Int J Adv Manuf Technol (2011) 55:755–762
DOI 10.1007/s00170-010-3111-7

ORIGINAL ARTICLE

Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer

J. J. Liang · Quan-Ke Pan · Chen Tiejun · Ling Wang

Received: 24 July 2010 / Accepted: 17 October 2010 / Published online: 30 December 2010

Abstract This paper presents a dynamic multi-swarm particle swarm optimizer (DMS-PSO) for solving the blocking flow shop scheduling problem with the objective to minimize makespan. To maintain good global search ability, small swarms and a regrouping schedule were used in the presented DMS-PSO. Each small swarm performed searching according to its own historical information, whereas the regrouping schedule was employed to exchange information among them. A specially designed local search phase was added into the algorithm to improve its local search ability. The experiments based on the well-known benchmarks were conducted. The computational results and comparisons indicated that the proposed DMS-PSO had a better performance on the blocking flow shop scheduling problems than some other compared algorithms in the literature.

J. J. Liang · C. Tiejun
School of Electrical Engineering, Zhengzhou University, Zhengzhou 450001, People's Republic of China

J. J. Liang
e-mail: liangjing@zzu.edu.cn

C. Tiejun
e-mail: tchen@zzu.edu.cn

Q.-K. Pan (✉)
College of Computer Science, Liaocheng University, Liaocheng 252059, People's Republic of China
e-mail: qkpan@gmail.com

L. Wang
Tsinghua National Laboratory for Information Science and Technology (TNList), Deparment of Automation, Tsinghua University, Beijing 100084, China

Keywords Flow shop scheduling · Makespan · Particle swarm optimization · Heuristics

1 Introduction

Among all types of scheduling problems, the blocking flow shop scheduling has important applications in the production environment where the processed jobs are sometimes kept in the machines because of the lack of intermediary storage [1–8] or stock is not allowed in some stages of the manufacturing process because of technology requirements [9]. This paper aims to minimize makespan for blocking flow shop scheduling problems. Minimizing makespan is important in situations where a simultaneously received batch of jobs is required to be completed as soon as possible. For example, a multi-item order submitted by a single customer needs to be delivered at the minimal possible time. The makespan criterion also increases the utilization of resources. Unlike the scheduling in the traditional flow shops, in a blocking flow shop, there are no buffers between the machines, hence a job having completed processing on a machine has to remain on this machine and block itself until the next machine is available for processing. For the computational complexity, it has been proven by NP-hard in the strong sense [10]. Therefore, heuristics and metaheuristics have been used to solve the problem in recent years because they can generally receive a near-optimal or optimal solution in acceptable computational time and memory requirements.

For the heuristics, McCormich et al. [11] considered the sequencing problems in an assembly line with blocking to minimize cycle time and developed a profile fitting method. Leisten [12] presented a comprehensive approach to deal with flow shops with finite and unlimited buffers to

maximize the use of buffers and to minimize the machine blocking. Ronconi [4] proposed a MinMax heuristic for blocking flow shop problems. Abadi et al. [13] proposed a heuristic for minimizing cycle time in a blocking flow shop. Recently, Ronconi and Henriques [14] studied the minimization of the total tardiness in blocking flow shops and presented some constructive heuristics.

With the development of computer technology, metaheuristic has been increasingly applied. Caraffa [15] developed a genetic algorithm for large size restricted slowdown flow shop problems where blocking flow shop problems were special cases, whereas Ronconi [3] proposed an interesting exact algorithm based on branch-and-bound method. Grabowski and Pempera [1] developed two tabu search approaches, called TS and TS+M, respectively. The algorithms utilize some properties of the problems associated with the blocks of jobs and multi-moves to accelerate the convergence to more promising areas of the solution space, and it was demonstrated by the authors that the algorithms outperformed both the genetic algorithm proposed by Caraffa et al. [14] and the branch-and-bound method by Ronconi [3]. Recently, Wang et al. [16] proposed an effective discrete differential evolution algorithm performing better than both TS and TS+M approaches. For the problem with total flow time minimization, Wang et al. [17] presented three harmony search algorithms providing high-quality solutions for the well-known Taillard's benchmark problems. Although metaheuristics generally provide better solutions than heuristics, it is often computationally expensive and easy to trap into local optima.

