Int J Adv Manuf Technol
DOI 10.1007/s00170-013-5351-9

ORIGINAL ARTICLE

A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time

M. Akhshabi · R. Tavakkoli-Moghaddam
F. Rahnamay-Roodposhti

Received: 11 August 2011 / Accepted: 26 September 2013
© Springer-Verlag London 2013

Abstract This paper proposes a particle swarm optimization (PSO) algorithm based on memetic algorithm (MA) that hybridizes with a local search method for solving a no-wait flow shop scheduling problem. The main objective is to minimize the total flow time. Within the framework of the proposed algorithm, a local version of PSO with a ring-shape topology structure is used as global search. In addition, a self-organized random immigrant's scheme is extended into our proposed algorithm in order to further enhance its exploration capacity for new peaks in search space. The experimental study over the moving peaks benchmark problem shows that the proposed PSO-based MA is robust. Finally, the analysis of the computational results and conclusion are given.

Keywords No-wait flow shop scheduling · Total flow time · Particle swarm optimization · Memetic algorithm

1 Introduction

A no-wait flow shop problem has important usages in different industries, such as chemical processing [1], food processing [2], concrete ware production [3], and pharmaceutical processing [4]. In a no-wait flow shop, each of n jobs must be of m operations possessing a predetermined processing time. The processing of each job has to be continuous. It means when a job is started on the first machine, it must be processed through all machines without any priority and interruption. Therefore, the start of a job on the first machine should be delayed in order to satisfy the no-wait requirement if required. Considering the processing time of each job on each machine, the objective of a no-wait flow shop scheduling problem is to detect a job sequence to reduce the makespan or total flow time. For the computational complexity of this problem, Garey and Johnson [5] confirmed that it was the NP hard problem. Thus, only small-sized instances can be resolved optimally within a reasonably computational time by using exact algorithms. As the problem size increases, the computational time of exact methods raises exponentially. However, heuristic algorithms can generally obtain an optimal (or near-optimal) solution in acceptable time and memory prerequisites.

For the makespan criterion, heuristics were presented by Bonney and Gundry [6], King and Spacins [7], Gangadharan and Rajendran [8], and Rajendran [1]. As for the flow time objective, Rajendran and Chaudhuri [9] suggested a simple construction heuristic with two priority rules and Fink and Voß [10] used several simple construction heuristics, including nearest neighbor, cheapest insertion, and pilot method. With the appearance of the computer technology, meta-heuristic algorithms for no-wait flow shop scheduling problems have been used quickly. The makespan criterion has been examined by some researchers applying different meta-heuristics, such as simulated annealing (SA) by Aldowaisan and Allahverdi [11], genetic algorithm (GA) by Aldowaisan and Allahverdi [11], hybrid GA and SA by Schuster and Framinan [12], variable neighborhood search by Framinan and Schuster [13], descending search by Grabowski and Pempera [14], tabu search (TS) by Grabowski and Pempera [14], particle swarm optimization (PSO) and differential evolution (DE) algorithms by Pan [15–18], and immune

M. Akhshabi (✉)
Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
e-mail: m.akhshabi@toniau.ac.ir

R. Tavakkoli-Moghaddam
School of Industrial Engineering and Center of Excellence for Intelligence Based Experimental Mechanics, College of Engineering, University of Tehran, Tehran, Iran

F. Rahnamay-Roodposhti
Faculty of Management and Economics, Science and Research Branch, Islamic Azad University, Tehran, Iran

Published online: 17 October 2013

algorithm by Amin-Tahmasbi and Tavakkoli-Moghaddam [19]. When studying the total flow time objective, Chen et al. [20] used a GA, while Fink and Voß [10] presented SA and TS algorithms to test the set of benchmark problems suggested by Taillard. In addition, a comprehensive study of no-wait flow shop scheduling problems can be found in Hall and Sriskandarayah [2].

