An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling

Bo Liu, Ling Wang, and Yi-Hui Jin

***Abstract*—This paper proposes an effective particle swarm optimization (PSO)-based memetic algorithm (MA) for the permutation flow shop scheduling problem (PFSSP) with the objective to minimize the maximum completion time, which is a typical non-deterministic polynomial-time (NP) hard combinatorial optimization problem. In the proposed PSO-based MA (PSOMA), both PSO-based searching operators and some special local searching operators are designed to balance the exploration and exploitation abilities. In particular, the PSOMA applies the evolutionary searching mechanism of PSO, which is characterized by individual improvement, population cooperation, and competition to effectively perform exploration. On the other hand, the PSOMA utilizes several adaptive local searches to perform exploitation. First, to make PSO suitable for solving PFSSP, a ranked-order value rule based on random key representation is presented to convert the continuous position values of particles to job permutations. Second, to generate an initial swarm with certain quality and diversity, the famous Nawaz-Enscore-Ham (NEH) heuristic is incorporated into the initialization of population. Third, to balance the exploration and exploitation abilities, after the standard PSO-based searching operation, a new local search technique named NEH_1 insertion is probabilistically applied to some good particles selected by using a roulette wheel mechanism with a specified probability. Fourth, to enrich the searching behaviors and to avoid premature convergence, a simulated annealing (SA)-based local search with multiple different neighborhoods is designed and incorporated into the PSOMA. Meanwhile, an effective adaptive meta-Lamarckian learning strategy is employed to decide which neighborhood to be used in SA-based local search. Finally, to further enhance the exploitation ability, a pairwise-based local search is applied after the SA-based search. Simulation results based on benchmarks demonstrate the effectiveness of the PSOMA. Additionally, the effects of some parameters on optimization performances are also discussed.**

***Index Terms*—Adaptive meta-Lamarckian learning, memetic algorithm (MA), particle swarm optimization (PSO), permutation flow shop scheduling.**

I. Introduction

PRODUCTION scheduling plays a key role in manufacturing systems of an enterprise for maintaining competitive position in the fast changing market. To account for this factor, it is important to develop effective and efficient advanced manufacturing and scheduling technologies and approaches [1], [2]. The flow shop scheduling problem (FSSP) is a class of widely studied problems based on ideas gleaned from engineering, which currently represent nearly a quarter of manufacturing systems, assembly lines, and information service facilities [3], and has earned a reputation for being difficult to solve [4]. As a simplification of FSSP, the permutation FSSP (PFSSP), i.e., the processing sequence of all jobs is the same for all machines, for minimizing the maximum completion time (i.e., makespan) is a typical and well-studied non-deterministic polynomial-time (NP) hard problem [4]. Due to its significance both in theory and engineering applications, it is important to develop effective and efficient approaches for PFSSP.

Since the pioneering work of Johnson [5], PFSSP has received considerable theoretical, computational, and empirical research work. To account for the difficulty of PFSSP, a large amount of optimization techniques has been proposed [6]. Yet, due to the complexity of PFSSP, exact solution techniques, such as branch and bound, and mathematical programming [7], are only applicable to small-scale problems. Thus, various heuristics have been proposed, including constructive heuristics, improvement heuristics, and their hybrids (sometimes called memetic algorithms (MAs) [8]). The constructive heuristics build a feasible schedule from scratch, mainly for solving the two- and three-machine scheduling problems. But, the solution qualities of the constructive heuristics are not satisfactory, although the process is very fast. Experimental results showed that the Nawaz-Enscore-Ham (NEH) heuristic [9] is currently one of the best constructive heuristics. The improvement heuristics start from one or a set of solutions generated by some sequence generators and try to improve the solution(s) by applying some specific problem knowledge to approach the "global" or "suboptimal" optimum. Improvement heuristics are generally based on metaheuristics such as simulated annealing (SA) [10], genetic algorithm (GA) [11], [12], tabu search (TS) [13], evolutionary programming [14], and variable neighborhood search (VNS) [15]. Improvement heuristics can obtain fairly satisfactory solutions, but they are often time-consuming and parameter dependent.

Recently, hybrid heuristics have been a hot topic in the fields of both computer science and operational research [2], [6], [12], [16]–[20]. It assumes that combining the features of different methods in a complementary fashion may result in more robust and effective optimization tools. Particularly, it is well known that the performances of evolutionary algorithms can be improved by combining problem-dependent local searches. MAs [8], [21] may be considered as a union of population-based global search and local improvements which are inspired by Darwinian principles of natural evolution and

Manuscript received April 29, 2005; revised August 23, 2005. This work was supported in part by the National Science Foundation of China under Grants 60204008, 60374060, and 60574072 as well as the 973 Program under Grant 2002CB312200. This paper was recommended by Guest Editor Y. S. Ong.

The authors are with the Institute of Process Control, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: liub01@mails.tsinghua.edu.cn; wangling@mail.tsinghua.edu.cn; jyh-dau@mail.tsinghua.edu.cn).

Digital Object Identifier 10.1109/TSMCB.2006.883272

Dawkins' notion of a meme, defined as a unit of cultural evolution that is capable of local refinements. So far, MAs have gained wide research on a variety of problems such as travelling salesman problem (TSP) [22], the graph bipartition problem [22], the quadratic assignment problem [22], cellular mobile networks [23], and scheduling problems [24]. In MAs, several studies [21], [25] have been focused on how to achieve a reasonable combination of global and local searches and how to make a good balance between exploration and exploitation. Recently, a multiagent metaheuristic architecture [26] was introduced as a conceptual and practical framework for designing MAs. Traditionally, most of the MAs rely on the use of one single evolutionary algorithm for globally rough exploration and one single local search for locally fine improvements. Some recent studies [21], [27]–[29] on the choice of local searches have shown that the choice significantly affects the searching efficiency. In order to avoid the negative effects of incorrect local search, Ong and Keane [28] coined the term "meta-Lamarckian learning" to introduce the idea of adaptively choosing multiple memes during a MA search in the spirit of Lamarckian learning and successfully solved continuous optimization problems by the proposed MA with multiple local searches. In [29], a classification of memes adaptation in adaptive MAs was presented on the basis of the mechanism used and the level of historical knowledge on the memes employed, and the global convergence properties of adaptive MAs were analyzed by means of finite Markov chain. In addition, a study in [30] focused on the surrogate-assisted MAs framework for computationally expensive problems and robust designing problems.

MAs have already been investigated for single-objective PFSSP in many studies. In [12], a multistep crossover fusion operator carrying a biased local search was introduced into GA. In [16], a hybrid GA was presented, where the standard mutation was replaced by SA and multiple crossover operators were applied to subpopulations. In [17], an ant-colony system was proposed, which was enhanced by a fast local search to yield high-quality solutions. In [18], two hybrid versions of GA, i.e., genetic SA and genetic local search, were presented, where an "improvement" phase with local search and SA, respectively, was performed before selection and crossover operation. In [19], a hybrid SA was integrated with features borrowed from the GA and local searches, which worked from a population of candidate schedules, and generated new populations by applying suitable small perturbation schemes. Moreover, during the annealing process, an iterated hill-climbing procedure was stochastically applied to the population. In [20], an ordinal optimization-based GA was presented to ensure the quality of the solution found with a reduction in computation effort. In addition, a hypothesis-test-based GA was proposed in [31] for stochastic PFSSP with uncertain processing time. As for the multiobjective scheduling problems, the hybridization with local search was first implemented in [32] as a multiobjective genetic local search (MOGLS), where a scalar fitness function with random weights was used for the selection of parents and the local search for their offspring. In [25], the former MOGLS [32] was modified by choosing only good individuals as initial solutions for local search and assigning an appropriate local search direction to each initial solution. Meanwhile, the importance of balance between genetic and local searches was stressed.

