INTELLIGENT AUTOMATION & SOFT COMPUTING, 2016
http://dx.doi.org/10.1080/10798587.2016.1261956

An Efficient Hybrid Algorithm for a Bi-objectives Hybrid Flow Shop Scheduling

S. M. Mousavi[a] and M. Zandieh[b]

[a]Faculty of Industrial Engineering, Department of Technical and Engineering, Islamic Azad University, Noshahr Branch, Postal code 41433—46511, Mazandaran, Iran; [b]Management and Accounting Faculty, Department of Industrial Management, Shahid Beheshti University, G. C., Tehran, Iran

ABSTRACT

This paper considers the problem of scheduling n independent jobs in g-stage hybrid flow shop environment. To address the realistic assumptions of the proposed problem, two additional traits were added to the scheduling problem. These include setup times, and the consideration of maximum completion time together with total tardiness as objective function. The problem is to determine a schedule that minimizes a convex combination of objectives. A procedure based on hybrid the simulated annealing; genetic algorithm and local search so-called HSA-GA-LS are proposed to handle this problem approximately. The performance of the proposed algorithm is compared with a genetic algorithm proposed in the literature on a set of test problems. Several performance measures are applied to evaluate the effectiveness and efficiency of the proposed algorithm in finding a good quality schedule. From the results obtained, it can be seen that the proposed method is efficient and effective.

KEYWORDS

Bi-objective scheduling; Hybrid flow shop; Genetic algorithm; Local search; Simulated annealing

1. Introduction

Scheduling is an important tool for manufacturing and engineering, where it can have a major impact on the productivity of a process. One of the most applied and recognized scheduling problem is hybrid flow shops (HFS), in which each job has to go through multiple stages with parallel machines instead of a single machine. The parallel machines at each stage are added for the objective of increasing productivity as well as flexibility. Ruiz and Vázquez-Rodríguez (2010) described the HFS problem in its "standard" form. This paper investigates an HFS problem in standard form with an additional feature to include setup times. The importance and applications of scheduling models with explicit considerations of setup times (costs) have been discussed in several studies (i.e. Andrés, Albarracín, Tormo, Vicens, & García-Sabater, 2005; Chang, Hsieh, & Wang, 2003). The setup times considered in this problem are classified into two types: (1) sequence-independent setup time (SIST); and (2) sequence-dependent setup times (SDST) (Naderi, Zandieh, & Roshanaei, 2008). Allahverdi, Gupta, and Aldowaisan (1999), Allahverdi, Ng, Cheng, and Kovalyov (2008) provided a comprehensive review of the literature on scheduling problems involving setup times (costs). Also, Jungwattanakit, Reodecha, Chaovalitwongse, and Werner (2008) and Kurz and Askin (2003) introduced the mathematical model of the HFS problem with sequence-dependent setup times.

The aim of scheduling is to assign jobs to the machines at the corresponding stages and determine the processing sequences on the machines so that one or some selected objectives are optimized. According to just-in-time concept, production managers should consider more than one criterion in scheduling problems. Therefore, simultaneous minimization of two conflicted objective functions that are maximum completion time (makespan or C_{max}) and total tardiness ($\bar{T}$). In fact, minimizing the makespan and the total tardiness will lead to increase internal and external efficiency respectively. The problem is configured as a bi-objective model, which is a sub-class of the multi-objective models.

The HFS scheduling problem is a strongly NP-hard problem (Ruiz & Maroto, 2006). In the literature of multi-objective HFS scheduling problem, different approaches of heuristics and meta-heuristics have been applied to solve these problems. Two types of solution methods are prevalent in scheduling: (1) develop a solution method that searches a set of non-dominated solutions; (2) develop a solution method to find a good quality schedule for convex sum of objectives. In the former, the set of Pareto optimal (or efficient) solutions is generated for the decision-maker, who then selects the most preferred among the alternatives. The inconveniences here are that the generation process is usually computationally expensive and sometimes in part, at least, difficult. On the other hand, it is hard for the decision-maker to select from a large set of alternatives. In the latter, a good quality schedule is generated for the decision-maker. Therefore, the aim of this paper is to develop a solution method for the proposed problem to find a good quality schedule. Now, some related research briefly reviewed.

Jungwattanakit et al. (2008), Jungwattanakit, Reodecha, Chaovalitwongse, and Werner (2009) considered the flexible flow shop with unrelated parallel machines and sequence - and machine -dependent setup times, release date and due date constraints to minimize a convex sum of makespan and the number of tardy jobs. They proposed the genetic algorithm (GA), simulated annealing (SA) and tabu search (TS) to find the near-optimal schedule for the problem. Davoudpour and Ashrafi (2009) considered the SDST HFS problems with identical parallel machines, and release date to minimize a weighted sum earliness, tardiness and completion time of jobs. They proposed a greedy randomized adaptive search procedure to solve this problem. Mousavi, Zandieh, and Yazdani (2012) considered

CONTACT S. M. Mousavi mousavi@iauns.ac.ir

the problem of scheduling n independent jobs in HFS environment with SDST to minimize the makespan and total tardiness. They developed a meta-heuristic based on SA/local search along with some basic improvement procedures to minimize convex combination of the makespan and total tardiness. Pargar and Zandieh (2012) investigated the HFS problems with SDST and learning effect of setup times for minimizing weighted sum of makespan and total tardiness. They proposed a novel meta-heuristic approach called water flow-like algorithm. Sheikh (2013) investigated a multi-objective flexible flow shop scheduling problem with limited time lag between stages and due windows. They formulated the problem by a mixed integer linear programming model with the objectives of maximizing the total profit gained from scheduled jobs and minimizing deviation from the due date. A GA procedure designed to solve this model efficiently. Tadayon and Salmasi (2013) investigated group scheduling in the flexible flow shop scheduling problem with release time and eligibility. They considered objectives as the minimization of the sum of the completion time of groups and the minimization of sum of the differences between the completion time of jobs and the delivery time of the group containing that job. A mathematical model and several meta-heuristic algorithms based on the particle swarm optimization (PSO) algorithm proposed to heuristically solve the problem. Behnamian and Zandieh (2013) proposed a novel hybrid meta-heuristic that hybridized the PSO, SA and variable neighborhood search (VNS) to solve the HFS scheduling with SDST and position-dependent learning effects. They considered objectives as tardiness and earliness penalties as objective function.

