An Effective Hybrid Ant Colony Optimization for Permutation Flow-Shop Scheduling

Xiaoxia Zhang*, Jiewei Tong and Yunyong Ma

College of Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, P.R. China

Abstract: This paper proposes a hybrid ant colony optimization algorithm to solve the permutation flow-shop scheduling (PFS). The hybridization of ant colony optimization (ACO) with path relinking (PR), which combines the advantages of two individual algorithms, is the key innovative aspect of the approach. Path relinking (PR) can be interpreted as an evolutionary method where the high quality solutions are generated by introducing features of the guiding solution gradually into the initial solution. Moreover, the effective hybrid algorithm is a method to integrate intensification and diversification in the search, and it adopts the criterion function restricting the frequencies of using the PR procedure to improve the convergence speed. Finally, the proposed algorithm is applied to PFS benchmark problems. The experimental results have shown that the hybrid method yields better results to solve the permutation flow-shop scheduling than well-known existing methods in terms of solution quality.

Keywords: Ant Colony Optimization, Permutation Flow-shop Scheduling, Path Relinking, Hybrid Algorithm.

1. INTRODUCTION

The permutation flow-shop scheduling (PFS) [1] has been becoming an important study field in both manufacturing systems and industrial process for improving the utilization of resources, and therefore it is crucial to develop efficient scheduling technologies. The classical PFS consists of a set $N= \{J_1,\ldots, J_n\}$ of n different jobs to be executed on a set $M =\{M_1,\ldots, M_m\}$of m machines with the objective of finding the permutation of jobs that minimizes the makespan. Each job J_j is composed of m stages, named operations, and every operation has a non-negative processing time. The processing time of job J_i on machine M_j is denoted by t_{ij} (i=1,2,…, n, j=1,2,…, m), while no operation can be preempted. Each job has exactly one operation to be processed on each machine and the sequence of processing a job on all machines is identical. At any time, each machine can process at most one job and each job can be processed on at most one machine. Preemption is not allowed; i.e., once an operation is started, it must be completed without interruption. Each job is available and ready for processing at time zero and the setup times are sequence independent. A schedule of this type, i.e., with the same job ordering on all machines, is called a permutation schedule and defined with a complete sequence of all jobs.

The permutation flow shop scheduling problem is often denoted by the symbols $n/m/T/C_{\max}$. n represents the number of jobs, m is the number of machines, T is the processing time and $C_{\max}$ is the makespan. A job permutation is denoted by $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, where n jobs will be sequenced through m machines. Let C(π_i, m) denote the completion time of job π_i, on machine m. The completion time of the permutation flow shop scheduling problem according to the processing sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ is shown as follows:

$$C(\pi_1,1)\cdot = t(\pi_1,1) \tag{1}$$

$$C(\pi_1,j)\cdot = C(\pi_1,j-1)+t(\pi_1,j) \qquad j=2,\cdots,m \tag{2}$$

$$C(\pi_i,1)\cdot = C(\pi_{i-1},1)+t(\pi_1,1) \qquad i=2,\cdots,n \tag{3}$$

$$C(\pi_i,j)\cdot = \max\{C(\pi_{i-1},j),C(\pi_i,j-1)\}+t(\pi_i,j)$$
$$i=2,\cdots n; j=2,\cdots m-1 \tag{4}$$

$$C(\pi_i,m)\cdot = \max\{C(\pi_{i-1},m),C(\pi_i,m-1)\}+t(\pi_i,m) \quad i=2,\cdots n \tag{5}$$

In the above recursion, the departure time of the first job on each machine is calculated first, then the second job, and so on until the last job. Eqs. (1) and (2) define the departure time of job π_1 through machine 1 to machine m, making sure that at any time, each machine can process at most one job and each job can be processed on one machine at most. Eq. (3) specifies the departure time of job π (i) on machine 1 (j = 2,3,…,m). Eq. (4) specifies the departure time of job π_i on machine j = 2,3, … ,m-1, which ensures no job passing is allowed. Eq. (5) gives the departure time of job π_i, i = 2,3, …,n, on the last machine. Finally, the makespan of the job permutation π= { π_1, π_2, … , π_n} can be defined as :

$$C_{\max}(\pi_n) = C(\pi_n, m) \tag{6}$$

The goal of the permutation flow shop problem is to find the most suitable arrangement of π^* in the set of all permutations $\Im$ such that