Recently, a new evolutionary technique, particle swarm optimization (PSO) [33], was proposed for unconstrained continuous optimization problems. Its development is based on observations of the social behaviors of animals such as bird flocking, fish schooling, and swarm theory. It is initialized with a population of random solutions. Each individual is assigned with a randomized velocity according to his own and his companions' flying experiences. The individuals, called particles, are then flown through the hyperspace. PSO has some attractive characteristics. Because it has memory, knowledge of good solutions is retained by all particles. Additionally, it has constructive cooperation between particles, and particles in the swarm share information between them. Due to the simple concept, easy implementation, and quick convergence, PSO has gained much attention and wide applications in different fields, mainly for continuous optimization problems [34]. However, the performance of a simple PSO depends on its parameters, and it often suffers from the problem of being trapped in local optima. Some studies have been carried out to prevent premature convergence and to balance the exploration and exploitation abilities [35]. In addition, most of the published work on PSO is for continuous optimization problems, whereas little research is found for combinatorial optimization problems. Obviously, it is a challenge to employ PSO to different areas of problems other than those areas the inventors originally focused on. To the best of our knowledge, there is little published work on PSO for scheduling problems. Recently, a hybrid PSO [36] based on VNS was proposed for PFSSP. In this paper, we will propose a PSO-based MA (PSOMA) for PFSSP. PSOMA applies the evolutionary searching mechanism of PSO, which is characterized by individual improvement, population cooperation, and competition to effectively perform exploration. On the other hand, PSOMA utilizes several adaptive local searches to perform exploitation. The features of PSOMA can be summarized as follows. First, to make PSO suitable for solving PFSSP, a ranked-order value (ROV) rule based on random key representation [37] is presented to convert the continuous position values of a particle to a job permutation. Second, the NEH heuristic is incorporated into random initialization of PSO to generate an initial population with certain quality and diversity. Third, a new local search named NEH_1 insertion is proposed and probabilistically applied to some good particles selected by using roulette wheel mechanism, with a specified probability to balance the exploration and exploitation abilities. Fourth, SA-based local search with multiple different neighborhoods is also designed and incorporated to enrich the searching behaviors and to avoid premature convergence, and an effective adaptive meta-Lamarckian learning strategy is employed to decide which neighborhood to be used. In addition, a pairwise-based local search is applied after the SA-based local search to further enhance the exploitation ability. Simulation results and comparisons demonstrate the effectiveness of the proposed PSOMA for PFSSP.

The remaining contents are organized as follows. In Sections II and III, the PFSSP and PSO are introduced, respectively. In Section IV, the PSOMA is proposed after presenting

solution representation, population initialization, NEH-based local search, SA-based local search combining adaptive meta-Lamarckian learning strategy, and pairwise-based local search. In Section V, experimental results on benchmarks together with comparisons to other previous metaheuristics are presented and analyzed. In Section VI, the effects of some parameters on optimization performances are discussed. Finally, in Section VII, we end this paper with some conclusions and suggestions of possible future work.

II. Permutation Flow Shop Scheduling Problem

The PFSSP can be described as follows: Each of the n jobs is to be sequentially processed on machine $1, \ldots, m$. The processing time $p_{i,j}$ of job i on machine j is given. 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. The objective is to find a sequence for the processing of the jobs in the machines so that a given criterion is optimized. In the literature, the criterion widely used is the minimization of the maximum completion time, i.e., makespan ($C_{\max}$) [9]–[14].

Let $\pi = \{j_1, j_2, \ldots, j_n\}$ denotes a permutation of all jobs, and $C(j_i, k)$ denotes the completion time of job j_i on the machine; then, the completion time $C(j_i, k)$ can be calculated as follows:

$$C(j_1, 1) = p_{j_1,1} \tag{1}$$

$$C(j_i, 1) = C(j_{j-1}, 1) + p_{j_i,1}, \qquad i = 2, \ldots, n \tag{2}$$

$$C(j_1, k) = C(j_1, k-1) + p_{j_1,k}, \qquad k = 2, \ldots, m \tag{3}$$

$$C(j_i, k) = \max\{C(j_{i-1}, k), C(j_i, k-1)\} + p_{j_i,k}, \quad i = 2, \ldots, n, \quad k = 2, \ldots, m \tag{4}$$

$$C_{\max}(\pi) = C(j_n, m). \tag{5}$$

The PFSSP is then to find a permutation π^* in the set of all permutation Π, such that

$$\pi^* = \arg\{C_{\max}(\pi) = C(j_n, m)\} \rightarrow \min \forall \pi \in \Pi. \tag{6}$$

III. Particle Swarm Optimization

In a PSO system, it starts with the random initialization of a population (swarm) of individuals (particles) in the search space and works on the social behavior in the swarm. The position and the velocity of the ith particle in the d-dimensional search space can be represented as $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]$ and $V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,d}]$, respectively. Each particle has its own best position (pbest) $P_i = [p_{i,1}, p_{i,2}, \ldots, p_{i,d}]$ corresponding to the personal best objective value obtained so far at time t. The global best particle (gbest) is denoted by P_g, which represents the best particle found so far at time t in the entire swarm. The new velocity of each particle is calculated as follows:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_1 (p_{i,j} - x_{i,j}(t)) + c_2 r_2 (p_{g,j} - x_{i,j}(t)), \qquad j = 1, 2, \ldots, d \tag{7}$$

where c_1 and c_2 are acceleration coefficients, w is inertia factor, r_1 and r_2 are two independent random numbers uniformly distributed in the range of [0, 1].

Thus, the position of each particle is updated in each generation according to the following equation:

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1), \qquad j = 1, 2, \ldots, d. \tag{8}$$

Generally, the value of each component in V_i can be clamped to the range $[-v_{\max}, v_{\max}]$ to control excessive roaming of particles outside the search space. Then, the particle flies toward a new position according to (8). This process is repeated until a user-defined stopping criterion is reached.

The procedure of standard PSO is summarized as follows.

Step 1) Initialize a population of particles with random positions and velocities, where each particle contains d variables (i.e., $d = n$).

Step 2) Evaluate the objective values of all particles; let pbest of each particle and its objective value equal to its current position and objective value; and let gbest and its objective value equal to the position and objective value of the best initial particle.

Step 3) Update the velocity and position of each particle according to (7) and (8).

Step 4) Evaluate the objective values of all particles.

Step 5) For each particle, compare its current objective value with the objective value of its pbest. If current value is better, then update pbest and its objective value with the current position and objective value.

Step 6) Determine the best particle of the current swarm with the best objective value. If the objective value is better than the objective value of gbest, then update gbest and its objective value with the position and objective value of the current best particle.

Step 7) If a stopping criterion is met, then output gbest and its objective value; otherwise go back to Step 3).

IV. PSOMA for PFSSP

In this section, we will explain the implementation of PSOMA for PFSSP in detail.

A. Solution Representation

Usually, a job-permutation-based encoding scheme [2] has been used in many papers for PFSSP. However, because of the continuous characters of the position of particles in PSO, the standard encoding scheme of PSO cannot be directly adopted for PFSSP. So, the most important issue in applying PSO to PFSSP is to find a suitable mapping between job sequence and positions of particles.

