International Journal of Innovative
Computing, Information and Control
Volume 8, Number 10(A), October 2012

ICIC International ©2012 ISSN 1349-4198
pp. 6599–6612

MINIMIZING TOTAL FLOW TIME IN PERMUTATION FLOWSHOP ENVIRONMENT

Shih-Wei Lin[1], Chien-Yi Huang[2], Chung-Cheng Lu[3] and Kuo-Ching Ying[2,*]

[1]Department of Information Management
Chang Gung University
No. 259, Wen-Hwa 1st Rd., Kwei-Shan, Tao-Yuan 333, Taiwan

[2]Department of Industrial Engineering and Management
[3]Institute of Information and Logistics Management
National Taipei University of Technology
No. 1, Sec. 3, Chung-hsiao E. Rd., Taipei 10608, Taiwan
*Corresponding author: kcying@ntut.edu.tw

Received June 2011; revised October 2011

Abstract. *The permutation flowshop scheduling problem with the objective of minimizing total flow time is known as a NP-hard problem, even for the two-machine cases. Because of the computational complexity of this problem, a multi-start simulated annealing (MSA) heuristic, which adopts a multi-start hill climbing strategy in the simulated annealing (SA) heuristic, is proposed to obtain close-to-optimal solutions. To examine the performance of the MSA algorithm, a set of computational experiments was conducted on a well-known benchmark-problem set from the literature. The experiment results show that the performance of the traditional single-start SA can be significantly improved by incorporating the multi-start hill climbing strategy. In addition, the proposed MSA algorithm is highly effective and efficient as compared with the other state-of-the-art meta-heuristics on the same benchmark-problem instances. In terms of both solution quality and computational expense, the proposed algorithm contributes significantly to this extremely challenging scheduling problem.*

Keywords: Scheduling, Permutation flowshop, Total flow time

1. **Introduction.** Scheduling is a decision-making process for optimally allocating resources [1]. Efficient scheduling has become essential for manufacturing firms to survive in today's intensely competitive business environment [2,3]. As one of the best known production scheduling problems, permutation flowshop sequencing problems (PFSPs) have long been a topic of interest for the researchers and practitioners in this field [4]. Since Johnson's pioneering work [5], more than one thousand papers on various aspects of PFSPs have been published in the operational research literature, most of which considered the objective of minimizing makespan [6]. Recently, the objective of minimizing total flow time or, equivalently, total completion time if all jobs are available for processing at the beginning, has attracted more attention from researchers. As per the standard three-field notation, introduced by Graham et al. [7], the PFSP with the criterion of total flow time can be denoted as $F \parallel \sum C_j$.

Total flow time is one of the most important performance measures, because, in practice, it can lead to stable utilization of resources, rapid turn-around of jobs, and minimization of work-in-process inventory costs [8]. Therefore, minimizing the total flow time is a very important objective for scheduling in many flowshop systems in, for instance, electronics, chemicals, textile, food and the service industries. Because the flowshop has been widely applied in many manufacturing and service systems, the $F \parallel \sum C_j$ problem has become a subject of continuing interest for researchers and practitioners. However, even a relatively simple $F \parallel \sum C_j$ problem involving only two machines is known as a strongly *NP*-hard problem [9]. Although exact methods, such as branch and bound [10-13] and mixed integer linear programming [14], have been developed to obtain optimal solutions to this problem, these techniques may be computationally intensive, even for moderate-size instances, and thus cannot be applied to solve the problem instances with practical sizes. Instead, heuristic methods that guarantee to obtain acceptable solutions with reasonable computational resources have the advantage in computational efficiency. Existing heuristic methods for solving this problem can be classified into two categories: constructive heuristics and improvement heuristics. In a constructive heuristic, once a job sequence is determined, it is fixed and cannot be reversed [15,16]. In the past decades, several constructive heuristic methods [8,17-29] have been proposed for solving the $F \parallel \sum C_j$ problem, such as FL, RZ, WY, H(2) and CH3, presented by Framinan and Leisten [8], Rajendra and Ziegler [26], Woo and Yim [27], Liu and Reeves [28], and Li and Wu [29], respectively. However, these constructive heuristics are simple methods that aim to quickly obtain feasible solutions without guaranteeing solution quality, especially for large-scale instances [30]. On the other hand, an improvement heuristic starts with an initial solution and then provides a scheme for iteratively obtaining an improved solution until reaching stopping criteria [31,32]. Computational results showed that FLR2, IH7-FL, and FLR1, proposed by Allahverdi and Aldowaisan [33], Framinan and Leisten [8], and Framinan et al. [34], respectively, significantly outperform other heuristics in the literature of the $F \parallel \sum C_j$ problem. For further extensive reviews on this problem, the reader is referred to Gupta and Stafford Jr. [6].

Recent interests in developing efficient improvement heuristics have resulted in considerable attention to meta-heuristics that are particularly attractive for large-scale instances [35]. Meta-heuristics typically refer to a general algorithmic framework that can be applied to different combinatorial optimization problems with minor modifications [36-39]. Meta-heuristics for the $F \parallel \sum C_j$ problem include genetic algorithm [40-42], ant colony optimization [43,44], particle swarm optimization [45,46], neural network algorithm [47], iterated local search [48], estimation of distribution algorithm [49], and differential evolutionary algorithm [50].

Among the modern meta-heuristics, the simulated annealing (SA) algorithm, proposed by Metropolis et al. [51], has emerged as a highly effective and efficient algorithmic approach to *NP*-hard combinatorial optimization problems. However, the search procedure in the SA algorithm typically requires some diversification mechanisms to escape from local optima. Without such mechanisms, the SA algorithm may be trapped in a small area of the solution space, missing the possibility of finding a global optimum [52]. One way to achieve diversification is to utilize the multi-start hill climbing strategy which performs the search procedure with several starting points. In light of this strategy, this study proposes the effective and efficient multi-start simulated annealing (MSA) heuristic to solve the $F \parallel \sum C_j$ problem. This novel MSA heuristic combines the advantages of the SA algorithm in effectively searching solution space and of the multi-start hill climbing strategy in escaping from local optima, and offers a significant contribution to the growing body of the literature for solving PFSPs.

The rest of this paper is structured as follows. After formulating the $F \parallel \sum C_j$ problem in Section 2, the proposed MSA heuristic is described in Section 3. Section 4 reports the computational results of empirically evaluating the effectiveness and efficiency of the proposed MSA algorithm by comparing its performance against the traditional single-start SA and the state-of-the-art algorithms on a benchmark-problem set from the literature.

Finally, conclusions are drawn together with recommendations for future research in Section 5.

