Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 976065, 7 pages
http://dx.doi.org/10.1155/2013/976065

Research Article

Inventory Based Bi-Objective Flow Shop Scheduling Model and Its Hybrid Genetic Algorithm

Ren Qing-dao-er-ji and Yuping Wang

School of Science, School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Yuping Wang; wyp9000@yahoo.com.cn

Received 31 August 2012; Accepted 5 March 2013

Academic Editor: Alexander Pogromsky

Flow shop scheduling problem is a typical NP-hard problem, and the researchers have established many different multi-objective models for this problem, but none of these models have taken the inventory capacity into account. In this paper, an inventory based bi-objective flow shop scheduling model was proposed, in which both the total completion time and the inventory capacity were as objectives to be optimized simultaneously. To solve the proposed model more effectively, we used a tailor-made crossover operator, and mutation operator, and designed a new local search operator, which can improve the local search ability of GA greatly. Based on all these, a hybrid genetic algorithm was proposed. The computer simulations were made on a set of benchmark problems, and the results indicated the effectiveness of the proposed algorithm.

1. Introduction

Scheduling is a major issue faced everyday in manufacturing systems and industrial settings, so it is significant to develop effective and efficient advanced manufacturing and scheduling technologies and approaches. The first research on the flow shop scheduling problem (FSSP) was completed by Johnson [1]. Johnson developed an exact algorithm to minimize makespan for the n-jobs and 2-machines FSSP. With the increase in the number of jobs and machines, FSSP becomes a strong NP-complete combinatorial optimization problem [2]. Most researches for single objective FSSPs result in a schedule to minimize the time required to complete all jobs, that is, to minimize the makespan. Historically, FSSP has been primarily treated by exact methods [3], such as linear programming, branch and bound, and Lagrangian relaxation. Recently, meta-heuristics have gained a wide attention [4], including the topics such as genetic algorithms (GAs), simulated annealing (SA), ant colony optimization (ACO), particle swarm optimization (PSO), tabu search (TS), and differential evolution (DE).

However, in many real world situations, several objectives must be considered. The purely single objective FSSP cannot totally reflect the needs of the modern practical applications, that is to say, multiple objectives must be optimized simultaneously. It is important to find out the trade-off solutions among these objectives, particularly in conflicting situations. So far, few researchers have attempted to tackle the multi-objective FSSPs (MOFSSPs). The most commonly used method in the current study of MOFSSPs is constant weight weighting method. It assigns constant weights to objectives according to the importance of each objective. By this way, an MOFSSP is transformed into a single objective FSSP through a linear combination of objectives and weights. The following works are examples of the representative works of this approach: Ravindran et al. [5] proposed some multi-criterion approaches to FSSP by considering makespan time and total flow time. Eren and Güner [6] considered a bicriteria scheduling problem with sequence-dependent setup times on a single machine. The objective function of the problem is the minimization of the weighted sum of total completion time and total tardiness.