Particle swarm optimization (PSO) is one of the evolutionary algorithms based on swarm intelligence. The PSO was first designed to simulate birds seeking food. Birds would find food through social cooperation with other birds within a neighborhood. Suppose there is a group of birds searching food in an area. There are small pieces of food near the food center and usually the nearer from the food center, the bigger the food becomes. No bird knows where the food center is. So what is the best strategy to find the food? The effective one is to follow the bird which has found the biggest pieces. PSO just simulates this scenario and uses it to solve optimization problems. PSO is simple but efficient, thus in this paper we developed a dynamic multi-swarm particle swarm optimizer (DMS-PSO for short) based on the previous work [18–20] to solve this blocking flow shop scheduling problem.

The rest of this paper is organized as follows: In Section 2, the blocking flow shop scheduling problem is stated and formulated. In Section 3, a brief introduction to PSO is presented. In Section 4 the DMS-PSO algorithm with local search is described in detail. The computational results and comparisons are provided in Section 5, and some conclusions are presented in Section 6.

2 The blocking flow shop scheduling problem

In the discussed blocking flow shop scheduling problem, there are a set of n jobs, $J=\{1, 2, ..., n\}$, and a set of m machines, $M=\{1, 2, ...m\}$. Each job j ($j \in J$) will be sequentially processed on machine 1, machine 2, and so on until machine m. Operation $O(j, k)$ corresponds to the processing of job j on machine k ($k \in M$) during an uninterrupted time $p(j, k)$. Setup time can be neglected or be included into the processing time. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine. Furthermore, due to the no intermediate buffer constraint, a job cannot leave a machine until its next machine downstream is free. That is, the job is blocked if the next machine is not free. The aim is then to find a sequence for processing all jobs on all machines with the minimal makespan.

Let a job permutation $\pi=\{\pi(1), \pi(2), ..., \pi(n)\}$ represent a schedule of jobs to be processed, and $e(\pi(j), k)$ be the departure time of the operation $O(\pi(j), k)$. Then, $e(\pi(j), k)$ can be calculated as follows [3]:

$$e(\pi(1), 0) = 0 \tag{1}$$

$$e(\pi(1), k) = e(\pi(1), k-1) + p(\pi(1), k), k = 1, \cdots, m-1 \tag{2}$$

$$e(\pi(j), 0) = e(\pi(j-1), 1), j = 2, \cdots, n \tag{3}$$

$$e(\pi(j), k) = \max\{e(\pi(j), k-1)\} + p(\pi(j), k), \quad e(\pi(j-1), k+1), j = 2, \cdots, n, k = 1, \cdots, m-1 \tag{4}$$

$$e(\pi(j), m) = e(\pi(j), m-1) + p(\pi(j), m), j-1, \cdots, n \tag{5}$$

where $e(\pi(j), 0), j = 1, \cdots, n$, denotes the starting time of job $\pi(j)$ on the first machine.

The above recursive equations first calculate the departure times for the jobs one by one. Since $C(\pi(j), m)=e(\pi(j), m)$ for all j, the makespan $C_{\max}(\pi)$ of the job permutation π can be given as follows:

$$C_{\max}(\pi) = e(\pi(n), m) \tag{6}$$

Obviously, the measure can be calculated in time $O(nm)$.

Thus, the objective of the paper is to find a permutation π^* in the set of all permutations Π such that

$$C_{\max}(\pi^*) \leq C_{\max}(\pi) \forall \pi \in \Pi. \tag{7}$$

3 Brief introduction to PSO algorithm

In PSO, each single solution is regarded as a "bird" in the search space and it is called a "particle." All particles have fitness values which are evaluated by the fitness function to be optimized and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current best positions found so far by the swarm. Therefore, the particles have a tendency to fly towards the potentially better search area over the course of the search process. Since its introduction in 1995 by Kennedy and Eberhart [21, 22], PSO has attracted a lot of attention as evidenced by the research results have been reported.