It can be seen that the GA and SA have been used in many studies to solve the HFS scheduling. According to the best of our knowledge, a method based on a hybrid of GA and SA has never been investigated in bi-objective HFS with SDST scheduling problems in the literature up to now. In this paper, a hybrid of GA proposed by Jungwattanakit et al. (2008), SA proposed by Mousavi et al. (2012) and local search is proposed to solve scheduling problem. The proposed algorithm and the details of it are explained in Section 2. The rest of the paper is organized as follows: Section 3 presents the computational results and numerical comparisons. Finally, Section 4 is devoted to conclusion and future works.

2. Proposed Hybrid Genetic Simulated Annealing

2.1. Genetic Algorithm and Simulated Annealing

A genetic algorithm developed by Holland (1975) is an iterative heuristic based on Darwin's evolutionary theory about "survival of the fittest and natural selection." Genetic algorithms are efficient, flexible, "intelligent" probabilistic search algorithms. They mimic the evolution process of biological organisms in nature. Genetic algorithms simulate the evolution process by generating an initial population of individuals (called chromosomes) and applying genetic operators on the fittest of those individuals in each reproduction cycle. A chromosome is represented by a string of numbers called genes. Each chromosome in the population is evaluated according to some fitness measure. Certain pairs of chromosomes (parents) are selected on the basis of their fitness. Each of these pairs combines to produce new chromosomes (offspring) and some of the offspring are modified. A new population is then formed replacing some of the original population by an identical number of offspring.

Simulated annealing is introduced to combinational optimization by Kirkpatrick (1983) in 1982. Simulated annealing is a neighborhood search approach designed to obtain a global optimum solution for combinatorial optimization problem. Simulated annealing starts with an initial solution and iteratively moves towards other existing solutions, while remembering the best solution found so far. In order to reduce the probability of getting trapped in local optima, simulated annealing accepts moves to inferior neighboring solution under the control of randomized scheme. More precisely, if a move from current solution S to another inferior neighboring solution S^* results in a change $\Delta E = f(S^*) - f(S)$ in the objective function value, the move is still accepted if $R < \exp(-\Delta E/T)$, where T is a control parameter, called temperature, and R is a uniform random number between interval (0, 1). Initially, the temperature T is high enough permitting many deteriorative moves to be accepted and it is lowered at a low speed of rate to a value so that inferior moves are approximately rejected. This algorithm investigates possible neighbors in each temperature sequentially and slowly in order to find the best solution.

2.2. The Proposed Algorithm

The proposed algorithm must seek to obtain effective and acceptable convex combination of objectives through the implementation of a simple method. Therefore, for solution$x$, the total objective function is given by Equation (1):

$$\begin{aligned} &\text{Total Objective Function} = \text{Minimizing } f(x) \\ &f(x) = \lambda \times f_1(x) + (1-\lambda) \times f_2(x) \\ &f_1(x) = \text{makespan } f_2(x) = \text{total tardiness } \ 0 \le \lambda \le 1 \end{aligned} \tag{1}$$

where λ values are the weighting coefficients representing the relative importance of makespan and the total tardiness

The proposed algorithm is structurally similar to the basic simulated annealing and genetic algorithm, but additional features have been proposed in the structure of algorithms. Now, the particular features of the proposed algorithm are described. In the initial simulated annealing, just a specific neighborhood search structure (NSS) used to generate a neighborhood solution. The NSS remains constant during the execution of algorithm. In this paper suggested to NSS will be randomly selected from among several different structures at each iteration procedure. This technique also has been proposed to generate offspring solutions through genetic operators. The important question can be expressed as follows: Why are several neighborhood search structure, crossover and mutation operators introduced to select among alternatives randomly in each generation?

In response to this question, the main reason of the application of this approach is that the algorithm is able to guide the search to another promising region through the types of moves (The algorithm is able to consider different search directions with the type of moves). This means that the features of several alternatives are applied to search space. Therefore, the performance of algorithm with cited characteristic can be better. This corresponded to the concept of the diversification.

Although the genetic algorithm has gained many applications, it is reported that the traditional genetic algorithm often suffers from the trouble of premature convergence. The proposed technique will help to avoid the premature convergence by the irregular selection among several alternatives.

The second feature is fitness function used in GA and SA. To prevail over the trap of dealing with different measurement sizes of objective values, we normalize the value of each

objective function by divided the minimum objectives with the actual objectives value. The normalized objectives can be obtained by Equation (2):

$$f_1'(x) = \frac{f_1}{f_1(x)} \text{ and } f_2'(x) = \frac{f_2}{f_2(x)} \quad (2)$$

f_1 and f_2 are the lowest observed of makespan and total tardiness values respectively, which can be updated after each iteration, therefore, the function of GA and SA are chosen from Equation (3):

$$\begin{cases} f'(x) = \left[\lambda \times f_1'(x) + (1-\lambda) \times f_2'(x)\right]^{-1} & \text{Used in SA} \\ f'(x) = \left[\lambda \times f_1'(x) + (1-\lambda) \times f_2'(x)\right] & \text{Used in GA} \end{cases} \quad (3)$$

The third feature is the improvement step that performed on the accepted solution in SA. The last feature is local search in the end of proposed algorithm. In this section, two local search procedures are applied on best solution archived to improve the final solution.