Lemesre et al. [7] proposed a parallel exact method to solve bi-objective permutation flow shop problem. Tseng and Liao [8] considered an n-job, m-machine lot-streaming problem in a flow shop with equal-size sublots where the objective is to minimize the total weighted sum of earliness and tardiness. Naderi et al. [9] applied a novel simulated annealing to hybrid FSSP to minimize both total completion time and total tardiness. The major disadvantage of this transformation is that the weight of each objective must be given subjectively in advance. It results in great gaps between the real and expected solutions if the decision maker is not experienced enough. An alternative method used to solve the MOFSSP does not specify the weights in advance. It provides various solutions to the decision makers for reference. These solutions are called Pareto optimal solutions. The number of Pareto optimal solutions is often not sole, and decision makers can select one of the solutions to meet their demands. It is a more flexible way than the constant weight weighting method. The representative works of this approach include the following: Qian et al. [10] proposed an effective differential evolution based hybrid algorithm for MOFSSP. Pasupathy et al. [11] proposed a multi-objective genetic algorithm named Pareto GA with an archive of nondominated solutions subjected to a local search. Melab et al. [12] proposed a grid-based parallel GA with the aim of obtaining an accurate Pareto front. They focused on FSSP with the minimization of the makespan and the total tardiness criteria. Tavakkoli-Moghaddam et al. [13] investigated a novel multi-objective model for a no-wait FSSP that minimizes both the weighted mean completion time and weighted mean tardiness. Li and Wang [14] solved the MOFSSP using a hybrid quantum-inspired GA. Chang et al. [15] presented a mining gene structures on subpopulation genetic algorithm which is combined with mining gene structure approach and subpopulation genetic algorithm. Dugardin et al. [16] focused on the multi-objective resolution of a reentrant hybrid FSSP. The two objectives are the maximization of the utilization rate of the bottleneck and the minimization of the maximum completion time. Karimi et al. [17] presented a multiphase approach to tackle hybrid flexible FSSP considering the minimization of makespan and total weighted tardiness simultaneously. Geiger [18] described the proposition and application of a local search metaheuristic for MOFSSP. Chiang et al. [19] considered the makespan and the total flow time as objectives and proposed a memetic algorithm to solve the FSSP. Dubois-Lacoste et al. [20] presented a new, carefully designed algorithm for five bi-objective permutation FSSPs that arise from the pair wise combinations of the objectives: makespan, the sum of the completion times of the jobs, and the weighted and nonweighted total tardiness of all jobs. Cho et al. [21] dealt with a scheduling problem for reentrant hybrid flow shop with serial stages where each stage consists of identical parallel machines.

The previous literature shows that the researchers have established many different multi-objective models for FSSP by considering different factors, but none of these models have taken the inventory capacity into account. The completion time, tardiness, flow time, lateness, earliness, and the number of tardy jobs are often used as the objectives in MOFSSP. Therefore, we considered both the inventory capacity and the completion situation of the jobs, established an inventory based MOFSSP model in this paper. To solve the proposed MOFSSP model more effectively, we used some tailor-made genetic operators and designed a new local search operator. Based on these genetic operators, we proposed a hybrid genetic algorithm (HGA). Finally, the efficiency of the proposed algorithm was verified by computer simulations on some typical FSSPs.

The remainder of the paper is organized as follows. In Section 2, we introduce the concepts of multi-objective problems. Then, we describe MOFSSP and its model in Section 3. In Section 4, a hybrid genetic algorithm to the MOFSSP is presented. Section 5 presents the experimental results. The conclusions are made in Section 6.

2. Multi-Objective Optimization and Pareto Optimality

Usually, a k objective multi-objective optimization problem can be described as follows [22]:

$$\begin{aligned} &\text{minimize} \quad y = f(X) = \left[f_1(X), f_2(X), \ldots, f_k(X)\right], \\ &\text{subject to} \quad g_i(X) \geq 0 \quad i = 1, 2, \ldots D, \end{aligned} \tag{1}$$

where $X = (x_1, x_2, \ldots, x_n)^T$ is the decision vector, $X \in \Theta \subset R^n$, Θ is the search space. Y is the objective space, and $y \in Y$ is the objective vector. g_i, $i = 1, 2, \ldots D$ is a constraint function.

The following concepts are often used in multi-objective optimization problems.

Definition 1. Let a decision vector $X_1 \in \Theta$.

(1) X_1 is said to dominate a decision vector $X_2 \in \Theta (X_1 \prec X_2)$ if and only if $f_i(X_1) \leq f_i(X_2)$ $i = 1, 2, \ldots, k$, and $\exists i \in \{1, 2, \ldots, k\}$ satisfying $f_i(X_1) < f_i(X_2)$.

(2) X_1 is said to be Pareto optimal if and only if $\neg \exists X_2 \in \Theta$ satisfying $X_2 \prec X_1$.