2. **Problem Definition.** The $F \parallel \sum C_j$ problem aims at scheduling n jobs on m machines, where each job has one operation on each machine and all jobs are processed in the same technological order on all machines. Meanwhile, the sequence in which each machine processes all jobs is identical on all machines. Besides, the following assumptions are considered in this study:

- Each machine can process at most one job at a time and each job can be processed on only one machine at any given time.
- The schedule is non-preemptive, meaning once a job starts to be processed on a machine, the process cannot be interrupted before completion.
- The number of jobs and their processing times on each machine are known in advance and are non-negative integers.
- The number of machines is known in advance, and all machines are persistently available to process all scheduled jobs as required.
- The individual operation setup times are small compared with their processing times, and are included in the processing times.
- The ready time of all jobs is zero, meaning that all jobs are available for processing at the beginning.

Based on the above definitions and assumptions, the objective is to identify a sequence $\sigma = (\sigma(1), \ldots, \sigma(n))$ for the n jobs so as to minimize the total flow time, $F = \sum_{i=1}^{n} C(\sigma(i), m)$, where σ ranges over all those permutations of n jobs. By imposing the condition that each operation is to be performed as soon as possible, the completion time of each job $\sigma(i)$ on machine $j, C(\sigma(i), j)$, is calculated using the recursive formula: $C(\sigma(i), j) = \max\{C(\sigma(i), j-1); C(\sigma(i-1), j)\} + p_{\sigma(i),j}, j = 1, \ldots, m$, where $p_{\sigma(i),j}$ denotes the processing time of job $\sigma(i)$ on machine j, and $C(\phi, j) = C(\sigma(i), 0) = 0$ for all i and j; ϕ is the initial null schedule.

3. **Proposed Multi-start Simulated Annealing Algorithm.** The proposed multi-start simulated annealing (MSA) algorithm combines the advantages of the SA algorithm and of the multi-start hill climbing strategy. The following subsections further illustrate the solution representation, initial solution, neighborhood, the parameters and the algorithmic procedures in the MSA algorithm.

3.1. **Solution representation and initial solution.** In this study, a sequence of jobs is represented by a string of numbers denoting a permutation of n jobs. For example, the permutation, [4 8 5 2 1 3 7 6], can be decoded as the operation sequence 4-8-5-2-1-3-7-6 of eight jobs on each machine. Half of the initial solutions (at least one) are generated by Nawaz-Enscore-Ham (NEH) heuristic [53], and the rest are generated by randomly ordering the jobs.

3.2. **Neighborhood.** Let S denote the set of feasible solutions and let $\sigma_k (k = 1, \ldots, P_{size})$ represent the current solutions, where $\sigma_k \in S$, and P_{size} denotes the number of starting points in the MSA algorithm. The set $N(\sigma_k)$ consists of the solutions neighboring σ_k, $(k = 1, \ldots, P_{size})$. $N(\sigma_k)$ can be defined by either a swap or an insertion operation. That is, a solution in $N(\sigma_k)$ is generated by randomly selecting a pair of jobs and swapping them, or by randomly selecting one job and inserting the chosen job immediately before another randomly selected job. The probabilities of performing the swap and insertion operations were fixed at 0.5 and 0.5, respectively.

3.3. **Parameters.** The proposed MSA algorithm has six parameters, namely I_{iter}, T_0, T_F, $N_{non\text{-}improving}$, α and P_{size}, where I_{iter} denotes the number of iterations performed at a particular temperature, T_0 represents the initial temperature, T_F is the final temperature (the MSA procedure terminates when the current temperature is below T_F), $N_{non\text{-}improving}$ is the maximum number of reductions in temperature when the incumbent total flow time is not improved, and α denotes the coefficient controlling the cooling schedule.

3.4. **MSA procedure.** The procedure of the proposed MSA algorithm is depicted in Figure 1. First, the current temperature T is set to T_0. Next, initial solutions σ_k ($k = 1, \ldots, P_{size}$) are randomly generated as the multi-start points. For each iteration, the next solutions σ'_k are chosen from their corresponding neighborhood, $N(\sigma_k)$, ($k = 1, \ldots, P_{size}$).

MSA (I_{iter}, T_0, T_F, $N_{non-improving}$, α and P_{size})

Step 1: Generate the initial solutions σ_k by NEH heuristic, $k = 1, 3, \ldots, P_{size}$-1;
    Generate the initial solutions σ_k randomly, $k = 2, 4, \ldots, P_{size}$;

Step 2: Let $T = T_0$; R=0; N=0; σ_{best} = the best σ_k among the P_{size} solutions;
    $TFT_{best} = obj(\sigma_{best})$;

Step 3: N=N+1;

Step 4: For $k = 1$ to P_{size} {
    Step 4.1 Generate a solution σ'_k based on σ_k;
    Step 4.2 If $\Delta_i = obj(\sigma'_k) - obj(\sigma_k) \leq 0$ {Let $\sigma_k = \sigma'_k$;}
        Else {
            Generate r~U(0,1);
            If $r < Exp(\Delta_k / T)$ { Let $\sigma_k = \sigma'_k$;}
            Else {Discard σ'_k;}
        }
    Step 4.3 If $obj(\sigma_k) < TFT_{best}$ {
        $\sigma_{best} = \sigma_k$; $TFT_{best} = obj(\sigma_k)$; R=0;
        Let $\sigma_j = \sigma_{best}$ ($j = 1, \ldots, P_{size}$)
        }
}

Step 5: If $N = I_{iter}$ {
    $T = T \times \alpha$; $N = 0$;
    Perform local search for each σ_k;
    If $obj(\sigma_k) < TFT_{best}$ {
        $\sigma_{best} = \sigma_k$; $TFT_{best} = obj(\sigma_k)$; R=0;
        Let $\sigma_j = \sigma_{best}$ ($j = 1, \ldots, P_{size}$)
    }
    Else {R= R +1;}
}
Else {Go to Step 3;}

Step 6: If $T < T_f$ or $R = N_{non-improving}$ {Terminate the MSA procedure;}
    Else {Go to Step 3;}

FIGURE 1. The pseudo-code of proposed MSA algorithm

Furthermore, let $obj(\sigma_k)$ denote the total flow time of σ_k, and let Δ_i denote the difference between $obj(\sigma_k)$ and $obj(\sigma'_k)$, that is $\Delta_k = obj(\sigma_k) - obj(\sigma'_k)$. The probability of replacing σ_k with σ'_k, given that $\Delta_k > 0$, is $Exp(-\Delta_k/T)$. This step is performed by generating a random number $r \in [0, 1]$ and replacing the solution σ_k with σ'_k if $r < Exp(-\Delta_k/T)$. Meanwhile, if $\Delta_i \leq 0$, the probability of replacing σ_k with σ'_k is 1.

