Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/081L26

An Exchange Local Search Heuristic based Scheme for Permutation Flow Shop Problems

Ruey-Maw Chen[*] *and Fu-Ren Hsieh*

Department of Computer Science and Information Engineering, NCUT, Taichung 41170, Taiwan, ROC.

Received: 2 May. 2013, Revised: 27 Aug. 2013, Accepted: 28 Aug. 2013
Published online: 1 Apr. 2014

Abstract: In manufacturing organization, one of the mechanisms to minimize *makespan* (the total completion time) requires machines and operations to be in accordance with a specific flow sequence, i.e., each production run has the same processing order. Determining the specific flow sequence to minimize the *makespan* is the well-known scheduling problem named permutation flow-shop problem (PFSP) which has been confirmed to be an NP problem. Hence, numerous researchers have devoted themselves to solve this problem effectively. However, many studies suffer from high computation complexity. This work suggests a simple exchange local search heuristic scheme with amended simulated annealing to efficiently solve PFSP. The simple exchange local search is adopted for generating the solution (product run) to reduce excessive computation complexity. Meanwhile, an amended simulated annealing strategy concerning relative energy change is applied to stable acceptance probability of the hill-climbing. Moreover, a threshold for relative energy change is designed to prevent divergence. Restated, a simple, stable and low computation complexity scheme is proposed to handle NP-complete PFSP. Experimental results illustrate that the proposed scheme provides an effective and efficient way to shorten *makespan* as required in production lines.

Keywords: PFSP, metaheuristics, exchange local search, simulated annealing

1 Introduction

Generally, scheduling problems involve the resources allocation (such as processors or machines) to run a set of activities (such as jobs, tasks, or processes) satisfying required constraints and optimizing desired criteria. Hence, scheduling algorithms must determine a schedule for a set of processes that satisfies the prerequisite constraints. There are many classes of real-world scheduling problems; however, this study concentrates on the permutation flow shop problem (PFSP), a particular problem of FSP, in which the processing order of the activities is always the same for every machine. Restated, the machines and operations are in accordance with specific flow sequence, i.e., each production run has the same processing order. PFPS applications can be found in a large number of real world environments including manufacturing, maintenance and warehousing operations, as well as in healthcare. Flowshop scheduling is common in a no-wait production line with cyclic scheduling, where multiple activities enter and leave the production line during a cycle; this type of system usually has a larger throughput rate than simpler ones. For example, MPEG-4 macroblock decoding is an application of a permutation flowshop problem for synchronization in co-processor systems while implementing tasks with low turnaround time [1]. A multi-degree cyclic scheduling of a permutation flowshop with two robots was studied by Che and Chu [2].

The goal of the scheduling algorithm in solving PFPS is to minimize the *makespan*, i.e., the total completion time. However, the PFSP has been confirmed to be NP-hard [3]. The solution space consists of n! possible solutions for n activities. However, finding the optimal solution for PFSP problems with exact algorithms such as the branch-and-bound method [4,5] is possible, but impractical when problem size (n) increases. Instead, many approximation algorithms and heuristics have been suggested for finding near optimal solutions with less effort, such as the CDS heuristic [6], the NEH algorithm [7], etc. However, all of these schemes require a substantial amount of computational effort to find solutions that are usually far from optimal. To efficiently obtain high quality solutions in reasonable time, many

[*] Corresponding author e-mail: raymond@ncut.edu.tw

metaheuristics have been proposed for solving PFSP, in particular genetic algorithms (GA) [8,9], simulated annealing (SA) [10], tabu search (TS) [11], ant colony optimization (ACO) [12,13], and particle swarm optimization (PSO) [14,15,16], etc. Additionally, metaheuristics are often integrated with a local search to enhance the algorithms efficiency. Many of these metaheuristics may result in acceptable schedules close to optimal solutions. However, they are often either very complex to implement or suffer from high computational complexity. Hence, Ruiz and Sttzle proposed the iterated greedy (IG) [17] and tried to provide a simple iterated greedy local search based on NEHT heuristic [7] to simplify implementation and reduce computational complexity. Nevertheless, destruction and construction phases are still needed for each IG iteration. During the destruction phase, d randomly chosen jobs are removed from the permutation; d jobs are then inserted back to finish a complete permutation based on the NEHT heuristic during the construction phase. Hence, the complexity of NEHT is still $O(n^2m)$ which is quite time consuming for large instances.