In this paper, a ROV rule based on random key representation [37] is presented to convert the continuous position $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ of particles in PSO to permutation of jobs $\pi = \{j_1, j_2, \ldots, j_n\}$, thus the performance of the particle can be evaluated. In particular, the position information $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ itself does not represent a sequence, whereas the rank of each position value of a particle represents

TABLE I
REPRESENTATION OF POSITION INFORMATION AND THE CORRESPONDING ROV (JOB PERMUTATION)

Dimension *j*	**1**	**2**	**3**	**4**	**5**	**6**
Position value	0.06	2.99	1.86	3.73	2.13	0.67
Ranked-order-value	1	5	3	6	4	2

TABLE II
SWAP-BASED LOCAL SEARCH FOR JOB PERMUTATION AND THE CORRESPONDING ADJUSTMENT FOR POSITION INFORMATION

Dimension *j*	**1**	**2**	**3**	**4**	**5**	**6**
Position value	0.06	***2.99***	1.86	***3.73***	2.13	0.67
Job permutation	1	**5**	3	**6**	4	2
Dimension *j*	**1**	**2**	**3**	**4**	**5**	**6**
Position value	0.06	***3.73***	1.86	***2.99***	2.13	0.67
Job permutation	1	**6**	3	**5**	4	2

a job index so as to construct a permutation of jobs. In our ROV rule, the smallest position value of a particle is first picked and assigned a smallest rank value of one. Then, the second smallest position value is picked and assigned a rank value of two. With the same way, all the position values will be handled to convert the position information of a particle to a job permutation. We provide a simple example to illustrate the ROV rule in Table I. In the instance $(n = 6)$, position is $X_i = [0.06, 2.99, 1.86, 3, 73, 2.13, 0.67]$. Because $x_{i,1} = 0.06$ is the smallest position value, $x_{i,1}$ is picked first and assigned a rank value of one; then, $x_{i,6} = 0.67$ is picked and assigned a rank value of two. Similarly, the ROV rule assigns a rank value of three–six to $x_{i,3}$, $x_{i,5}$, $x_{i,2}$, and $x_{i,4}$, respectively. Thus, based on the ROV rule, the job permutation is obtained, i.e., $\pi = [1, 5, 3, 6, 4, 2]$.

In our PSOMA, several local searches are not directly applied to position information but to job permutation. So, when a local search procedure is completed, the particle's position information should be repaired to guarantee that the permutation that will result from the ROV rule for the new position information is the same as the permutation that will result from the local search. That is to say, when applying these local searches to job permutation, position information should be adjusted correspondingly. Fortunately, the adjustment is very simple due to the mechanism of ROV rule. The process based on some local searches to position information is the same as the process to permutation. For example, in Table II, if a SWAP operator [16] is used as a local search for job permutation, obviously swapping of job 5 and job 6 is corresponding to swapping of position values 2.99 and 3.73. As for other local search operators such as INVERSE and INSERT [16], the adjustment is similar.

B. Population Initialization

In the standard PSO, initial swarm is often generated randomly. To guarantee an initial population with certain quality and diversity, the NEH heuristic [9] is applied to generate a solution (i.e., permutation of jobs), whereas the rest of the particles are initialized with random position values and velocities in a certain interval. Since the result produced by the NEH heuristic is a job permutation, it should be converted to position values of a certain initial particle to perform the PSO-based searches. The conversion is performed using the following equation:

$$x_{\mathrm{NEH},j} = x_{\min,j} + (x_{\max,j} - x_{\min,j}) \cdot (s_{\mathrm{NEH},j} - 1 + \mathrm{rand})/n, \qquad j = 1, 2, \ldots, n. \quad (9)$$

where $x_{\mathrm{NEH},j}$ is the position value in the jth dimension of the particle, $s_{\mathrm{NEH},j}$ is the job index in the jth dimension of the permutation by the NEH heuristic, $x_{\max,j}$ and $x_{\min,j}$ are the upper and lower bounds of the position value, respectively, rand denotes a random number uniformly distributed in the interval [0, 1], and n represents the number of dimensions of a position, which is equal to the number of jobs. Table III provides an example of the above conversion from job permutation to position information. Obviously, such a conversion obeys ROV rule. That is, the permutation can be obtained from position information using ROV rule.

TABLE III
CONVERSION OF NEH SOLUTION (JOB PERMUTATION) TO PARTICLE POSITION INFORMATION

Dimension *j*	**1**	**2**	**3**	**4**	**5**	**6**
Permutation by NEH	2	4	3	6	5	1
Position value	0.97	2.01	1.88	3.63	3.08	0.53

C. PSO-Based Search

In this paper, the PSO-based searches, i.e., (7) and (8), are applied for exploration. That is, the position information of the particles in the current swarm is evolved by PSO-based searching operators. Note that, the PSO-based evolution is performed on a continuous space. So, when evaluating the performance of a particle, the position information should be converted to job permutation using the above ROV rule. On the other hand, PSO provides a parallel evolutionary framework for the optimization of hard problems, and it is also easy to incorporate local searches in PSO to develop hybrid algorithms. In the following contents, we will present some local searches that are incorporated in PSO to propose a PSO-based MA.

D. NEH-Based Local Search

In MAs, local searches are very important for exploitation. For PFSSP, Aldowaisan and Allahverdi [38] designed a local search named NEH_2 insertion to improve the quality of a job permutation, where two consecutive jobs were considered as a block and inserted in the sequence in a similar manner as in NEH. The NEH_2 insertion is described as follows.

Step 1) Given a sequence of jobs π.
Step 2) $k = 1$. Pick the first two jobs from π and schedule them in order to minimize the partial makespan. Select the better sequence as a current sequence.

Step 3) Set $k = k + 1$. Generate candidate sequences by selecting the next two jobs from π, inserting this kth block of two jobs into each slot of the current sequence, and interchanging the order of the two jobs within the block. Among these candidates, select the best one with the least partial makespan. Set the best one as a current sequence.

Step 4) Repeat step 3) until all jobs in π are assigned. If the last assignment is a single job, treat that job as a block.

Inspired by the NEH_2 insertion, we propose a local search named NEH_1 insertion for permutation-based solutions. We will demonstrate the superiority of NEH_1 over NEH_2 in the later simulation. The NEH_1 is described as follows.

Step 1) Given a sequence of jobs π.

Step 2) The first two jobs from π are taken, and the two partial possible schedules are evaluated. Select the better sequence as a current sequence.

Step 3) Take job k, $k = 3, \ldots, n$, and find the best schedule by placing it in all the possible k positions in the sequence of jobs that are already scheduled. The best partial sequence is selected for the next iteration.

In addition, in PSOMA, based on the performances of pbest solutions of all particles in current swarm, pbest solution of each particle is assigned a probability to be selected by the rank-based fitness assignment technique [39]. Then, the roulette wheel mechanism [39] is used to decide which pbest solutions will be selected. Subsequently, the selected pbest solutions will perform the above NEH_1 or NEH_2 insertion with a predefined probability p_{ls}. Due to the mechanism of roulette wheel rule, a good solution will gain more chance for exploitation. Besides, it is easy to control such an exploitation process by adjusting the value of p_{ls}. For example, if $p_{ls} = 1$, the selected pbest should perform NEH_1 or NEH_2; whereas if $p_{ls} < 1$, the selected pbest will perform NEH_1 or NEH_2 with the certain probability of p_{ls}.

E. SA-Based Local Search Combining Adaptive Meta-Lamarckian Learning Strategy

In SA, starting from an initial state, the algorithm randomly generates a new state in the neighborhood of the original one, which causes a change of ΔE in the objective function value. For minimization problems, the new state is accepted with probability $\min\{1, \exp(-\Delta E/T)\}$, where T is a control parameter. SA provides a mechanism to probabilistically escape from local optima, and the search process can be controlled by the cooling schedule [40].

In this paper, we design a SA-based local search with multiple different neighborhoods to enrich the local searching behaviors and to avoid premature convergence. Moreover, the adaptive meta-Lamarckian learning strategy in [28] is employed to decide which neighborhood to be used.

1) Neighborhoods in SA-Based Local Search: In order to maintain diversity of a population and enrich the local searching behaviors, three different kinds of neighborhoods, i.e., SWAP, INSERT, and INVERSE are utilized.

SWAP: Select two distinct elements from an n-job permutation randomly and swap them.

INSERT: Choose two distinct elements from an n-job permutation randomly and insert the back one before the front.

INVERSE: Invert the subsequence between two different random positions of an n-job permutation.