T is decreased after running I_{iter} iterations from the previous reduction in temperature, according to the formula $T \leftarrow \alpha T$, where $0 < \alpha < 1$. Whenever T is decreased, a local search procedure that sequentially performs swap and insertion operations is used to improve the current best solution among all the current solutions σ_k $(k = 1, \ldots, P_{size})$. If T is less than T_F, the algorithm is terminated. If the incumbent solution, σ_{best}, is not improved in $N_{non\text{-}improving}$ successive reductions in temperature, the algorithm is also terminated. If a new σ_{best} solution is obtained, then all the current solutions σ_k $(k = 1, \ldots, P_{size})$ are set to be the same as this new solution, σ_{best}, and the MSA procedure is continued. Following the termination of the MSA procedure, the near-optimal sequence can be derived from σ_{best}.

4. **Experimental Results.** This section describes the computational experiments conducted to evaluate the performance of the proposed MSA algorithm for solving the $F \parallel \sum C_j$ problem. The test problems, parameters selection, and computational results are further detailed in the following subsections.

4.1. **Test problems.** The proposed MSA algorithm was tested on the benchmark-problem set, provided by Taillard [54]. This set was also adopted to evaluate the state-of-the-art meta-heuristics in the literature. In order to conduct a systematic evaluation of the performance of different algorithms, Taillard [54] generated the test instances based on the characteristics of real flowshop systems. The benchmark instances have 12 different sizes, the number of jobs ranges from 20 to 500, and the number of machines varies from 5 to 20. Moreover, for each job i $(i = 1, \ldots, n)$ on each machine j $(j = 1, \ldots, m)$, an integer was generated from the uniform distribution $[1, 99]$ as the processing time $p_{i,j}$. The levels of variations in problem size and job processing times could accommodate a wide spectrum and well-balanced test instances. In order to generate instances that are more challenging, for each problem size, Taillard selected the most difficult ten instances to form the basic problem set. Thus, there are 120 instances in all. The job processing times in each of these instances were randomly generated using a different random seed. The files containing the instances are available to be downloaded on Taillard's web site (URL: http://www.idsia.ch/∼eric) or can be downloaded from the OR-Library web site (URL: http://mscmga.ms.ic.ac.uk/jeb/orlib/&owshopinfo.html).

4.2. **Parameters selection.** The parameters' values influence the effectiveness and efficiency of the MSA algorithm. To determine the best set of parameter values in terms of solution quality and computational efficiency, an extensive set of preliminary experiments were conducted. Based on the results, the following parameters were used in the final computational experiments in this study: $I_{iter} = n \times 4000/P_{size}$, $T_0 = 2.5 \times n$, $T_F = 0.0025 \times n$, $N_{non\text{-}imprving} = 40$ and $\alpha = 0.90$, where P_{size} ranges from 1 to 9, and n is the number of jobs requiring scheduling.

With this parameter setting, in all the experiments, the algorithm terminates when current temperature is less than $0.0025 \times n$ or when the best objective function value was not improved in 40 successive temperature reductions. Since the number of iterations (I_{iter}) is inversely proportion to the value of P_{size}, the number of solutions evaluated for the instances with the same size are almost the same for each P_{size} value, making the comparison on a fair basis. To determine the best values of P_{size} for the experiments, each instance was solved five times (each of which used different random seeds) under each P_{size} value and the best solution out of the five solutions was chosen as the output in the experiment. The average relative percentage deviation ($ARPD$) from the best objective values over the 120 benchmark-problem instance is calculated according to

$$ARPD = \sum_{l=1}^{N} \frac{(Obj_l - BKS_l)/BKS_l}{N} \times 100\%$$

where OBJ_l denotes the objective value of instance l obtained by the algorithm being evaluated, and BKS_l is the best objective value instance l obtained by existing algorithms, including $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$ [27,28], $\mathrm{BEST}_{\mathrm{LWW}}$ [30], M-MMAS [43], PACO [44], $\mathrm{PSO}_{\mathrm{VNS}}$ [45], C-PSO [46], ILS [47], $\mathrm{HGA}_{\mathrm{ZLW}}$ [42], $\mathrm{HGA}_{\mathrm{TL}}$ [41], QDEA [50], EDA and EDA-VNS [49].

Specifically, $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$ denotes three composite heuristics and an insetion-based heuristic, proposed by Liu and Reeves [28] and Woo and Yim [27]. M-MMAS and PACO denotes the two ant-colony-based approaches, proposed by Rajendran and Ziegler [43,44]. $\mathrm{BEST}_{\mathrm{LWW}}$ [30] represents the best objective value obtained by $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$, $\mathrm{PSO}_{\mathrm{VNS}}$, $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$ and three composite heuristics, presented by Li et al. [30]. $\mathrm{PSO}_{\mathrm{VNS}}$ represents the particle swarm optimization approach with variable neighborhood search, developed by Tasgetiren et al. [45]. ILS is the iterated local search approach, proposed by Dong et al. [48]. $\mathrm{HGA}_{\mathrm{ZLW}}$ is the hybrid genetic algorithm, proposed by Zhang et al. [42]. $\mathrm{HGA}_{\mathrm{TL}}$ is another hybrid genetic algorithm, proposed by Tseng and Lin [41]. C-PSO is the combinatorial particle swarm optimization approach, developed by Jarboui et al. [46]. QDEA is the quantum differential evolutionary algorithm, proposed by Zheng and Yamashiro [50]. EDA and EDA-VNS are the estimation-of-distribution-algorithm-based approaches, developed by Jarboui et al. [49].

The above-mentioned approaches represent the effective algorithms currently available for solving the $F \parallel \sum C_j$ problem. Thus, the performance of the proposed MSA algorithm was compared with that of these state-of-the-art algorithms and of the traditional single-start SA, on the same benchmark-problem instances. The $ARPD$ of each P_{size} value for the first 90 benchmark-problem problems (with the number of jobs less than or equal to 100) is shown in Figure 2. As shown in Figure 2, the smallest average $ARPD$ of the proposed MSA algorithm is obtained by setting $P_{size} = 2$. Thus, the solutions of the benchmark-problem instances obtained by the proposed MSA algorithm with $P_{size} = 2$ were used for further comparison.

4.3. **Results and discussion.** The proposed MSA algorithm was implemented using C language and evaluated on a PC with an Pentium 4 (3.2GHz) CPU and 2048 MB memory.