This study proposes a simple exchange with amended simulated annealing (SEASA) scheme to reduce the excessive computational complexity and simplify the implementation, while the method still generalizes to other flow-shop variants. The SEASA applies a simple exchange local search to explore the local neighborhood of each PFSP solution. As in artificial intelligence, this local search is easy to integrate into trajectory meta-heuristics, like simulated annealing, tabu search, and others. Basically, simulated annealing is a memory-less operation. Therefore, an amended simulated annealing strategy is also included to exclude the memory requirement. Restated, the acceptance criterion of the hill-climbing in simulated annealing is modified to stabilize the acceptance probability for PFSP based on relative energy change instead of energy difference. The acceptance criterion is the key factor of simulated annealing which enables the simulated annealing to escape from local minima. Simulated annealing has proven to be a good choice to existing algorithms for hard combinatorial optimization problems.

This study is organized as follows: Section 2 introduces the definition of the studied PFSP problem. Section 3 presents the details of the proposed scheme, simple exchange with amended simulated annealing, for solving PFSP. Section 4 illustrates the simulated cases and experimental results as well as the effectiveness and efficiency of the compared to that of other state-of-the-art schemes. Finally, Section 5 presents the conclusions.

2 Permutation Flow Shop Problem

In this study, a named permutation flow shop problem (PFSP) [3] is investigated. In PFSP, the processing order of the activities has to be the same (in accordance with specific flow sequence) for every machine. The PFSP can be defined as follows:

•At any time, each machine can only process one activity. Meanwhile, the set-up times of activities on machines are included in the processing times.

•A set $N = (1,...,n)$ is composed of n independent activities. And a set $M = (1,...,m)$ consists of m independent machines in the system. These n activities need to be processed on the m machines.

•Each activity j ($j \in N$) which contains m operations $o_{j,k}$ ($k=1,..., m$) to be processed with the processing time $p_{j,i}$ on every machine i ($i \in M$); all of the activities are non-preemptive and non-segmented. The operation $o_{j,k}$ ($k = 2,..., m$) is not allowed to process until $o_{j,k-1}$ is finished.

•All activities should be processed with the same processing order with a permutation, i.e., $\pi=\{\pi(1),...,\pi(n)\}$ from the fist machine to the last machine, where $\pi(r)$ indicates the r^{th} order processed activity. Here, π illustrates the solution to the PFSP.

•The target of PFSP is widely defined as obtaining the minimum *makespan*. The processing time is critical for the minimization of the *makespan*.

For example, imagine a scenario with 3 activities and 5 machines where the processing time $p_{j,i}$ for activity j processing on machine i is shown in Table 2.1. There are 3! solutions in the solution space. With all possible permutation sequences of this scenario: S_1={1, 2, 3}, S_2={1, 3, 2}, S_3={2, 1, 3}, S_4={2, 3, 1}, S_5={3, 1, 2} and S_6={3, 2, 1} as examples, six resulting makespans are illustrated in Table 2.2.

Table 2.1: Processing time ($p_{j,i}$) for representative example.

Machine/Activity	1	2	3
1	2	3	4
2	3	1	1
3	1	2	4
4	2	4	2
5	4	2	1

Table 2.2: Possible solutions with different *makespans*

Solution	*makespan*
S_1={1,2,3}	**17**
S_2={1,3,2}	19
S_3={2,1,3}	**17**
S_4={2,3,1}	20
S_5={3,1,2}	18
S_6={3,2,1}	21

Table 2.2 indicates that the S1 and S3 accomplish the scheduling with shorter *makespans*. Restated, different activities processing orders yield different *makespans*. The objective of this work is to generate a flow sequence which finds the minimum *makespan*.

3 Solving PFSP by Simple Exchange with Simulated Annealing

This section outlines the details of the proposed SEASA scheme including the procedures of SEASA, exchange local search and amended simulated annealing. The procedures of the proposed simple exchange with simulated annealing (SEASA) for PFSP is summarized in Figure 3.1.

•Step 1 is the initialization phase. In this work, NEHT heuristic is applied to generate initial solution S_0.

•At step 2.1, the exchange local search is used for selecting the candidate solution S_t' from the neighbor of S_t (solution at iteration t).

•Steps 2.2 through step 2.5 are the amended simulated annealing process utilized in this study, where $E(S_t)$ and $E(S_t')$ are the related energy values (*makespans*) corresponding to solutions S_t and S_t' respectively.