Memetic algorithms (MAs) have been used widely for many complex optimization problems, including scheduling problems by Ishibuchi et al. [21], Liu et al. [22], and Man et al. [23], combinatorial optimization problems by Gallardo et al. [24], multi-objective problems Goh and Tan [25], and so on. However, it is observed that the problems are mainly static problems for which MAs have been used. Araujo and Nagano [26] also developed a new constructive heuristic, namely GAPH based on a structural property for the m-machine no-wait flow shop with sequence-dependent setup times with the makespan criterion. Framinan et al. [27] addressed the problem of scheduling jobs in a no-wait flow shop with the objective of minimizing the total completion time. This problem is well-known for being non-deterministic polynomial-time hard, and therefore, most contributions to the topic focus on developing algorithms are able to obtain good approximate solutions for the problem in a short CPU time. Framinan and Nagano [28] proposed scheduling jobs in an m-machine no-wait with the objective of minimizing the maximum completion time of jobs. The problem is known to be NP hardness for more than two machines, so they focused on obtaining good (although not necessarily optimal) solutions in a short-time interval. To do so, they proposed a new heuristic based on an analogy between the problem under consideration and the well-known traveling salesman problem. Ribeiro et al. [29] addressed a novel approach to solve the m-machine no-wait flow shop scheduling problem. A continuous flow shop problem with the total flow time is considered applying a hybrid evolutionary algorithm.

Pan and Wang [30] proposed an effective composite heuristic for the blocking flow shop scheduling with makespan criterion. Nagano et al. [31] addressed the m-machine no-wait flow shop problem in which the setup time of a job is separated from its processing time. The performance measure considered is the total flow time. A new hybrid meta-heuristic method, namely genetic algorithm cluster search, is proposed to solve the scheduling problem. Gao et al. [32] presented two constructive heuristics, namely improved standard deviation heuristic and improved Bertolissi heuristic, by combining the standard deviation heuristic method and the proposed composite heuristics in order to further improve the presented constructive heuristics for the no-wait flow shop scheduling problem with the total flow time criterion. Nagano et al. [31] proposed a new constructive heuristic method, namely GAPH based on a structural property to solve the m-machine no-wait flow shop problem. Araujo and Nagano [33] addressed the problem of scheduling jobs in a no-wait flow shop problem with sequence-dependent setup times with the objective of minimizing makespan. This problem is well-known to be NP hardness, and small contribution to the problem has been made.

PSO, which simulates the social behavior of a group of fishes or birds to solve problems, has become another rapidly growing active topic over the last decades. PSO has been used for many optimization problems with desired results. In the recent years, PSO has been used to refer dynamic environments [34, 35]. On the basis of the different learning approaches of particles, PSO presents with two versions, the global version, and the local version. In the global PSO, each particle learns from the best particle in the whole swarm while in the local version each particle learns from the best particle in its neighborhood. Of these two versions, the local PSO has a slower convergence speed, and thus, it may adapt to a changing environment more easily. In many manufacturing cells, there is the constraint that all the jobs have to follow the same path, and that once a job is started, it must be processed until completion without any interruption either on or between machines. Such a flow shop is usually called no-wait flow shop. This type of scheduling is necessary where operations are required to follow one immediately after the other due to production constraints, such as during the production of steel or plastic molding. In addition, modern manufacturing environments (e.g., agile manufacturing) can frequently be modeled as a no-wait scheduling problem in which robots and industrial machines provide a highly coordinated process.

In this paper, a new PSO-based MA is proposed to solve a no-wait flow shop scheduling problem minimizing the total flow time. The results of a comprehensive computational and statistical analysis show that the proposed algorithm is very effective in terms of the solution quality and computational time. Despite its simplicity, we show that our proposed algorithm is able to improve five out of 120 best known solutions of the Taillard's flow shop benchmark considering the total flow time criterion. The remainder of this paper is outlined as follows. Section 2 reviews and presents the no-wait flow shop scheduling problem. Sections 3 and 4 describe PSO and the proposed PSO-based MA in details, respectively. Section 5 evaluates empirically the proposed algorithm on a set of benchmarks in comparison with PSO and GA. Finally, Section 6 concludes this paper with discussions on future work.

2 No-wait flow shop scheduling problem

The no-wait flow shop scheduling problem considered in this paper can be explained as follows. Each of n jobs from a set $J=\{1,2,\ldots,n\}$ is sequenced through m machines ($k=1,2,\ldots,m$). Job $j \in J$ has a sequence of m operations ($c_{j1},c_{j2},\ldots,c_{jm}$).

