Research Article

A Discrete Group Search Optimizer for Hybrid Flowshop Scheduling Problem with Random Breakdown

Zhe Cui and Xingsheng Gu

Key Laboratory of Advanced Control and Optimization for Chemical Process, Ministry of Education, Shanghai 200237, China

Correspondence should be addressed to Xingsheng Gu; xsgu@ecust.edu.cn

Received 5 December 2013; Accepted 24 January 2014; Published 27 February 2014

Academic Editor: Huaicheng Yan

Copyright © 2014 Z. Cui and X. Gu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The scheduling problems have been discussed in the literature extensively under the assumption that the machines are permanently available without any breakdown. However, in the real manufacturing environments, the machines could be unavailable inevitably for many reasons. In this paper, the authors introduce the hybrid flowshop scheduling problem with random breakdown (RBHFS) together with a discrete group search optimizer algorithm (DGSO). In particular, two different working cases, preempt-resume case, and preempt-repeat case are considered under random breakdown. The proposed DGSO algorithm adopts the vector representation and several discrete operators, such as insert, swap, differential evolution, destruction, and construction in the producers, scroungers, and rangers phases. In addition, an orthogonal test is applied to configure the adjustable parameters in the DGSO algorithm. The computational results in both cases indicate that the proposed algorithm significantly improves the performances compared with other high performing algorithms in the literature.

1. Introduction

The hybrid flowshop scheduling (HFS) problem, being also referred to as multiprocessor or flexible flowshop, is one kind of production scheduling problems, which has been widely used in the process industry such as the paper, oil, petrochemical, and pharmaceutical industries [1]. In the HFS problem, it is assumed that the jobs have to pass through all stages in the same order and that there is at least one stage that must have multiple machines. Each job is processed on only one machine at each stage. The HFS problem minimizing the makespan or maximum completion time is denoted as $HFC/C_{\max}$ using the three-field notation by Graham et al. [2], and it has been proved to be strongly NP-hard when the number of machines is no less than two [3]. The research efforts on the HFS problem generally presume a static environment and no unexpected events [4]. However, the real natural environments are stochastic and dynamic [5, 6], and the real manufacturing processes tend to suffer a wide range of uncertainties, such as machine breakdown, job cancellation, due-date changes, shortage of materials, and changes in job priority. In this paper, the authors

concentrate only on the hybrid flowshop scheduling problem with random breakdown (RBHFS), which is referred to as $HFC/brkdn/C_{\max}$. In the RBHFS problem, the breakdown of machines may happen at any time and the job being processed on the machine must be stopped until the machine has been repaired. On the basis of the characters of jobs and the breakdown of machines, the RBHFS problem falls into two categories: the preempt-resume case and the preempt-repeat case. In the preempt-resume case, the completed part of the job will not be lost and the processing time of the job is cumulative after the machine breakdown is repaired. While in the preempt-repeat case, the completed part will be lost so that the operation of the job should be started from the beginning, which is commonly used in the preprocessing phase, for example, the heating and cooling processes.

Compared with a lot of works on the HFS problem, the studies on the RBHFS problem are still in their infancy. Alcaide et al. [7] considered the stochastic flowshop scheduling problem subject to random breakdowns where the objective is to minimize the expected makespan and proposed a dynamic procedure that enables us to consider a random breakdowns stochastic problem as a sequence of

without-breakdowns stochastic problems. In [8] the dynamic scheduling problems in random flexible manufacturing systems subject to machine breakdowns are addressed and an adaptive scheduling approach is proposed to make coupled decisions about the part/machine scheduling and operation/tool assignments on a rolling horizon basis. Allaoui and Artiba [9] studied the hybrid flowshop scheduling problem to minimize flow time and due date based criteria under maintenance constraints. In this model, the setup, cleaning, and transportation time are also considered. Tang et al. [10] studied the single machine stochastic just-in-time scheduling problem subject to machine breakdowns for the preempt-resume and the preempt-repeat. The objective function of the problem is the sum of squared deviations of the job expected completion time from the due date. Gholami et al. [11] examined a heuristic to find schedules minimizing expected makespan in the hybrid flowshop problem with sequence-dependent setups and machines with random breakdowns. This method employed the random keys genetic algorithm approaches (RKGA) to find the optimum solutions and utilized the simulator algorithm to evaluate the objective function under this condition. From the analysis results of the Taguchi parameter design, the number of job, the number of stage, and the population size play important roles in the algorithm. And the same problem was studied later by Zandieh and Gholami [12]. Based on a clonal selection principle and an affinity maturation mechanism of the immune response, they proposed an immune algorithm (IA) and applied the Taguchi parameter design method to analyze the proposed algorithm. Safari and Sadjadi [13] explored flowshop configuration under the assumption of condition-based maintenance to minimize expected makespan and proposed a condition-based maintenance (CBM) strategy and a hybrid algorithm based on genetic algorithm and simulated annealing. Their simulation results showed its superiority. Wang and Choi [14] proposed a decomposition-based approach that integrates the completely reactive approach with the predictive reactive approach to minimize the makespan of the flexible flowshop scheduling problem under machine breakdown. Mirabi et al. [15] considered a two-stage hybrid flowshop scheduling problem where the machine may not always be available during the scheduling period. The objective is to find the optimal job combinations and the job schedules such that the makespan is minimized.