•At step 2.3, the variance of energy $\triangle E$ and the relative energy change $rel\_\triangle E$ are calculated. The details of the scheme are as follows.

1. Initialization: t=0, temperature T_0, cooling rate α, initial solution S_0 generated by NEHT.
2. Loop
   2.1 Simple exchange local search determines the neighbor of S_t as S_t' at exchange stage t.
   2.2 Calculate the energy $E(S_t')$ of S_t'
   2.3 Calculate $\Delta E = E(S_t')- E(S_t)$, $rel\_\Delta E= \Delta E /E(S_t)$
   2.4 if $\Delta E \leq 0$
   then $S_t := S_t'$,
   else if a random variable p, $p \leq e^{-rel\_\Delta E / T_t} \mid p \in U(0,1)$
   and $rel\_\Delta E$< threshold *(Th)*
   then $S_t := S_t'$
   2.5 $T_{t+1} := T_t \times \alpha$
   2.6 t:=t+1
3. Until the End condition is satisfied, return S_t

Fig. 1: The procedures of SEASA.

Initial solution generation. According to the search mechanism, starting with a good initial state (using some heuristics such as greedy) would assist the algorithm in obtaining optimal solutions such as in GA, ACO and PSO. Meanwhile, a good initial solution would also facilitate speeding-up convergence. Most studies apply the initial solution heuristic for improving the initial solution. There are many heuristics which can be effectively utilized as initial solution heuristics for PFSP. NEHT, proposed by Taillard [3], is a variation of NEH with complexity improvements. Moreover, the NEH looks to be the best known polynomial-time heuristic in practice for FSP-class problems. Hence, this study applies NEHT to obtain the initial solution S_0.

Exchange local search. The local search is a way of neighbourhood search. Among many local search schemes, NEH is one famous neighbourhood search method based on insertion; the computational efficiency and solution quality of the NEH have been verified [3]. However, the NEH requires a total of $[n(n+1)/2]-1$ schedules, with n of these schedules being complete sequences. Thus, the time-complexity of the algorithm rises to $O(n^{3m})$ [7]. In NEHT, a data structure is applied to reduce the time-complexity of NEH to $O(n^{2m})$ [3]. Additionally, a variant of IG includes iterative insertion neighborhood [17], but the time-complexity is still $O(n^2)$. Thus, this work applies exchange local search rather than insertion local search for searching the neighborhood of an existing solution. Restated, this investigation further simplifies the neighborhood search operation by a simple exchange which further reduces the time-complexity to $O(n)$.

The suggested exchange local search is easy to implement. Assume that an existing activity processing order is denoted by the permutation . The exchange operation switches the activity at the i^{th} position and the activity at the j^{th} position in π, where $i \neq j$, and i, j are randomly generated. Once a permutation π is obtained as a PFSP solution, in the case of: π=$\{\pi(1),..., \pi(i-1), \boldsymbol{\pi(i)}, \pi(i+1),..., \pi(j-1), \boldsymbol{\pi(j)}, \pi(j+1),..., \pi(n)\}$ is the permutation before exchange local search is applied; the new permutation π'=$\{\pi(1),..., \pi(i-1), \boldsymbol{\pi(j)}, \pi(i+1),..., \pi(j-1), \boldsymbol{\pi(i)}, \pi(j+1),..., \pi(n)\}$ can be obtained after applying the exchange operation. Table 3.1 shows an example of a permutation before exchange local search is applied and three permutations after exchange local search is performed from π. The operation of this simple exchange local search is listed as step 2.1 of Figure 3.1.

Amended simulated annealing. The well-known simulated annealing (SA) algorithm was first proposed for combinatorial optimization (CO) problems by Kirkpatrick *et al.* [18], and it can be classified as a trajectory metaheuristic. The simulated annealing might (with an acceptance probability) allow moving to a neighboring solution with worse quality than the current solution for

escaping from local minima. Restated, simulated annealing relies on an acceptance criterion which is related to an acceptance probability for accepting worse solutions. This probability decreases as time passes.
There are three phases involved in amended SA as displayed from steps 2.2 through 2.5 of Figure 3.1.