The original PSO algorithm is described as below:

$$v_i(j) \leftarrow v_i(j) + c_1\text{*rand1}_i(j)\text{*}(\text{pbest}_i(j) - x_i(j)) + c_2\text{*rand2}_i(j)\text{*}(\text{gbest}(j) - x_i(j)) \quad (8)$$

$$x_i(j) \leftarrow x_i(j) + v_i(j) \quad (9)$$

where $X_i = \{x_i(1), x_i(2), ..., x_i(n)\}$ in the equation represents the position of the ith particle; $\text{Pbest}_i = \{\text{pbest}_i(1), \text{pbest}_i(2), ..., \text{pbest}_i(n)\}$ represents the best previous position (the position giving the best fitness value) of the ith particle; $\text{Gbest} = \{\text{gbest}(1), \text{gbest}(2), ..., \text{gbest}(n)\}$ represents the best previous position of the population; $V_i = \{v_i(1), v_i(2), ..., v_i(n)\}$ represents the rate of the position change (velocity) for particle i; c_1 and c_2 are the acceleration constants, which represent the weighting of stochastic acceleration terms that pull each particle toward Pbest and Gbest positions. $\text{rand1}_i(j)$and $\text{rand2}_i(j)$are two random numbers in the range [0,1].

Shi and Eberhart introduced one new parameter, the inertia weight, into the original PSO algorithm [23]. The velocity updating equation of PSO with inertia weight is:

$$v_i(j) \leftarrow \omega\text{*}v_i(j) + c_1\text{*rand1}_i(j)\text{*}(\text{pbest}_i(j) - x_i(j)) + c_2\text{*rand2}_i(j)\text{*}\left(\text{gbest}^d - x_i(j)\right) \quad (10)$$

Equation 10 is the same as Eq. 9 except the inertia weight ω. The inertia weight is an important parameter which is used to balance the global and local search capabilities. A large inertia weight facilitates global search, while a small inertia weight facilitates local search.

4 The proposed DMS-PSO for blocking flow shop

In this section, we will present a DMS-PSO for the blocking flow shop scheduling problem after detailing solution representation, initialization method, DMS-PSO search, and local search. The key components of the presented DMS-PSO algorithm are elucidated as follows.

4.1 Solution representation

In PSO, a particle, $X_i = \{x_i(1), x_i(2), .., x_i(n)\}$, is represented as an n-dimensional real-number vector. It is necessary to convert it to a job permutation for evaluating the objective value. Let each index of the dimensions of the vector represents a typical job from $J = \{1, 2, .., n\}$, and then n indexes denote n different jobs. Thereafter, the largest position value rule is employed to obtain a job permutation $\pi = \{\pi(1), \pi(2), .., \pi(n)\}$ by ordering the jobs in their non-increasing position value of X_i. A simple example is illustrated in Fig. 1.

4.2 Initialization

In order to improve the stability and efficiency of the algorithm, an initial particle swarm with a certain level of quality and diversity would be helpful, thus we utilize the NEH heuristic [24] and its variants to produce an initial population. The NEH heuristic is a famous method for permutation flow shop scheduling with makespan criterion, whose procedure is given as follows [24]:

Step 1 Sort the jobs according to the ascending sums of their process times. Let the obtained sequence be $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$.

Step 2 The first two jobs of π are taken and the two possible partial sequences of these two jobs are evaluated. Then, the better partial sequence is chosen as the current sequence.

Step 3 Take job $\pi(j)$, $j = 3, 4, .., n$, and find the best partial sequence by placing it in all possible positions of the partial sequence of jobs that have been already scheduled. Then the best partial sequence is selected for the next generation. (If several partial sequences have the same minimum fitness value, randomly select one as the best partial sequence). Repeat this step until all jobs are sequenced and the final sequence of n jobs is constructed.

Framinan et al. [25] investigated the phases of the NEH heuristic [24] and their contribution to its excellent performance regarding makespan minimization. The results show that the excellent performance of NEH is primarily due to the large number of subsequence evaluated in the step 3 of the heuristics. Thus, we propose a variant of the NEH heuristics where the initial job sequence in step 1 is generated randomly and then step 2 and step 3 are sequentially used to improve the seed sequence. In this way, we can obtain a large number of permutations with high quality.