Operation c_{jk} conforms to the processing of job j on machine k during an uninterrupted processing time c_{jk} at any time, in which each machine can process at most one job, and each job can be processed on at most one machine. To follow the no-wait limitation, the completion time of the operation c_{jk} must be equal to the earliest begin time of the operation $c_{j,k+1}$ (k=1,2,…,$m-1$). In other words, there must be no waiting time between the processing for any sequential operations of each of n jobs. The problem is to detect a schedule in such a way that the total flow time is optimized.

Imagine that a job prioritization $p=V\{1,2,\ldots,i+1,\ldots,n\}$ shows a schedule of jobs to be processed. Let $e(\pi_{j-1},\pi_j)$ be the minimum delay on the first machine between the beginning of job (π_{j-1} and job π_j limited by no-wait constraint) when the job π_j is directly processed after the job (π_{j-1}). Then, completion time $C(\pi_j, m)$ of job π_j on machine m can be estimated by the following formula [14].

$$C(\pi_1, m) = \sum\nolimits_{k=1}^{m} p(\pi_1, k) \tag{1}$$

$$C(\pi_j, m) = \sum\nolimits_{i=2}^{j} e(\pi_{i-1}, \pi_i) + \sum\nolimits_{k=1}^{m} p(\pi_j, k); j = 2, 3, \ldots, n \tag{2}$$

$$e(\pi_{j-1}, \pi_j) = F_{j-1,j}(\pi_j, m) - \sum\nolimits_{k=1}^{m} p(\pi_j, k); j = 2, 3, \ldots, n \tag{3}$$

Where $F_{j-1,j}(\pi_j, \mathrm{m})$ suggests the makespan of jobs π_j-1 and π_j in the two-machine prioritization flow shop scheduling problem. So, the total flow time of the schedule $\pi=\{1,2,\ldots,j, i+1,\ldots,n\}$ is presented below.

$$F(\pi) = \sum\nolimits_{i=1}^{n-1} (n-i) \times e(\pi_{i-1}, \pi_{i-1}) + \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{m} p(\pi_j, j) \tag{4}$$

Therefore, the no-wait flow shop scheduling problem with the total flow time criterion is to find a permutation π^* in which the total flow time is minimized in the set of all permutations Π.

$$F(\pi^*) \leq F(\pi); \forall \pi \in \Pi \tag{5}$$

3 Particle swarm optimization

The PSO was first presented by Kennedy and Eberhart in 1995. In PSO, each potential solution is a point in the search space and may be regarded as a particle. Initially, a set of particles is randomly created and be moved through motion through the multidimensional problem space. In their movement, the particles have memory and each particle regulates its position on the base of its own experience as well as the experience of a next particle by using the best position faced by itself and its neighbors. The previous best value is known as the *pbest* of the particle, and the best value of all the particles' *pbest* in the swarm is known as the *gbest*. In each repetition, the velocity and the position of each particle will be updated on the basis of its *pbest* and *gbest*.

Imagine that the search space has d-dimensional, and the position of the ith particle at the tth iteration is shown by $X_i^t=(x_{i1}^t, x_{i2}^t, \ldots, x_{iD}^t)$. The velocity of the ith particle at the tth repetition is suggested as $V_i^t=(v_{i1}^t, v_{i2}^t, \ldots, v_{iD}^t)$. $P_i^t=(p_{i1}^t, p_{i2}^t, \ldots p_{iD}^t)$ is the best position of the ith particle at the tth iteration. The best position detected for all swarm is denoted as $P_g^t=(p_{g1}^t, p_{g2}^t, \ldots, p_{gD}^t)$ at the tth iteration. The velocity and position of particles are updated by Eqs. (6) and (7) in the standard PSO.

$$v_{id}^{t+1} = w \times v_{id}^{t+i} + c_1 \times \mathrm{rand}() \times \left(p_{id}^t - x_{id}^t\right) + c_2 \times \mathrm{rand}() \times \left(p_{gd}^t - x_{id}^t\right) \tag{6}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \tag{7}$$

Where X_i^t and V_i^t indicate the position and velocity of the dth dimension of the ith particle at the tth iteration, respectively. w is the inertia weight represented to balance between the global and local search capabilities, and rand() is a random number creator with a uniform distribution over [0, 1]. c_1 and c_2 are the acceleration constants, which suggest the weighting of stochastic acceleration terms that force each particle toward *pbest* and *gbest*, respectively.