Table 3.1: Three exchange examples

exchange operation	determined i^{th} and j^{th}	new permutation
origin	π	{2,1,3,5,4}
1^{st}	(i=3,j=1)	{3,1,2,5,4}
2^{nd}	(i=3,j=5)	{3,1,4,5,2}
3^{rd}	(i=5,j=4)	{3,1,4,2,5}

•At steps 2.2 and 2.3, the energy value related to the solution S_t' is calculated; $E(S_t')$ is denoted as an energy function which corresponds to the makespan of the solution. The variance of energy ($\triangle E$=$E(S_t')$- $E(S_t)$) and the relative energy change ($rel\_\triangle E$=$\triangle E$ /$E(S_t)$) are also calculated.

•At step 2.4, the acceptance criterion is applied, if the variance of energy δE is less than zero, the S_t would be replaced by S_t', *i.e.*, S_t' is accepted. Otherwise, S_t' with worse quality can be accepted with a probability p, where p is a uniform distributed random number. An acceptance probability is calculated using the acceptance probability function which is conventionally defined as in Eq. (1).

$$P_{acp}(S_t, S_t', T_t) = e^{-\Delta E/T_t} = e^{-\frac{E(S_t')-E(S_t)}{T_t}} \quad (1)$$

Here, T_t is the cooling temperature of the t^{th} iteration, and it decreases as the iteration proceeds. Meanwhile, the acceptance probability decreases as the temperature decreases based on Eq. (1). A worse solution will be accepted when the randomly generated number p is smaller than the determined acceptance probability (P_{acp}). However, in PFSP, the energy correlates with the *makespan* of the solution and its scale depends on the problem scale, and so does the variance of energy δE. Consequently, the acceptance probability is greatly affected by the problem scale. Restated, the acceptance probability is sensitive to the problem scale. In this study, the δE in the acceptance probability function is replaced by the relative energy change, $rel\_\Delta E$ as listed in Eqs. (2) and (3):

$$rel\_\Delta E = \frac{\Delta E}{E(S_t)} = \frac{E(S_t') - E(S_t)}{E(S_t)} \quad (2)$$

$$P_{acp}(S_t, S_t', T_t) = e^{-rel\_\Delta E/T_t} \quad (3)$$

Hence, this acceptance probability is less sensitive to the problem scale, since $rel\_\Delta E$ will not dramatically change with the variation of problem dimensions.
Additionally, to prevent accepting an extremely worse solution, a threshold Th based on $rel\_\Delta E$ is included. When $\Delta E > 0$ with the acceptance probability and $rel\_\Delta E$ <threshold (Th) are met, S_t is then accepted as indicated in step 2.4 of the proposed algorithm in Figure 1. Restated, with the help of the threshold, the suggested scheme is able to prevent divergence.

•At step 2.5, the temperature cooling schedule follows an exponential cooling scheme (ECS) with cooling rate as listed in Eq. (4):

$$T_{t+1} = T_t \times \alpha,\ \alpha \in [0,1] \quad (4)$$

4 Experiment

To verify the effectiveness of the proposed simple exchange with simulated annealing scheme (SEASA), the simulation instances from Taillard [19] were tested. Taillard generated a set of PFSP scheduling problems with different combinations of activity (n) and machine (m), $n \in \{20,50,80,\ldots,440,470,500\}$ and $m \in \{5,10,15,20\}$, and there are 10 instances for each problem case. The processing time of each activity is distributed uniformly in the interval [1, 99]. This test suite is obtainable from http://mistic.heig-vd.ch/taillard/ and http://people.brunel.ac.uk/ mastjjb/jeb/orlib/files /flowshop2.txt. In this work, simulations based on computation time were performed and compared with the state-of-the-art algorithms listed in [17]. The termination condition (on the basis of computation time) in [17] was $n \times (m/2) \times 60$, and the algorithm was running on an Athlon XP 1600+ (1400 MHz) system. The proposed scheme was running on a Pentium-M (1.73 GHz) PC. Therefore, the computation time of this work was $n \times (m/2) \times 60 \times (1.4/2.4)$. Additionally, parameter settings of this simulation are: T_0=1, α=0.999, $Th = makespan \times 0.005$ for $n \leqslant 50$, otherwise, $Th = makespan \times 0.001$. The objective of the lower threshold value for large size problems is to reduce the acceptance rate of worse solutions in a large solution space.

