithms, Evol. Comput. 13 (2) (2005) 241–277.
[17] S. Mostaghim, J. Teich, Covering Pareto-optimal fronts by subswarms in multiobjective particle swarm optimization, Evol. Comput. 2 (2004) 1404–1411.
[18] P.-C. Chang, S-H. Chen, K.-L. Lin, Two-phase sub population genetic algorithm for parallel machine-scheduling problem, Expert Syst. Appl. 29 (3) (2005) 705–712.
[19] P.-C. Chang, S-H. Chen, J-Ch. Hsieh, A global archive sub-population genetic algorithm with adaptive strategy in multi-objective parallel-machine scheduling problem, Lect. Notes Comput. Sci. 4221 (2006) 730–739.
[20] I. Lee, R. Sikora, M.J. Shaw, A genetic algorithm based approach to flexible flow-line scheduling with variable lot sizes, IEEE Trans. Syst. Man Cyb. Part B 27 (1) (1997) 36–54.
[21] R.Z. Rios-Mercado, J.F. Bard, Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups, Comput. Oper. Res. 25 (5) (1998) 351–366.
[22] V. Botta-Genoulaz, Hybrid flowshop scheduling with precedence constraints and time lags to minimize maximum lateness, Int. J. Prod. Econ. 64 (1–3) (2000) 101–111.
[23] M. Azizoglu, E. Cakmak, S. Kondakci, A flexible flow shop problem with total flow time minimization, Eur. J. Oper. Res. 132 (2001) 528–538.
[24] J.N.D. Gupta, K. Kruger, V. Lauff, F. Werner, Y.N. Sotskov, Heuristics for hybrid flow shops with controllable processing times and assignable due dates, Comput. Oper. Res. 29 (2002) 1417–1439.
[25] I. Harjunkoski, I.E. Grossmann, Decomposition techniques for multistage scheduling problems using mixed integer and constraint programming methods, Comput. Chem. Eng. 26 (11) (2002) 1533–1552.
[26] M.E. Kurz, R.G. Askin, Comparing scheduling rules for flexible flow lines, Int. J. Prod. Econ. 85 (2003) 371–388.
[27] B. Wardono, Y. Fathi, A tabu search algorithm for the multi-stage parallel machine problems with limited buffer capacities, Eur. J. Oper. Res. 155 (2004) 380–401.
[28] C. Andrés, J.M. Albarracín, G. Tormo, E. Vicens, J.P. García-Sabater, Group technology in a hybrid flowshop environment: a case study, Eur. J. Oper. Res. 167 (2005) 272–281.
[29] L.X. Tang, H. Xuan, J. Liu, A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time, Comput. Oper. Res. 33 (2006) 3344–3359.
[30] M. Zandieh, S.M.T. Fatemi Ghomi, S.M. Moattar Husseini, An immune algorithm approach to hybrid flowshops scheduling with sequence-dependent setup times, Appl. Math. Comput. 180 (2006) 111–127.
[31] Z. Jina, Z. Yang, T. Ito, Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem, Int. J. Prod. Econ. 100 (2) (2006) 322–334.
[32] D.S. Johnson, C.R. Aragon, L.A. Mcgeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation. Part I: graph partitioning, Oper. Res. 37 (6) (1989) 865–892.
[33] A. Janiak, E. Kozan, M. Lichtenstein, C. Oguz, Metaheuristic approaches to the hybrid flowshop scheduling problem with a cost-related criterion, Int. J. Prod. Econ. 105 (2007) 407–424.
[34] K. Alaykiran, O. Engin, A. Döyen, Using ant colony optimization to solve hybrid flowshop scheduling problems, Int. J. Adv. Manuf. Technol. 35 (5–6) (2007) 541–550.
[35] B-C. Choi, S-H. Yoon, S-J. Chung, Minimizing maximum completion time in a proportionate flow shop with one machine of different speed, Eur. J. Oper. Res. 176 (2007) 964–974.
[36] K-C. Ying, An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks, J. Oper. Res. Soc. 60 (6) (2009) 810–817.
[37] Y.-D. Kim, B-J. Joo, J-H. Shin, Heuristics for a two-stage hybrid flowshop scheduling problem with ready times and a product-mix ratio constraint, J. Heuristics 15 (1) (2009) 19–42.
[38] M. Gholami, M. Zandieh, A. Alem-Tabriz, Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns, Int. J. Adv. Manuf. Technol. 42 (1–2) (2009) 189–201.
[39] X. Wang, L. Tang, A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers, Comput. Oper. Res. 36 (2009) 907–918.
[40] B. Naderi, M. Zandieh, A. Khaleghi Ghoshe Balagh, V. Roshanaei, An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness, Expert Syst. Appl. 36 (2009) 9625–9633.
[41] A. Setämaa-Kärkkäinen, K. Miettinen, J. Vuori, Heuristic for a new multiobjective scheduling problem, Optim. Lett. 1 (2007) 213–225.
[42] H. Luo, G.Q. Huang, Y. Zhang, Q. Dai, X. Chen, Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm, Robot. Comput.-Int. Manuf. 25 (6) (2009) 962–971.
[43] J. Behnamian, S.M.T. Fatemi Ghomi, M. Zandieh, A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic, Expert Syst. Appl. 36 (8) (2009) 11057–11069.
[44] S. Khalouli, F. Ghedjati, A. Hamzaoui, A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop, Eng. Appl. Artif. Intel. 23 (5) (2010) 765–771.
[45] C. Kahraman, O. Engin, Í. Kay, R.E. Öztürk, Multiprocessor task scheduling in multistage hybrid flow-shops: a parallel greedy algorithm approach, Appl. Soft Comput., in press, doi:10.1016/j.asoc.2010.03.008.
[46] K. Miettinen, Nonlinear Multiobjective Optimization, Kluwer Academic Publishers, Boston, 1999.
[47] C. Hwang, S. Paidy, K. Yoon, Mathematical programming with multiple objectives: a tutorial, Comput. Oper. Res. 7 (1) (1980) 5–31.
[48] A. Osyczka, Multicriterion Optimization in Engineering with FORTRAN Programs, Wiley, New York, 1984.
[49] R. Steuer, Multiple Criteria Optimization: Theory, Computation and Application, Wily, New York, 1986.
[50] J. Andersson, A survey of multiobjective optimization in engineering design, Technical Report LiTH-IKP-R-1097, Department of Mechanical Engineering, Linköping University, Linköping, Sweden, 2000.
[51] A. Carlos, C. Coello, A.D. Christiansen, An approach to multiobjective optimization using genetic algorithms, Fuzzy Logic Evol. Program. 5 (1995) 411–416.
[52] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, USA, Ann Arbor, 1975.
[53] C. Low, Y. Yuling, Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated, Robot. Comput.-Int. Manuf. 2 (25) (2009) 314–322.

[54] S-W. Lin, K-C. Ying, Z-J. Lee, Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times, Comput. Oper. Res. 36 (4) (2009) 1110–1121.
[55] Z. Michalewicz, Genetic Algorithms + Data Structure = Evolution Program, third ed., Springer-Verlag, Berlin, 1996.
[56] M. Rocha, M. Gómez Ravetti, G.R. Mateus, P.M. Pardalos, Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighborhood search, IMA J. Manage. Math. 18 (2007) 101–115.
[57] V.J. Leon, B. Ramamoorthy, An adaptable problem-space based search method for flexible flow line scheduling, IIE Trans. 29 (1997) 115–125.