2.3. The Structure of the Proposed Algorithm

Encoding

A scheme using integers is applied to display a solution. For example, one solution of a hypothetical problem with five jobs as [3 1 2 5 4] denotes that job 3 is processed first, and then job 1, job 2, job 5, job 4 are processed successively. For determining the order of jobs, from the second stage to next, the first in first out (FIFO) rule has been used.

Initialization

- **Input parameters**: Initial temperature (T_0); Final temperature (T_f); Number of stages for reach of T_0 to T_f (N); The weighting coefficients ($\lambda \in \{0.25, 0.5, 0.75\}$); Number of initial population (np); Probability of crossover (P_c); Probability of mutation (P_m); Probability of reproduction (P_r).
- **Initialize an initial population randomly**
- **Evaluate** $f_1(x_i)$ **and** $f_2(x_i)$: Where $f_1(x_i)$ is the makespan, $f_2(x_i)$ is the total tardiness of solution ith in population.
- **Set** $T = T_0$; $it = 1$; $q = 1$; Archive(q) = {best solution in population}; $f_1 = \min\{f_1(x_i)\ i = 1, 2, .., np\}$, $f_2 = \min\{f_2(x_i)\ i = 1, 2, \ldots, np\}$.

While $T > T_f$
% ***Start parallel simulated annealing in temperature T***
For $i = 1$: np

Step 1: *yi* = Move *xi* by considering random integer so-called *K* in the range 1 to 4;

(i) Perform the swap moves (K = 1), random insertion scheme (K = 2), inversion moves (K = 3) or shift moves (K = 4) (Prandtstetter & Raidl, 2007) on x_i.
(ii) Evaluate $f_1(y_i)$ and $f_2(y_i)$ new solutions in the neighborhood of x_i.
(iii) Update f_1 and f_2 as $f_1 = \min\{f_1, f_1(y_i)\}, f_2 = \min\{f_2, f_2(y_i)\}$.

Step 2: Calculate f_{SA} and ΔEi as follows:

$$f_{SA}(x_i) = \left[\lambda \times \frac{f_1}{f_1(x_i)} + (1-\lambda) \times \frac{f_2}{f_2(x_i)}\right]^{-1} \text{ and}$$

$$f_{SA}(y_i) = \left[\lambda \times \frac{f_1}{f_1(y_i)} + (1-\lambda) \times \frac{f_2}{f_2(y_i)}\right]^{-1}$$

$$\Delta E_i = f_{SA}(y_i) - f_{SA}(x_i)$$

Step 3: Decision-making;

If $\Delta E_i < 0$ Then *%we accept the new solution*
$q = q + 1$; Archive(q) = $\{y_i\}$; $x_i = y_i$;
Else *%we accept the new solution with a certain probability*
If random<exp ($-\Delta E_i$ /T) Then
$x_i = y_i$;
Endif
Endif

Step 4: Improvement of accepted solution in previous step as follows:

(i) Generate (n-g) new solutions (called z_j) in the neighborhood of x_i with NSS in step 1.
(ii) Evaluate $f_1(z_j)$ and $f_2(z_j)$ new solutions in the neighborhood of x_i.
(iii) **Update** f_1 and f_2 as $f_1 = \min\{f_1, f_1(z_j); j = 1, 2, .., n\text{-}g\}$, $f_2 = \min\{f_2, f_2(z_j)\ j = 1, 2, ..., n\text{-}g\}$.
(vi) Accept solution with minimal $f_{SA}(z_j)$; $q = q + 1$; Archive(q) = $\{z_j\}$; $x_i = z_j$;

$$f_{SA}(z_j) = \left[\lambda \times \frac{f_1}{f_1(z_j)} + (1-\lambda) \times \frac{f_2}{f_2(z_j)}\right]^{-1} \quad j = 1, 2, ..., n-g$$

End for
% ***Start genetic algorithm in temperature T***

Step 1: Calculate the f_{GA} and *prob* of solutions in population as follows:

$$f_{GA}(x_i) = \left[\lambda \times \frac{f_1}{f_1(x_i)} + (1-\lambda) \times \frac{f_2}{f_2(x_i)}\right] \qquad i = 1, ..., np$$

$$prob(x_i) = \frac{f_{GA}(x_i)}{\sum_{i=1}^{np} f_{GA}(x_i)} \qquad i = 1, ..., np$$

Step 2: Crossover operator

(i) Select $np \times P_c$ pairs of parents based on roulette wheel selection;
(ii) Choose randomly integer number (K) in the range 1 to 5, and
(iii) Generate an offspring of 1PX (if K = 1), OPX (if K = 2), CX (if K = 3), OBX (if K = 4) or PBX (if K = 5).

Step 3: Mutation operator

(i) Select $np \times P_m$ chromosome based on purely random selection.
(ii) Choose randomly integer number (K), in the range 1 to 4, and
(iii) Perform the swap moves (K = 1), random insertion scheme (K = 2), inversion moves (K = 3) or shift moves (K = 4) on chromosome.

Step 4: Reproduction

Select $np \times P_r$ solutions from current population based on elitist selection.