Dimension/job j	1	2	3	4	5	6
$x(j)$	-0.4	0.6	0.3	0.5	-0.1	0.0

A harmony vector X

Job permutation π	2	4	3	6	5	1
$x(j)$ in non-increasing order	0.6	0.5	0.3	0.0	-0.1	-0.4

A job permutation π

Fig. 1 An example for the LPV rule

Since the initial solutions generated by the NEH heuristic and its variants are job sequences, they should be converted to particles represented by real-number vectors. Let $\pi=\{\pi(1), \pi(2), .., \pi(n)\}$ be a job sequence, and X be a particle. A simple conversion is performed by using the following equation.

$$x(\pi(j)) = x^{\max} - \frac{x^{\max}(j) - x^{\min}(j)}{n-1} \times (j-1), \quad j = 1, 2, \cdots n \tag{11}$$

where $x_{\min}(j)=-1.0$ and $x_{\max}(j)=1.0$ for each dimension j and r is a uniform random number between 0 and 1 for each dimension.

Finally, the initial procedure is described below.

Step 1 Generate a sequence π_1 using the NEH heuristic, and yield the rest NP-1 sequences π_i, i=2, 3, ..., NP by the variant of the NEH heuristic. Calculate the makespan $f(\pi_i)$ for π_i, i=1, 2, ..., NP.

Step 2 Convert the NP sequences to NP particles using Eq. 11. Let the fitness of X_i, $f(X_i)$, be equal to $f(\pi_i)$ for i=1, 2, ..., NP.

Step 3. Step 3 For each particle X_i, the velocity $V_i=\{v_i(1), v_i(2), ..., v_i(n)\}$ is generated by $v_i(j) = v_{\min} + (v_{\max} - v_{\min}) \times \text{rand}()$

Step 4 Set $\text{Pbest}_i=X_i$ and $f(\text{Pbest}_i)=f(X_i)$ for i=1, 2, ..., NP.

Step 5 Find the particle with minimum makespan value and set it as Gbest.

In our procedure, we also keep the sequences implied by the pbest in the memory for facility our local search algorithm.

4.3 DMS-PSO search

There are two versions of PSO, namely the global and the local PSO. In the global version of PSO, each particle dynamically adjusts its velocity and position according to the best solution found so far by itself and the best solution found so far by the whole swarm. While in the local version of PSO, each particle adjusts its velocity and position according to its personal best and the best solution achieved so far within its neighborhood and comparing with the global version, the local version of particle swarm optimizer has better global search ability [26].

The dynamic multi-swarm particle swarm optimizer (DMS-PSO) [18–20] is constructed based on the local version of PSO and a periodically changed neighborhood structure is used. In this new neighborhood structure, small neighborhoods are used. In order to slow down the population's convergence velocity and increase the diversity, we divide the population into small sized sub-swarms in the DMS-PSO. Each swarm uses its own members to search for a better area in the search space, and since the small sized sub-swarms are searching using their own best historical information, they are easy to converge to a local optimum because of the convergence property of PSO. In order to exchange information among the sub-swarms, a randomized regrouping schedule is employed. An example of the regrouping schedule for three swarms with three particles in each swarm is shown in Fig. 2. Every R generations, the population is regrouped randomly and starts searching using a new configuration of small swarms. Here, R is called regrouping period. In this way, the good information obtained by each swarm is exchanged among the swarms and the diversity of the population is increased simultaneously. The new neighborhood structure has more freedom when compared with the classical neighborhood structure. It is not surprising that it performs better on complex multimodal problems.