*Address correspondence to this author at the College of Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, P.R. China; Tel: 0086-0412-5929812; Fax: 0086-0412-5929818;
E-mail: aszhangxx@163.com

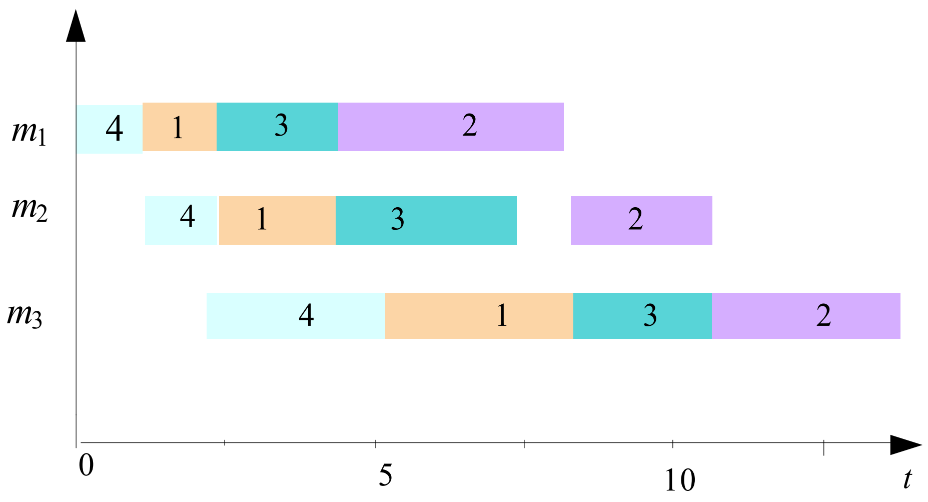

Fig. (1). An example for calculating makespan.

$$C_{\max}(\pi^*) \le C(\pi_n, m) \qquad \forall \pi \in \Im \qquad (7)$$

The following example illustrates the calculation of makespan in detail with a permutation $\pi = \{4, 1, 3, 2\}$. Suppose there are four jobs and three machines, the processing time $t_{i,j}$ is given by

$$[t_{i,j}]_{4\times 3} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

Then the departure time $C(\pi_i, j)$ is calculated as follows (See in Fig. (**1**)):

$$C(\pi_1,1) = t_{\pi_{1,1}} = 1$$

$$C(\pi_1,2) = C(\pi_1,1) + t_{\pi_{1,2}} = 2$$

$$C(\pi_1,3) = C(\pi_1,2) + t_{\pi_{1,3}} = 5$$

$$C(\pi_2,1) = C(\pi_1,1) + t_{\pi_{2,1}} = 5$$

$$C(\pi_2,2) = \max\{C(\pi_2,1), C(\pi_1,2)\} + t_{\pi_{2,2}} = 4$$

$$C(\pi_2,3) = \max\{C(\pi_2,2), C(\pi_1,3)\} + t_{\pi_{2,3}} = 8$$

$$C(\pi_3,1) = C(\pi_2,1) + t_{\pi 3,1} = 4$$

$$C(\pi_3,2) = \max\{C(\pi_3,1), C(\pi_2,2)\} + t_{\pi_{3,2}} = 7$$

$$C(\pi_3,3) = \max\{C(\pi_3,2), C(\pi_2,3)\} + t_{\pi_{3,3}} = 10$$

$$C(\pi_4,1) = C(\pi_3,1) + t_{\pi_{4,1}} = 8$$

$$C(\pi_4,2) = \max\{C(\pi_4,1), C(\pi_3,2)\} + t_{\pi_{4,2}} = 10$$

$$C(\pi_4,3) = \max\{C(\pi_4,2), C(\pi_3,3)\} + t_{\pi_{4,3}} = 13$$

Thus, the makespan is:

$C(\pi_4,3) = 13$

The permutation flow-shop scheduling (PFS) is well-known NP-hard problem, and provides a challenging area for both exact algorithms and metaheuristic approaches. Many researchers have worked on this problem and experimented with many different approaches of exact and approximate algorithms over the years to solve the permutation flow-shop scheduling problem with the objectives of minimizing makespan and total flow-time of jobs, considered either separately or simultaneously. Exact solution methods can only be adopted to solve very small instances, so for real-world problems, to obtain better results, researchers have recently focused on the use of metaheuristic algorithms in solving the problem [2, 3].