2) Adaptive Meta-Lamarckian Learning Strategy: Inspired by the idea of Ong and Keane's work [28], we not only apply multiple neighborhoods but also adopt the effective adaptive meta-Lamarckian learning strategy to decide which neighborhood to be chosen for local search to reward the utilization times of those neighborhoods, resulting in solution improvement.

The adaptive meta-Lamarckian learning strategy in our PSOMA is illustrated as follows. We divide SA-based search into training phase and nontraining phase. During the meta-Lamarckian training phase, each SA-based local search with different neighborhood is applied for the same times, i.e., $n(n-1)$ steps (a generation of Metropolis sampling). Then, the reward η of each neighborhood is determined using the following equation:

$$\eta = |\mathrm{pf} - \mathrm{cf}| / \left(n(n-1)\right) \tag{10}$$

where n denotes the number of jobs, pf is the objective value of the old permutation, and cf is the objective value of the best permutation found by the local search based on different neighborhood during consecutive $n(n-1)$ steps.

After the reward of each neighborhood is determined, the utilizing probability p_{ut} of each neighborhood is adjusted using the following equation:

$$p_{\mathrm{ut},i} = \eta_i / \sum_{j=1}^{K} \eta_j \tag{11}$$

where η_i is the reward value of the ith neighborhood and K is the total number of neighborhoods.

At nontraining phase, according to the utilizing probability of each neighborhood, a roulette wheel rule [39] is used to decide which neighborhood to be used for SA-based local search. If the ith neighborhood is used, its reward will be updated by $\eta_i = \eta_i + \Delta\eta_i$, where $\Delta\eta_i$ is the reward value of the ith neighborhood calculated during a nontraining phase (i.e., SA-based searching based on the ith neighborhood with consecutive $n(n-1)$ steps). Of course, the utilizing probability p_{ut} of each neighborhood should be adjusted again for the next generation of PSOMA.

3) Other Notes About SA-Based Local Search: A proper initial temperature should be high enough so that all states of the system have an equal probability of being visited, and at the same time, it should not be rather high so that a lot of unnecessary searches in high temperature will be avoided. In this paper, we set an initial temperature by trial and error.

Moreover, exponential cooling schedule $t_k = \lambda t_{k-1}$, $0 < \lambda < 1$ is applied, which is often believed to be an excellent cooling recipe, since it provides a rather good compromise between a computationally fast schedule and the ability to reach a low-energy state [41]. To provide a good compromise

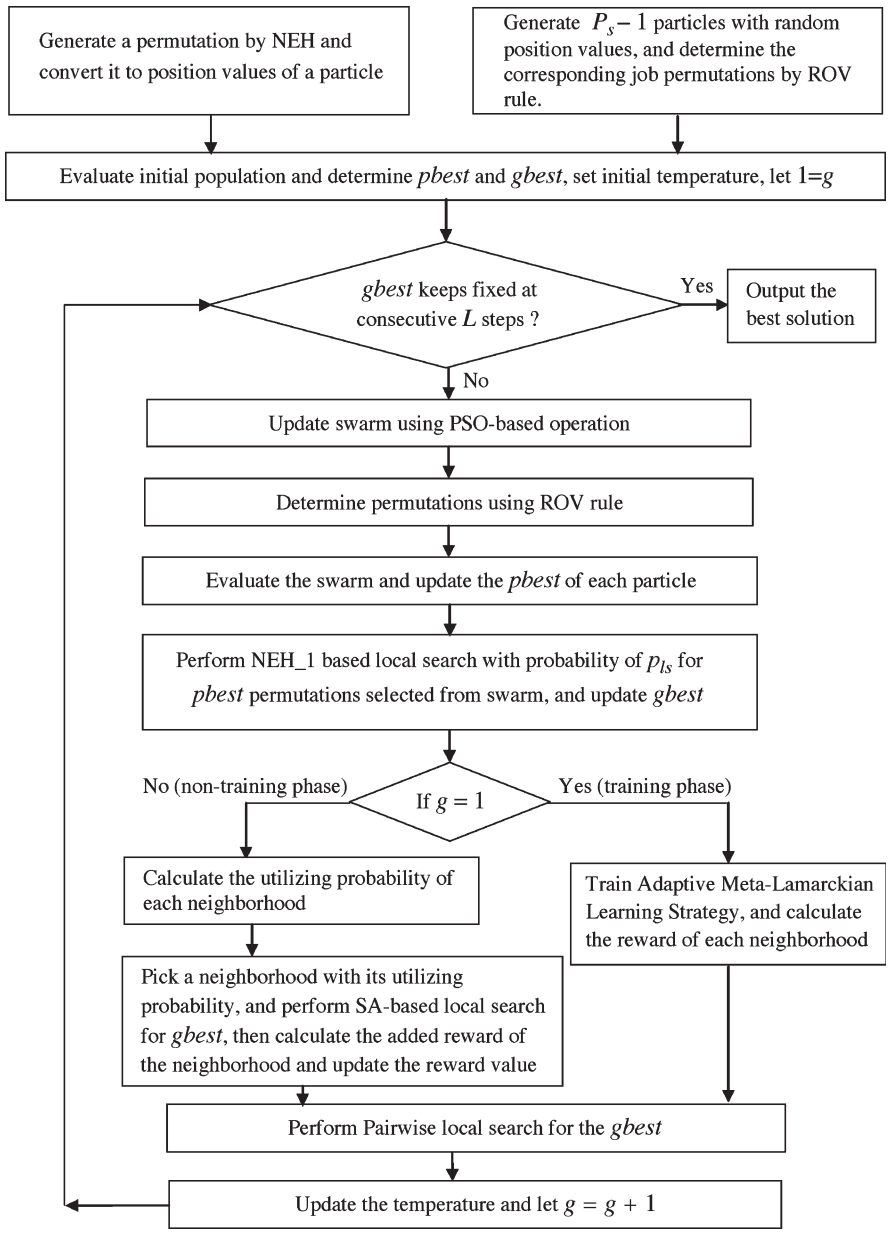


Fig. 1. Outline of the PSOMA.

between solution quality and search efficiency, the step of Metropolis sampling process is set to $n \cdot (n-1)$.

In addition, the above SA-based local search is only applied to the best solution found so far, i.e., gbest. Thus, at a high temperature, the algorithm performs exploration with certain jumping probability as well as local improvement from gbest, whereas at a low temperature, the algorithm stresses the exploitation from gbest. On the other hand, not too much computational effort will be paid in such local search since it is only applied to gbest. After the SA-based local search is completed, gbest of the whole swarm should be updated if a solution with better quality is found during the local search.

F. Pairwise-Based Local Search

As a complementary searching process, a pairwise-based local search [38] is employed for the gbest, whose procedure is described as follows: It examines each possible pairwise interchange of the job in first position, with the jobs in all other positions. Whenever there is an improvement in the objective function, the jobs are interchanged. Similarly, the jobs in the second and other subsequent positions are examined. Pairwise-based search can be viewed as a detailed neighborhood searching process, so it is applied after SA-based local search to further enhance the exploitation ability.

G. PSO-Based MA

Based on the proposed ROV rule, population initialization, NEH-based local search, SA-based local search combining adaptive meta-Lamarckian learning strategy, and pairwise-based local search, a PSOMA outline is proposed, which is illustrated in Fig. 1.

It can be seen that PSOMA not only applies the PSO-based evolutionary searching mechanism to effectively perform exploration for promising solutions within the entire region but also applies problem-dependent local searches to perform exploitation for solution improvement in subregions. Due to the hardness of PFSSP, PSOMA applies several local searches simultaneously, including NEH-based initialization, NEH_1 or NEH_2 insertion, SA-based local search combining adaptive meta-Lamarckian learning strategy, and pairwise-based local search. Since both exploration and exploitation are stressed and balanced, it is expected to achieve good results for PFSSP. In the next section, we will investigate the performances of PSOMA.