Step 5: Replacement

(i) Combine solutions obtained from the previous steps (include steps 2, 3 and 4) as new population.
(ii) Evaluate $f_1(x_i)$ and $f_2(x_i)$ solutions in the population.
(iii) Update f_1 and f_2 as f_1=min $\{f_1, f_1(x_i)\ i = 1, 2, .., np\}$, f_2 = min $\{f_2, f_2(x_i)\ i = 1, 2, .., np\}$.

T **= temperature reduction by a linear schedule**
Endwhile

Local Search: The best solution in the archive (solution corresponding with minimal $f(x)$) is now subjected to two local search schemes, namely, neighborhood swapping (Prandtstetter & Raidl, 2007) and random insertion perturbation scheme (RIPS) (Prandtstetter & Raidl, 2007). Then, solution with minimal $f(x)$ is selected.

$$f(x) = \left[\lambda \times \frac{f_1}{f_1(x)} + (1-\lambda) \times \frac{f_2}{f_2(x)}\right]^{-1}$$

3. Computational Experiments

This section contains the method of generating data sets and run these data sets by proposed algorithm, and algorithm in the literature, performance criteria, and then expressing the results of the efficiency of the proposed algorithm.

3.1. Generation of a Test Problem

The numerical data should be created to test the performance of the algorithm. Data required for a problem consist of the range of processing times, range of setup times, number of stages (g), number of jobs (n), range in number of machines per stage and range of due date. Processing times are distributed uniformly over two ranges with a mean of 60: [50–70] and [20–100]. The setup times are uniformly distributed from 12 to 24, which are 20% to 40% of the mean of the processing time. We used problems with 15 jobs × 5 stages, 25 jobs × 10 stages, and 40 jobs × 20 stages. Numbers of machines are distributed uniformly over two ranges [1–4] and [1–10]. Due dates can be generated from a composite uniform distribution based on R and τ; with probability τ the due date is uniformly distributed over the interval $[\bar{d},\ \bar{d} + (C_{\max} - \bar{d})R]$ and with probability (1–τ) over the interval $[\bar{d},\ \bar{d} + (C_{\max} - \bar{d})R]$, where τ and R are two parameters called the tardiness factor ($\tau = 1 - \bar{d}/C_{\max}$) and the due date range ($R = (d_{\max} - d_{\min})/C_{\max}$), respectively. It should be noted that $d_{\max}$, $d_{\min}$ and $\bar{d}$ are maximum, minimum and average due date, respectively.

Values of τ close to 1 indicate that the due dates are tight, and values close to 0 indicate that the due dates are loose. A high value of R indicates a wide range of due dates, whereas a low value indicates a narrow range of due dates (Eren & Güner, 2008). The values of τ and R are taken as 0.2, 0.5 and 0.8. For each problem structure, data based on five different τ and R combinations are used: (0.2, 0.2), (0.2, 0.8), (0.5, 0.5), (0.8, 0.2), (0.8, 0.8).

3.2. Performance Criteria

The use of performance measures (or metrics) allows a researcher to assess (in a quantitative way) the performance of their algorithms. In this paper, four metrics are used to evaluate the quality of solutions. The metrics applied in this paper are described as follows:

The first and second metrics is computed a convex combination of objective functions. In fact, one of the views of decision-makers is to minimize a convex combination of objective functions. The definitions of the metrics are given as follows:

$$M_1 = \left[\lambda \times \frac{f_1}{f_1(x)} + (1-\lambda) \times \frac{f_2}{f_2(x)}\right]^{-1} \quad (4)$$

$$M_2 = \left[\lambda \times f_1(x) + (1-\lambda) \times f_2(x)\right] \quad (5)$$

The objectives are not normalized in second metric. Consequently, this criterion is sensitive to increase and decrease in objective function with a larger value. Lower values for these metrics (M_1 and M_2) represent better solution.

The decision-makers require schedules with respect to the trade-off between the various objectives. Figure 1 presents the acceptable trade-off between the objectives by angle (α). The angle of the solution (x), called M_3 is computed as given in Equation (6). According to this metric, M_3 in the interval 35 to 55 (45 ± 10) degrees is proper.

$$M_3 = \arctan\left(\frac{\frac{f_2(x)}{f_2}}{\frac{f_1(x)}{f_1}}\right) \quad (6)$$

This metric can be justified as follows:

A multi-objective optimization problem (MOP) differs from a single-objective optimization problem, because it contains several objectives that require optimization. A suitable solution should provide acceptable performance for all objectives. If there is a solution just along an individual axis (in one corner of the solution space), it won't be appropriate, because this solution is suitable only for an objective (similar to a single objective problem). This means that angles near zero and 90 degrees are not suitable in bi-objective problem. It is better to have solutions away from the angles 0 and 90. Note tolerance (10) can be any number from 0 to 45.

The hope of decision-makers is to find schedules close to the ideal point (0, 0). For this reason, the obtained solutions should converge towards the ideal point. Figure 2 represents convergence to the ideal point by distance. The distance between ideal point and the solution x, called M_4 is computed as given in Equation (7). Lower values of the fourth metric represent better solution.

$$M_4 = \sqrt{\left(\frac{f_1(x)}{f_1}\right)^2 + \left(\frac{f_2(x)}{f_2}\right)^2} \quad (7)$$

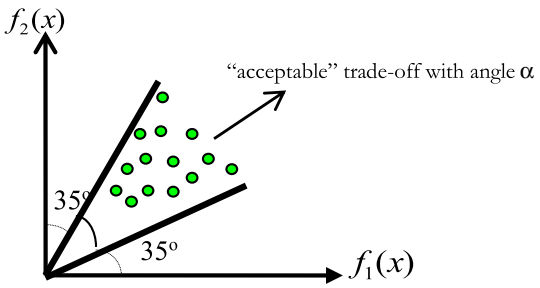

Figure 1. A Range of Angle as Acceptable Trade-off Between the Two Objectives.