The updating formula forces particles moving toward compound vector of local and global optimal solutions. Therefore, opportunity for particles to obtain the optimal solution will be increased.

4 PSO-based MA

Clearly, how to choose suitable particles from the current population for local promotion and how to carry out an effective local search (LS) for the selected particles must be referred when we design a PSO-based MA. Moreover, the convergence problem must be also considered when it is used. Many diversity keeping approaches have confirmed to be good choices for evolutionary algorithms to refer this problem. However, how to balance between LS and diversity keeping plans under the framework of a PSO-based MA becomes a very interesting work.

The algorithms examined in this paper are a class of PSO-based MAs, which hybridize PSO algorithms with LS methods. The PSO-based MA for a minimizing problem can be stated by the pseudo-code in Fig. 1, where *s_size* shows the population size. Within this MA, a population of *s_size* particles is first initiated with random positions and velocities and calculated via a fitness function f(.). The *pbest* of each particle is set equal to itself at the initial step. Then, some particles, chosen from the current population using a certain choice strategy, are promoted by a meme LS method, and finally,

```
Procedure: General PSO-based MA
begin
   for $i$ :=0 to s-size -1 do
      Initialize particle $i$ with random position $x_i$ and velocity $v_i$;
      Evaluate f ($x_i$);
      $p_i = x_i$
   endfor
   if LS is applied then
      Select particle from the current population for local refinement;
      Apply LS upon each of selected particles;
   endif
   for $i$ :=0 to s-size -1 do
      Update $P_{gi}$;
   endfor
   repeat
      for $i$ :=0 to s-size -1 do
         Update $v_i$ and $x_i$ according to Eqs. (6) and (7), respectively;
         Evaluate f ($x_i$);
         if f ($p_i$) > f ($x_i$) then $p_i = x_i$ endif
      endfor
      if LS is applied then
         Select particles from the population for local refinement;
         Apply LS upon each of selected particles;
      endif
      for $i$ :=0 to s-size -1 do
         Update $P_{gi}$;
      endfor
   until a stop condition is met
end
```

Fig. 1 Pseudo-code for a general PSO-based MA

the *gbest* of each particle is updated at each sequential repetition. Particles can perform their update of velocities and positions according to Eqs. (6) and (7), respectively.

If one particle can obtain a better solution than its *pbest*, its *pbest* will be replaced by the newly achieved solution. Finally, some particles are chosen to carry out local improvement before the *gbest* of each particle is updated.

4.1 Solution representation

The job-permutation-based showing has been widely applied in the literature survey for flow shop scheduling problems, and it is easy to decode to decrease the cost of the algorithm [36]. So, in this paper, we select this representation. An example is represented Table 1.

Table 1 An example of the job permutation-based representation

Dimension j	1	2	3	4	5	6	7	8	9	10
x_{ij}	6	4	2	3	1	7	9	10	8	7
Job permutation	6	4	2	3	1	7	9	10	8	7

4.2 Initial population

Each particle has its own position and velocity. The algorithm first begins by creating a population P, which is made of N_p particles. Each individual x_i also has d-dimensional parameter vectors. The index t denotes the iteration number of the algorithm. The population P is applied to show the positions of particle. The initial position population can be created randomly by using Eq. (10). m shows the number of identical parallel machines, and rand() is a random value between 0 and 1.

$$p(t) = \{x_1(t), x_2(t), \ldots . x_i(t), \ldots \ldots, x_{N_p}(t)\} \tag{8}$$

$$x_i(t) = \{x_{1,i}(t), x_{2,i}(t), \ldots \ldots \ldots, x_{j,i}(t), \ldots \ldots .., x_{D,i}(t)\} \tag{9}$$

$$x_{ij}(0) = \text{rand}() \times m + 1 \ ; \ i = 1, 2, \ldots, N_p \ ; \quad j = 1, 2, \ldots, d \ ; \ t = 1, 2, \ldots, T \tag{10}$$