Recently, a population-based optimization algorithm group search optimizer (GSO) has been proposed by He et al. [16], which is inspired by the animal social foraging behavior and group living theory. Some recent researches indicate that the GSO algorithm is better than some other evolutionary algorithms for solving the large-scale multimodal benchmark functions [17, 18]. At the same time, the GSO algorithm has been successfully applied in a variety of fields, such as artificial neural network training [19, 20], power system [21, 22], and mechanical design [23].

Considering the successes of the GSO algorithm, the authors proposed to use a discrete group search optimizer (DGSO) algorithm for the production scheduling problem. The proposed DGSO algorithm maintains the optimization mechanism of the basic GSO algorithm and abandons the

angle evolution strategy to improve the effectiveness and efficiency of the algorithm. In the DGSO algorithm, an encoding scheme based on the vector representation is introduced in order to adapt the GSO algorithm to the discrete problems. Meanwhile, an improved variable neighborhood search, a novel differential evolution operation, and the destruction and construction procedures are proposed in the producer, scrounger, and ranger phases, respectively. In addition, in order to achieve a good performance of the DGSO algorithm, an orthogonal experiment design is carried out for getting a guideline on tuning of the parameters in the algorithm. In both the preempt-resume case and the preempt-repeat case, our proposed DGSO algorithm shows the state-of-the-art results on benchmarks.

The rest of the paper is organized as follows. In Section 2, the RBHFS problem statement and the approach to dealing with random breakdown are formulated. Section 3 presents the details of the proposed DGSO algorithm. The tests on parameter selection and the simulation results are provided in Section 4. Finally, the authors draw conclusions in Section 5.

2. Problem Statement

2.1. Description of the Problem. The RBHFS problem with makespan criterion can be described as follows. There are n jobs $J = \{1, 2, \dots, i, \dots, n-1, n\}$ that have to be performed on s stages $S = \{1, 2, \dots, j, \dots, s-1, s\}$, and each stage j has m_j identical machines. At least one stage j must have more than one machine. Each job consists of s operations, each of which must be sequentially processed at stage 1 to stage s and by exactly one machine at every stage. At any time, a machine can process at most one job and a job can be processed by at most one machine. The processing time required for job i at stage j is given as p_{ij} . These identical machines are not available at all times, and they suffer random breakdown. Once the machine begins to work, it will continue to run until breakdown occurs or all jobs are finished. The setup and release time of all jobs are negligible. Preemption is not allowed and infinite intermediate buffers exist between two successive stages. The scheduling problem is to choose a machine at each stage for each job and determine the sequence of jobs on each machine so as to minimize the makespan [14].

2.2. An Approach to Dealing with Random Breakdown. It is assumed that a machine keeps processing the jobs sequentially until it break down or it has finished all the jobs, and machine breakdowns may arise at any time in working periods. The authors find that the possible positions where the breakdown may happen in are shown in Figure 1, where A means the breakdown happened during machine's processing time interval, B , C , and D represent the breakdown happened during machine's idle time interval.

This paper considers two different cases while dealing with an interrupted job. Case 1 is the preempt-resume case and Case 2 is the preempt-repeat case. In Case 1, after repairing, only the unprocessed part of the interrupted job needs to be processed. And in Case 2, after repairing, the

FIGURE 1: The Gantt chart of the RBHFS problem.

whole interrupted job needs to be reprocessed. It is assumed that st_{ijk} and C_{ijk} are the starting time and the completing time of job i on machine k ($k \in \{1, 2, \dots, \sum_{j=1}^s m_j\}$) at stage j , respectively, and r_k denotes the repairing time of machine k . Then C_{ijk} can be calculated as follows:

A: if $st_{ijk} \leq A < C_{ijk}$

Case 1: $st_{ijk} = st_{ijk}, C_{ijk} = st_{ijk} + r_k + p_{ij}$

Case 2: $st_{ijk} = A + r_k, C_{ijk} = st_{ijk} + p_{ij}$

B: if $B < st_{ijk}$

Case 1 and Case 2: $st_{ijk} = \max\{st_{ijk}, B + r_k\}, C_{ijk} = st_{ijk} + p_{ij}$

C: if $C_{(i-1)jk} \leq C < st_{ijk}$

Case 1 and Case 2: $st_{ijk} = \max\{st_{ijk}, C + r_k\}, C_{ijk} = st_{ijk} + p_{ij}$

D: if $D \geq C_{ijk}$

Case 1 and Case 2: $st_{ijk} = st_{ijk}, C_{ijk} = C_{ijk}$.