A lot of research work has been carried out and most approximate or heuristic methods have been proposed in literature. Most of the published studies for the permutation flow-shop scheduling have focused on the development of heuristics. The majority of the studies have concentrated particularly on designing efficient and effective heuristics as well as taking into account high-quality solutions in short computational times. Ant colony optimization (ACO) algorithm is a population-based, cooperative search procedure which is derived from the behavior of real ants. The main idea in ant colony optimization is to mimic the pheromone trail used by real ants, which are capable of finding the shortest path to food sources from their nests as they forage for food in nature. Ant colony optimization (ACO) algorithm was first proposed by Dorigo, Maniezzo and Colorni [4] and successfully used to solve the traveling salesman problem [5,6]. Stutzle [7] has proposed the first ACO algorithm to solve the PFS with the objective of minimizing the makespan. The ACO algorithm MMAS has been an implementation of the max-min ant system. Rajendran and Ziegler [8] have proposed two ACO algorithms for the PFS with the objective of minimizing the makespan/total flowtime of jobs. Ahmadizar [9] proposed a new ant colony algorithm to solve the PFS with the objective of minimizing the makespan. Because the permutation flow-shop scheduling problem is very complicated, the solution obtained in the basic ACO construction phase is not guaranteed to be locally optimal and becomes the starting point for the local search phase. It is a trend to adopt hybrid ACO to solve very large scale of the permutation flow-shop scheduling that makes use of other heuristics as an intensification and diversification mechanism. The development of modern metaheuristics has led to considerable progress, but each metaheuristic has its own strength and weakness. Therefore, much research has tried to develop the quest for the performance of hybrid algorithms in an effort to achieve the effectiveness and efficiency.

The distinctive characteristics of this paper are different from all the literatures described above on following points: First, the main feature of this hybrid algorithm (ACO&PR) is the utilization of the ACO construction solution mechanism and the use of an improvement method based on path relinking (PR) without losing their unique features. The PR method is embedded into the ACO algorithm as a local search to improve the solutions. Moreover, since some of the parameters in our proposed algorithm can be adjusted dynamically, the hybrid algorithm can search different solution space in order to ensure to escape from the local optimum. Finally, the computational results on the benchmark instances have shown that the hybrid algorithm is comparable to other existing methods in terms of solution quality.

The rest of this paper is organized as follows. Section 2 introduces ant colony optimization (ACO) for the permutation flow-shop scheduling (PFS). Section 3 gives path relinking (PR) for the permutation flow-shop scheduling. Section 4 describes our ACO&PR algorithm to the permutation flow-shop scheduling. Section 5 provides the computational experiments. Finally, Section 6 presents some conclusions.

2. ANT COLONY OPTIMIZATION FOR PFS

Ant colony optimization (ACO) [4] is based on the foraging behavior of ant colonies in nature. As an ant travels, it lays pheromone trail that other ants can follow. The more pheromone trail on the paths can enhance the probability of the next ants to choose. Over time, as more ants are able to complete the shorter route, pheromone accumulates faster on shorter paths and longer paths are less reinforced. A greater amount of pheromone on the path gives an ant a stronger stimulation and thus a higher probability to follow it. Ants are capable of not only finding the best path from their nests to food sources, but also modifying according to the environment as the old path is no longer the best due to a new obstacle. The Ant colony optimization method consists of solution construction and pheromone trail updating.