A population V, which is composed of N_p individuals, suggests the velocity of each particle. It also has d-dimensional parameter vectors. The initial velocity population can be formed using Eq. (13). The velocity of each particle is equal to zeros. It means that all particles by velocity in the initial generation.

$$V(t) = \{v_1(t), v_2(t), \ldots ., v_i(t), \ldots \ldots, v_{N_p}(t)\} \tag{11}$$

$$v_i(t) = \{v_{1,i}(t), v_{2,i}(t), \ldots \ldots \ldots, v_{j,i}(t), \ldots \ldots .., v_{D,i}(t)\} \tag{12}$$

$$V(0) = \text{zeros}(N_{p,} D) \tag{13}$$

4.3 MA operations

In the following section, MA operation involving mutation operation and selection operation are described in detail.

4.3.1 Mutation

Mutation is an important operator in MA. It can maintain the diversity in the population and allow the algorithm to avoid local minima by preventing the individuals in a population from becoming too similar. The degree of maturation is inversely proportional to their parent's affinity, meaning that the greater the affinity, the lower the mutation.

In this paper, we apply a swap mutation operator to the problem. This operator just changes two genes. Randomly generate two points and the values in two positions are exchanged. An example of how to implement the swap mutation is depicted in Fig. 2.

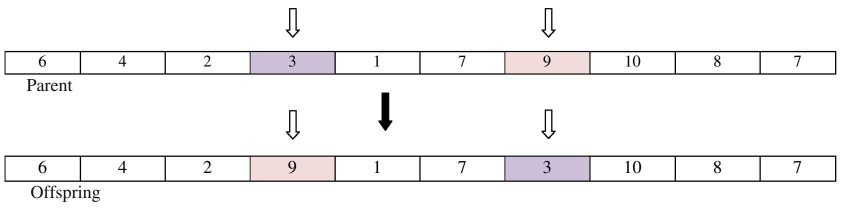

Fig. 2 Swap mutation

4.3.2 Selection

In GA, we use the roulette wheel selection mechanism. However, this operator does not warrant that each individual selected to enter the next generation is better than the last generation. In order to overcome this shortcoming, we adopt another selection strategy here. For each individual, we select the best individual to enter the next generation. In this article, we set the parameter values for MA selection as follows.

$$L = 50\% \times N_p \ ; \ \beta = 0.9 \ ; \ P_{hm} = 1$$

4.4 Updating the velocity and position

After performing the MA selection operation, the population will be done by PSO. According to Eqs. (7) and (8), the velocity and position of the particles are updated. It is important to be mention that, on the basis of updating the velocity and the position, the main difference between PSO and PSO-based MA is the promotion of the current population. As we know, in PSO, the update plan permits each particle in the swarm to regulate its own velocity and position and will not be replaced by other potential particle. However, regarding PSO-based MA, because MA can yield a more diverse solution region, some particles in the swarm may be replaced by better solutions before performing PSO. Moreover, the local best positions will be promoted due to the contribution of MA. Therefore, it is possible for all particles in the swarm to benefit from their individual, as well as the swarm community, discoveries about the solution space.

5 Computational results

There are several meta-heuristics in the literature for solving no-wait flow shop scheduling problems. Tavakkoli-Moghaddam et al. [37] proposed an immune algorithm for solving a multi-objective no-wait flow shop scheduling problem. Qian et al. [38] suggested the MA on the basis of DE to solve multi-objective no-wait flow shop scheduling problem. To match this problem, they also designed some rules, such as the largest-order-value rule, the concept of Pareto dominance, several local searchers, and a speedup computing method. Qian et al. [39] also developed an efficient hybrid DE-based algorithm for the MFSP with limited buffers between sequential machines. Javadi et al. [40] developed a fuzzy multi-objective linear programming method to resolve the multi-objective flow shop scheduling problem with particular constraint of no-wait. Pan et al. [16] suggested a discrete