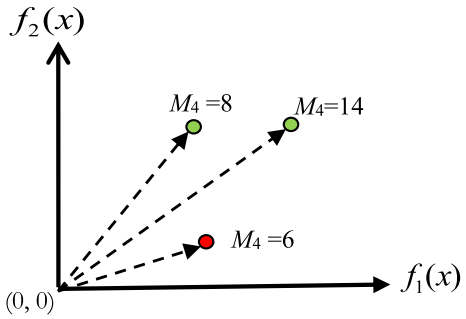

Figure 2. A Hypothetical Example of Distance.

Note the third and fourth metrics are complementary metrics. A solution is appropriate when results from both metrics are acceptable (close to the origin and with the desired angle).

3.3. Numerical Result

In this paper, a hybrid algorithm of GA (Jungwattanakit et al., 2008), SA (Mousavi et al., 2012) and local search is proposed to solve a scheduling problem. The aim is that the benefits of both effective algorithms are applied in the design of the proposed algorithm. The performance of the proposed HSA-GA-LS is compared with genetic algorithm proposed by Jungwattanakit et al. (2008). Now, the reasons of this choice are explained. According to the literature review, a variety of heuristics and meta-heuristics have been applied to find a convex combination of objectives. It is well known that algorithms are characterized by a parallel search (i.e. GA, PSO) or a point-by-point search (i.e. SA, TS) of the state space. In this research, method is proposed to handle problem by a parallel search. Consequently, a GA is selected to search space similar to HAS-GA-LS. It is noticeable that all of algorithms are implemented in MATLAB 2009a and run on a PC with 2.30 GHz Intel Core and 4 GB of RAM memory. To show the efficiency and effectiveness of the proposed algorithm in comparison with a GA, computational experiments were done on various test problems. The three replications for each problem size have been performed since there are some random conditions when applying the algorithm. The following parameters value has been used in tests: Initial temperature (T_0): 2.9; Final temperature (T_f): 0.05; Number of stages for reach of T_0 to T_f (N): 55; Number of initial population (np): 30; Probability of crossover (P_c): 0.80; Probability of mutation (P_m): 0.10; Probability of reproduction (P_r): 0.10. It is noted that these parameters are set according to Jungwattanakit et al. (2008) and Mousavi et al. (2012).

Time cost is an important factor when comparing different algorithms. In this paper, the running time for each problem is recorded by HSA-GA-LS. Then, these computational times are used in the GA as stopping criterion. Therefore, CPU time is almost the same for both algorithms (Figure 3).

Tables 1–3 represent the values of the four metrics for various problems. Based on the results of given in Tables, the following observations can be made. Due to M_1, M_2 and M_4 metrics, the proposed algorithm is able to outperform other algorithm on all problems. Due to M_3 metric, two algorithms

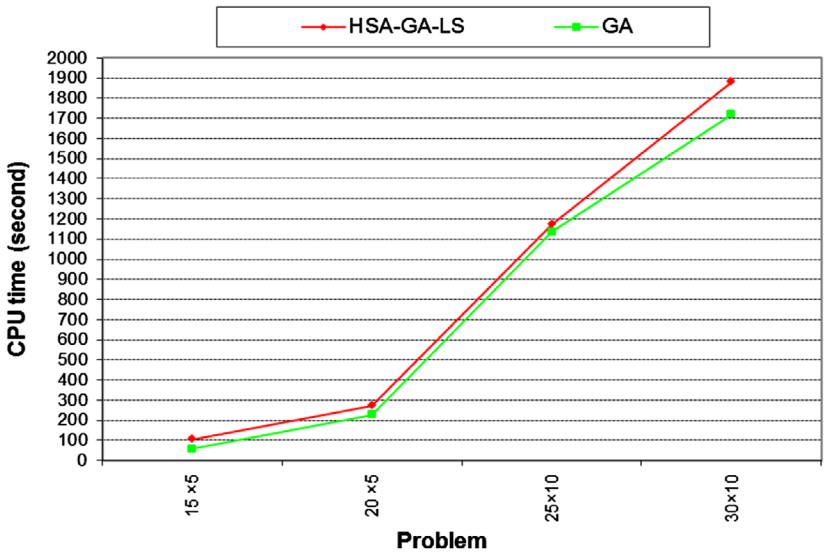

Figure 3. The Computational Times of HAS-GA-LS and GA.

Table 1. Results of Metrics for λ = 0.25 ($M_2 \times 10^3$).