3. The Discrete Group Search Optimizer (DGSO) for the RBHFS Problem

The basic GSO algorithm adopts the framework of the biological model, Producer-Scrounger (PS) model [24], which uses two foraging strategies within groups: producing and scrounging. In order to avoid entrapping in local minima, GSO also employs rangers foraging strategies which perform random walks [17]. The population of GSO is called a group and each individual in the group is called a member. In the GSO scheme, a group contains three types of members, namely, the producers, the scroungers, and the rangers [18]. Each individual in the group has its own position, search angle, and search direction. At each iteration cycle, the producers perform producing strategy to search for the optimal positions; the scroungers perform scrounging strategy to join resources found by the producers; the remaining members are the rangers which walk randomly in the searching space to find their new positions. In the GSO algorithm, a position of the individual corresponds to a solution of the optimization problem, and the fitness of the position corresponds to the fitness of the solution. The processes of the basic GSO algorithm can be shown in pseudocode as Algorithm 1.

Following the procedure for the continuous function optimization, in this paper the authors propose a discrete version of the GSO algorithm for the RBHFS problem. It is discussed in detail below.

3.1. Individual Representation and Initialization. Owing to its continuous nature, the GSO algorithm does not directly fit for the discrete flowshop scheduling problem. So it is important to find a suitable mapping which can conveniently convert individuals to solutions. In the HFS problem, there are two formats to represent a solution: the matrix representation and the vector representation [25]. In this paper, the RBHFS problem is formulated by the vector representation, which considers the sequence of jobs only at stage one and the sequences of jobs at other stages are decoded by employing the List Scheduling (LS) algorithm [26, 27]. As a result, the vector representation is also adopted in the proposed DGSO algorithm. In such a case, the individual in the DGSO algorithm can be represented by a permutation of jobs at stage one $\pi = \{\pi(1), \pi(2), \dots, \pi(n)\}$. The population size of the proposed DGSO algorithm is determined by parameter PS and the initial population is generated randomly in the search space.

3.2. Producer. Recently, Couzin et al. [28] indicated that the larger the group, the smaller the proportion of informed individuals needed to guide the group, and that only a very small proportion of informed individuals is required to achieve great accuracy. As a result, for the great accuracy and the simplicity, there is only one producer in the DGSO algorithm, which means that the best individual is the producer and the remaining members in the population are scroungers or rangers. Furthermore, considering the efficiency of the algorithm, the producing strategy is executed on the producer only if it is changed after the evolutionary procedure.

Aiming at improving the efficiency of the proposed DGSO algorithm, the authors introduce an improved variable neighborhood search (IVNS) [29] as the producing strategy in this paper. The IVNS contains two structures of neighborhoods: the insert local search and the swap local search. The pseudocodes of the insert local search and the swap local search are given in Algorithm 2. The IVNS keeps doing the insert local search followed by the swap local search until there is no improvement. If the newly generated individual is better than the current producer, the new individual is set to be the producer. Otherwise the old producer is retained.

3.3. Scrounger. In the population, the left individuals except the producer are divided into the scroungers and the rangers. Each individual is set to be scrounger or ranger with the probability of P and $(1 - P)$, respectively.

As for each scrounger, a novel discrete differential evolution scheme is employed for improving the scrounging performance, which consists of three steps: mutation, crossover, and selection.

In the mutation part, the *bestrandinsert* operation [30] is introduced. The procedure of the *bestrandinsert* is described

```

Procedure the GSO algorithm
  Initial population
  While (criterion)
    Choose the producers and perform the producing
    Choose the scroungers and perform the scrounging
    Disperse the rest individuals to perform ranging
    Evaluate the individuals
  End while
End procedure

```

ALGORITHM 1: The procedure of the basic GSO algorithm.

as follows. It first randomly chooses $pr1$ jobs without repetition; then it inserts each chosen job into other $n - 1$ position, and $pr2$ better permutations with relatively smaller makespan are remembered; finally one of the $pr2$ permutations is randomly chosen to replace the incumbent scrounger. For each scrounger, the producer undergoes the *bestrandinsert* operation with a mutation rate (MR) to obtain the mutant individual. On the other hand, the mutant individual is the same as the producer with a probability of about $(1 - MR)$. That is to say, the mutant individual is obtained as

$$V_i^t = \begin{cases} \text{bestrandinsert}(\text{producer}), & \text{if rand}(0, 1) < MR \\ \text{producer}, & \text{otherwise,} \end{cases} \quad (1)$$

where V_i^t denotes the mutant individual at generation t and $\text{rand}(0, 1)$ is a random function returning a number between 0 and 1 with uniform distribution.

Next a crossover operation called CRO is employed in the crossover part, and it will be able to work effectively even though the individuals in the population are very close to each other in the later stage of evolution. The crossed individual $U_i^t = \text{CRO}(\text{scrounger}, V_i^t)$. The authors first randomly select a part of job permutation in the scrounger and insert it on the front or back of the mutant individual V_i^t and then delete the jobs in V_i^t which are already in the selected segment. In this case, two valid crossed individuals will be obtained: one is the front one and the other the back one.

Following the crossover operation, the selection is conducted. The best one which has the lowest objective value among the two crossed individuals and the incumbent scrounger is accepted. In other words, if either of these two crossed individuals yields a better makespan than the scrounger, then the better individual becomes the scrounger; otherwise the old scrounger is retained.