(3) $P_S = \{X_1 \in \Theta \mid \neg \exists X_2 \in \Theta$ satisfying $X_2 \prec X_1\}$ is said to be Pareto optimal set of all Pareto optimal decision vectors.

(4) $P_F = \{f(X) = (f_1(X), f_2(X), \ldots, f_k(X)) \mid X \in P_S\}$ is said to be Pareto optimal front of all objective function values corresponding to the decision vectors in P_S.

(5) X_1 is said to be non-dominated regarding a given set if X_1 is not dominated by any decision vectors in the set.

Pareto optimal decision vector cannot be improved in any objectives without causing degradation in at least one other objective. When a decision vector is non-dominated on the whole search space, it is Pareto optimal.

3. Problem Definition and Mathematical Modeling

In classical FSSP [23], we have a set of n jobs and a set of m machines. Each job consists of m operations that have to be processed in a specified sequence. All jobs visit machines in the same processing route, starting from machine 1 until finishing on machine m. The operation of job i on machine j lasts for a fixed and predetermined amount of time, denoted by $t_{i,j}$. Other assumptions commonly characterized for FSSP are as follows.

(i) Each machine can only process one operation at a time, and each job can be processed by one machine at a time.

(ii) A job can visit a machine once and only once, and preemption of operations is not allowed.

(iii) There are no precedence constraints among the operations of different jobs.

(iv) Neither release times nor due dates are specified, and setup times are included in processing times.

(v) There is no machine failure; therefore, machines are always available.

The objective of FSSP is to find the optimal schedule, that is, the schedule of the operation sequences and starting time on each machine so that one or more given criteria are optimized.

In the most of previous MOFSSPs, the completion time, flow time, lateness, earliness, tardiness, and the number of tardy jobs are often used as the optimization criteria. These objectives are established based on the completion situation of the jobs and are almost not taking the inventory capacity into account. As we all know, the completion situation of each job is the most concerned problem in FSSP, but the inventory capacity is also an important issue which should not be ignored. Especially for those factories with high inventory costs but very low transportation costs, if the volume (or size) of each operation is large, then we need a very large depot to put all the operations, and this will increase inventory cost. In order to reduce the inventory cost, we can try to use as little space of depot as possible by delivering the operations required in batches to the depot and transported them away after process. Of course, the operations are different for different schedules at a fixed time, and what we need to do is finding a reasonable scheduling scheme with the inventory capacity needed as small as possible. Thus, we can construct an inventory based MOFSSP model to minimize the makespan and the inventory capacity simultaneously.

Given a feasible schedule, we can know the operation sequences and the completion time on each machine. We introduce several notations at first:

(i) n: the total number of jobs to be scheduled;

(ii) m: the total number of machines in the process;

(iii) $t_{i,j}$: the processing time for job i on machine j;

(iv) $C_{i,j}$: the completion time for job i on machine j;

(v) $S = (s_{i,j})_{m\times n}$: the processing sequence matrix of all jobs, where $(s_{i,j})$ means the jth processed job number on machine i.

(vi) W_{ij}: the volume of the operation for job i on machine j.

We use $C_{\max} = C_{s_{m,n},m}$ to represent the maximum makespan and $S_{\max} = \max_{0\le p\le C_{\max}} \sum_{i=1}^{n}\sum_{j=1}^{m}(N_{ijp}\times W_{ij})$ to represent the maximum inventory capacity needed, where

$$N_{ijp} = \begin{cases} 1, & C_{ij} - t_{ij} = p \\ 0, & C_{ij} - t_{ij} \neq p, \end{cases} \quad (2)$$
$$i = 1, 2, \ldots, n,\ \ j = 1, 2, \ldots, m,\ \ p = 0, 1, \ldots C_{\max}.$$