Test problem				HSA-GA-LS				Genetic algorithm			
n	g	τ	R	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
15	5	0.2	0.2	0.5693	0.6383	44.2437	1.4145	0.5951	0.6710	47.2922	1.4932
		0.2	0.8	0.5907	0.4306	45.2554	1.4966	0.6267	0.4384	49.9702	1.6179
		0.5	0.5	0.5772	1.0057	44.9759	1.4397	0.5855	1.0281	46.0563	1.4633
		0.8	0.2	0.5732	5.3740	44.7476	1.4252	0.5751	5.4063	44.7660	1.4334
		0.8	0.8	0.5757	4.8990	45.1755	1.4304	0.5767	4.9046	44.8334	1.4394
20	5	0.2	0.2	0.5897	0.9605	46.0926	1.4817	0.6007	0.9800	48.0424	1.5114
		0.2	0.8	0.5739	0.8921	44.1122	1.4366	0.5745	0.8955	43.8985	1.4422
		0.5	0.5	0.5764	2.1361	44.8303	1.4381	0.5799	2.1627	45.8979	1.4404
		0.8	0.2	0.5727	9.0537	44.7095	1.4233	0.5747	9.0820	44.2597	1.4381
		0.8	0.8	0.5729	6.6178	44.4673	1.4275	0.5760	6.6955	44.8187	1.4365
25	10	0.2	0.2	0.5765	1.7405	45.0554	1.4358	0.5969	1.8404	47.0108	1.5043
		0.2	0.8	0.5734	6.6350	44.3840	1.4307	0.5953	7.1470	45.7634	1.5113
		0.5	0.5	0.5767	8.3525	44.9212	1.4382	0.5841	8.5450	44.9759	1.4702
		0.8	0.2	0.5744	16.448	44.9884	1.4273	0.5776	16.540	44.4118	1.4488
		0.8	0.8	0.5748	4.7460	43.8165	1.4444	0.5871	4.9471	44.7845	1.4864
30	10	0.2	0.2	0.5762	3.0012	44.2416	1.4450	0.6045	3.2775	47.4793	1.5342
		0.2	0.8	0.5744	6.7755	43.9286	1.4414	0.6011	7.4867	47.4602	1.5190
		0.5	0.5	0.5741	7.3899	44.0631	1.4379	0.6007	8.0954	45.9544	1.5339
		0.8	0.2	0.5735	21.4460	44.4774	1.4300	0.5812	21.9240	44.2671	1.4669
		0.8	0.8	0.5755	6.3420	43.7736	1.4484	0.6103	7.1991	47.9444	1.5568

Table 2. Results of Metrics for λ = 0.5 ($M_2 \times 10^3$).

Test problem				HSA-GA-LS				Genetic algorithm			
n	*g*	*τ*	R	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
15	5	0.2	0.2	0.6808	0.8950	46.0667	1.4604	0.6823	0.8972	46.1501	1.4653
		0.2	0.8	0.6752	0.7557	43.8869	1.4425	0.7318	0.7682	51.3944	1.6631
		0.5	0.5	0.6681	1.1309	44.8092	1.4189	0.6936	1.1886	47.1065	1.5048
		0.8	0.2	0.6699	4.0437	44.9040	1.4245	0.6719	4.0526	44.6958	1.4311
		0.8	0.8	0.6692	3.7141	44.7998	1.4223	0.6723	3.7369	44.7201	1.4323
20	5	0.2	0.2	0.6731	1.2187	45.0328	1.4349	0.7069	1.2761	49.3071	1.5595
		0.2	0.8	0.6726	1.1969	44.2331	1.4335	0.6796	1.1998	45.7451	1.4562
		0.5	0.5	0.6751	2.0311	44.8112	1.4413	0.6809	2.0674	46.6572	1.4619
		0.8	0.2	0.6706	6.6700	44.6903	1.4269	0.6757	6.7475	44.6994	1.4433
		0.8	0.8	0.6722	5.0681	45.0152	1.4320	0.6782	5.1256	44.8284	1.4513
25	10	0.2	0.2	0.6713	2.0423	44.3959	1.4294	0.6903	2.1124	46.9384	1.4934
		0.2	0.8	0.6726	5.3715	44.2270	1.4337	0.6843	5.5754	45.4342	1.4711
		0.5	0.5	0.6734	6.3925	44.7797	1.4357	0.6792	6.4950	45.0628	1.4544
		0.8	0.2	0.6695	11.8410	44.9420	1.4232	0.6717	11.8040	44.3489	1.4306
		0.8	0.8	0.6721	3.9911	44.5784	1.4317	0.6855	4.1171	44.7751	1.4749
30	10	0.2	0.2	0.6708	3.0248	44.4644	1.4276	0.6900	3.1638	46.4165	1.4913
		0.2	0.8	0.6713	5.6407	44.6292	1.4290	0.6968	6.0747	46.5921	1.5140
		0.5	0.5	0.7005	6.2640	46.9198	1.5272	0.7127	6.5235	48.2925	1.5736
		0.8	0.2	0.6708	15.5460	44.9971	1.4275	0.6823	15.6920	43.8312	1.4655
		0.8	0.8	0.6796	5.3127	45.6853	1.4561	0.7021	5.6332	46.7405	1.5321

Table 3. Results of Metrics for λ = 0.75 ($M_2 \times 10^3$).