2.1. Solution Construction

In the PFS, when building a schedule with ant colony optimization, each artificial ant constructs a complete solution by iteratively applying a transition rule. Each ant starts with an empty sequence and chooses one of the jobs. The ant successively selects an unscheduled job to the partial sequence constructed so far until a complete solution is built, i.e., and all jobs have been selected. The ant k at the current position of job i chooses the next job j according to the following probabilistic formula:

$$j = \begin{cases} \arg\max_{l \notin M_k} \{\tau_{il}\}, & \text{if } q \le q_0 \\ S \quad , & \text{otherwise} \end{cases} \tag{8}$$

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}}{\sum_{l \notin M_k} \tau_{il}} & , j \notin M_k \\ 0 \quad , & \text{otherwise} \end{cases} \tag{9}$$

where τ_{il} is the amount of pheromone trail of placing job l in the position i of a sequence. This pheromone trail represents the desirability of processing the job in the given position. The value q is a random number uniformly distributed over [0,1], and q_0 is a tunable parameter in the interval [0,1], which evaluates the relative importance of exploitation versus exploration. If $q \le q_0$ then the best jobs are selected according to Eq. (8); otherwise an edge is chosen according to S, which represents the candidate list and is chosen according to the above probabilistic rule as in Eq. (9). p_{ij}^{k} is the probability distribution with which ant k selects to search from job i to job j. The ants working memory M_k is the set of jobs already selected by an ant and are not considered for choice.

2.2. Pheromone Trail Updating

The pheromone trails are dynamically modified at runtime in order to form a kind of adaptive memory of previously found solution to guide exploring high-quality regions. The pheromone information is modified by both local trail updating after individual solutions have been constructed and global trail updating of the best solution. During constructing a solution, local updating is changed to reduce the amount of pheromone on all chosen jobs in order to simulate the natural evaporation of pheromone and to ensure that no job becomes too dominant. This is performed with the following rule:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\beta \big/ C_k \tag{10}$$

where ρ is a pheromone decay parameter, β is a parameter that determines the relative importance of this sequence. C_k is the makespan of the complete sequence of ant k.

On the other side, to intensify the search in the neighborhood of the best solution, global updating rule is applied after all ants finish their schedules. Only the best schedule is adopted to globally adjust the pheromone trail. Global trail updating provides a greater amount of pheromone trail between adjacent jobs of best schedule. Global trail updating is accomplished according to the following equation,

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho \big/ C_{best} \tag{11}$$

where C_{best} is the makespan of the best sequence produced by ants until the current iteration.

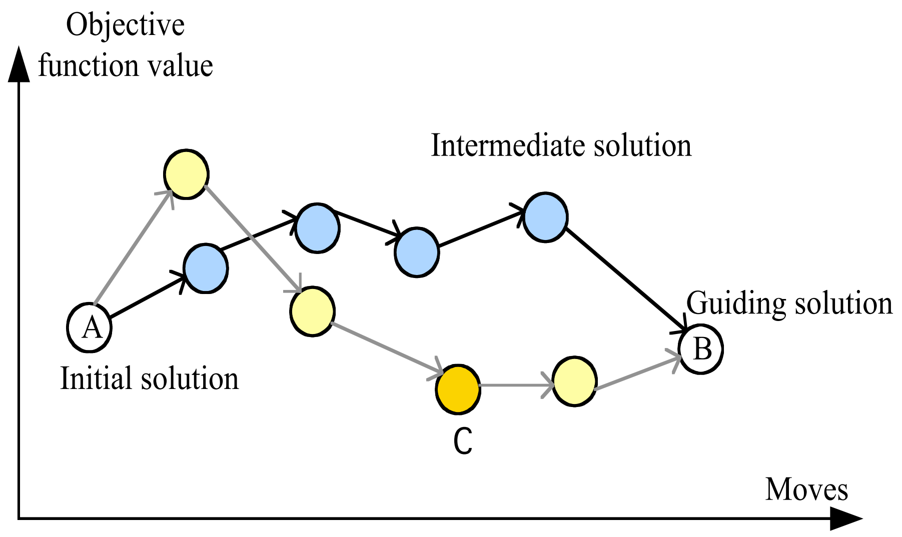

Fig. (2). Process flow of path relinking.