The performance is compared with the percentage of deviation from Taillards upper bound (the lowest known *makespan*). Each problem instance was repeated for five independent trials ($R = 5$). Instead of choosing the best trial, the average of the five trials was calculated and all the 10 instances for every dimension problem (*n activities / m machines*) were averaged. The performance criterion used for comparison is the average relative percentage deviation ($\overline{RPD}$) as displayed in Eq. (5) which is defined in [17].

$$\overline{RPD} = \sum_{i=1}^{R} \left(\frac{Sol_i - Best_{sol}}{Best_{sol}} \times 100\%\right)/R \quad (5)$$

where Sol_i is the *makespan* of a solution yielded by any of the R repetitions of the compared algorithms and $Best_{sol}$ is the *makespan* of the optimal solution or upper

bound solution for Taillards instances. The simulation results are displayed in Table 4.1 on the basis of computation times.

As shown in Table 4.1, the proposed scheme is better than some complex meta-heuristics and worse than the best IG_RSLS (proposed by Ruiz and Sttzle [17], HGA_RMA, PACO, and M_MMAS. Although the overall performance of this study is worse than IG_RSLS, HGA_RMA, PACO and M_MMAS, it is much simpler than them and hence very simple and easy to implement.

Moreover, to verify the stability of the proposed scheme, the relative deviation intervals (Avg Dev, average relative percentage deviation; and Max Dev, maximum average relative percentage deviation) are displayed in Table 4.2. The average and maximal deviations are less than 2.2% and 5.1%, respectively; hence, the suggested algorithm is stable for finding solutions. In this work, more than 8500, 3500, 2000, 1000 and 400 schedules (for one instance of 20/20, 50/20, 100/20, 200/20 and 500/20 cases, respectively) can be yielded by our scheme in one second. That is, the consumed CPU time to find a feasible solution for every instance for the largest case (500/20 case) would be less than 2.5 ms. Hence, the proposed scheme is effective for solving application problems which are similar to PFSP.

Additionally, the performance of the different algorithms is compared based on the number of solutions generated. Each problem instance was tested for five independent trials, where the best trial was chosen and the 10 different instances for every problem size were averaged.

Comparisons with the ant colony system (ACS) for PFSP proposed by Ying *et al.* [20] were conducted using 20 ants and 5,000 iterations or the lowest bound as terminating conditions. SEASA was set to terminate when the number of solutions reached 205,000=100,000, that is, 100,000 iterations. Meanwhile, comparisons are made to ACS and ant colony optimization based on interchange-moves local search (ACS+) [12], where the ACS+ uses 10 ants and 10 interchange-moves for 5,000 iterations. Thus, the comparison criterion (*on the basis of solutions generated*) is computed as the relative percentage deviation ($\overline{RPD}$) as defined above. The simulation results demonstrate that SEASA has better performance than the ACS, conventional SA, and ACS+ as shown in Table 4.3.

5 Conclusions

Many applications are an extension of PFSP. However, PFSP has been confirmed to be an NP problem, thus, this investigation suggested a strategy named simple exchange with amended simulated annealing (SEASA) with the objective of achieving a scheme that it is simple and easy to implement.

The required computational complexity for constructing PFSP is reduced to *O(n)*. Meanwhile, an acceptance criterion design of the amended simulated annealing is based on the relative energy change which can help prevent from acceptance probability turbulence. Additionally, a threshold is introduced to prevent from accepting undesirable solutions. Moreover, only two parameters (α and *Th*) are required to release the laborious experiment test. The threshold value (*Th*) should be low (0.1% of the *makespan*) for large problems to reduce the acceptance of poor solutions.

The simulation results exhibit that, despite the simplicity of the proposed SEASA, SEASA outperforms many complex metaheuristics. The resulting relative percentage deviation by applying SEASA is also compared to state-of-art algorithms (such as Ruiz and Sttzle [17]); SEASA ranks as the 5th on average results as shown in Table 4.1. Meanwhile, the maximum relative percentage deviation is less than 5.1% as indicated in Table 4.2, indicating the stability of the designed algorithm. Hence, SEASA is an effective and efficient algorithm to be used for solving PFSP problems. Moreover, the proposed scheme is able to find out the lowest makespan for all instances of 20/5, 50/5 and 100/5 cases as listed in Table 4.3.

SEASA doesnt seem to perform very well on numerous machine case problems (*m* more than 50). However, many real-world problems utilize fewer machines. Still, one avenue of future work is to improve PFSP for numerous machine case problem instances.

Acknowledgement

This work was partly supported by the National Science Council, Taiwan, under contract NSC 99-2221-E-167-007.