3.4. Ranger. In the basic GSO algorithm, the rangers search randomly in the predefined space to increase the population diversity and avoid getting trapped in local optima. Here the rangers employ the destruction and construction procedures of the iterated greedy (IG) algorithm with one parameter: destruction size (d) [31]. Regardless of whether the newly generated individual is better than the original ranger, the ranger is replaced to enhance the global search ability. Throughout

```

(a) the insert local search
  improve = 1;
  while (improve = 1)
    do {
      improve = 0;
      for (i = 1 : n)
        remove job j at random from producer
         $\pi'$  = best permutation obtained by inserting
        job j in any possible positions of producer
        if ( $C_{\max}(\pi') < C_{\max}(\text{producer})$ )
          producer =  $\pi'$ ;
          improve = 1;
        endif
      endfor
    }
  }
(b) the swap local search
  improve = 1;
  while (improve = 1)
    do {
      improve = 0;
      for (i = 1 : n)
         $\pi'$  = best permutation obtained by swapping
        job j with other jobs of producer
        if ( $C_{\max}(\pi') < C_{\max}(\text{producer})$ )
          producer =  $\pi'$ ;
          improve = 1;
        endif
      endfor
    }
  }

```

ALGORITHM 2: The procedure of improved variable neighborhood search.

the course of evolution, once the new individual generated by scroungers or rangers is superior to the producer, the producer will be updated.

3.5. Computational Procedure. Based on the above operations, the procedure of the DGSO algorithm for the RBHFS problem is summarized as follows.

Step 1. Set the algorithm parameters PS, P , MR, $pr1$, $pr2$, and d , and initialize the population.

Step 2. The producer conducts an improved variable neighborhood search to search for a better solution if the producer is changed.

Step 3. The scroungers employ the discrete differential evolution operation to keep searching for the high-quality solutions, which includes mutation, crossover, and selection.

Step 4. The rangers produce new solutions by using the destruction and construction procedures to avoid local optimum.

Step 5. Evaluate each member in the population and utilize the best individual to update the producer.

Step 6. If the given termination criterion is satisfied, end the procedure and return the producer; otherwise go back to Step 2.

Considering the character of the RBHFS problem, some discrete operators are introduced for the producers, scroungers, and rangers. The DGSO algorithm has the producer and the scroungers to play the part of exploitation and employs the rangers to play the part of exploration. Since both the exploitation and exploration are improved and well balanced, it is expected to generate good results for the RBHFS problem under the criterion of makespan minimization. In the next section, the performance of the DGSO algorithm is investigated based on simulation results and comparisons.

4. Simulation Results and Comparisons

4.1. Experimental Setup. To fully examine the performance of the DGSO algorithm, a parameter discussion, a preliminary experiment, and an extensive experimental comparison with other powerful methods are provided. As no test instances are available for the RBHFS problem, some benchmark problems for the HFS problem are modified. Ten benchmark problems proposed by Liao et al. [32] have been selected. In each instance of Liao's benchmark problems, there are 30 jobs and 5 stages. At each stage, the machine number has a uniform distribution in the range of [3, 5]. The processing time in these problems is within [1, 100]. It is assumed that the breakdown time point and repairing time obey the normal distributions and there are five breakdown time points. The mean values of the five breakdown time points are equal to 200, 400, ..., 1000, respectively, and the variances of all the breakdown points are 10. The mean value of repairing time of each machine is 20, and the variance is 5. All the algorithms were coded in Visual C++ and run on an Intel Pentium 3.06 GHz PC with 2 GB RAM under Windows 7 operating system. The maximum computation time is fixed at 200 seconds for all the parameter tests, the preliminary experiment, and compared algorithms in the following subsections.

4.2. Parameter Discussion. Tuning parameters properly is critical for an evolutionary algorithm to achieve a good

TABLE 1: Factors and levels for orthogonal experiment.

Factor	Level 1	Level 2	Level 3	Level 4	Level 5
PS(1)	10	20	30	40	50
P(2)	0.6	0.7	0.8	0.9	1
MR(3)	0.2	0.4	0.6	0.8	1
pr1(4)	2	3	4	5	6
pr2(5)	1	2	3	4	5
d(6)	1	2	3	4	5

performance [33, 34]. In the proposed DGSO algorithm, there are six main parameters: PS, P, MR, pr1, pr2, and d. All the six parameters are regarded as factors at five different levels, as illustrated in Table 1. If the authors carry out the full factorial experiment design in this case, it requires $5^6 = 15,625$ experiments which is not necessary and economical. Therefore, an orthogonal experiment design [35] is applied to provide a receipt for tuning the adjustable parameters of the DGSO algorithm, which just needs $5^2 = 25$ experiments.