Test problem				HSA-GA-LS				Genetic algorithm			
n	*g*	*τ*	R	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
15	5	0.2	0.2	0.8116	1.1366	45.3140	1.4440	0.8217	1.1421	47.4122	1.4972
		0.2	0.8	0.8220	1.0875	43.6934	1.4476	0.8815	1.1035	54.1699	1.7972
		0.5	0.5	0.8113	1.2511	45.7799	1.4495	0.8249	1.2729	47.6848	1.5091
		0.8	0.2	0.8031	2.6920	44.8177	1.4188	0.8057	2.7085	44.9806	1.4265
		0.8	0.8	0.8034	2.5375	45.0958	1.4230	0.8079	2.5686	45.5147	1.4384
20	5	0.2	0.2	0.8122	1.5006	45.4764	1.4474	0.8229	1.5184	46.5414	1.4865
		0.2	0.8	0.8156	1.4941	45.6829	1.4577	0.8496	1.5348	51.8557	1.6528
		0.5	0.5	0.8173	1.9256	45.7909	1.4630	0.8223	1.9406	45.7967	1.4745
		0.8	0.2	0.8073	4.2687	45.8452	1.4413	0.8159	4.2553	44.7727	1.4467
		0.8	0.8	0.8085	3.4067	44.5144	1.4270	0.8176	3.4729	44.9223	1.4524
25	10	0.2	0.2	0.8044	2.2424	44.7360	1.4208	0.8347	2.3321	48.5797	1.5474
		0.2	0.8	0.8137	3.9914	45.1167	1.4461	0.8345	4.1270	45.2713	1.4949
		0.5	0.5	0.8089	4.4305	44.7822	1.4313	0.8234	4.5862	45.8302	1.4774
		0.8	0.2	0.8041	7.1763	45.3976	1.4282	0.8085	7.1780	45.0148	1.4332
		0.8	0.8	0.8123	3.2759	44.3183	1.4332	0.8120	3.2875	44.9728	1.4404
30	10	0.2	0.2	0.8210	3.0158	45.5817	1.4686	0.8386	3.1125	48.5993	1.5569
		0.2	0.8	0.8028	4.3688	45.0336	1.4208	0.8181	4.5001	45.6885	1.4634
		0.5	0.5	0.8173	4.7662	47.2930	1.4852	0.8390	4.9725	48.1181	1.5493
		0.8	0.2	0.8076	9.3050	44.9189	1.4300	0.8216	9.3487	43.9698	1.4499
		0.8	0.8	0.8087	4.2870	44.8112	1.4312	0.8279	4.4689	46.1834	1.4927

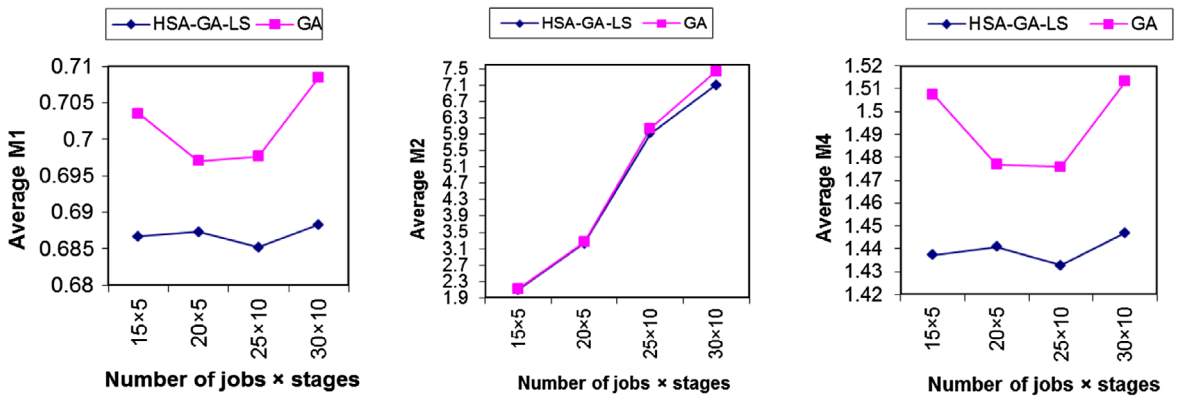

Figure 4. Metrics Plot Increasing the Number of Jobs and Stages.

are suitable. It means that solutions obtained of the algorithms are in the trade-off surface. These results show that the proposed algorithm works effectively in all size of problems.

To demonstrate the behavior of the algorithms in the different situations, the average M1, M2 and M4 of the algorithms is plotted in the different levels of the number of jobs and stages (Figure 4). It is observed in this figure that, the HSA-GA-LS algorithm has better performance regarding increasing the number of jobs and stages than genetic algorithm in the all cases (15 × 5, 20 × 5, 25 × 10 and 30 × 10).

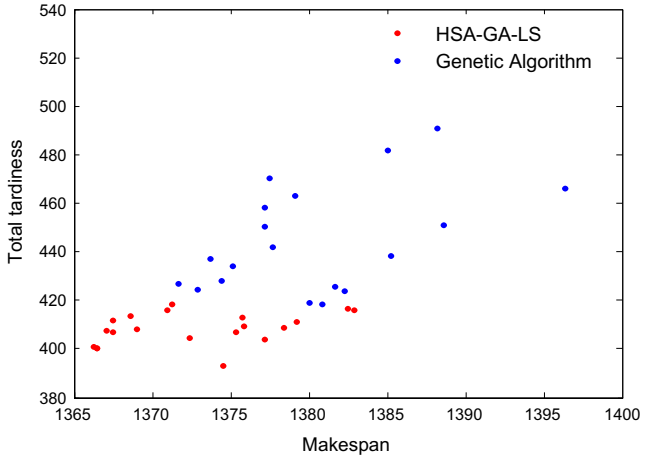

Figure 5. Convex Combination of Algorithms for Problems with 15 Jobs × 5 Stages.

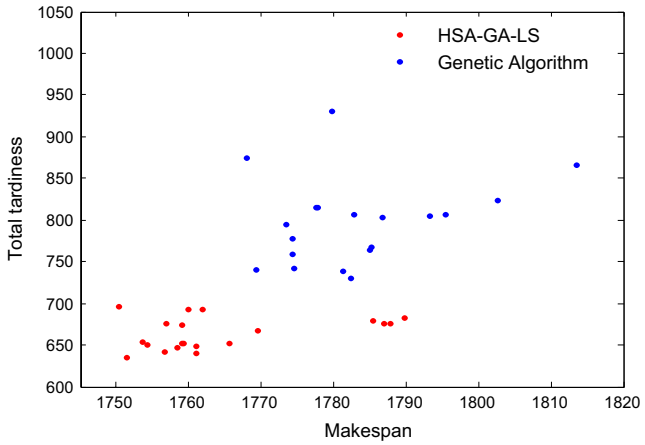

Figure 6. Convex Combination of Algorithms for Problems with 20 Jobs × 5 Stages.