Then, a two objective optimization model for FSSP can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & f\left(\vec{X}\right) = \left[C_{\max}, S_{\max}\right], \\ \text{subject to} \quad & C_{s_{1,1},1} = t_{s_{1,1},1} \\ & C_{s_{1,i},1} = C_{s_{1,i-1},1} + t_{s_{1,i},1}, \quad i = 2, \ldots, n \\ & C_{s_{j,1},j} = C_{s_{j,1},j-1} + t_{s_{j,1},j}, \ \ j = 2, \ldots, m \\ & C_{s_{j,i},j} = \max\left\{C_{s_{j,i-1},j}, C_{s_{j,i},j-1}\right\} + t_{s_{j,i},j}, \\ & i = 2, \ldots, n;\ \ j = 2, \ldots, m, \end{aligned} \quad (3)$$

where (3) is to minimize a vector of objective functions, in which $\vec{X}$ is a feasible schedule.

4. A Hybrid Genetic Algorithm for MOFSSP

Genetic algorithms were proposed by Holland [24] and have been successfully used in a variety of combinatorial optimization problems. In general, GA works by keeping a fixed number of candidate individuals (population). Each corresponds to a solution of the problem. To describe or formulate a solution, an appropriate representation of individual must be defined, which encodes a set of genes and joins them together to form a string of values. In each generation of GA, a fitness function is required to evaluate the fitness value of each individual. Then, a scheme which favors fitter individuals is used to randomly select parents from the population to form the mating pool. Next, crossover and mutation operators are used to generate better offspring. Furthermore, the offspring are inserted into the current population to form new population for next generation. The process is iterated until a specified stopping criterion is reached.

Our proposed hybrid genetic algorithm adopts the general structure described earlier. We will explain the details of our implementation of the genetic algorithm for the MOFSSP in the following.

4.1. Encoding and Decoding. Defining an appropriate encoding scheme is one of the most important and critical issues for constructing an efficient GA. The following encoding

scheme proposed by [25] is adopted in this paper: a chromosome is composed of m substrings, corresponding to m different machines. Each substring represents the sequence of the operations processed on each machine, denoted by a permutation of integers from 1 to n. The total length of a chromosome is $n \times m$. Consider a 4-job and 3-machine FSSP as an example. Suppose a chromosome is: [(2 1 4 3) (3 4 1 2) (2 1 3 4)], then the substrings (2 1 4 3), (3 4 1 2), (2 1 3 4) represent, respectively, the sequences of the operations processed on machine 1, 2, and 3.

4.2. Fitness Function and Selection Operation. The fitness function is defined as follows in this study.

Suppose the current population set is $P(t)$. Each individual $X_i \in P(t)$ is compared with all other individuals in $P(t)$ first and then we get a set $C(X_i) = \{X_j \mid X_j \in P(t)(X_j \succ X_i)\}$. Let $f(X_i) = |C(X_i)|$, where $f(X_i)$ is the fitness value of individual $X_i \in P(t)$, and $|C(X_i)|$ represents the number of the individuals in $C(X_i)$.

Suppose N is the population size and $f(X_i)$ means the fitness value of individual $X_i \in P(t)$, then the selection probability of X_i is defined by $p(X_i) = f(X_i)/\sum_{k=1}^{N} f(X_k)$.

4.3. The Crossover Operator. Information is swapped among the chromosomes presenting in the mating pool to create new chromosomes in crossover operator. A crossover operator proposed by [26] is adopted in this paper. Suppose that x, y are two parent individuals, the crossover operator can be described as follows.

Algorithm 2 (crossover operator).

Step 1. Divide the machines $1, 2, \ldots, m$ into two complementary sets A and B.

Step 2. For two parent individuals x and y exchange the genes in the substrings that belong to set A with probability of p_c and get two children x' and y'.

Exchanging the genes in the substrings that belong to set A is equivalent to exchanging the sequences of the operations processed on machines which belong to set A. In this way, the children can effectively inherit the sequences of the operations processed on machines.

From the proposed crossover operator, we can see that this operator is easy to implement, and the children can inherit the sequences of the operations processed on some machines of their parents.