Parameter experiments are conducted on ten Liao's benchmark problems. The breakdowns in the parameter discussion are subject to preempt-resume case (Case 1) for simplicity. Taking the test on instance j30c5e1, for example, Table 2 shows the orthogonal parameter table $L_{25}(5^6)$ including 25 groups of the parameter test samples. Each group of parameters is trialed for ten times. The unabridged result tables similar to Table 2 in other nine instances are omitted here. After the detailed analyses of the orthogonal design, the authors set the parameters PS = 10, P = 0.6, MR = 0.8, pr1 = 4, pr2 = 2, and d = 2 in the following experiments.

4.3. Preliminary Experiment. Comparing with the group search optimizer (GSO) algorithm, there are three new elements in the proposed DGSO algorithm: the improved variable neighborhood search in the producer phase, the differential evolution in the scrounger phase, and the destruction and construction procedures in the ranger phase. The following abbreviations represent the variants considered: the GSO-IVNS (GSO with improved variable neighborhood search) and the GSO-IVNS-DE (DHS with improved variable neighborhood search and differential evolution). To verify the effect of each element in the algorithm for solving the RBHFS problem, the authors test four algorithms: GSO, GSO-IVNS, GSO-IVNS-DE, and DGSO in the preempt-repeat case (Case 2). Each method was run twenty times for each Liao's benchmark problem and its performance, including the average and minimum value, was recorded in Table 3.

In Table 3, the overall mean values of AVE and MIN yielded by the DGSO algorithm are equal to 710.9 and 701, respectively, and the DGSO algorithm is the best one among the four variants for all the problems with the same computational time. It is shown that the utilization of the improved variable neighborhood search, the differential evolution, and the destruction and construction procedures is the key to stress the balance of exploration and exploitation so as to improve the performance of GSO algorithm.

TABLE 2: Orthogonal parameter table $L_{25}(5^6)$ and results of j30c5e1.

Test	Factor						Mean value
	PS	P	MR	$pr1$	$pr2$	d	
1	10(1)	0.6(1)	0.2(1)	2(1)	1(1)	1(1)	505.7
2	10(1)	0.7(2)	0.4(2)	3(2)	2(2)	2(2)	503.5
3	10(1)	0.8(3)	0.6(3)	4(3)	3(3)	3(3)	504.5
4	10(1)	0.9(4)	0.8(4)	5(4)	4(4)	4(4)	503.8
5	10(1)	1(5)	1(5)	6(5)	5(5)	5(5)	509.8
6	20(2)	0.6(1)	0.4(2)	4(3)	4(4)	5(5)	503.2
7	20(2)	0.7(2)	0.6(3)	5(4)	5(5)	1(1)	506.5
8	20(2)	0.8(3)	0.8(4)	6(5)	1(1)	2(2)	504.7
9	20(2)	0.9(4)	1(5)	2(1)	2(2)	3(3)	505.4
10	20(2)	1(5)	0.2(1)	3(2)	3(3)	4(4)	512.7
11	30(3)	0.6(1)	0.6(3)	6(5)	2(2)	4(4)	503.5
12	30(3)	0.7(2)	0.8(4)	2(1)	3(3)	5(5)	503.4
13	30(3)	0.8(3)	1(5)	3(2)	4(4)	1(1)	506.6
14	30(3)	0.9(4)	0.2(1)	4(3)	5(5)	2(2)	506.3
15	30(3)	1(5)	0.4(2)	5(4)	1(1)	3(3)	512.3
16	40(4)	0.6(1)	0.8(4)	3(2)	5(5)	3(3)	503.4
17	40(4)	0.7(2)	1(5)	4(3)	1(1)	4(4)	506.4
18	40(4)	0.8(3)	0.2(1)	5(4)	2(2)	5(5)	506.0
19	40(4)	0.9(4)	0.4(2)	6(5)	3(3)	1(1)	507.2
20	40(4)	1(5)	0.6(3)	2(1)	4(4)	2(2)	512.0
21	50(5)	0.6(1)	1(5)	5(4)	3(3)	2(2)	504.0
22	50(5)	0.7(2)	0.2(1)	6(5)	4(4)	3(3)	505.3
23	50(5)	0.8(3)	0.4(2)	2(1)	5(5)	4(4)	504.5
24	50(5)	0.9(4)	0.6(3)	3(2)	1(1)	5(5)	507.1
25	50(5)	1(5)	0.8(4)	4(3)	2(2)	1(1)	510.4
k_1	505.46	503.96	507.20	506.20	507.24	507.28	
k_2	506.50	505.02	506.14	506.66	505.76	506.10	
k_3	506.42	505.26	506.72	506.16	506.36	506.18	
k_4	507.00	505.96	505.14	506.52	506.18	506.18	
k_5	506.26	511.44	506.44	506.10	506.10	505.90	
std	0.558	2.946	0.770	0.247	0.554	0.544	

TABLE 3: Performance of HDHS with different versions.