V. Numerical Test and Comparisons

A. Experimental Setup

To test the performance of the proposed PSOMA, computational simulation is carried out with some well-studied benchmarks. In this paper, 29 problems that were contributed to the OR-Library are selected. The first eight problems are called car1, car2 through car8 by Carlier [42]. The other 21 problems are called rec01, rec03 through rec41 by Reeves [11], who used them to compare the performances of some metaheuristics and found these problems to be particularly difficult. So far, these problems have been used as benchmarks for study with different methods by many researchers.

In our simulation, PSOMA is coded in MATLAB 7.0 (The Math Works, Inc., 2004), and experiments are executed on Mobile Pentium IV 2.2-GHz processor with 512 MB RAM. The following parameters are used in PSOMA: swarm size $p_s = 20$, $w = 1.0$, $c_1 = c_2 = 2.0$, $x_{\min} = 0$, $x_{\max} = 4.0$, $v_{\min} = -4.0$, $v_{\max} = 4.0$, $p_{ls} = 0.1$, initial temperature $T_0 = 3.0$, annealing rate $\lambda = 0.9$, and stopping parameter $L = 30$. Each instance is independently run 20 times for every algorithm for comparison.

To validate the effectiveness of each part in PSOMA, variants of PSOMA are compared. The following abbreviations represent the variants considered:

1) PM_NEH1: PSOMA with NEH_1;
2) PM_NEH2: PSOMA with NEH_2 replacing NEH_1;
3) PM_NONEH: PSOMA without NEH-based local search (either NEH_1 or NEH_2);
4) PM_NOSA: PSOMA without SA-based local search;
5) PM_NOPAIR: PSOMA without pairwise-based search.

B. Simulation Results and Comparisons

1) Comparison of PM_NEH1, PM_NEH2, and NEH Heuristic: The statistical performances of 20 independent runs are listed in Table I, including the best relative error to C^* (BRE), average relative error to C^* (ARE), the worst relative error to C^* (WRE), and average CPU executing time (Tavg). Since the NEH heuristic is a deterministic heuristic, the relative error (RE) values to C^* are given. In Table IV, C^* is the optimal makespan or lower bound value known so far.

From Table IV, it can be seen that the PM_NEH1 (i.e., our proposed PSOMA) provides the best optimization performance among the methods studied almost for all benchmarks. On the one hand, the optimal results obtained by the PM_NEH1 are close to C^* (that is to say, the BRE values are very small), which demonstrates the effective searching quality of PSOMA. On the other hand, the ARE and WRE values that result from PM_NEH1 are also very small, which demonstrates the robustness of PSOMA on initial conditions.

TABLE IV
Comparisons of PM_NEH1, PM_NEH2, and NEH

Problem	*n, m*	*C**	PM_NEH1				PM_NEH2				NEH
			BRE	ARE	WRE	Tavg	BRE	ARE	WRE	Tavg	RE
Car1	11,5	7038	**0**	**0**	**0**	0.68	**0**	0	**0**	**0.62**	0
Car2	13,4	7166	**0**	**0**	**0**	0.95	**0**	0	**0**	**0.88**	2.931
Car3	12,5	7312	**0**	**0**	**0**	1.06	**0**	0	**0**	**0.94**	1.19
Car4	14,4	8003	**0**	**0**	**0**	1.22	**0**	0	**0**	**0.95**	0
Car5	10,6	7720	**0**	**0.018**	**0.375**	**0.70**	**0**	0.024	0.389	**0.70**	1.49
Car6	8,9	8505	**0**	0.114	**0.764**	0.49	**0**	**0.038**	**0.764**	**0.44**	3.151
Car7	7,7	6590	**0**	**0**	**0**	**0.30**	**0**	**0**	**0**	0.31	0
Car8	8,8	8366	**0**	**0**	**0**	0.42	**0**	**0**	**0**	**0.40**	2.367
Rec01	20,5	1247	**0**	**0.144**	**0.160**	2.60	0.160	0.160	**0.160**	**2.48**	4.491
Rec03	20,5	1109	**0**	0.189	0.721	**2.50**	**0**	**0.176**	**0.271**	2.77	2.074
Rec05	20,5	1242	**0.242**	**0.249**	**0.402**	**2.39**	**0.242**	0.250	0.403	2.53	3.14
Rec07	20,10	1566	**0**	**0.986**	**1.149**	**2.81**	**0**	1.041	1.277	3.03	3.831
Rec09	20,10	1537	**0**	**0.621**	**1.691**	4.23	**0**	1.002	1.757	**3.70**	2.993
Rec11	20,10	1431	**0**	**0.129**	0.978	**3.79**	**0**	0.315	**0.909**	3.92	8.316
Rec13	20,15	1930	**0.259**	**0.893**	**1.502**	**4.64**	0.311	1.217	2.021	5.04	3.731
Rec15	20,15	1950	0.051	0.628	**1.076**	**5.23**	**0**	**0.564**	1.641	5.63	3.231
Rec17	20,15	1902	**0**	**1.330**	**2.155**	**4.67**	0.631	1.498	2.997	5.28	6.151
Rec19	30,10	2093	**0.43**	**1.313**	**2.102**	**10.49**	1.051	1.455	2.389	10.82	4.396
Rec21	30,10	2017	**1.437**	**1.596**	**1.636**	**8.41**	1.438	1.621	1.884	8.81	5.652
Rec23	30,10	2011	**0.596**	**1.310**	**2.038**	**9.36**	0.895	1.713	2.435	12.34	7.161
Rec25	30,15	2513	**0.835**	**2.085**	**3.223**	12.64	1.990	2.501	3.263	**11.89**	5.213
Rec27	30,15	2373	1.348	**1.605**	**2.402**	12.15	**1.264**	2.088	3.287	**11.04**	5.268
Rec29	30,15	2287	1.442	**1.888**	**2.492**	**11.31**	**0.962**	2.396	3.498	13.42	4.547
Rec31	50,10	3045	**1.510**	**2.254**	**2.692**	**37.15**	1.872	2.652	3.153	**37.15**	4.204
Rec33	50,10	3114	**0**	**0.645**	**0.834**	**36.07**	0.514	0.734	0.963	43.78	4.078
Rec35	50,10	3277	**0**	**0**	**0**	**29.92**	**0**	0.005	0.092	30.44	1.099
Rec37	75,20	4951	**2.101**	**3.537**	**4.039**	170.2	2.828	3.821	4.787	**158.4**	5.575
Rec39	75,20	5087	**1.553**	**2.426**	**2.830**	155.7	2.064	2.797	3.440	**154.2**	4.344
Rec41	75,20	4960	**2.641**	**3.684**	**4.052**	164.3	3.448	4.237	5.101	**147.1**	6.694

Compared with NEH heuristic, PM_NEH1 can obtain much better results, and the WRE values that result from PM_NEH1 and PM_NEH2 (two variants of PSOMA) are better than RE values that result from NEH heuristic, which demonstrates the significant improvement of PSOMA over the famous NEH heuristic.

Compared with PM_NEH2, the BRE, ARE, and WRE values that result from PM_NEH1 are better than those that result from PM_NEH2 for most instances, which demonstrates the effectiveness in terms of searching quality and robustness of NEH_1 insertion with respect to NEH_2 insertion. As for the computational time performance, NEH_1-based PSOMA is competitive to NEH_2-based PSOMA. Thus, it is concluded that NEH_1 is superior to NEH_2 in our PSOMA.

2) Comparison of PM_NEH1, PM_NONEH, PM_NOSA, and PM_NOPAIR: We demonstrate the effectiveness of local search by comparing PM_NEH1 (PSOMA) with PM_NONEH, PM_NOSA, and PM_NOPAIR. The statistical performances of each algorithm with 20 independent runs are listed in Table V.