3. PATH RELINKING FOR PFS

Path-relinking (PR) [10] is a metaheuristic that was originally proposed as an approach to integrate diversification and intensification in the context of tabu search [11]. The PR can produce a series of new solutions by searching paths that link initial solutions and guiding solutions in a reference set *RefSet*. The reference set *RefSet* is a set of feasible solutions. It is critical to balance solution quality and solution diversification in a reference set *RefSet.* Therefore, the reference set consists of high quality solutions and diverse solutions. Path-relinking generates new solutions by exploring paths that connect high-quality solutions starting from one of the solutions, called an initial solution, and generating a path in the neighborhood space that leads toward the other solution, named a guiding solution. This path is explored by selecting moves that introduce features contained in the guiding solutions with the aim to systematically reach the guiding solution. In this process, the features of the guiding solution are progressively increased, and the features which are not included in the guiding solution are gradually deleted. Unlike genetic algorithms, where randomness is a key factor in the creation of offsprings from parent solutions to produce new solutions, path relinking adopts systematic, deterministic rules for combining solutions [12]. The path relinking process can generate a series of intermediate solutions with fewer features from the initiating solution and more from the guiding solution. The generating path (i.e., sequences of intermediate solutions) could reasonably hope to find better solutions than the initiating or guiding solution. Process flow of path relinking is illustrated for instance in Fig. (**2**). Two solutions A and B represent an initiating solution and a guiding solution respectively. The path from A to B (dashed line) can yield two better solutions, while the path from A to B (solid line) cannot generate solutions improving A or B.

The Path relinking approach produces paths (trajectories) connecting elite solutions in the neighborhood space. The character of such paths consists in reference to by introducing in the initial solution features of the guiding solution features. The approach may be regarded as a strategy that seeks better moves. At each step, all moves that incorporate features of guiding solutions are analyzed and the best move is selected by creating inducements to favor these features from the guiding solutions. Therefore, it is important to select initial and guiding solutions which denote the starting and ending points of paths during the path relinking phase, since the quality of the intermediate solutions is highly determined by these solutions. Usually the guiding solution is of high quality. To generate the desired trajectory, the initial and guiding solutions are chosen from *RefSet* according to the following three selection criteria.

C1: The guiding solution is chosen to be the best solution in *RefSet*, while the initial solution is the second best one.

C2: The guiding solution is chosen as the best solution in *RefSet*, while the initial solution is defined as the solution with the most diversity.

C3: The initial solution is chosen randomly in *RefSet* and the guiding is defined as the best solution.

For illustration, an example will be described to explain the path relinking process. Consider the following two solutions π^A =(4,5,6,1,3,2) and π^B =(3,1,5,4,2,6) and the path from π^A to π^B. We start with location 1 in solution π^A. Since the job of this location is 4 in π^A (π_1^A=4) and 3 in π^B (π_1^B=3), we then exchange jobs 3 and 4 in solution π^A corresponding to the locations 5 and 1, respectively. So, solution π^A has changed to π^{A_1} =(3,5,6,1,4,2) (i.e. we have performed the move from π^A to π^{A_1}). The job of location 2 in solution π^{A_1} is 5 and in solution π^B is 1. Job 1 is assigned to location 4 in solution π^{A_1}, and therefore we exchange jobs 5 and 1 in solution π^{A_1}, corresponding to locations 2 and 4, respectively. We obtain solution π^{A_2} =(3,1,6,5,4,2). In this way the relinking process continues up to matching solution π^B in five steps.

Table 1. The Comparison of CT_ACO with ACO Algorithm.

Problem	Optimal Solution	ACO	ACO&PR	Improvement
Car1	7038	7038	7038	0
Car2	7166	7166	7166	0
Car3	7312	7312	7312	0
Car4	8003	8102	8003	1.24
Car5	7720	7786	7770	0.20
Car6	8505	8565	8548	0.20
Car7	6590	6670	6590	1.2
Car8	8366	8483	8385	1.17