Graphical representation is provided to demonstrate output results of the HSA-GA-LS and genetic algorithm (Figs. 5 and 6). These figures show the obtained solutions (convex combination) of HSA-GA-LS and genetic algorithm over twenty runs for problem with 15 jobs × 5 stages and 20 jobs × 5 stages. It is observed in these figures that obtained solutions of genetic algorithm have the trade-off between the various objectives (M3 metric), but other metrics (M_1, M_2 and M_4 metrics) are not satisfied. These figures illustrate and confirm the conclusion derived from the numerical results based on the performance criteria. Therefore, the proposed algorithm is effective in minimizing makespan and total tardiness for the hybrid flow shop problem with sequence-dependent setup times.

4. Conclusion and Future Work

This paper considers the HFS sequence-dependent job setup times scheduling problem. Our objective is to determine a schedule that minimizes a convex combination of makespan and the total tardiness. The hybrid method is applied to solve this problem, which belongs to NP-hard class. Several computational tests are used to evaluate the effectiveness and efficiency of the proposed algorithm in finding good quality schedules. Computational results show that the proposed algorithm provides better results than genetic algorithm in the literature. For future study, the scheduling with other system characteristics, which have not been included in this paper, such as release date, limited intermediate buffers, machine availability constraints, and unrelated parallel machines at each stage can be a practical extension, although the problem would be very difficult to solve.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Sayyed Mostafa Mousavi obtained a B.Sc. in Industrial Engineering at University of Science and Industry, Behshar Branch, Iran (1999–2003), and an M.Sc. in Industrial Engineering at Islamic Azad University, Qazvin branch, Iran (2006–2008). He obtained a Ph.D. in Industrial Engineering from Mazandaran University of Science and Technology, Iran (2011–2016). Currently, he is an assistant professor at Industrial Engineering Department, Islamic Azad University, Noshahr Brach, Iran. His research interests are production planning and scheduling, applied operations research.

Mostafa Zandieh accomplished a B.Sc. in Industrial Engineering at Amirkabir University of Technology, Tehran, Iran (1994–1998), and an M.Sc. in Industrial Engineering at Sharif University of Technology, Tehran, Iran (1998–2000). He obtained a Ph.D. in Industrial Engineering from Amirkabir University of Technology, Tehran, Iran (2000–2006). Currently, he is an associate professor at Industrial Management Department, Shahid Beheshti University, Tehran, Iran. His research interests are production planning and scheduling, financial engineering, quality engineering, applied operations research, simulation, and artificial intelligence techniques in the areas of manufacturing systems design.

References

Allahverdi, A., Gupta, J.N.D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega, 27*, 219–239.

Allahverdi, A., Ng, C.T., Cheng, T.C.E., & Kovalyov, M.Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*, 985–1032.

Andrés, C., Albarracín, J.M., Tormo, G., Vicens, E., & García-Sabater, J.P. (2005). Group technology in a hybrid flowshop environment: A case study. *European Journal of Operational Research, 167*, 272–281.

Behnamian, J., & Zandieh, M. (2013). Earliness and tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic. *Arabian Journal for Science and Engineering, 38*, 1229–1242.

Chang, P.C., Hsieh, J.C., & Wang, Y.W. (2003). Genetic algorithms applied in BOPP film scheduling problems: minimizing total absolute deviation and setup times. *Applied Soft Computing, 3*, 139–148.

Davoudpour, H., & Ashrafi, M. (2009). Solving multi-objective SDST flexible flow shop using GRASP algorithm. *The International Journal of Advanced Manufacturing Technology, 44*, 737–747.

Eren, T., & Güner, E. (2008). The tricriteria flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 36*, 1210–1220.

Holland, J.H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, Michigan: University of Michigan.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *The International Journal of Advanced Manufacturing Technology, 37*, 354–370.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research, 36*, 358–378.

Kirkpatrick, S. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Kurz, M.E., & Askin, R.G. (2003). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research, 159*, 66–82.

Mousavi, S.M., Zandieh, M., & Yazdani, M. (2012). A simulated annealing/ local search to minimize the makespan and total tardiness on a hybrid flowshop. *International Journal of Advanced Manufacturing Technology, 64*, 369–388.

Naderi, B., Zandieh, M., & Roshanaei, V. (2008). Scheduling hybrid flow shops with sequence dependent setup times to minimize makespan and maximum tardiness. *International Journal of Advanced Manufacturing Technology, 41*, 1186–1198.

Pargar, F., & Zandieh, M. (2012). Bi-criteria SDST hybrid flow shop scheduling with learning effect of setup times: Water flow-like algorithm approach. *International Journal of Production Research, 50*, 2609–2623.

Prandtstetter, M., & Raidl, G.R. (2007). An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research, 191*, 1004–1022.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*, 781–800.

Ruiz, R., & Vázquez-Rodríguez, J.A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*, 1–18.

Sheikh, S. (2013). Multi-objective flexible flow lines with due window, time lag, and job rejection. *The International Journal of Advanced Manufacturing Technology, 64*, 1423–1433.

Tadayon, B., & Salmasi, N. (2013). A two-criteria objective function flexible flowshop scheduling problem with machine eligibility constraint. *The International Journal of Advanced Manufacturing Technology, 64*, 1001–1015.