4.4 Local search

A larger diversity and a faster convergence velocity are always a trade-off problem. Since we achieve a larger

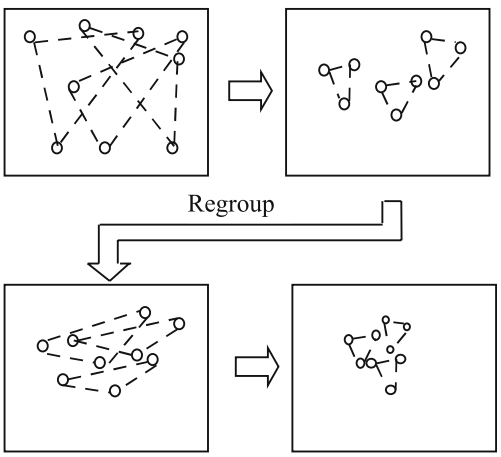

Fig. 2 DMS-PSO's search

diversity using DMS-PSO, at the same time, we lose the fast convergence velocity. In order to alleviate this weakness and give a better search in the better local areas, a problem-dependent local search is added into DMS-PSO. Every R generations, before we regroup the sub-swarms, we will refine the Lbest of each group using the problem-dependent local search method and replace the Lbest with the new solution found by the local search if it is better than Lbest. Since we have the sequences implied by the Lbests in the memory, we can easy perform permutation-based neighborhood search to it. According to [16], an insert operator is the most suitable for performing a fine local search. Thus, an insertion heuristic is presented for stressing exploitation. The insertion heuristic is based on a first-improvement type of pivoting rule. Let $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$ be the job permutation to undergo the insertion heuristic, $\alpha = \{\alpha(1), \alpha(2), ..., \alpha(n)\}$ be a reference job permutation generated randomly, and $\phi = \{\phi(1), \phi(2), ..., \phi(n)\}$ be the final sequence by the insertion heuristic. The procedure for the insertion heuristic is given as follows:

Step 1 Set Cnt = 0, j=0, and ϕ is obtained by performing two or three insertion moves to π.

Step 2 Set $j = j + 1$. If $j > n$, then let $j = j - n$.

Step 3 Remove the job that corresponds to job $\alpha(j)$ from its original position in the permutation ϕ, and insert it into all the other possible slots, respectively. Denote ω as the best obtained sequence.

Step 4 If ω is better than ϕ, then let $\phi = \omega$ and Cnt=0; Otherwise, let Cnt = Cnt + 1.

Step 5 If Cnt$\leq n$, then go back to step 2, otherwise stop the procedure and return ϕ.

Note that for the blocking flow shop scheduling problem with the makespan criterion, the speedup for the NEH

Fig. 3 DMS-PSO

ps: Each sub-swarm's population size ps=3

sn: Sub-swarms' number $sn = ceil(\sqrt{n})$

R: Regrouping period $R = 20$

$c_1 = 2$ $c_2 = 2$ $v_{max} = 1$

Initialize *ps*sn* particles (position and velocity)

Divide the population into *sn* swarms randomly, with *ps* particles in each swarm.

gen = 0

While $t < T$ (T is set to $5mn$ milliseconds in the experiments)

gen = *gen*+1;

ω = 0.9-max(0.5*gen/5000, 0)

For $i = 1 : ps*sn$

Find $Lbest_i$

For $j = 1{:}n$

$$v_i(j) = w * v_i(j) + c_1 * rand1_i(j) * (pbest_i(j) - x_i(j)) + c_2 * rand2_i(j) * (lbest_i(j) - x_i(j))$$

$$v_i(j) = \min(\max(v_i(j), -v_{\max}), v_{\max})$$

$$x_i(j) = x_i(j) + v_i(j)$$

End

Calculate the fitness value

Update *Pbest*

End

If *mod* (*gen*, *R*) = 0,

For $j = 1 : sn$

Do the local search for sub-swarm j

Update *Lbest* for sub-swarm j

End

Regroup the swarms randomly

End

End

heuristic developed by Taillard [27] can be adapted here to reduce the CPU time requirements.

4.5 The procedure for the proposed DMS-PSO

Based on the above design, we present a DMS-PSO algorithm for blocking flow shop scheduling problem, of which the computational complexity is $O(n^{3/2}m)$, and its procedure is given as follows (Fig. 3).