Table 2 Computational results in terms of the RDI for the proposed PSO-based MA and PSO

$n \times m$	PSO-based MA	PSO
20×5	1.394	1.123
20×10	0.065	3.245
20×20	1.028	0.584
50×5	0.48	3.478
50×10	0.119	2.239
50×20	0.048	1.357
100×5	0.150	1.893
100×10	0.049	2.328
100×20	0	3.200
200×10	0	2.480
200×20	0	3.248
500×20	0	3.487
Average	0.2777	2.388

Table 3 Statistical computation results of the PSO-based MA and GA

$n \times m$	PSO-based MA		GA	
	ARDI	SD	RDI	SD
20×5	0.621	0.582	10.736	1.814
20×10	0.948	0.726	7.145	1.215
20×20	0.368	0.214	6.796	1.789
50×5	1.032	0.714	18.956	2.345
50×10	0.789	0.987	14.785	1.809
50×20	0.459	0.514	16.025	1.687
100×5	0.314	0.219	18.357	1.548
100×10	0.226	0.232	14.214	1.235
100×20	0.617	0.410	18.245	1.125
200×10	0.219	0.379	9.436	1.015
200×20	0.579	0.711	8.014	1.587
500×20	1.093	1.004	7.356	1.587
Average	0.605	0.557	2.505	1.563

Table 4 Improved OFV (in italics) in terms of the total flow time for the Taillard's benchmarks

$n\times m$	Best solution	$n\times m$	Best solution	$n\times m$	Best solution	$n\times m$	Best solution
20×5	14033	50×5	64802	100×5	253266	200×10	1046314
	15151		68051		242281		1034195
	13301		63162		237832		1046902
	15447		68226		227738		1030481
	13529		69351		240301		1034027
	13123		66841		232342		1006195
	13548		66253		240366		1053051
	13948		64332		230945		1044875
	14298		62981		247921		1026137
	12943		68770		242933		1030299
20×10	20911	50×10	87114	100×10	298358	200×20	1227733
	22440		82820		274384		1245271
	19833		79931		288114		*1254162*
	18710		86446		301044		1238349
	18641		86377		284681		1227214
	19245		86587		269686		1227604
	18363		88750		279463		1243707
	20241		86727		290908		*1235460*
	20330		85441		301970		1234936
	21320		87998		291283		1250596
20×20	33623	50×20	125831	100×20	365463	500×20	6698656
	31587		119247		372449		*6723548*
	33920		116459		370027		6739645
	31661		120261		372393		*6743598*
	34557		118184		368915		6729468
	32564		120586		370908		6724085
	32922		122880		373408		6691468
	32412		122489		384525		*6755489*
	33600		121872		374423		6711305
	32262		123954		379296		6755722

differential evolution and compared the related with another differential evolution for better Pareto solutions

In this paper, we compare the proposed PSO-based MA with PSO. We present a test bed consisting of a total of 120 problems of various sizes. These problems are divided into 12 subsets, and each subset consists of ten instances with the same size. These subsets are denoted as 20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20, 200×10, 200×20, and 500×20, representing the number of jobs and machines, respectively. In this paper, we use this test bed to test our algorithms.

The proposed PSO-based MA is coded in C^{++} on a Laptop with Pentium IV Core 2 Duo 2.53 GHz CPU. Each instance is independently run ten repetitions, in which in each replication, we compute the relative deviation index (RDI) by:

$$\mathrm{RDI}_k = \frac{F_k - \mathrm{Min}_k}{\mathrm{Max}_k - \mathrm{Min}_k} \times 100 \qquad (14)$$

Where F_k is the objective function value (OFV) obtained for the kth instance. Min_k and Max_k are the best and worst solutions obtained for each instance, respectively.

Obviously, the smaller the RDI value, the better result the algorithm supplies [41]. In addition, we calculate the average relative percentage increase (ARDI) and standard deviation (SD) over the 10 replication as statistics for the solution quality.

In this section, we compare the proposed PSO-based MA with PSO. The relative percentage increase created by both heuristics is reported in Table 2. It is clear that PSO-based MA outperforms PSO since the proposed PSO-based MA produces the much smaller overall average RDI value (i.e., 0.2777) than PSO (i.e., 2.388). We can conclude that the convergence and performance of our proposed PSO-based MA is better that PSO in terms of the RDI.