From Table V, it is shown that the BRE and ARE values obtained by PM_NEH1 are much better than those obtained by PM_NONEH, PM_NOSA, and PM_NOPAIR, which demonstrates the effectiveness by incorporating local searches into PSO. That is to say, the superiority in terms of searching quality and robustness of PSO owns to the combination of global search and local searches, i.e., the balance of exploration and exploitation. Since local search elements are employed, the computational time of PSOMA is larger than that of algorithms without local search, while it can be regarded as the cost to

TABLE V
COMPARISONS OF PM_NEH1, PM_NONEH, PM_NOSA, AND PM_NOPAIR

Problem	PM_NEH1			PM_NONEH			PM_NOSA			PM_NOPAIR		
	BRE	ARE	Tavg	BRE	ARE	Tavg	BRE	ARE	Tavg	BRE	ARE	Tavg
Car1	**0**	**0**	0.68	**0**	**0**	0.445	**0**	**0**	**0.088**	**0**	**0**	0.400
Car2	**0**	**0**	0.95	**0**	**0**	0.703	**0**	**0**	**0.127**	**0**	**0**	0.555
Car3	**0**	**0**	1.06	**0**	0.170	0.602	**0**	0.624	**0.141**	**0**	0.037	0.570
Car4	**0**	**0**	1.22	**0**	**0**	0.728	**0**	**0**	**0.136**	**0**	**0**	0.620
Car5	**0**	0.018	0.70	**0**	0.045	0.422	**0**	**0**	**0.084**	**0**	**0**	0.285
Car6	**0**	0.114	0.49	**0**	0.038	0.229	**0**	0.038	**0.066**	**0**	**0**	0.188
Car7	**0**	**0**	0.30	**0**	**0**	0.136	**0**	**0**	**0.047**	**0**	**0**	0.124
Car8	**0**	**0**	0.42	**0**	0.032	0.234	**0**	**0**	**0.061**	**0**	**0**	0.174
Rec01	**0**	**0.144**	2.60	**0**	0.176	1.605	**0**	0.241	**0.415**	0.160	0.168	1.244
Rec03	**0**	0.189	2.50	**0**	0.131	1.690	**0**	0.135	**0.402**	**0**	**0.086**	1.438
Rec05	**0.242**	0.249	2.39	**0.242**	**0.242**	1.702	**0.242**	0.322	**0.442**	0.242	**0.242**	1.454
Rec07	**0**	0.986	2.81	**0**	**0.884**	2.155	0.511	1.118	**0.382**	0.128	1.012	1.572
Rec09	**0**	0.621	4.23	**0**	0.765	2.377	0.390	1.340	**0.471**	**0**	**0.228**	2.171
Rec11	**0**	**0.129**	3.79	**0**	0.720	2.260	**0**	0.391	**0.536**	**0**	0.228	2.167
Rec13	0.259	**0.893**	4.64	0.415	1.521	2.491	0.622	1.264	**0.581**	**0.104**	0.899	2.080
Rec15	**0.051**	**0.628**	5.23	1.077	1.318	2.630	0.564	1.174	**0.600**	0.462	1.028	2.208
Rec17	**0**	1.330	4.67	0.894	2.308	2.155	0.946	1.777	**0.642**	**0**	**1.173**	2.771
Rec19	**0.43**	**1.313**	10.49	1.147	1.634	6.176	1.386	2.057	**1.206**	0.669	1.398	5.990
Rec21	**1.437**	1.596	8.41	1.438	**1.584**	5.365	1.487	1.787	**1.197**	1.438	1.589	4.328
Rec23	**0.596**	**1.310**	9.36	0.945	1.584	6.352	1.542	2.708	**1.300**	0.597	**1.504**	4.851
Rec25	**0.835**	**2.085**	12.64	1.671	2.712	7.451	1.711	2.915	**1.820**	1.711	2.161	6.589
Rec27	1.348	**1.605**	12.15	1.306	2.103	7.928	1.517	2.199	**1.475**	**1.054**	1.669	6.773
Rec29	**1.442**	**1.888**	11.31	1.662	2.825	6.666	1.618	2.383	**1.820**	1.443	2.086	6.353
Rec31	**1.510**	**2.254**	37.15	2.168	2.555	24.17	2.430	3.069	**5.846**	1.806	2.381	23.98
Rec33	**0**	0.645	36.07	0.129	0.724	26.33	0.353	1.008	**5.467**	0.129	**0.596**	20.68
Rec35	**0**	**0**	29.92	**0**	0.005	25.05	**0**	0.018	**5.177**	**0**	**0**	18.63
Rec37	**2.101**	**3.537**	170.2	3.696	4.463	83.80	3.959	4.553	**21.47**	3.232	3.689	100.9
Rec39	**1.553**	**2.426**	155.7	2.477	3.090	98.98	2.929	3.468	**22.17**	1.809	2.473	98.36
Rec41	**2.641**	**3.684**	164.3	3.629	4.615	93.55	4.355	4.888	**22.42**	3.488	3.857	96.23

TABLE VI
COMPARISONS OF PSOMA AND PSOVNS

Problem	PSOMA			PSOVNS		
	BRE	ARE	WRE	BRE	ARE	WRE
Car1	**0**	**0**	**0**	**0**	**0**	**0**
Car2	**0**	**0**	**0**	**0**	**0**	**0**
Car3	**0**	**0**	**0**	**0**	0.420	1.189
Car4	**0**	**0**	**0**	**0**	**0**	**0**
Car5	**0**	**0.018**	**0.375**	**0**	0.039	0.389
Car6	**0**	0.114	**0.764**	**0**	**0.076**	**0.764**
Car7	**0**	**0**	**0**	**0**	**0**	**0**
Car8	**0**	**0**	**0**	**0**	**0**	**0**
Rec01	**0**	**0.144**	**0.160**	0.160	0.168	0.321
Rec03	**0**	0.189	0.721	**0**	**0.158**	**0.180**
Rec05	**0.242**	**0.249**	**0.402**	**0.242**	**0.249**	**0.402**
Rec07	**0**	**0.986**	**1.149**	0.702	1.095	1.405
Rec09	**0**	**0.621**	1.691	**0**	0.651	**1.366**
Rec11	**0**	**0.129**	**0.978**	0.071	1.153	2.656
Rec13	**0.259**	**0.893**	**1.502**	1.036	1.790	2.643
Rec15	**0.051**	**0.628**	**1.076**	0.769	1.487	2.256
Rec17	**0**	**1.330**	**2.155**	0.999	2.453	3.365
Rec19	**0.43**	**1.313**	**2.102**	1.529	2.099	2.532
Rec21	**1.437**	**1.596**	**1.636**	1.487	1.671	2.033
Rec23	**0.596**	**1.310**	**2.038**	1.343	2.106	2.884
Rec25	**0.835**	**2.085**	**3.223**	2.388	3.166	3.780
Rec27	**1.348**	**1.605**	**2.402**	1.728	2.463	3.203
Rec29	**1.442**	**1.888**	**2.492**	1.968	3.109	4.067
Rec31	**1.510**	**2.254**	**2.692**	2.594	3.232	4.237
Rec33	**0**	**0.645**	**0.834**	0.835	1.007	1.477
Rec35	**0**	**0**	**0**	**0**	0.038	0.092
Rec37	**2.101**	**3.537**	**4.039**	4.383	4.949	5.736
Rec39	**1.553**	**2.426**	**2.830**	2.850	3.371	3.951
Rec41	**2.641**	**3.684**	**4.052**	4.173	4.867	5.585

avoid premature convergence so as to achieve better optimization performances.

3) Comparison of PSOMA and PSOVNS: To show the effectiveness of PSOMA, we carry out a simulation to compare our PSOMA with the hybrid PSO based on VNS (PSOVNS) [36]. In our simulation, the parameters of PSOVNS are the same as those used in [36], while the PSOVNS uses the same number of evaluation with PSOMA for each instance. The statistical results of the two algorithms are listed in Table VI.