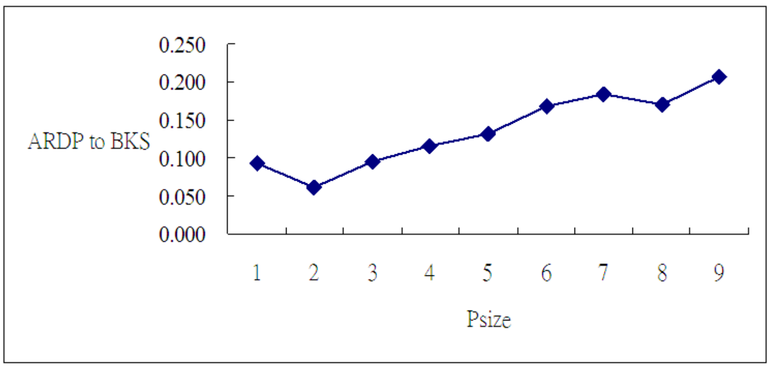

FIGURE 2. The $ARDP$ of each P_{size} value for the instances with the number of jobs n is no more than 100

Each of the test instances was solved five times by the MSA and the other algorithms mentioned in Section 4.2 using different random seeds. For each test instance and each evaluated algorithm, the best solution out of the five runs is shown in Tables 1-4 for different combination of number of jobs and machines. Note that the original papers did not report the results of evaluating M-MMAS, PACO, PSO_{VNS}, ILS, DE, HGA_{TL}, C-PSO, EDA and EDA-VNS on the larger instances with more than 100 jobs (i.e., they were

TABLE 1. Results on the instances with $n = 20$, and $m = 5$, 10 and 20

n	m	BKS	$BEST_{LR\&WY}$	$BEST_{Lww}$	M-MMAS	PACO	PSO_{VNS}	ILS	HGA_{ZLW}	DE	HGA_{TL}	C-PSO	VNS	EDA-VNS	SA	MSA
20	5	14033	14226	14041	14056	14056	**14033***	**14033**	**14033**	**14033**	**14033**	**14033**	**14033**	**14033**	**14033**	**14033**
		15151	15446	**15151**	**15151**	15214	**15151**	**15151**	**15151**	**15151**	**15151**	**15151**	**15151**	**15151**	**15151**	**15151**
		13301	13676	13386	13416	13403	**13301**	**13301**	**13301**	**13313**	**13301**	**13301**	**13301**	**13301**	**13301**	**13301**
		15447	15750	15486	15486	15505	**15447**	**15447**	**15447**	**15447**	**15447**	**15447**	**15447**	**15447**	**15447**	**15447**
		13529	13633	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**	**13529**
		13123	13265	**13123**	13139	**13123**	**13123**	**13123**	**13123**	**13123**	**13123**	**13123**	**13123**	**13123**	**13123**	**13123**
		13548	13774	13559	13559	13674	**13548**	**13548**	**13548**	13557	**13548**	**13548**	**13548**	**13548**	**13548**	**13548**
		13948	13968	13968	13968	14042	**13948**	**13948**	**13948**	**13948**	**13948**	**13948**	**13948**	**13948**	**13948**	**13948**
		14295	14456	14317	14317	14383	**14295**	**14295**	**14295**	**14295**	**14295**	**14295**	**14295**	**14295**	**14295**	**14295**
		12943	13036	12968	12968	13021	**12943**	**12943**	**12943**	**12943**	**12943**	**12943**	**12943**	**12943**	**12943**	**12943**
20	10	20911	21207	20958	20980	20958	**20911**	**20911**	**20911**	**20911**	**20911**	**20911**	**20911**	**20911**	**20911**	**20911**
		22440	22927	22440	22440	22591	**22440**	**22440**	**22440**	**22440**	**22440**	**22440**	**22440**	**22440**	**22440**	**22440**
		19833	20072	19833	19833	19968	**19833**	**19833**	**19833**	**19833**	**19833**	**19833**	**19833**	**19833**	**19833**	**19833**
		18710	18857	18724	18724	18769	**18710**	**18710**	**18710**	**18710**	**18710**	**18710**	**18724**	**18710**	18717	**18710**
		18641	18939	18644	18644	18749	**18641**	**18641**	**18641**	**18641**	**18641**	**18641**	**18641**	**18641**	**18641**	**18641**
		19245	19608	**19245**	**19245**	**19245**	19249	**19245**	**19245**	**19245**	**19245**	**19245**	19249	**19245**	**19245**	**19245**
		18363	18723	18376	18376	18377	**18363**	**18363**	**18363**	**18363**	**18363**	**18363**	**18363**	**18363**	**18363**	**18363**
		20241	20504	**20241**	**20241**	20377	**20241**	**20241**	**20241**	**20241**	**20241**	**20241**	**20241**	**20241**	**20241**	**20241**
		20330	20561	**20330**	**20330**	20330	**20330**	**20330**	**20330**	**20330**	**20330**	**20330**	**20330**	**20330**	**20330**	**20330**
		21320	21506	**21320**	**21320**	21323	**21320**	**21320**	**21320**	**21320**	**21320**	**21320**	**21320**	**21320**	**21320**	**21320**
20	20	33623	34119	**33623**	**33623**	**33623**	34975	**33623**	**33623**	**33623**	**33623**	**33623**	**33623**	**33623**	**33623**	**33623**
		31587	31918	31597	31604	31597	32659	**31587**	**31587**	**31587**	**31587**	**31587**	**31587**	**31587**	**31587**	**31587**
		33920	34552	33920	33920	34130	34594	**33920**	**33920**	**33920**	**33920**	**33920**	**33920**	**33920**	**33920**	**33920**
		31661	32159	31698	31698	31753	32716	**31661**	**31661**	**31661**	**31661**	**31661**	**31661**	**31661**	**31661**	**31661**
		34557	34990	34593	34593	34642	35455	**34557**	**34557**	**34557**	**34557**	**34557**	**34557**	**34557**	**34557**	**34557**
		32564	32734	32594	32637	32594	33530	**32564**	**32564**	**32564**	**32564**	**32564**	**32564**	**32564**	**32564**	**32564**
		32922	33449	32922	33038	32922	33733	**32922**	**32922**	**32922**	**32922**	**32922**	**32922**	**32922**	**32922**	**32922**
		32412	32611	32444	32444	32533	33008	**32412**	**32412**	**32412**	**32412**	**32412**	**32412**	**32412**	**32412**	**32412**
		33600	34084	33623	33625	33623	34446	**33600**	**33600**	**33600**	**33600**	**33600**	**33600**	**33600**	**33600**	**33600**
		32262	32537	32317	32317	32317	33281	**32262**	**32262**	**32262**	**32262**	**32262**	**32262**	**32262**	**32262**	**32262**

*Bold font means the best solution among various algorithms.