5 Computational results

There are several metaheuristics existing in the literature for solving the blocking flow shop scheduling problems with makespan criterion, e.g., the genetic algorithm (GA) proposed by Caraffa et al. [14] and a branch-and-bound based heuristic algorithm (RON) by Ronconi [3]. Grabowski and Pempera [1] developed two tabu search approaches (TS and TS+M) which provided better results than the GA of Caraffa et al. [14] and the RON of Ronconi [3]. Recently, Qian et al. [2] presented a hybrid DE-based (HDE) algorithm for multi-objective flow shop scheduling problems with limited buffers between consecutive machines, and demonstrated the effectiveness and efficiency of their HDE algorithm. In this paper, we compare the proposed DMS-PSO with TS/TS+M and HDE algorithms.

For the classic permutation flow shop scheduling problem with makespan criterion, Taillard [27] presented 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 respectively 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, where the first number is the job size and the second is the machine size. In this paper, we use this test bed to test our methods.

The DMS-PSO algorithm is coded in C++ and the experiments are executed on a Pentium P-IV 3.0 GHz PC with 512 MB memory. The maximum computational time is set as T=5 mn milliseconds. Each instance is independently run ten replications and in each replication we compute the percentage relative difference (PRD) as follows:

$$\mathrm{PRD} = \frac{100 \times (C^{\mathrm{Ron}} - C^{A})}{C^{\mathrm{Ron}}} \tag{12}$$

where C^{Ron} is the makespan provided by Ronconi [3], and C^{A} is the makespan found by our DMS-PSO algorithm. Furthermore, average percentage relative difference (APRD), maximum percentage relative difference, minimum percentage relative difference, and the standard

Table 1 Computation results of the DMS-PSO and HDE algorithms

	DMS-PSO					TS		TS+M		HDE				
$n \times m$	APRD	MinPRD	MaxPRD	SD	T (s)	APRD	T	APRD	T (s)	APRD	MinPRD	MaxPRD	SD	T (s)
20×5	**0.30**	0.09	0.36	0.16	0.50	−1.64	2.4	−0.34	2.7	0.26	0.00	0.44	0.16	0.50
20×10	**2.32**	1.55	2.39	0.11	1.00	1.45	4.1	1.76	4.6	2.30	2.13	2.39	0.10	1.00
20×20	**3.26**	3.02	3.30	0.04	2.00	2.88	7.1	2.94	7.6	3.25	3.14	3.30	0.06	2.00
50×5	2.78	2.36	3.12	0.30	1.25	−0.55	6.0	0.55	6.2	**3.09**	2.59	3.62	0.36	1.25
50×10	4.09	3.77	3.45	0.29	2.50	1.98	10.6	3.52	10.8	**4.57**	4.10	5.06	0.31	2.50
50×20	4.91	4.56	5.09	0.26	5.00	3.68	19.0	4.26	19.3	**5.00**	4.58	5.51	0.30	5.00
100×5	**1.01**	0.58	1.52	0.39	2.50	−3.03	12.2	−2.62	12.4	−0.32	−0.84	0.31	0.36	2.50
100×10	**4.43**	3.91	4.87	0.38	5.00	1.71	21.9	2.66	22.1	3.40	2.88	3.99	0.35	5.00
100×20	**4.72**	4.21	5.14	0.37	10.00	2.01	39.2	3.03	39.4	3.05	2.69	3.47	0.26	10.00
200×10	**1.93**	1.43	2.36	0.38	10.00	−0.60	44.1	0.58	44.3	0.33	−0.14	0.88	0.34	10.00
200×20	**2.94**	2.49	3.31	0.33	20.00	1.24	79.2	2.31	79.4	1.36	1.00	1.82	0.26	20.00
500×20	**2.00**	1.74	2.28	0.21	50.00	0.63	207	1.47	209	0.25	−0.06	0.59	0.20	50.00
Mean	**2.89**	2.48	3.1	0.27	9.15	0.81	37.73	1.68	38.15	2.21	1.84	2.62	0.26	9.15

CPU times of TS and TS+M on Pentium IV, 1,000 MHZ

The minimum APRD values are in bold, *MaxPRD* maximum percentage relative difference, *MinPRD* minimum percentage relative difference

deviation (SD) of PRD are calculated as the statistics for the performance measures. For the HDE algorithm, we modify it for the blocking flow shop scheduling problem and recode it in C++. The parameters is consistent with those in the original paper [2] and the maximum computation time is also set as T=5 mn milliseconds. For the TS/TS+M algorithm, we directly adopt the results reported in the original paper since the technique of the blocks of jobs and multi-move is adopted by the authors and it is very difficult to re-realize [28]. Table 1 reports the experiment results.