4. ACO&PR ALGORITHM

In this section, we present our proposed algorithm to solve the permutation flow-shop scheduling. Since the permutation flow-shop scheduling (PFS) is a well-known NP-hard problem, metaheuristics are recommended for searching good solutions within reasonable computation times. We present for this problem a hybrid ant colony optimization algorithm (ACO&PR), in which the search diversification and the search intensification are improved by effectively combining the solution construction mechanism of the ACO with path relinking (PR) without losing their unique features. During the initial population generation, it is critical to balance solution quality and solution diversification. So, we first generate initial solutions by adopting ant colony algorithm and a greedy randomized procedure based on the NEH heuristic [13]. At each iteration, artificial ants probabilistically construct solutions to the problem under consideration using artificial pheromone trails. The PR method is embedded into the ACO algorithm as a local search to improve the solutions. To save computation time, we limit the certain frequencies of applying the PR procedure such as at every 10 iterations, or we execute it only if the best solution is improved. The termination criterion is solution quality or the maximum number of iterations. Repeat the procedure until some termination criterion is satisfied, that is, usually when a sufficient solution is reached or when the number of iterations is satisfied. The general framework of our proposed ACO&PR algorithm can be described as follows:

Step 1: Generate initial solutions. Initialize parameters and generate distinct solutions obtained by ant colony optimization (ACO) and a randomized rule based on the NEH heuristic.

Step 2: Build and maintain the reference set (*RefSet*). The reference set includes quality solutions and diverse solutions. Select b_1 high quality solutions with best objective function values and b_2 diverse solutions with highest diversity from initial feasible solutions, and then let $|RefSet|= b_1 + b_2$ be the reference set.

Step 3: Build a feasible solution with ant colony optimization (ACO).

Step 4: Improve the solution quality by the local search. To transform the feasible solution into one or more enhanced feasible solutions, the PR procedure is executed to enhance the solution quality, after a feasible solution has been generated.

Step 5: Update the best solution *Bestsol.* Calculate the objective function values of the solution, and determine the best solution with the least objective function in the reference set. If the best local optimum is better than the best solution *Bestsol* found so far, replace the best solution *Bestsol* generated so far.

Step 6: Update the reference set. Both building and updating can take the objective function value and the diversity of the solutions into account.

Step 7: Repeat Steps 4-6 until a stopping condition is met.

5. COMPUTATIONAL RESULTS

In this section, the proposed hybrid ant colony optimization has been coded in the visual C++ and run on a LENOVO computer with 2.0GB memory and 2.50Ghz CPU Speed. To evaluate performance of our proposed algorithm for the permutation flow-shop scheduling, the performance of our algorithms was tested on benchmark problems and the test problems can be downloaded from the OR-library web site. The objective of the computational experiments is to evaluate the performance of the algorithms in terms of solution quality. Several experiments were conducted on test problems in order to determine the tendency for the values of parameters. We tested several values for each parameter while all the others were held constant. In most cases, the parameter values have been set through direct numerical experiments. On the basic of a set of preliminary experiments, the algorithm parameters are set with the following settings: q_0=0.85, β=3~5 and ρ=0.05. Other settings were: $|RefSet|$=10, $\tau_0 = 0.01$. Moreover, the algorithm terminates when the maximum iterations reaches 2000, i. e . $CN_{\max}$=2000.

Table 2. Comparison of Heuristics for the Permutation Flow-Shop Scheduling.

Problem	Optimal Solution	NEH	IGA	ACO&PR
Car1	7038	0	0	0
Car2	7166	2.93	2.10	0
Car3	7312	1.19	1.31	0
Car4	8003	1.57	0	0
Car5	7720	3.83	0.98	0.65
Car6	8505	3.15	0.80	0.50
Car7	6590	0	0.16	0
Car8	8366	2.37	0.47	0.23

Table **1** summarizes the results of two algorithms on the problem instances. From the results in Table **1**, where ACO refers to basic ACO, and ACO&PR is the algorithms we proposed, in terms of solution quality. Table **1** reports the comparison between ACO&PR and the basic ACO algorithm without combination with other algorithms. It has shown that ant colony optimization (ACO) can find the better solutions, but our proposed ACO&PR outperforms the basic ACO. This demonstrates that the path-relinking procedure can improve the quality and stability of the solutions obtained by ACO algorithm.