TABLE 2. Results on the instances with $n = 50$, and $m = 5$, 10 and 20

n	m	BKS	$BEST_{LR\&WY}$	$BEST_{Lww}$	M-MMAS	PACO	PSO_{VNS}	ILS	HGA_{ZLW}	DE	HGA_{TL}	C-PSO	VNS	EDA-VNS	SA	MSA
50	5	64817	65663	65546	65768	65546	65058	64980	64956	65058	64853	64838	64841	**64817***	64901	64937
		68066	68664	68485	68828	68485	68298	68316	68298	68396	68173	68223	**68066**	68151	68179	68132
		63240	64378	64149	64166	64149	63577	63653	63513	63652	63367	63436	**63240**	63258	63337	63378
		68281	69795	69113	69113	69359	68571	68751	68571	68701	**68281**	68590	68459	68287	68416	68339
		69478	70841	70154	70331	70154	69698	69701	69562	69696	69551	69584	69478	69495	**69462**	69541
		66882	68084	67563	67563	67664	67138	67167	67111	67197	67013	67062	66997	66882	66987	**66865**
		66274	67186	66600	67014	66600	66338	66576	66318	66334	66294	66375	66335	**66274**	66420	66345
		64418	65582	64863	64863	65123	64638	64754	**64418**	64601	64560	64531	64424	64429	64423	64547
		62981	63968	63483	63735	63483	63227	63509	64157	63217	63029	63157	**62981**	63055	63049	63057
		68843	70273	69831	70256	69831	69195	69476	69141	69229	69037	69121	**68843**	68960	69017	69076
50	10	87238	88770	88770	89599	88942	88031	87979	87593	88028	87599	87672	87299	**87238**	87532	87494
		83001	85600	83612	83612	84549	83624	83531	83517	83525	83001	83199	83187	83116	82983	**82907**
		80132	82456	81338	81655	81338	80609	80408	80316	80388	80224	80311	**80132**	80249	80234	80268
		86725	89356	87924	87924	88014	87053	86749	86953	86972	86787	87037	86725	86850	86797	**86688**
		86626	88482	87801	88826	87801	87263	**86508**	86905	87194	86646	86819	86626	86674	86809	86806
		86735	89602	88269	88394	88269	87255	87194	87008	87024	86826	**86735**	86785	86782	86879	86900
		88996	91422	89984	90686	89984	89259	89359	89155	89279	88996	89014	89192	89243	89220	**88909**
		86860	89549	88281	88595	88281	87192	87445	87192	87790	**86860**	87336	87123	87025	87287	87385
		85688	88230	86975	86975	86995	86102	86127	86086	86125	85841	85964	85806	85688	86079	**85628**
		88149	90787	89238	89470	89238	88631	88588	88363	88937	88293	88149	88319	88338	88255	**88103**
50	20	125831	129095	126962	127348	126962	128622	126338	125950	126898	126073	126126	**125831**	126129	126051	125878
		119247	122094	120728	121208	121098	122173	119558	119442	119373	119300	119936	**119247**	**119247**	119345	119441
		116696	121379	117524	118051	117524	118719	117258	117315	117248	116856	117210	116950	**116696**	116947	116888
		120834	124083	122807	123061	122807	123028	121158	121114	122003	121028	121540	120978	**120834**	120997	121074
		118457	122158	119221	119920	119221	121202	118534	118975	118950	118736	118783	118833	**118457**	118700	118589
		120820	124061	122262	122369	122262	123217	121100	120955	121507	121066	120914	120851	**120820**	121014	121008
		123271	126363	125351	125609	125351	125586	123666	123740	123607	123580	123756	123526	**123271**	123772	123672
		122770	126317	124374	124543	124374	125714	123428	122940	123451	**122770**	122900	122962	122820	122862	122860
		121872	125318	123646	124059	123646	124932	122623	122123	122398	121872	122281	122315	**121872**	122138	122145
		124354	127823	125767	126582	125767	126311	124647	124936	125149	124354	124529	124537	124486	124253	**124215**

*Bold font means the best solution among various algorithms.

TABLE 3. Results on the instances with $n = 100$, and $m = 5$, 10 and 20

n	m	BKS	$BEST_{LR\&WY}$	$BEST_{Lww}$	M-MMAS	PACO	PSO_{VNS}	ILS	HGA_{ZLW}	DE	HGA_{TL}	C-PSO	VNS	EDA-VNS	SA	MSA
100	5	254250	256789	256066	257025	257886	254762	256061	254762	256051	254619	255520	**254250***	254859	254444	254279
		243227	245609	244885	246612	246326	245315	245114	243850	245173	243817	244511	243365	**243227**	243542	243498
		238580	241013	239536	240537	241271	239777	239409	239173	240183	239075	239843	**238580**	238809	238780	238757
		228291	231365	230376	230480	230376	228872	229386	228705	229794	228291	229481	228542	228520	**228278**	228544
		241255	244016	243013	243013	243457	242245	243004	241432	242166	241255	242229	241397	241528	241442	**241149**
		233161	235793	235793	236225	236409	234082	234282	233698	235302	233583	234394	**233161**	233721	233685	233544
		241213	243741	243741	243935	243854	242122	242306	241650	242018	241458	242779	241213	241270	**241138**	241550
		231865	235171	234164	234813	234579	232755	233335	232734	234224	232283	232889	231865	232091	232235	**231850**
		249038	251291	251291	252384	253325	249959	250345	249920	249838	249269	250294	249038	249166	**248656**	249038
		243647	247491	246261	246261	246750	244275	245828	244131	245139	243879	244903	243902	**243647**	243923	244094
100	10	300507	306375	305004	305004	305376	303142	301453	300507	303028	300634	302971	301413	301001	300837	**299799**
		275601	280928	278921	279094	278921	277109	277819	277109	278160	277209	277408	**275601**	276077	276069	275675
		288943	296927	294239	297177	294239	292465	292250	290468	292243	290198	291669	**288943**	289961	289799	289661
		303443	309607	306739	306994	306739	304676	305245	303443	305722	303669	305663	303597	303555	304106	**302313**
		286647	291731	289676	290493	289676	288242	286824	286647	288208	287136	287761	286911	287201	286559	**286514**
		271956	276751	275420	276449	275932	272790	273728	272764	273376	273172	274152	**271956**	272052	272099	272443
		281090	288199	284846	286545	284846	282440	281692	282373	282231	281306	282602	**281090**	282115	281938	281328
		293067	296130	296130	297454	297400	293572	293400	**293067**	293432	293628	295430	293460	293842	293127	293671
		303893	312175	307043	309664	307043	305605	304912	304330	304622	304276	305819	303893	304479	304530	**303576**
		293465	298901	296869	296869	297182	295173	294775	293932	294951	**293465**	296425	293584	293492	293643	293551
100	20	368641	383865	372630	373756	372630	374351	370035	369652	371146	370603	372480	**368641**	369391	369015	368933
		374838	383976	381124	383614	381124	379792	376184	376067	377418	375982	376476	376108	**374838**	376116	375707
		372423	383779	379135	380112	379135	378174	373102	373199	375280	373554	375733	373092	**372423**	372566	373542
		374832	384854	380201	380201	380765	380899	376972	374832	376456	376236	379273	375534	375382	**374655**	375447
		371268	383802	377268	377268	379064	376187	373098	**371268**	373118	373524	374416	371461	372479	371550	371820
		374705	387962	380464	381510	380464	379248	374987	375463	376466	374705	378380	375348	375550	375093	**374596**
		376353	384839	381963	381963	382015	380912	377145	376353	377139	376998	379395	376366	377143	376503	**376336**
		387189	397264	393075	393617	393075	392315	388118	**387189**	388127	388058	390587	388028	387481	388451	387802
		377729	387831	380359	385478	380359	382212	378968	378657	378676	378474	380762	377729	378057	377445	**377201**
		381623	394861	387948	387948	388060	386013	382721	382004	382139	383283	384734	381623	382623	381779	**381044**

*Bold font means the best solution among various algorithms.