Problem	GSO		GSO-IVNS		GSO-IVNS-DE		DGSO	
	AVE	MIN	AVE	MIN	AVE	MIN	AVE	MIN
j30c5e1	564.2	555	561.9	558	561.3	555	552.7	549
j30c5e2	721.4	716	720.0	716	722.5	716	700.7	697
j30c5e3	752.8	743	741.9	736	740.5	732	723.3	712
j30c5e4	704.5	695	702.5	695	700.5	688	685.3	674
j30c5e5	748.1	726	733.6	726	727.6	726	725.1	709
j30c5e6	743.2	737	737.3	730	731.1	721	718.1	708
j30c5e7	751.9	745	748.0	743	737.2	733	732.1	723
j30c5e8	859.0	824	842.1	801	822.2	788	786.8	771
j30c5e9	799.9	786	796.0	779	791.8	770	766.9	760
j30c5e10	750.2	738	738.4	718	729.6	715	718.5	710
Average	739.5	726	732.2	720	726.4	714	710.9	701

TABLE 4: Comparison results on Liao's benchmark problems of Case 1.

Problem	PSO			RKGA			IA			DGSO		
	AVE	MIN	STD	T (s)	AVE	MIN	STD	T (s)	AVE	MIN	STD	T (s)
j30c5e1	511.7	507	2.4	50.9	513.0	505	3.9	60.4	509.0	506	1.5	53.6
j30c5e2	670.3	670	0.4	29.8	670.5	670	0.8	64.9	670.0	670	0.0	44.9
j30c5e3	663.6	657	3.7	66.0	667.5	655	7.5	83.4	659.6	655	1.7	75.9
j30c5e4	631.3	628	2.1	85.3	630.7	626	3.1	72.4	629.2	627	1.2	80.4
j30c5e5	668.5	666	1.7	64.1	671.7	666	4.5	68.5	662.8	662	0.4	70.0
j30c5e6	675.4	670	3.3	67.6	679.0	670	5.6	66.3	669.1	665	2.5	62.7
j30c5e7	686.1	684	1.8	49.7	686.0	683	2.5	74.2	683.9	682	0.9	54.8
j30c5e8	737.9	732	3.2	95.4	738.3	731	5.1	101.1	734.2	731	1.8	86.3
j30c5e9	708.8	705	2.0	89.8	709.8	703	5.1	103.7	705.4	702	1.6	90.2
j30c5e10	650.5	641	5.3	82.2	648.8	636	7.2	84.9	642.5	637	2.2	71.2
Average	660.4	656	2.6	68.1	661.5	655	4.5	78.0	656.6	654	1.4	69.0
									652.1	651	1.1	60.1

TABLE 5: Comparison results on Liao's benchmark problems of Case 2.

Problem	PSO			RKGa			IA			DGSO		
	AVE	MIN	STD	T (s)	AVE	MIN	STD	T (s)	AVE	MIN	STD	T (s)
j30c5e1	561.6	557	5.1	80.2	562.1	555	8.7	107.0	559.6	554	2.4	87.0
j30c5e2	720.0	701	4.4	86.8	723.8	701	7.8	102.3	716.3	701	3.2	97.7
j30c5e3	747.5	739	6.0	69.6	748.6	735	6.2	89.0	743.0	736	4.5	77.4
j30c5e4	702.2	696	7.5	101.9	701.3	694	8.1	121.6	697.5	688	5.9	69.7
j30c5e5	733.7	726	8.7	54.3	734.1	726	13.5	105.3	731.5	726	5.9	54.2
j30c5e6	737.1	730	3.6	114.4	737.0	720	6.0	71.8	730.7	718	3.1	101.8
j30c5e7	752.2	745	4.3	91.7	751.8	743	6.9	110.1	747.9	741	3.8	84.9
j30c5e8	836.7	802	10.4	96.8	839.2	792	14.9	84.6	826.4	790	9.5	83.4
j30c5e9	799.4	791	6.8	98.2	800.2	789	15.4	107.3	793.6	788	5.9	106.5
j30c5e10	748.9	740	3.2	88.7	746.2	715	11.6	108.8	738.1	712	9.3	93.5
Average	733.9	723	6.0	88.3	734.4	717	9.9	100.8	728.5	715	5.4	85.6
									710.9	701	5.3	80.2

4.4. Comparisons among PSO, RKGA, IA, and DGSO in Different Breakdown Cases. Several metaheuristics have been applied to the RBHFS problem. To evaluate the performance of the proposed DGSO algorithm in solving the RBHFS problem under the criterion makespan, the DGSO algorithm is compared with a PSO algorithm proposed by Liao et al. [32], a RKGA by Gholami et al. [11], and an IA by Zandieh and Gholami [12]. In order to establish more accurate and objective comparisons, the parameters of these algorithms are set following the corresponding literature. All the algorithms were run twenty independent replications for each problem. The comparison results on Liao's benchmark problems of Case 1 and Case 2 are summarized in Tables 4 and 5. In these two tables, AVE, MIN, STD, and T indicate the values of average, minimum, standard deviation, and the convergence time, respectively.