From Table 1, we can observe that the proposed DMS-PSO gives the best performance in terms of the overall solution quality, since it yields the largest overall mean ARPD value equal to 2.89%, which is much better than those by the HDE (2.21%), TS (0.81%), and TS+M (1.68%). More specifically, the DMS-PSO produces much better APRD than HDE for nine out of 12 group instances, and provides much larger APRD than both TS and TS+M for all the groups. As the size of instances increases, the superiority of the DMS-PSO algorithm over the HDE algorithm increases. In addition, the mean SD value resulting from the DMS-PSO algorithm is very small, demonstrating the robustness of the DMS-PSO algorithm on the initial conditions. So it is clear from the these results that the proposed DMS-PSO outperforms the HDE, TS and TS+M algorithms by a considerable margin for blocking flow shop scheduling problems to minimize makespan.

Next, we compare DMS-PSO, DMS-PSO without local search (DMS-PSOnL for short), and the PSO algorithm [23]. The computational results are given in Table 2.

It can be seen from Table 2 that the DMS-PSOnL algorithm generates much worse results than the DMS-PSO algorithm, which demonstrates the effectiveness of incorporating a local search into DMS-PSO variant. The superiority of the DMS-PSO in terms of search ability and efficiency should be attributed to the combination of global search and local search as well as the balance between exploration and exploitation. However, DMS-PSOnL produces better APRD value than PSO, suggesting the effectiveness of the small swarms and a regrouping schedule mechanism used in the presented DMS-PSO algorithm. That is to say, the performance of the PSO algorithm can be improved by dividing the whole swarm into a number of small sub-swarms and performing evolution independently and exchanging information after a fixed period.

6 Conclusions

We extended the DMS-PSO algorithm, which has proven its good global search ability in continuous problems, to solve blocking flow shop scheduling problems in this paper. In the proposed algorithm, we not only use small swarms and regrouping scheme, but also employ a special designed local search operator to improve the local search ability of the algorithm. The experiments show it is a promising algorithm on this kind of problems. It is better than some other algorithms on most group instances, and it is especially better and more stable on the high dimensional problems. In the future, we will consider adding some discrete operators in DMS-PSO to improve its performance and design some better solution representation methods to improve its efficiency.

Table 2 Computation results of DMS-PSO, DMS-PSOnL, and PSO algorithms

	DMS-PSO				DMS-PSOnL				PSO			
$n\times m$	APRD	MinPRD	MaxPRD	SD	APRD	MinPRD	MaxPRD	SD	APRD	MinPRD	MaxPRD	SD
20×5	0.30	0.09	0.36	0.16	−2.35	−2.95	−1.86	0.52	−3.39	−3.86	−2.93	0.39
20×10	2.32	1.55	2.39	0.11	0.57	0.04	1.19	0.50	−1.03	−1.64	−0.25	0.55
20×20	3.26	3.02	3.30	0.04	1.91	1.71	2.46	0.31	1.33	0.90	1.83	0.37
50×5	2.78	2.36	3.12	0.30	−1.40	−2.11	−0.88	0.50	−1.66	−2.13	−1.06	0.44
50×10	4.09	3.77	3.45	0.29	0.76	−0.26	1.12	0.55	0.20	−0.19	0.75	0.38
50×20	4.91	4.56	5.09	0.26	0.92	0.22	1.49	0.52	0.72	0.44	0.97	0.22
100×5	1.01	0.58	1.52	0.39	−3.84	−4.16	−3.61	0.22	−3.92	−4.13	−3.63	0.20
100×10	4.43	3.91	4.87	0.38	0.19	−0.19	0.32	0.20	0.10	−0.16	0.35	0.21
100×20	4.72	4.21	5.14	0.37	1.05	0.95	1.29	0.13	1.14	0.98	1.32	0.14
200×10	1.93	1.43	2.36	0.38	−1.18	−1.40	−1.15	0.10	−1.19	−1.30	−1.05	0.10
200×20	2.94	2.49	3.31	0.33	0.37	0.27	0.48	0.09	0.45	0.33	0.60	0.11
500×20	2.00	1.74	2.28	0.21	0.20	0.11	0.26	0.02	0.18	0.14	0.23	0.04
Mean	2.89	2.48	3.10	0.27	−0.23	−0.65	0.09	0.31	−0.59	−0.89	−0.24	0.262