TABLE 4. Results on the instances with $n = 200$ and 500, and $m = 10$ and 20

n	m	BKS	$BEST_{LR\&WY}$	$BEST_{Lww}$	M-MMAS	PACO	PSO_{VNS}	ILS	HGA_{ZLW}	DE	HGA_{TL}	C-PSO	VNS	EDA-VNS	SA	MSA
200	10	1050564	1063976	1057244	N/A	N/A	N/A	N/A	**1050564***	1061356	N/A	N/A	N/A	N/A	1052491	1052058
		1040604	1049076	1049076	N/A	N/A	N/A	N/A	**1040604**	1045405	N/A	N/A	N/A	N/A	1042851	1042806
		1050785	1059765	1059765	N/A	N/A	N/A	N/A	**1050785**	1055712	N/A	N/A	N/A	N/A	1051427	1051625
		1038885	1051335	1042435	N/A	N/A	N/A	N/A	1038885	1049792	N/A	N/A	N/A	N/A	**1035920**	1037395
		1041510	1055823	1051089	N/A	N/A	N/A	N/A	1041510	1052792	N/A	N/A	N/A	N/A	1040586	**1040209**
		1013421	1023054	1018049	N/A	N/A	N/A	N/A	1013421	1021684	N/A	N/A	N/A	N/A	1011654	**1011018**
		1061285	1071471	1068762	N/A	N/A	N/A	N/A	1061285	1070612	N/A	N/A	N/A	N/A	1059562	**1059194**
		1049007	1054500	1051772	N/A	N/A	N/A	N/A	**1049007**	1052870	N/A	N/A	N/A	N/A	1051724	1050631
		1026991	1045183	1036608	N/A	N/A	N/A	N/A	**1026991**	1041209	N/A	N/A	N/A	N/A	1028996	1029734
		1038016	1044888	1044888	N/A	N/A	N/A	N/A	1038016	1042602	N/A	N/A	N/A	N/A	**1034819**	1035628
200	20	1235238	1247352	1243608	N/A	N/A	N/A	N/A	1235238	1244599	N/A	N/A	N/A	N/A	**1231607**	1232832
		1254529	1271603	1269391	N/A	N/A	N/A	N/A	1254529	1260944	N/A	N/A	N/A	N/A	1249520	**1249029**
		1274798	1297768	1292511	N/A	N/A	N/A	N/A	1274798	1282514	N/A	N/A	N/A	N/A	1272495	**1271136**
		1245656	1272199	1265740	N/A	N/A	N/A	N/A	1245656	1262987	N/A	N/A	N/A	N/A	**1243433**	1244300
		1236246	1255708	1245394	N/A	N/A	N/A	N/A	1236246	1249789	N/A	N/A	N/A	N/A	**1229370**	1230550
		1237754	1251817	1248909	N/A	N/A	N/A	N/A	1237754	1245824	N/A	N/A	N/A	N/A	**1232910**	1233043
		1248821	1275658	1265967	N/A	N/A	N/A	N/A	1248821	1264431	N/A	N/A	N/A	N/A	1249017	**1246752**
		1249644	1273142	1268583	N/A	N/A	N/A	N/A	1249644	1269343	N/A	N/A	N/A	N/A	1249430	**1248323**
		1237428	1259311	1249406	N/A	N/A	N/A	N/A	1237428	1251134	N/A	N/A	N/A	N/A	1240528	**1236393**
		1253075	1273354	1269812	N/A	N/A	N/A	N/A	**1253075**	1263586	N/A	N/A	N/A	N/A	1256725	1256193
500	10	6723143	6746310	6732747	N/A	N/A	N/A	N/A	6723143	6739564	N/A	N/A	N/A	N/A	**6692467**	6696073
		6770735	6868018	6858362	N/A	N/A	N/A	N/A	6844840	**6770735**	N/A	N/A	N/A	N/A	6821192	6814417
		6772110	6793698	6778236	N/A	N/A	N/A	N/A	6772110	6784318	N/A	N/A	N/A	N/A	6755556	**6736622**
		6803343	6812857	6812857	N/A	N/A	N/A	N/A	6809460	6803343	N/A	N/A	N/A	N/A	**6785213**	6785246
		6742209	6767964	6760655	N/A	N/A	N/A	N/A	6742209	6753596	N/A	N/A	N/A	N/A	6742202	**6736622**
		6729388	6774423	6736190	N/A	N/A	N/A	N/A	**6729388**	6765837	N/A	N/A	N/A	N/A	6756331	6750667
		6706950	6739792	6713386	N/A	N/A	N/A	N/A	**6706950**	6724431	N/A	N/A	N/A	N/A	6711769	6708140
		6795769	6821619	6801698	N/A	N/A	N/A	N/A	6795769	6809832	N/A	N/A	N/A	N/A	6788717	**6776403**
		6736573	6753839	6746074	N/A	N/A	N/A	N/A	6736573	6750574	N/A	N/A	N/A	N/A	**6711373**	6712900
		6764295	6778403	6771808	N/A	N/A	N/A	N/A	6764295	6769981	N/A	N/A	N/A	N/A	6759759	**6759547**

*Bold font means the best solution among various algorithms.

tested on only 90 instances in Taillard's benchmark-problem set), so the corresponding results are not shown in Table 4.

As shown in Tables 1-4, out of the 120 instances, the proposed MSA algorithm outperforms the other algorithms on 38 instances, and obtains the same objective values on 31 instances as the other algorithms. These results reveal that the proposed MSA algorithm could obtain better solutions than the state-of-the-art algorithms. If $P_{size} = 1$, the proposed MSA algorithm reduces to the traditional single-start SA algorithm. In this case, the proposed MSA algorithm obtains a better solution on 28 instances, and the same objective value on 29 instances. The results indicate that the proposed MSA algorithm is relatively more effective in minimizing total flow time than the traditional single-start SA.