From Table VI, it is shown that the BRE values obtained by PSOMA are much better than those obtained by PSOVNS for all the instances, and the ARE and WRE values obtained by PSOMA are much better than those obtained by PSOVNS almost for all the instances. So, it is concluded that our proposed local search methods, especially their utilization in hybrid senses, are more effective than the pure VNS-based local search in [36]. In brief, our PSOMA is more effective than PSOVNS.

4) Comparison of PSOMA and Other Popular Methods: To further show the effectiveness of PSOMA, we carry out some comparisons with several other popular and effective metaheuristics such as simple GA (SGA), SGA+NEH, and hybrid GA (HGA) [16], whose performance are comparable or even superior to those of many other metaheuristics in the previous papers. Table VII lists the statistical results of each algorithm for solving the above 29 problems. From the results, it can be concluded that our PSOMA is more effective than the HGA [16].

TABLE VII
COMPARISONS OF PSOMA WITH SGA, SGA+NEH, AND HGA

Problem	PSOMA		SGA		SGA+NEH		HGA	
	BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE
Car1	**0**	**0**	**0**	0.27	**0**	**0**	**0**	**0**
Car2	**0**	**0**	**0**	4.07	2.93	2.93	**0**	**0**
Car3	**0**	**0**	1.19	2.95	0.82	1.21	**0**	**0**
Car4	**0**	**0**	**0**	2.36	**0**	0.07	**0**	**0**
Car5	**0**	0.018	**0**	1.46	**0**	1.14	**0**	**0**
Car6	**0**	0.114	**0**	1.86	**0**	2.82	**0**	**0.04**
Car7	**0**	**0**	**0**	1.57	**0**	1.36	**0**	**0**
Car8	**0**	**0**	**0**	2.59	**0**	0.03	**0**	**0**
Rec01	**0**	0.144	2.81	6.96	2.25	6.13	**0**	**0.14**
Rec03	**0**	0.189	1.89	4.45	1.26	4.27	**0**	**0.09**
Rec05	0.242	**0.249**	1.93	3.82	2.33	2.90	**0**	0.29
Rec07	**0**	0.986	1.15	5.31	3.38	5.27	**0**	**0.69**
Rec09	**0**	**0.621**	3.12	4.73	0.39	2.13	**0**	0.64
Rec11	**0**	**0.129**	3.91	7.39	1.19	3.66	**0**	1.10
Rec13	**0.259**	**0.893**	3.68	5.97	1.92	4.41	0.36	1.68
Rec15	**0.051**	**0.628**	2.21	4.29	2.87	4.02	0.56	1.12
Rec17	**0**	**1.330**	3.15	6.08	2.16	4.02	0.95	2.32
Rec19	**0.43**	**1.313**	4.01	6.07	2.05	4.35	0.62	1.32
Rec21	**1.437**	1.596	3.42	6.07	3.52	3.58	1.44	**1.57**
Rec23	0.596	1.310	3.83	7.46	3.63	5.12	**0.40**	**0.87**
Rec25	**0.835**	**2.085**	4.42	7.20	3.14	4.89	1.27	2.54
Rec27	1.348	**1.605**	4.93	6.85	3.16	5.21	**1.10**	1.83
Rec29	1.442	**1.888**	6.21	8.48	3.32	4.93	**1.40**	2.70
Rec31	1.510	2.254	6.17	8.02	5.94	6.66	**0.43**	**1.34**
Rec33	**0**	**0.645**	3.08	5.12	2.70	3.38	**0**	0.78
Rec35	**0**	**0**	1.46	3.30	1.89	2.58	**0**	**0**
Rec37	**2.101**	**3.537**	7.89	10.07	7.14	7.94	3.75	4.90
Rec39	**1.553**	**2.426**	7.32	8.51	6.25	7.09	2.20	2.79
Rec41	**2.641**	**3.684**	8.51	10.03	7.49	8.47	3.64	4.92

VI. EFFECTS OF PARAMETERS

In this section, we will investigate the effects of swarm size and the utilizing probability of NEH_1 insertion on the performances of PSOMA.

A. Effect of Swarm Size p_s

As we have known, the performance of PSO greatly depends on swarm size. So, we investigate the effect of swarm size on

TABLE VIII
EFFECTS OF SWARM SIZE

Problem/*n, m/C**	PSOMA				
	Swarm Size	**BRE**	**ARE**	**WRE**	**Tavg**
Rec25/30,15/2513	10	1.036	2.158	2.745	7.695
	20	0.835	2.085	3.223	12.64
	30	0.950	2.103	2.904	16.13
	40	1.036	2.156	2.865	18.13
	50	0.863	2.113	2.785	24.00
	60	0.719	2.005	2.507	26.33
	70	0.691	2.021	2.745	30.77
	80	0.777	2.071	2.626	34.83
	90	0.662	1.949	2.785	38.19
	100	1.007	1.902	2.546	47.40

TABLE IX
EFFECTS OF P_{ls}

Problem/*n, m/C**	PSOMA				
	P_{ls}	**BRE**	**ARE**	**WRE**	**Tavg**
Rec25/30,15/2513	0	1.671	2.712	3.701	7.451
	0.1	0.835	2.085	3.223	12.64
	0.2	1.080	2.220	3.053	16.34
	0.3	0.663	2.205	3.307	19.36
	0.4	1.105	2.275	3.265	26.06
	0.5	0.835	2.134	3.223	25.81
	0.6	0.884	2.115	3.307	33.33
	0.7	0.982	2.228	3.138	35.49
	0.8	0.761	1.983	3.265	45.97
	0.9	0.859	2.159	3.138	51.93
	1.0	1.031	2.195	3.138	73.33

the performance of PSOMA. The effect of p_s on searching quality and computational time when solving Rec25 problem is illustrated in Table VIII.

From Table VIII, it can be seen that, with the increase of swarm size, the computational time increases, while the searching quality varies with small magnitude. That is to say, the swarm size does not affect the searching quality of PSOMA too much. So, it is recommended to adopt a swarm size of 20 for problems with small and medium size and is suggested to suitably increase the swarm size for problems with large size.

B. Effect of p_{ls}

In PSOMA, the parameter p_{ls} determines the probability to use NEH_1 insertion for the selected pbest permutation. So, it affects the exploitation element as well as the computational time. The effect of p_{ls} on searching quality and computational time when solving Rec25 problem is illustrated in Table IX.

It can be seen that the performances obtained by PSOMA when $p_{ls} = 0$ (that means the NEH-based local search is not used) are worse than those obtained when $p_{ls} > 0$, which demonstrates the effectiveness of incorporating the NEH-based local search into PSOMA. With the increase of p_{ls}, the computational time increases, and the searching quality varies with a small magnitude. That is to say, the p_{ls} does not affect the searching quality of PSOMA too much when $p_{ls} > 0$. So, to compromise between the performances and computational time, it is recommended that $p_{ls} = 0.1$ in our studies.

VII. CONCLUSION AND FUTURE RESEARCH

In this paper, by hybridizing the population-based evolutionary searching ability of PSO with local improvement abilities of some local searches to balance exploration and exploitation, an effective PSOMA has been proposed for PFSSP with the objective to minimize the makespan. Our improvement work can be summarized as the following five aspects. First, to make PSO suitable for solving PFSSP, a ROV rule was presented to convert the continuous positions to job permutations. Second, the NEH heuristic was incorporated into random initialization to generate an initial population with certain quality and diversity. Third, NEH_1 insertion was proposed and probabilistically applied to some good particles selected by using the roulette wheel mechanism to balance the exploration and exploitation abilities. Fourth, SA-based local search was designed and incorporated into the MA to enrich the searching behaviors and to avoid premature convergence, and an effective adaptive meta-Lamarckian learning strategy was employed to decide which neighborhood to be used. Besides, a pairwise-based local search was also applied to enhance the exploitation ability. Simulation results and comparisons demonstrate the superiority of the PSOMA in terms of searching quality and robustness. The future work is to investigate the parameter adaptation of PSOMA and to develop other PSO-based MAs for job shop scheduling problem and multiobjective scheduling problems.