4.4. The Mutation Operator. The mutation operator avoids convergence to local optima solution and diversifies the search directions.

We use a mutation operator designed in [27]. It can be described as follows. Given chromosome x, mutation generates y by the following procedure:

Algorithm 3 (a mutation operator based on the critical path).

Step 1. Calculate the critical path of individual x.

Step 2. Swapping two successive operations v and w processed on the same machine with probability p_m, where v and w are two successive operations on a critical path of parent 1. Then, we can get the offspring, denoted by y.

As can be seen from [28], for any feasible solution x, it is possible to construct a finite sequence of mutation operations, which will lead x to a globally minimal solution. Thus, the mutation operator used in this paper improved the possibility of production of the excellent individuals and ensured the convergence of the algorithm.

4.5. A New Local Search Operator. To improve the convergence speed of the algorithm, a new local search operator is designed in this paper. For individual x, suppose x' is the final offspring through our local search operator, and k is the number of the new solutions to be generated by our local search operator.

Algorithm 4 (a new local search operator).

Step 1. Let $i = 1$, $N_\varepsilon = \phi$ (ϕ represents an empty set).

Step 2. If $i \leq \kappa$, go to Step 3, else, go to Step 5.

Step 3. Randomly select a substring of x, randomly generate a sequence of the selected substring and get a new solution N_i.

Step 4. Put solution $N_i$into set N_ε, $i = i + 1$, and then turn to Step 2.

Step 5. Choose the best individual from N_ε as offspring x'.

4.6. The Framework of the HGA. We proposed a hybrid genetic algorithm (briefly HGA) to solve the MOFSSP based on aforementioned genetic operators. The framework of the HGA can be described as follows.

Algorithm 5 (HGA).

Step 1. Generate initial population $P(0)$, and let $t = 0$. Calculate the fitness value of each individual in $P(0)$.

Step 2. Randomly select two individuals from $P(t)$ and use Algorithm 2 to get two offsprings. Repeat the above procedure $N/2$ times and get N offspring, denoted the set of these offspring by $P'(t)$.

Step 3. For each individuals in $P'(t)$, use Algorithm 3 to get a set of offspring, denoted by $P''(t)$.

Step 4. Calculate the fitness value of each individual in $P''(t)$.

Step 5. For each individual in $P''(t)$, use Algorithm 4 with probability p_l and get a set of offspring, denoted by $P'''(t)$.

Step 6. Select N individuals from $P'''(t)$ to get a tentative population, still denoted as $P'''(t)$, then use the elitist strategy

to the union of $P'''(t)$ and $P(t)$ to get the next generation population $P(t+1)$.

Step 7. If the stop criterion is satisfied, then stop. Otherwise, let $t = t + 1$, and turn to Step 2.

5. Simulation Results

5.1. Algorithm Description and Test Problems. We selected several benchmarks with different scales to test the performances of HGA in this Section. Benchmarks Rec01-Rec41 are selected from [29]. The proposed HGA was compared with a non-dominated sorting multi-objective evolutionary algorithm, called NSGA-II reported in [30], which is recently used by many researchers to illustrate the superiority of their algorithms.

5.2. Measures for Comparing Nondominated Sets. Multiple and uniformly distributed solutions are required to form a Pareto front in multi-objective problems. It is not easy to compare the non-dominated solution sets between two algorithms. The following three comparison metrics are considered in this paper [31]. Suppose algorithm j, $j = 1, 2$ obtains a non-dominated solution set S_j. The union of two non-dominated solution sets (i.e., $S = S_1 \cup S_2$) is denoted by S.

(1) The number of the obtained non-dominated solutions can be defined as

$$\mathrm{NDSN}\left(S_j\right) = \left|S_j - \left\{x \in S_j \mid \exists y \in S : y \prec x\right\}\right|. \quad (4)$$

The larger the value NDSN(S_j) is, the better the solution set S_j is.