In Tables 4 and 5, the smallest values of AVE, MIN, STD, and T in the rows are shown in bold. It can be noted that in the preempt-resume case, the overall mean values of AVE, MIN, and STD yielded by the DGSO algorithm are equal to 652.1, 651, and 1.1, respectively, which are much better than those generated by PSO, RKGA, and IA. The DGSO algorithm can converge to better solutions than other three compared algorithms in about the same amount of time, and the same performance is provided in the preempt-repeat case. From these observations, it is concluded that the DGSO algorithm is more effective and efficient in comparison with other methods for the RBHFS problem in both cases.

To confirm whether the observed differences are indeed statistically significant, the authors carry out an analysis of variance (ANOVA). ARE is analyzed by multicompare method using least significant difference (LSD) procedure, where ARE denotes the average relative error to the best solution found by any of the compared algorithms. Obviously, the smaller ARE value is the better result the algorithm yields. The means plots of ARE for all the compared algorithms with LSD intervals at a 95% confidence level in Case 1 and Case 2 are shown in Figures 2 and 3. Note that if the LSD intervals for two means are not overlapping, then the means are significantly different. In Figures 2 and 3, the four algorithms can be divided into two homogenous groups: (PSO, RKGA, and IA) and DGSO, where no statistically significant differences can be found within each group. From these two figures, it is clear that the proposed DGSO algorithm is statistically better than PSO, RKGA, and IA. Considering the stability and robustness [36, 37] of the algorithm, these figures also demonstrate the superiority of the DGSO algorithm under the random breakdown cases.

4.5. Comparisons of the RBHFS Problem in Different Breakdown Cases. Figure 4 is the computational result for different breakdown cases of Liao's benchmark problems, which are achieved by the DGSO algorithm. The authors found that the problem without breakdown takes less time than Case 1, and Case 1 takes less time than Case 2, which is consistent with the theoretical fact. It means that the job without interrupt needs less processing time than the job being interrupted by breakdown, and the job being interrupted by

FIGURE 2: Means plot of ARE with 95% LSD intervals for different algorithms in Case 1.

FIGURE 3: Means plot of ARE with 95% LSD intervals for different algorithms in Case 2.

breakdown needs less processing time than the job to be reprocessed. In conclusion, the machine breakdown affects the processing time of the interrupted jobs and influences the scheduling solution, and different breakdown cases have different scheduling schemes.

5. Conclusions

This paper models the hybrid flowshop scheduling problem with random breakdown by analyzing the random breakdown time point and offering an approach to dealing with the breakdown. Then a discrete group search optimizer algorithm is proposed to minimize the makespan of the RBHFS problem. Several efficient operators are introduced for the producers, scroungers, and rangers in the DGSO algorithm. In addition, an orthogonal test is applied to configure the

FIGURE 4: The computational results under two breakdown cases and no breakdown case.

algorithm parameters after a small number of experiments. Compared with some state-of-the-art algorithms on the same benchmark problems, the simulation results demonstrate the effectiveness and efficiency of the proposed DGSO algorithm. Future studies can focus on the replication of the DGSO algorithm for other kinds of scheduling problems, such as stochastic scheduling problem and multiobjective scheduling problem.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are grateful to the anonymous reviewers for giving us helpful suggestions. This work is supported by National Natural Science Foundation of China (Grant nos. 61174040, 61104178) and Fundamental Research Funds for the Central Universities, Shanghai Commission of Science and Technology (Grant no. 12JC1403400).

References

- [1] M. L. Pinedo, *Scheduling: Theory Algorithms and Systems*, Prentice Hall, Englewood Cliffs, NJ, USA, 2002.
- [2] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy-Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, no. 1, pp. 287–326, 1979.
- [3] J. A. Hoogeveen, J. K. Lenstra, and B. Veltman, "Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard," *European Journal of Operational Research*, vol. 89, no. 1-2, pp. 172–175, 1996.
- [4] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, 2010.
- [5] H. Zhang, Q. Chen, H. Yan, and J. Liu, "Robust H_∞ filtering for switched stochastic system with missing measurements," *IEEE Transactions on Signal Processing*, vol. 57, no. 9, pp. 3466–3474, 2009.
- [6] H. Yan, Z. Su, H. Zhang, and F. Yang, "Observer-based H_∞ control for discrete-time stochastic systems with quantisation and random communication delays," *IET Control Theory & Applications*, vol. 7, no. 3, pp. 372–379, 2013.
- [7] D. Alcaide, A. Rodriguez-Gonzalez, and J. Sicilia, "An approach to solve the minimum expected makespan flow-shop problem subject to breakdowns," *European Journal of Operational Research*, vol. 140, no. 2, pp. 384–398, 2002.
- [8] J. Chen and F. F. Chen, "Adaptive scheduling in random flexible manufacturing systems subject to machine breakdowns," *International Journal of Production Research*, vol. 41, no. 9, pp. 1927–1951, 2003.
- [9] H. Allaoui and A. Artiba, "Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints," *Computers and Industrial Engineering*, vol. 47, no. 4, pp. 431–450, 2004.
- [10] H. Tang, C. Zhao, and C. Cheng, "Single machine stochastic JIT scheduling problem subject to machine breakdowns," *Science in China A*, vol. 51, no. 2, pp. 273–292, 2008.
- [11] M. Gholami, M. Zandieh, and A. Alem-Tabriz, "Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns," *International Journal of Advanced Manufacturing Technology*, vol. 42, no. 1-2, pp. 189–201, 2009.
- [12] M. Zandieh and M. Gholami, "An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns," *International Journal of Production Research*, vol. 47, no. 24, pp. 6999–7027, 2009.
- [13] E. Safari and S. J. Sadjadi, "A hybrid method for flowshops scheduling with condition-based maintenance constraint and machines breakdown," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2020–2029, 2011.
- [14] K. Wang and S. H. Choi, "A decomposition-based approach to flexible flow shop scheduling under machine breakdown," *International Journal of Production Research*, vol. 50, no. 1, pp. 215–234, 2012.
- [15] M. Mirabi, S. M. T. F. Ghomi, and F. Jolai, "A two-stage hybrid flowshop scheduling problem in machine breakdown condition," *Journal of Intelligent Manufacturing*, vol. 24, no. 1, pp. 193–199, 2013.
- [16] S. He, Q. H. Wu, and J. R. Saunders, "A novel group search optimizer inspired by animal behavioural ecology," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1272–1278, Vancouver, Canada, July 2006.
- [17] S. He, Q. H. Wu, and J. R. Saunders, "Group search optimizer: an optimization algorithm inspired by animal searching behavior," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 973–990, 2009.
- [18] L. Wang, X. Zhong, and M. Liu, "A novel group search optimizer for multi-objective optimization," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2939–2946, 2012.