To show the efficiency of our proposed PSO-based MA, we conduct a simulation to compare the results obtained by our proposed algorithm with the GA suggested by Chen [40]. The values of parameters in the GA are the same as given in Chen's paper. The statistical performances of two algorithms are illustrated in Table 3. It can be easily seen that the PSO-based MA is the winner since it produces smaller over average ARDI (i.e., 0.605) and SD (i.e., 0.557) than the GA (i.e., 12.50 and 1.56). More attracting, the proposed PSO-based MA produces significantly better results for all the problem sizes. Hence, it is resulted that our proposed PSO-based MA is clearly superior to the GA [20] in order to solve the no-wait flow shop scheduling problem with the total flow time criterion under the same parameters and computational time.

In order to facilitate and follow up our study, we report the best known solutions found so far in Table 4. We compare the best solution found by our proposed algorithm with the solutions reported by Pan and Ruiz [42]. The best solution for each of the 120 Taillard's instances [43] is calculated by closely examining all existing results. They are applied to these 120 benchmark instances ranging from 20 jobs with five machines to 500 jobs with 20 machines by considering the processing times from the first five machines. It is interesting to see that our proposed algorithm further improves five out of 120 instances.

6 Conclusions

This paper has proposed an efficient hybrid meta-heuristic algorithm, namely PSO-based MA in order to solve the no-wait flow shop scheduling problem minimizing the total flow time. The performance of the proposed PSO-based MA has been calculated in comparison with the results obtained by the standard GA and PSO algorithms individually. The results have clearly proved that the proposed PSO-based MA has substantially outperformed GA and PSO. Future research directions can be recommenced as follows: (1) using the PSO-based MA under an uncertain environment (e.g., searching for an appropriate schedule where it is robust against some predefined interruptions), (2) using other famous meta-heuristics to compare the associated results with the ones reported by the proposed PSO-based MA, (3) considering other objective functions at different practical constraints, and (4) expanding the proposed algorithm to other similar scheduling problems

References

1. Rajendran C (1994) A no-wait flowshop scheduling heuristic to minimize makespan. J Oper Res Soc 45:472–478
2. Hall NG, Sriskandarayah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44:510–525
3. Grabowski J, Pempera J (2000) Sequencing of jobs in some production system. Eur J Oper Res 125:535–550
4. Raaymakers W, Hoogeveen J (2000) Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. Eur J Oper Res 126:131–151
5. Garey MR, Johnson DS (1979) Computers and intractability, a guide to the theory of NP-completeness. Freeman, San Francisco
6. Bonney MC, Gundry SW (1976) Solutions to the constrained flow shop sequencing problem. Oper Res Quart 27(4):869–883
7. King JR, Spachis AS (1980) Heuristics for flow shop scheduling. Int J Prod Res 18(3):343–357
8. Gangadharan R, Rajendran C (1993) Heuristic algorithms for scheduling in no-wait flow shop. Int J Prod Econ 32(3):285–290
9. Rajendran C, Chaudhuri D (1990) Heuristic algorithms for continuous flow-shop problem. Nav Res Logist 37(5):695–705
10. Fink A, Voß S (2003) Solving the continuous flow-shop scheduling problem by metaheuristics. Eur J Oper Res 151(2):400–414
11. Aldowaisan T, Allahverdi A (2003) New heuristics for no-wait flow shops to minimize makespan. Comput Oper Res 30(8):1219–1231
12. Schuster CJ, Framinan JM (2003) Appreciative procedure for nowait job shop scheduling. Oper Res Lett 31(4):308–318
13. Framinan JM, Schuster CJ (2006) An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach. Comput Oper Res 33(5):1200–1213
14. Grabowski J, Pempera J (2005) Some local search algorithms for no-wait flow-shop problem with makespan criterion. Comput Oper Res 32(4):2197–2212
15. Pan QK, Fatih Tasgetiren M, Liang YC (2008) A discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem. Comput Oper Res 35(9):2807–2839
16. Pan QK, Wang L, Qian B (2008) A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problem. Comput Oper Res 36(9):2498–2511
17. Pan QK, Wang L (2008) No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. Int J Adv Manuf Technol 39:796–807
18. Pan QK, Wang L, Tasgetirend MF, Zhao BH (2008) A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. Int J Adv Manuf Technol 38:337–347
19. Amin-Tahmasbi H, Tavakkoli-Moghaddam R (2011) Solving a bi-objective flowshop scheduling problem by a multi-objective immune system. Adv Eng Softw 42(10):772–779
20. Chen CL, Neppalli RV, Aljaber N (1996) Genetic algorithms applied to the continuous flow shop problem. Comput Ind Eng 30(4):919–929
21. Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 7(2): 204–223
22. Liu B, Wang L, Jin YH (2007) An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Trans Syst Man Cybern B 37(1):18–27
23. Man S, Liang Y, Leung KS, Lee KH, Mok TSK (2007) A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. IEEE Trans Syst Man Cybern B 37(1):84–91
24. Gallardo JE, Cotta C, Ferandez AJ (2007) On the hybridization of memetic algorithms with branch-and bound techniques. IEEE Trans Syst Man Cybern B 37(1):77–83
25. Goh CK, Tan KC (2009) A competitive-cooperation coevolutionary paradigm for dynamic multi-objective optimization. IEEE Trans Evol Comput 13(1):103–127
26. Araujo DC, Nagano MS (2010) An effective heuristic for the no-wait flowshop with sequence-dependent setup times problem. Lect Notes Comput Sci 6437:187–196