(2) The distance of the obtained non-dominated front to Pareto front (if known) can be defined as

$$\mathrm{NDSAD}\left(S_j\right) = \frac{\sum_{i=1}^{|S_j|} D\left(x_i, x^*\right)}{\left|S_j\right|}. \quad (5)$$

NDSAD(S_j) denotes the average distance from each solution x_i ($x_i \in S_j$) to an ideal solution x^*, where $D(x_i, x^*)$ is the distance between x_i and an ideal solution x^* in the objective space,

$$D\left(x_i, x^*\right) = \sqrt{\left(f_1\left(x_i\right) - f_1\left(x^*\right)\right)^2 + \left(f_2\left(x_i\right) - f_2\left(x^*\right)\right)^2}. \quad (6)$$

Since x^* is unknown, $D(x_i, x^*)$ cannot be directly measured. We use an approximated value $f_1(x^*) = \min_j f_1(x_j)$ and $f_2(x^*) = \min_j f_2(x_j)$ to replace $f_1(x^*)$ and $f_2(x^*)$. Obviously, the smaller the value NDSAD(S_j) is, the better the solution set S_j is.

(3) The spread and distribution of the obtained non-dominated solutions can be defined as

$$\mathrm{SM}\left(S_j\right) = \sqrt{\frac{\sum_{i=1}^{n}\left(d_i - \overline{d}\right)}{(n-1)}}, \quad (7)$$
$$d_i = \min_j\left(\left|f_1^i(X) - f_1^j(X)\right| + \left|f_2^i(X) - f_2^j(X)\right|\right),$$

where $j = 1, 2, \ldots n$ and $i \neq j$. $\overline{d}$ are the mean value of all d_i, and n is the number of solutions in set S_j.

5.3. Test Results and Discussions. The algorithms are implemented in Visual C++ on a computer with a 3.20 GHz Pentium(R) Dual-Core CPU. The parameters used in simulations are as follows: for both the NSGA-II and HGA, the population size $N = 100$, the crossover probability $p_c = 0.7$, and the mutation probability $p_m = 0.1$. For HGA, the local search probability $p_l = 0.5$ and $\sigma = 10$. For all benchmarks, suppose the volume of operation O_{ik} is $W_{ik} = i$, where O_{ij} represents job i when processed on machine j, $i = 1, 2, \ldots n$, $j = 1, 2, \ldots m$. The algorithms were run 10 independent runs on each benchmark. The proposed HGA is applied to a number of benchmarks, and its performance is compared with NSGA-II.

Table 1 lists the values of metrics NDSN(S_j), NDSAD(S_j), and SM(S_j) obtained by HGA and NSGA-II, respectively. From the values of NDSN(S_j), we can know that the set of the non-dominated solutions obtained by NSGA-II are dominated by that obtained by HGA for all test problems. It indicates that HGA tends to find non-dominated solutions with higher quality than NSGA-II. From the values of NDSAD(S_j), we can know that the values of NDSAD(S_j) obtained by HGA are smaller than those of NSGA-II for all the test problems. It shows that the distance between the Pareto front and the non-dominated solutions obtained by HGA is smaller than those by NSGA-II. From Table 1 we also can know that the values of SM(S_j) obtained by HGA are smaller than those of NSGA-II for all the test problems. It shows that uniformity of the distribution of the non-dominated solutions obtained by HGA is better than that of NSGA-II.

By the above comparison, we can know that both the quality and the distribution of non-dominated solutions obtained by our algorithm are better than those by NSGA-II. This shows that the genetic operator designed in this paper is effective, and HGA performs better than NSGA-II.

6. Conclusions

We considered both the completion time of the jobs and the inventory capacity as objectives and constructed an inventory based MOFSSP model in this paper. To solve the proposed model, a local search operator was designed in order to improve the quality of the solutions. Based on these, a hybrid genetic algorithm was proposed. The experimental results show that the proposed algorithm is effective and performs better than the compared algorithm.