- [19] S. He, Q. H. Wu, and J. R. Saunders, "A group search optimizer for neural network training," in *Computational Science and Its Applications-ICCSA*, Lecture Notes in Computer Science, pp. 934–943, Springer, Berlin, Germany, 2006.
- [20] D. Chen, J. Wang, F. Zou, W. Hou, and C. Zhao, "An improved group search optimizer with operation of quantum-behaved swarm and its application," *Applied Soft Computing*, vol. 12, no. 2, pp. 712–725, 2012.
- [21] Q. H. Wu, Z. Lu, M. S. Li, and T. Y. Ji, "Optimal placement of FACTS devices by a group search optimizer with multiple producer," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 1033–1039, Hong Kong, June 2008.
- [22] M. Moradi-Dalvand, B. Mohammadi-Ivatloo, A. Najafi, and A. Rabiee, "Continuous quick group search optimizer for solving non-convex economic dispatch problem," *Electric Power Systems Research*, vol. 93, pp. 93–105, 2012.
- [23] H. Shen, Y. Zhu, B. Niu, and Q. H. Wu, "An improved group search optimizer for mechanical design optimization problems," *Progress in Natural Science*, vol. 19, no. 1, pp. 91–97, 2009.
- [24] C. J. Barnard and R. M. Sibly, "Producers and scroungers: a general model and its application to captive flocks of house sparrows," *Animal Behaviour*, vol. 29, no. 2, pp. 543–550, 1981.
- [25] B. Wardono and Y. Fathi, "A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities," *European Journal of Operational Research*, vol. 155, no. 2, pp. 380–401, 2004.
- [26] C. Oğuz, Y. Zinder, V. H. Do, A. Janiak, and M. Lichtenstein, "Hybrid flow-shop scheduling problems with multiprocessor task systems," *European Journal of Operational Research*, vol. 152, no. 1, pp. 115–131, 2004.
- [27] C. Oğuz and M. F. Ercan, "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *Journal of Scheduling*, vol. 8, no. 4, pp. 323–351, 2005.
- [28] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.
- [29] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.
- [30] G. L. Deng, Z. Cui, and X. S. Gu, "A discrete artificial bee colony algorithm for the blocking flow shop scheduling problem," in *Proceedings of the 10th World Congress on Intelligent Control and Automation (WCICA '12)*, pp. 518–522, Beijing, China, July 2012.
- [31] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.
- [32] C.-J. Liao, E. Tjandradjaja, and T.-P. Chung, "An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem," *Applied Soft Computing Journal*, vol. 12, no. 6, pp. 1755–1764, 2012.
- [33] R. Zhang and C. Wu, "A hybrid differential evolution and tree search algorithm for the job shop scheduling problem," *Mathematical Problems in Engineering*, vol. 2011, Article ID 390593, 20 pages, 2011.
- [34] H. Zhang, H. Yan, F. Yang, and Q. Chen, "Distributed average filtering for sensor networks with sensor saturation," *IET Control Theory & Applications*, vol. 7, no. 6, pp. 887–893, 2013.
- [35] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, NY, USA, 7th edition, 2008.
- [36] H. Zhang, H. Yan, F. Yang, and Q. Chen, "Quantized control design for impulsive fuzzy networked systems," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 6, pp. 1153–1162, 2011.
- [37] H. Yan, H. Shi, H. Zhang, and F. Yang, "Quantized H_∞ control for networked systems with communication constraints," *Asian Journal of Control*, vol. 15, no. 5, pp. 1468–1476, 2013.