For the instances with n less than or equal to 100 (i.e., the first 90 instances), the 14 approaches (i.e., $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$, $\mathrm{BEST}_{\mathrm{LWW}}$, M-MMAS, PACO, $\mathrm{PSO}_{\mathrm{VNS}}$, ILS, $\mathrm{HGA}_{\mathrm{ZLW}}$, DE, $\mathrm{HGA}_{\mathrm{TL}}$, C-PSO, VNS, EDA-VNS, SA, MSA) obtain the best solution on 0 (0/90 = 0.00%), 12 (12/90 = 13.33%), 10 (10/90 = 11.11%), 6 (6/90 = 6.67%), 19 (19/90 = 21.11%), 31 (31/90 = 34.44%), 34 (34/90 = 37.77%), 28 (28/90 = 31.11%), 35 (35/90 = 38.89%), 31 (31/90 = 34.44%), 43 (43/90 = 47.78%), 44 (44/90 = 48.89%), 34 (34/90 = 37.78%) and 47 (47/90 = 52.22%) instances, respectively. By far, the MSA obtains the most number of best solutions among all of the approaches under comparison. For the larger instances (n is more than 100), the six approaches (i.e., $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$, $\mathrm{BEST}_{\mathrm{LWW}}$, $\mathrm{HGA}_{\mathrm{ZLW}}$, DE, SA, MSA) obtains the best solution on 0 (0/30 = 0.00%), 0 (0/30 = 0.00%), 8 (8/30 = 26.67%), 1 (1/30 = 3.33%), 9 (9/30 = 30.00%), and 12 (12/30 = 40.00%) instances, respectively. The MSA also obtains the most number of best solutions among the six approaches under evaluation.

The *ARPD* over 10 different benchmark-problem instances for each problem size is reported in Table 5. For the instances with the number of jobs less than or equal to 100, the total *ARPD* is 0.061% for the proposed MSA algorithm, whereas the total *APRD* for $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$, $\mathrm{BEST}_{\mathrm{LWW}}$, M-MMAS, PACO, $\mathrm{PSO}_{\mathrm{VNS}}$, ILS, $\mathrm{HGA}_{\mathrm{ZLW}}$, DE, $\mathrm{HGA}_{\mathrm{TL}}$, C-PSO, VNS, EDA-VNS and SA are 1.953%, 0.823%, 1.052%, 0.990%, 0.910%, 0.302%, 0.182%, 0.361%, 0.114%, 0.326%, 0.063%, 0.072% and 0.093%, respectively. The total *ARPD* for all the 120 instances is 0.020% for the proposed MSA algorithm, whereas, for $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$, $\mathrm{BEST}_{\mathrm{LWW}}$, $\mathrm{HGA}_{\mathrm{ZLW}}$, DE, and SA, the values are 1.727%, 0.790%, 1.146%, 0.435% and 0.054%, respectively. According to the results, the performance of the proposed MSA algorithm is better than that of the state-of-art algorithms as well as the traditional single-start SA on solving the $F \parallel \sum C_j$ problem.

Table 6 lists the average computational time (CPU time in seconds) over 10 different benchmark-problem instances for each problem size. The heuristics of $\mathrm{BEST}_{\mathrm{LR\&\ WY}}$ were

Table 5. Performance comparison on Taillard's benchmark-problems (RPD)

Prob.	$\mathrm{BEST}_{\mathrm{LR\&WY}}$	$\mathrm{BEST}_{\mathrm{Lww}}$	M-MMAS	PACO	$\mathrm{PSO}_{\mathrm{VNS}}$	ILS	$\mathrm{HGA}_{\mathrm{ZLW}}$	DE	$\mathrm{HGA}_{\mathrm{TL}}$	C-PSO	VNS	EDA-VNS	SA	MSA (P_{size}=2)
20|5	1.361	0.152	0.197	0.454	0.000	0.000	0.000	0.016	0.000	0.000	0.000	0.000	0.000	0.000
20|10	1.433	0.039	0.049	0.324	0.002	0.000	0.000	0.000	0.000	0.000	0.010	0.000	0.004	0.000
20|20	1.224	0.068	0.119	0.189	2.828	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
50|5	1.679	0.980	1.257	1.072	0.371	0.544	0.424	0.422	0.132	0.246	0.057	0.049	0.137	0.142
50|10	2.819	1.413	1.821	1.558	0.581	0.448	0.356	0.606	0.121	0.257	0.134	0.136	0.236	0.123
50|20	2.850	1.191	1.530	1.222	2.088	0.342	0.276	0.528	0.122	0.317	0.155	0.038	0.159	0.134
100|5	1.157	0.858	1.112	1.231	0.401	0.603	0.230	0.642	0.125	0.512	0.033	0.096	0.067	0.074
100|10	2.037	1.253	1.627	1.338	0.571	0.468	0.213	0.600	0.215	0.733	0.062	0.179	0.141	0.002
100|20	3.020	1.451	1.752	1.519	1.344	0.312	0.136	0.437	0.315	0.868	0.115	0.154	0.094	0.076
200|10	1.039	0.659	–	–	–	–	0.000	0.798	–	–	–	–	-0.010	-0.008
200|20	1.640	1.170	–	–	–	–	0.000	0.979	–	–	–	–	-0.146	-0.198
500|10	0.463	0.248	–	–	–	–	0.118	0.189	–	–	–	–	-0.029	-0.100
Avg. for $n \leq 100$	1.953	0.823	1.052	0.990	0.910	0.302	0.182	0.361	0.114	0.326	0.063	0.072	0.093	0.061
Avg. for $n \leq 500$	1.727	0.790	–	–	–	–	0.146	0.435	–	–	–	–	0.054	0.020

TABLE 6. Performance comparison on Taillard's benchmark-problems (time in seconds)

Prob.	[1]BEST$_{LR\&WY}$	[1]BEST$_{LWW}$	[2]M-MMAS	[2]PACO	PSO$_{VNS}$	ILS	HGA$_{ZLW}$	[1]DE	HGA$_{TL}$	C-PSO	VNS	EDA-VNS	SA	MSA (P_{size}=2)
20|5	N/A	N/A	<3600	<3600	3.18	0.25	2.18	N/A	1.11	2.13	0.13	0.30	3.74	3.74
20|10	N/A	N/A	<3600	<3600	7.21	0.53	4.14	N/A	3.45	9.21	0.30	1.29	6.81	6.80
20|20	N/A	N/A	<3600	<3600	11.93	1.00	7.57	N/A	6.83	9.88	0.74	1.51	11.42	11.55
50|5	N/A	N/A	<3600	<3600	41.71	3.45	40.85	N/A	6.55	54.80	31.98	57.22	27.28	27.95
50|10	N/A	N/A	<3600	<3600	74.49	7.43	111.74	N/A	36.54	132.12	56.18	105.45	54.20	54.57
50|20	N/A	N/A	<3600	<3600	143.32	14.52	189.98	N/A	106.51	277.98	142.08	240.96	100.11	100.60
100|5	N/A	N/A	<3600	<3600	222.28	25.32	461.33	N/A	16.540	121.63	174.26	124.55	99.55	102.87
100|10	N/A	N/A	<3600	<3600	407.88	55.98	826.76	N/A	88.605	274.69	324.37	266.02	202.86	208.95
100|20	N/A	N/A	<3600	<3600	824.41	111.22	2014.96	N/A	299.92	268.80	644.98	570.27	407.52	422.53
200|10	N/A	N/A	N/A	N/A	N/A	N/A	8515.01	N/A	N/A	N/A	N/A	N/A	721.86	762.49
200|20	N/A	N/A	N/A	N/A	N/A	N/A	17849.58	N/A	N/A	N/A	N/A	N/A	1462.45	1536.80
500|10	N/A	N/A	N/A	N/A	N/A	N/A	50000.00	N/A	N/A	N/A	N/A	N/A	8863.99	9261.40
Avg. for $n \leq 100$	N/A	N/A	N/A	N/A	192.93	24.41	406.61	N/A	62.89	127.92	152.78	151.95	101.50	104.39
Avg. for $n \leq 500$	N/A	N/A	N/A	N/A	N/A	N/A	6668.68	N/A	N/A	N/A	N/A	N/A	996.82	1041.69