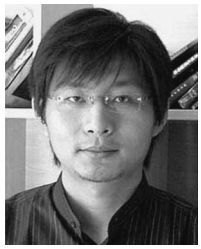
ACKNOWLEDGMENT

The authors would like to thank the editors of this special issue on MAs and the anonymous referees for their constructive comments on the earlier manuscript of this paper.

REFERENCES

[1] H. Stadtler, "Supply chain management and advanced planning-basics, overview and challenges," *Eur. J. Oper. Res.*, vol. 163, no. 3, pp. 575–588, Jun. 2005.
[2] L. Wang, *Shop Scheduling with Genetic Algorithms*. Beijing, China: Tsinghua Univ. Press, 2003.
[3] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.
[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
[5] S. M. Johnson, "Optimal two- and three- stage production schedules with setup times included," *Nav. Res. Logist. Q.*, vol. 1, no. 1, pp. 61–68, Mar. 1954.
[6] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 479–494, Sep. 2005.
[7] B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan, "A general bounding scheme for the permutation flow-shop problem," *Oper. Res.*, vol. 26, no. 1, pp. 53–67, 1978.
[8] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," in "Caltech Concurrent Computation Program," California Inst. Technol., Pasadena, CA, Tech. Rep. 826, 1989.
[9] M. Nawaz, E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.
[10] I. H. Osman and C. N. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.
[11] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, Jan. 1995.
[12] C. R. Reeves and T. Yamada, "Genetic algorithms, path relinking and the flowshop sequencing problem," *Evol. Comput.*, vol. 6, no. 1, pp. 45–60, 1998.

[13] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flow-shop problem," *Eur. J. Oper. Res.*, vol. 91, no. 1, pp. 160–175, May 1996.
[14] L. Wang and D. Z. Zheng, "A modified evolutionary programming for flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 22, no. 7/8, pp. 522–527, Nov. 2003.
[15] P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 449–467, May 2001.
[16] L. Wang and D. Z. Zheng, "An effective hybrid heuristic for flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 21, no. 1, pp. 38–44, 2003.
[17] T. Stutzle, "An ant approach to the flow shop problem," in *Proc. 6th Eur. Congr. Intell. Tech. and Soft Comput.*, Aachen, Germany, 1998, pp. 1560–1564.
[18] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 1061–1071, Sep. 1996.
[19] A. C. Nearchou, "A novel metaheuristic approach for the flow shop scheduling problem," *Eng. Appl. Artif. Intell.*, vol. 17, no. 4, pp. 289–300, Apr. 2004.
[20] L. Wang, L. Zhang, and D. Z. Zheng, "A class of order-based genetic algorithm for flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 22, no. 11/12, pp. 828–835, Dec. 2003.
[21] W. E. Hart, N. Krasnogor, and J. E. Smith, *Recent Advances in Memetic Algorithms*. Heidelberg, Germany: Springer-Verlag, 2004.
[22] P. Merz, "Memetic algorithms for combinatorial optimization problems: fitness landscapes and effective search strategies," Ph.D. dissertation, Univ. Siegen, Siegen, Germany, 2000.
[23] A. Quintero and S. Pierre, "Sequential and multi-population memetic algorithms for assigning cells to switches in mobile networks," *Comput. Netw.*, vol. 43, no. 3, pp. 247–261, Oct. 2003.
[24] P. M. França, A. Mendes, and P. Moscato, "A memetic algorithm for the total tardiness single machine scheduling problem," *Eur. J. Oper. Res.*, vol. 132, no. 1, pp. 224–242, Jul. 2001.
[25] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.
[26] M. Milano and A. Roli, "MAGMA: A multiagent architecture for metaheuristics," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 925–941, Apr. 2004.
[27] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, Univ. West England, Bristol, U.K., 2002.
[28] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.
[29] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb. 2006.
[30] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA J.*, vol. 41, no. 4, pp. 687–696, 2003.
[31] L. Wang, L. Zhang, and D. Z. Zheng, "A class of hypothesis-test based genetic algorithm for flow shop scheduling with stochastic processing time," *Int. J. Adv. Manuf. Technol.*, vol. 25, no. 11/12, pp. 1157–1163, Jun. 2005.
[32] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, Aug. 1998.
[33] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers, 2001.
[34] B. Liu, L. Wang, H. Jin, and D. X. Huang, "Advances in particle swarm optimization algorithm," *Control Instruments Chemical Industry*, vol. 32, no. 3, pp. 1–6, 2005.
[35] B. Liu, L. Wang, Y. H. Jin, F. Tang, and D. X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons Fractals*, vol. 25, no. 5, pp. 1261–1271, Sep. 2005.
[36] M. F. Tasgetiren, M. Sevkli, Y. C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for permutation flowshop sequencing problem," in *Lecture Notes in Computer Science*, vol. 3172, M. Dorigo, M. Birattari, and C. Blum, Eds. New York: Springer-Verlag, 2004, pp. 382–389.
[37] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, 1994.
[38] T. Aldowaisan and A. Allahverdi, "New heuristics for no-wait flowshops to minimize makespan," *Comput. Oper. Res.*, vol. 30, no. 8, pp. 1219–1231, Jul. 2003.
[39] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[40] S. Kirkpatrick, C. D. Gelat, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[41] L. Wang and D. Z. Zheng, "An effective hybrid optimization strategy for job-shop scheduling problems," *Comput. Oper. Res.*, vol. 28, no. 6, pp. 585–596, May 2001.
[42] J. Carlier, "Ordonnancements a contraintes disjonctives," *R.A.I.R.O. Recherche Operationelle/Oper. Res.*, vol. 12, no. 4, pp. 333–351, 1978.



Bo Liu received the B.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2001. He was an exchange student with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore, in 2001. He is currently working toward the Ph.D. degree at Tsinghua University, Beijing, China.

His research interests include particle swarm optimization, differential evolution, chaotic dynamic, and production scheduling.

Ling Wang received the B.Sc. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1995 and 1999, respectively.

Since 1999, he has been with the Department of Automation, Tsinghua University, where he became an Associate Professor in 2002. His current research interests are mainly in optimization theory and algorithms, and production scheduling. He has published two books: *Intelligent Optimization Algorithms with Applications* (Tsinghua University and Springer Press, 2001) and *Shop Scheduling with Genetic Algorithms* (Tsinghua University and Springer Press, 2003). He has also authored over 100 refereed international and domestic academic papers. He is an Editorial Board Member of the International Journal of Automation and Control (IJAAC). He was also a reviewer for many international journals such as the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, *European Journal of Operational Research*, *Computers and Operations Research*, *Computers and Industrial Engineering*, *Artificial Intelligence in Medicine*, and so on.

Dr. Wang received the Outstanding Paper Award in the International Conference on Machine Learning and Cybernetics (ICMLC'02) organized by the IEEE SMC Society in 2002 and the National Natural Science Award (first place prize) nominated by the Ministry of Education of China in 2003. He was the Young Talent of Science and Technology of Beijing City awardee in 2004.

Yi-Hui Jin received the B.S. degree in power mechanism from Tsinghua University, Beijing, China, in 1959 and as a graduated student, received a degree from the Mechanical School, Tsinghua University, in 1963.

Since 1963, she joined the Department of Power Mechanism and Department of Automation. Now, she is a Full Professor with Tsinghua University. Her current research fields are in modeling, control, and optimization theory and applications, production planning and scheduling, and supply chain management. She has published two books: *Process Control* (Tsinghua University Press, 1993) and *Control and Management for the Process Systems* (China Petrifaction Press, 1998). She has also published more than 200 papers in international and domestic journals and conferences.

Prof. Jin obtained several awards from Ministries of China and some conferences, such as the First Grade Prize of Advancement in Science and Technology in 2001. She is the Vice Chairman of the Process Control Committee and a member of the Expert Consulting Committee of the Chinese Association of Automation.