Table 4.1: Comparison of different algorithms based on computation time in [17] (deviation%)

n/m	20/5	20/10	20/20	50/5	50/10	50/20	100/5	100/10	100/20	200/10	200/20	500/20	Average
IG_RSLS	0.04	0.06	0.03	0	0.56	0.94	0.01	0.2	1.3	0.12	1.26	0.78	0.44
HGA_RMA	0.04	0.13	0.09	0.02	0.72	1.28	0.02	0.29	1.66	0.2	1.48	0.96	0.574
PACO	0.21	0.37	0.24	0.01	0.85	1.59	0.03	0.27	2.09	0.27	1.92	1.09	0.75
M_MMAS	0.04	0.15	0.06	0.03	1.4	2.18	0.04	0.47	2.59	0.23	2.26	1.15	0.88
This work	0.24	0.88	0.87	0.19	0.76	2.19	0.09	0.65	1.52	0.66	1.43	1.81	0.94
ILS	0.49	0.59	0.36	0.20	1.48	2.20	0.18	0.68	2.55	0.56	2.24	1.25	1.06
GA_RMA	0.26	0.73	0.43	0.07	1.71	2.74	0.07	0.62	2.75	0.43	2.31	1.40	1.13
GA_REEV	0.62	2.04	1.32	0.21	2.06	3.56	0.17	0.85	3.41	0.55	2.84	1.66	1.61
GA_AA	0.94	1.54	1.43	0.36	3.72	4.69	0.32	1.72	4.91	1.27	4.21	2.23	2.28
SA_OP	1.09	2.63	2.38	0.52	3.51	4.52	0.30	1.48	4.63	1.01	3.81	2.52	2.37
GA_MIT	0.80	2.14	1.75	0.30	3.55	5.09	0.27	1.63	4.87	1.14	4.18	3.34	2.42
NEHT	3.35	5.02	3.73	0.84	5.12	6.26	0.46	2.13	5.23	1.43	4.41	2.24	3.35
GA_CHEN	4.15	5.18	4.26	2.03	6.54	7.74	1.35	3.80	8.15	2.76	7.24	4.72	4.83
SPRIT	4.33	6.07	4.44	2.19	6.04	7.63	1.06	3.01	6.74	2.07	4.97	12.58	5.09

Table 4.2: Deviation interval of the proposed scheme (%).

n/m	20/5	20/10	20/20	50/5	50/10	50/20	100/5	100/10	100/20	200/10	200/20	500/20
Avg Dev	0.24	0.88	0.87	0.19	0.76	**2.19**	0.09	0.65	1.52	0.66	1.43	1.81
Max Dev	1.33	**5.06**	4.71	0.71	3.14	4.37	0.77	2.20	4.11	1.95	3.46	2.64

Table 4.3: Comparison of different algorithms based on solutions generated

n/m	ACS	SA	ACS+	**SEASA**
20/5	1.19	1.27	0.247	**0.00**
20/10	1.70	1.71	1.225	0.11
20/20	1.60	0.86	1.400	0.10
50/5	0.43	0.78	0.107	**0.00**
50/10	1.89	1.98	1.432	0.26
50/20	2.71	2.86	2.732	1.39
100/5	0.22	0.56	0.118	**0.00**
100/10	1.22	1.33	0.874	0.29
100/20	2.22	2.32	2.307	0.87
200/10	0.64	0.83	0.709	0.36
200/20	1.30	1.74	2.011	0.88
500/20	1.68	0.85	1.832	0.97
Average	1.40	1.42	1.250	**0.436**

References

[1] J. Boutellier, S.S. Bhattacharyya and O. Silven, Low-Overhead Run-Time Scheduling for Fine-Grained Acceleration of Signal Processing Systems. Proceedings of the 2007 IEEE Workshop on Signal Processing Systems, Shanghai, China, 457-462 (2007).

[2] A. Che and C. Chu, Multi-degree cyclic scheduling of two robots in a no-wait flowshop, IEEE Transactions on Automation Science and Engineering, **2**,173-182 (2005).

[3] E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem, European Journal of Operational Research, **47**, 65-74 (1990).

[4] U. Dorndorf, E. Pesch and T. Phan-Huy, A branch-and-bound algorithm for the resource-constrained project scheduling problem, Mathematical Methods of Operations Research, **52**, 413-439 (2000).