Table 1: Comparisons of metrics NDSN(S_j), NDSAD(S_j), and SM $\left(S_j\right)$.

Problem	Size	$f_1(x^*)$	$f_2(x^*)$	NDSN (S_j)		NDSAD (S_j)		SM $\left(S_j\right)$	
				NSGA-II	HGA	NSGA-II	HGA	NSGA-II	HGA
Rec01	20 × 5	1245	20	0	11	69.5866	23.6526	15.0548	6.3485
Rec03	20 × 5	1093	20	0	16	89.6586	34.2652	9.1562	4.2596
Rec05	20 × 5	1228	20	1	11	54.5394	21.0078	9.1548	3.0051
Rec07	20 × 10	1545	20	1	7	58.3495	19.3526	8.2549	2.5486
Rec09	20 × 10	1530	20	0	6	102.6843	18.8546	13.2385	4.5786
Rec11	20 × 10	1402	20	0	12	148.6842	34.5263	8.2548	4.4532
Rec13	20 × 15	1930	20	0	9	125.6841	32.1532	7.1548	2.1354
Rec15	20 × 15	1950	20	0	8	57.2593	33.3562	14.2596	6.5489
Rec17	20 × 15	1902	20	1	13	45.2985	28.1354	22.4519	8.4865
Rec19	30 × 10	2093	30	1	13	85.4872	48.9564	14.1258	4.9515
Rec21	30 × 10	2017	30	0	15	111.1531	58.1465	13.6529	2.4631
Rec23	30 × 10	2008	30	0	14	123.6521	62.5852	9.4582	3.5647
Rec25	30 × 15	2513	30	0	11	98.3643	45.6287	7.5628	2.5864
Rec27	30 × 15	2373	30	0	12	96.3325	37.1524	19.1548	7.6531
Rec29	30 × 15	2287	30	1	8	86.1452	32.1654	18.6543	8.6525
Rec31	50 × 10	3045	50	1	12	106.5842	57.1491	22.6532	4.9846
Rec33	50 × 10	3093	50	0	15	88.6452	36.5843	24.1592	12.0356
Rec35	50 × 10	3262	50	0	17	94.6321	49.0150	18.6345	9.4521
Rec37	75 × 20	4951	75	2	9	84.2564	38.1572	9.5482	3.5986
Rec39	75 × 20	5087	75	0	12	134.5263	64.8132	26.3468	11.0248
Rec41	75 × 20	4960	75	0	8	128.4524	78.3154	33.2158	14.9854

Acknowledgment

The work is supported by the National Natural Science Foundation of China (61272119).

References

[1] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[2] J. K. Lenstra, A. H. G. R. Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.

[3] L. Hege, M. Minoux, and W. van Canneyt, "A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-wait constraints," *Computers & Operations Research*, vol. 39, no. 3, pp. 629–636, 2012.

[4] K. Gokbayrak and O. Selvi, "Service time optimization of mixed-line flow shop systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 395–404, 2010.

[5] D. Ravindran, S. J. Selvakumar, R. Sivaraman, and A. N. Haq, "Flow shop scheduling with multiple objective of minimizing makespan and total flow time," *International Journal of Advanced Manufacturing Technology*, vol. 25, no. 9, pp. 1007–1012, 2005.

[6] T. Eren and E. Güner, "A bicriteria scheduling with sequence-dependent setup times," *Applied Mathematics and Computation*, vol. 179, no. 1, pp. 378–385, 2006.

[7] J. Lemesre, C. Dhaenens, and E. G. Talbi, "An exact parallel method for a bi-objective permutation flowshop problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1641–1655, 2007.

[8] C. T. Tseng and C. J. Liao, "A discrete particle swarm optimization for lot-streaming flowshop scheduling problem," *European Journal of Operational Research*, vol. 191, no. 2, pp. 360–373, 2008.