We compare the hybrid ACO&PR algorithm with the better existing approaches available for the PFS, and the results of some problems are shown in Table **2**, where NEH denotes NEH heuristic by Nawaz *et al.* [13], IGA to improved genetic algorithm by Lyer [14], and ACO&PR is the hybrid algorithm we present. The results have indicated that, in terms of solution quality, the hybrid ACO&PR algorithm actually performs even better than the existing methods. Moreover, the computational results show that the hybrid ACO&PR algorithm can obtain the optimal or best known solutions published so far for some benchmark problems. In Table **2**, the last column describes the average percentage deviation. The deviation of solution value is defined as the percentage increase in the objective function value compared to the lower bound. A zero deviation indicates that the algorithms can produce best known solutions. In comparisons to the best known solutions, performance of the hybrid ACO&PR algorithms are statistically excellent in terms of solution quality, and the total average deviation is only 0.17%. These average improvements indicate that our proposed method is more efficient and competitive to solve the permutation flow-shop scheduling (PFS) than other existing methods with respect to solution quality.

6. CONCLUSIONS

In this paper, an effective hybrid ant colony optimization ACO&PR algorithm is developed for solving the permutation flow-shop scheduling (PFS). In this hybrid algorithm, the search intensification and the search diversification are improved by adopting the solution construction mechanism of the ACO and the PR. The PR method is embedded into the ACO algorithm as a local search to enhance the solution quality. The computational results on the benchmark instances have indicated that the hybrid ACO&PR algorithm can obtain the optimal or best known solutions in a short timeframe and it actually performs even better than the existing methods on the permutation flow-shop scheduling.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGMENTS

This research is supported by National Natural Science Foundation of China (Grant No. 51174110).

REFERENCES

[1] M. Widmer and A. Hertz, "A new heuristic method for the flow shop sequencing problem", *European Journal of Operational Research*, vol. 41, no. 3, pp. 186-193, 1989.

[2] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for the permutation flow-shop problem", *European Journal of Operational Research*, vol. 91, no. 1, pp. 160-175, 1996.

[3] H.S. Woo and D.S. Yim, "A heuristic algorithm for mean flowtime objective in flowshop scheduling", *Computers and Operations Research*, vol. 25, no. 3, pp. 175-182, 1998.

[4] A. Clolrni, M. Dorigo and V. Maniezzo, "Distributed optimization by ant colonies", In Proceedings of the First European Conference of Artificial Life (ECAL'91), Elsevier, pp. 134-142, 1991.

[5] M. Dorigo, V. Maniezzo and A.Colorni, "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, *Cybernetics*, vol. 26 , no. 1, pp. 29-41, 1996.

[6] M. Dorigo and L.M. Gambardella, "Ant colonies for the traveling salesman problem", *BioSystems*, vol. 43, no.2, pp. 73-81, 1997.

[7] T. Stutzle, "An ant approach for the flow shop problem", Proceedings of the 6th European congress on intelligent techniques and soft computing, EUFIT'98, vol. 3, pp. 1560-1564, 1998.

[8] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs", *European Journal of Operational Research*, vol. 155, no. 2, pp. 426-438, 2004.

[9] F.Ahmadizar, "A new ant colony algorithm for makespan minimization in permutation flow shops", *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 355-361, 2012.

[10] M. Marti, M. Laguna and F. Glover, "Principles of scatter search", *European Journal of Operational Research*, vol. 169, no. 2, pp. 359-372, 2006.

[11] F. Glover, M. Laguna and M. Marti, "Fundamentals of Scatter Search and Path Relinking", *Control and Cybernetics*, vol. 39, no. 3, pp. 653-684, 2000.

[12] S.C. Ho and M. Gendreau, "Path relinking for the vehicle routing problem", *Journal of Heuristics*, vol. 12, no. 1-2, pp. 55-72, 2006.

[13] M. Nawaz, E. Enscore Jr 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.

[14] S.K. Lyer and B. Saxena, "Improved genetic algorithm for the permutation flowshop scheduling problem", *Computers & Operations Research*, vol. 31, no. 4, pp. 593-606, 2004.

Received: August 13, 2013 Revised: August 28, 2013 Accepted: August 28, 2013