27. Framinan JM, Nagano MS, Moccellin JV (2010) An efficient heuristic for total flowtime minimisation in no-wait flowshops. Int J Adv Manuf Technol 46:1049–1057
28. Framinan JM, Nagano MS (2008) Evaluating the performance for makespan minimisation in no-wait flowshop sequencing. J Mater Process Technol 197:1–9
29. Ribeiro Filho G, Nagano MS, Lorena LAN (2007) Hybrid evolutionary algorithm for flowtime minimisation in no-wait flowshop scheduling. Lect Notes Comput Sci 4827:1099–1109
30. Pan Q-K, Wang L (2012) Effective heuristics for the blocking flow shop scheduling problem with makespan minimization. Omega 40: 218–229
31. Nagano MS, da Silva AA, Nogueira Lorena LA (2012) A new evolutionary clustering search for a no-wait flow shop problem with set-up times. Eng Appl Artif Intel 25:1114–1120
32. Gao K, Pan Q, Suganthan PN, Li J (2013) Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization. Int J Adv Manuf Technol 66:1563–1572
33. Araujo DC, Nagano MS (2011) A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem. Int J Ind Eng Comput 2(1):155–166
34. Blackwell TM, Branke J (2006) Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans Evol Comput 10(4):459–472
35. Parrott D, Li X (2006) Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans Evol Comput 10(4):440–458
36. Wang L, Zheng DZ (2003) An effective hybrid heuristic for flow shop scheduling. Int J Adv Manuf Technol 21(1):38–44
37. Tavakkoli-Moghaddam R, Rahimi-Vahed A, Mirzaei AH (2008) Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm. Int J Adv Manuf Technol 36:969–981
38. Qian B, Wang L, Huang DX, Wang X (2009) Mutli-objective no-wait flow-shop scheduling with a memetic algorithm based on differential evolution. Soft Comput 13:847–869
39. Qian B, Wang L, Huang DX, Wang WL, Wang X (2009) An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. Comput Oper Res 36:209–233
40. Javadi B, Saidi-Mehrabad M, Haji A, Mahdavi I, Jolai F, Mahdavi-Amiri N (2008) No-wait flow shop scheduling using fuzzy multiobjective linear programming. J Franklin Inst 345:452–467
41. Li JQ, Pan QK, Suganthan PN, Chua TJ (2010) A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. Int J Adv Manuf Technol 52:683–697
42. Pan Q-K, Ruiz R (2012) Local search methods for the flowshop scheduling problem with flowtime minimization. Eur J Oper Res 222:31–433
43. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285