[9] B. Naderi, M. Zandieh, A. Khaleghi Ghoshe Balagh, and 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 Systems with Applications*, vol. 36, no. 6, pp. 9625–9633, 2009.

[10] B. Qian, L. Wang, D. X. Huang, and X. Wang, "Multi-objective flow shop scheduling using differential evolution," *Lecture Notes in Control and Information Sciences*, vol. 345, pp. 1125–1136, 2006.

[11] T. Pasupathy, C. Rajendran, and R. K. Suresh, "A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs," *International Journal of Advanced Manufacturing Technology*, vol. 27, no. 7-8, pp. 804–815, 2006.

[12] N. Melab, M. Mezmaz, and E. G. Talbi, "Parallel cooperative meta-heuristics on the computational grid. A case study: the bi-objective Flow-Shop problem," *Parallel Computing*, vol. 32, no. 9, pp. 643–659, 2006.

[13] R. Tavakkoli-Moghaddam, A. Rahimi-Vahed, and A. H. Mirzaei, "A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness," *Information Sciences*, vol. 177, no. 22, pp. 5072–5090, 2007.

[14] B. B. Li and L. Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 37, no. 3, pp. 576–591, 2007.

[15] P. C. Chang, S. H. Chen, and C. H. Liu, "Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems," *Expert Systems with Applications*, vol. 33, no. 3, pp. 762–771, 2007.

[16] F. Dugardin, F. Yalaoui, and L. Amodeo, "New multi-objective method to solve reentrant hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 203, no. 1, pp. 22–31, 2010.

[17] N. Karimi, M. Zandieh, and H. R. Karamooz, "Bi-objective group scheduling in hybrid flexible flowshop: a multi-phase approach," *Expert Systems with Applications*, vol. 37, no. 6, pp. 4024–4032, 2010.

[18] M. J. Geiger, "Decision support for multi-objective flow shop scheduling by the Pareto Iterated Local Search methodology," *Computers & Industrial Engineering*, vol. 61, no. 3, pp. 805–812, 2011.

[19] T. C. Chiang, H. C. Cheng, and L. C. Fu, "NNMA: an effective memetic algorithm for solving multiobjective permutation flow shop scheduling problems," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5986–5999, 2011.

[20] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems," *Computers & Operations Research*, vol. 38, no. 8, pp. 1219–1236, 2011.

[21] H. M. Cho, S. J. Bae, J. Kim, and I. J. Jeong, "Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm," *Computers & Industrial Engineering*, vol. 61, no. 3, pp. 529–541, 2011.

[22] D. Lei, "A Pareto archive particle swarm optimization for multi-objective job shop scheduling," *Computers & Industrial Engineering*, vol. 54, no. 4, pp. 960–971, 2008.

[23] M. Khalili and R. Tavakkoli-Moghaddam, "A multi-objective electromagnetism algorithm for a bi-objective flow shop scheduling problem," *Journal of Manufacturing Systems*, vol. 31, no. 2, pp. 232–239, 2012.

[24] J. H. Holland, "Genetic algorithm," *Scientific American*, vol. 266, pp. 44–50, 1992.

[25] T. Murata, H. Ishibuchi, and H. Tanaka, "Multi-objective genetic algorithm and its applications to flowshop scheduling," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 957–968, 1996.

[26] R. Qing-dao-er-ji and Y. Wang, "A new hybrid genetic algorithm for job shop scheduling problem," *Computers & Operations Research*, vol. 39, no. 10, pp. 2291–2299, 2012.

[27] Q. D. E. J. Ren, Y. Wang, and X. Si, "An improved genetic algorithm for job shop scheduling problem," in *Proceedings of the International Conference on Computational Intelligence and Security (CIS '10)*, pp. 113–116, December 2010.

[28] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Annals of Operations Research*, vol. 41, no. 3, pp. 231–252, 1993.

[29] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & Operations Research*, vol. 22, no. 1, pp. 5–13, 1995.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[31] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.