[5] A. Jalilvand, S. Khanmohammadi and F. Shabaninia, Scheduling of sequence-dependant jobs on parallel multiprocessor systems using a branch and bound-based Petri net, Proceedings of the 2005 IEEE Symposium onEmerging Technologies, Islamabad, Pakistan, 334-339 (2005).

[6] H. G. Campbell, R. A. Dudek and M. L. Smith, A heuristic algorithm for the n-job, m-machine sequencing problem, Management Science, **16**, B630-B637 (1970).

[7] M. Nawaz, E. E. Enscore and I. Ham, A heuristic algorithm for the m-machine, n-job flowshop sequencing problem, OMEGA, The International Journal of Management Science, **11**, 91-95 (1983).

[8] S. H. Chen, P. C. Chang, and Q. Zhang, A self-guided genetic algorithm for flowshop scheduling problems, Proceedings of the IEEE Congress on Evolutionary Computation, Trondheim, Norway, 471-478 (2009).

[9] M. C. Wang, Y. Q. Rao and K. P. Wang, A niche genetic algorithm for two-machine flowshop scheduling with family sequence-dependent setup times and a common due window, Proceedings of the 2010 IEEE International Conference on Industrial Engineering and Engineering Management, Macao, 296-300 (2010).

[10] N. Javadian, A. Mozdgir, E. G. Kouhi, D. Qajar and M.E. Shiraqai, Solving assembly flowshop scheduling problem with parallel machines using Variable Neighborhood Search, Proceedings of the 2009 International Conference on Computers & Industrial Engineering, Troyes, France, 102-107 (2009).
[11] S. Liu, J.H. Cui and Y. Li, Heuristic-Tabu Algorithm for Hybrid Flowshop Scheduling with Limited Waiting Time, Proceedings of the 2008 International Symposium on Computational Intelligence and Design, Wuhan, China, 233-237 (2008).
[12] R. M. Chen, S. T. Lo, C. L. Wu and T. H. Lin, An effective ant colony optimization-based algorithm for flow shop scheduling, Proceedings of the 2008 IEEE Conference on in Soft Computing in Industrial Applications, Muroran, Japan, 101-106 (2008).
[13] W. Yang, Y. Zhou and K. Li, A novel Ant algorithm for permutation flow shop problem, Proceedings of the 5th International Conference on Bio-Inspired Computing: Theories and Applications, Changsha, China, 1093-1097 (2010).
[14] J. Kennedy and R. C. Eberhart, Particle swarm optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks,; Perth, WA , Australia, 1942-1948 (1995).
[15] H. B. Tang and C. M. Ye, Permutation flow shop scheduling algorithm based on a hybrid particle swarm optimization, Proceedings of the IEEE 17th International Conference on Industrial Engineering and Engineering Management, Xiamen, China, 557-560 (2010).
[16] L. Tang and X. Wang, An Improved Particle Swarm Optimization Algorithm for the Hybrid Flowshop Scheduling to Minimize Total Weighted Completion Time in Process Industry, IEEE Transactions on Control Systems Technology, **18**, 1303-1314 (2010).
[17] R. Ruiz and T. Sttzle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research, **177**, 2033-2049 (2007).
[18] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, **220**, 671-680 (1983).
[19] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research, **64**, 278-285 (1993).
[20] K. C. Ying and C. J. Liao, An ant colony system for permutation flow-shop sequencing, Computers and Operations Research, **31**, 791-801 (2004).

Ruey-Maw Chen received the B. S., the M. S. and the PhD degrees in engineering science from National Cheng Kung University of Taiwan R.O.C. in 1983, 1985 and 2000, respectively. From 1985 to 1994 he was a senior engineer on avionics system design at Chung Shan Institute of Science and Technology (CSIST). He was a networking engineer at Chinyi Institute of Technology during 1994 to 2002. Since 2002, he has been with the Department of Computer Science and Information Engineering, National Chinyi University of Technology (NCUT), where he is an associate professor. His research interests include meta-heuristics, scheduling optimization with applications and computer networks.

Fu-Ren Hsieh was born at Taipei, Taiwan, R.O.C. He received the B.S. degree in computer science and information engineering from National Chin-Yi University of Technology of Taiwan R.O.C. in 2010. Since 2010, he is a graduate student, his major investigations focus on resource-constrained project scheduling problem, flow-shop scheduling problem and vehicle routing problem.