MaxPRD maximum percentage relative difference, *MinPRD* minimum percentage relative difference

Acknowledgements This research is partially supported by the National Science Foundation of China (60874075, 60905039, 70871065) and Science Foundation of Shandong Province, China (BS2010DX005).

References

1. Grabowski J, Pempera J (2007) The permutation flow shop problem with blocking. A tabu search approach. OMEGA, The international Journal of Management Science 35:302–311
2. Qian B, Wang L, Huang DX, Wang WL, Wang X (2007) An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. Comput Oper Res. doi:10.1016/j.cor.2007.08.007
3. Ronconi DP (2005) A branch-and-bound algorithm to minimize the makespan in a flowshop problem with blocking. Ann Oper Res 138(1):53–65
4. Ronconi DP (2004) A note on constructive heuristics for the flowshop problem with blocking. Int J Prod Econ 87:39–48
5. 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(7–8):796–807
6. Pan Q, Wang L, Zhao B (2008) An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. Int J Adv Manuf Technol 38(7–8):778–786
7. Pan Q, 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(3–4):337–347
8. Pan Q, Wang L, Gao L, Li J (In press) An effective shuffled frog-leaping algorithm for lot-streaming flow shop scheduling problem. Int J Adv Manuf Tech
9. Grabowski J, Pempera J (2000) Sequencing of jobs in some production system. Eur J Oper Res 125:535–550
10. Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44:510–525
11. McCormich ST, Pinedo ML, Shenker S, Wolf B (1989) Sequencing in an assembly line with blocking to minimize cycle time. Oper Res 37:925–936
12. Leisten R (1990) Flowshop sequencing problems with limited buffer storage. Int J Prod Res 28:2085–2100
13. Abadi INK, Hall NG, Sriskandarajh C (2000) Minimizing cycle time in a blocking flowshop. Oper Res 48:177–180
14. Ronconi DP, Henriques LRS (2007) Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. OMEGA-Int J Manage S. doi:10.1016/j.omega.2007.01.003
15. Caraffa V, Ianes S, Bagchi TP, Sriskandarajah C (2001) Minimizing makespan in a blocking flowshop using genetic algorithms. Int J Prod Econ 70:101–115
16. Wang L, Pan Q, Suganthan PN, Wang W, Ya-Min (2010) A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Computers & Operations Research 37 (3):509–520
17. Wang L, Pan Q, Tasgetiren MF (2010) Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. Expert Syst Appl 37(12):7929–7936
18. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer. Proceedings 2005 IEEE Int. Swarm Intelligence Symposium, pp 124–129. Pasadena, CA, USA, June 2005
19. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer with local search. Proceedings of IEEE Congress on Evolutionary Computation CEC 1:522–528
20. Liang JJ, Suganthan PN (2006) Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. Proceedings of IEEE Congress on Evolutionary Computation 1:9–16
21. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, pp 39–43
22. Kennedy J, Eberhart RC (1995) Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, pp 1942–1948
23. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. Proceedings of the IEEE Congress on Evolutionary Computation, pp 69–73
24. Nawaz M, Enscore EEJ, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. Omega 11:91–95
25. Framinan JM, Leisten R, Ruiz-Usano R (2002) Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimization. Eur J Oper Res 141:559–569
26. Kennedy J, Mendes R (2002) Population structure and particle swarm performance. Proceedings of the IEEE Congress on Evolutionary Computation, pp 1671–1676
27. Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problems. Eur J Oper Res 47:65–74
28. Ruiz R, Maroto CA (2005) Comprehensive review and evaluation of permutation flowshops heuristics. Eur J Oper Res 165:479–494