1: Executing time is not provided in the paper.
2: Run on a PC that has a Pentium III 800 MHz processor and the computing time for each problem is less than one hour. No detailed time information is provided in their original paper.

run on a PC with a Pentium 200 CPU and a CRAY Y/MP system. All the heuristics of BEST$_{LWW}$ were implemented using Visual Basic 6.0 and evaluated on an IBM PC with a 2.0GHz CPU and 256MB RAM. The M-MMAS and PACO were coded using FORTRAN and run on a PC with a Pentium III 800MHz CPU. The only information, regarding computational time, provided in the original papers of M-MMAS and PACO is that every instance can be solved in less than one hour. The PSO$_{VNS}$ algorithms were coded using C and run on a PC with a Pentium IV 2.6GHz CPU and 256MB memory. ILS was implemented using C++, and run on a PC with an AMD Sempron 3200+ (1.8GHz) CPU and 512M memory. The computational time of DE was not provided in the original paper. HGA$_{ZLW}$ was implemented using Java, and run on a PC with a Pentium IV 2.93GHz CPU and 512MB DRAM. HGA$_{TL}$ was implemented using C++, and run on a PC with an AMD K7 1.83GHz CPU and 512MB DRAM. C-PSO was implemented using C++, and run on a PC with a Pentium IV 3.2GHz CPU. VNS and EDA-VNS were implemented using C++, and run on a PC with a Pentium IV 3.2GHz CPU and 1GB Memory. Note that since some of the literature did not report detailed computational times that depend heavily on the testing environment (e.g., hardware, software) and programming skills, it may not be fair to compare directly the computational efficiency of all the algorithms. Moreover, for the smaller instances, the parameter setting in the experiments could result in a large number of iterations for the proposed MSA to obtain the solution quality which can be attained with less number of iterations (and hence less computational times). For example, for the instances with the number of jobs equal to 20, only 0.57, 0.96 and 1.41 seconds are required to converge to the best solution with number of machines equals 5, 10 and 20, respectively.

Indeed, according to the results, the proposed MSA outperforms the other algorithms in terms of solution quality (i.e., effectiveness). In addition to the outstanding computational performance on the smaller instances, for the large instances with 500 jobs and 10 machines, the proposed MSA algorithm still obtains better solutions than the other algorithm, using reasonable computational expenses. Judging from the experimental results, it is clear that the proposed MSA algorithm has made a step towards establishing an effective and efficient approach for solving the $F \| \sum C_j$ problem.

To make a more rigorous comparison, a set of one-sided paired-samples t-tests, with

Table 7. Results of the paired-samples t-tests with respect to RPD

	MSA vs. $\text{BEST}_{\text{LR\&WY}}$	MSA vs. BEST_{Lww}	MSA vs. M-MMAS	MSA vs. PACO	MSA vs. PSO_{VNS}	MSA vs. ILS	MSA vs. HGA	MSA vs. DE	MSA vs. HGA_{TL}	MSA vs. CPSO	MSA vs. VNS	MSA vs. EDA-VNS	MSA vs. SA
Paired difference (*RPD*)	-1.706	-0.770	-0.991	-0.928	-0.849	-0.241	-0.126	-0.414	-0.053	-0.2648	-0.176	-0.011	-0.024
t-Value	-21.746	-15.421	-13.120	-16.297	-8.292	-8.257	-5.623	-10.797	-3.074	-7.583	-0.114	-0.646	-2.931
Degree of freedom	119	119	89	89	89	89	119	119	89	89	89	89	119
P-Value	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0. 0000	0.0000	0.0014	0.000	0.4546	0.2599	0.002

respect to the relative percentage deviation ($RPD = (Obj - BKS)/BKS \times 100\%$), were performed to examine the difference in solution quality between the proposed MSA algorithm and $\text{BEST}_{\text{LR\& WY}}$, BEST_{LWW}, M-MMAS, PACO, PSO_{VNS}, ILS, HGA, C-PSO and SA. The test results are listed in Table 7. At confidence level $\alpha = 0.05$, Table 7 shows that the proposed MSA algorithm significantly outperforms all the other approaches. Besides, the test results also show that the performance of the single-start SA algorithm can be significantly improved by incorporating the multi-start hill climbing strategy. Although the MSA only slightly outperforms the VNS and EDA-VNS, the MSA spends less computational time and obtains the best solution in more instances than VNS and EDA-VNS.

5. Conclusions and Future Research.

The MSA heuristic which encompasses the main properties of the SA (e.g., effective convergence, efficient use of memory, and easy implementation) and of the multi-start hill climbing strategies (e.g., sufficient diversification, excellent capability of escaping local optima, and efficient sampling in the solution space) is proposed for solving the $F \| \sum C_j$ problem. In light of the comparison of the computational results with the best known solutions on a wide range of benchmark-problem instances, the proposed MSA algorithm is relatively more effective in minimizing total flow time than the existing algorithms and the traditional single-start SA. Given that the $F \| \sum C_j$ problem is an extremely challenging *NP*-hard problem, the proposed MSA algorithm contributes significantly to the research of the optimization techniques for solving this problem.

There are several possible research directions based on this research. One of them is to develop other efficient and effective meta-heuristics for solving this problem. Another interesting research direction is to apply the proposed MSA algorithm to solve other complex scheduling problems, such as hybrid flowshop scheduling problems. Finally, the proposed MSA algorithm can be used to solve this problem with other performance criteria or with multiple criteria.

Acknowledgements. This research was partially supported by the National Science Council of Taiwan, under Contract Nos. NSC99-2628-E-027-003 and NSC99-2410-H-182-023-MY2.

REFERENCES

[1] K. C. Ying and S. W. Lin, Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems, *International Journal of Advanced Manufacturing Technology*, vol.33, no.7-8, pp.793-802, 2007.

[2] Y. Li, Y. Yang, L. Zhou and R. Zhu, Observations on using problem-specific genetic algorithm for multiprocessor real-time task scheduling, *International Journal of Innovative Computing, Information and Control*, vol.5, no.9, pp.2531-2540, 2009.

[3] S. W. Lin and Y. C. Ying, Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems, *International Journal of Production Research*, vol.47, no.5, pp.1411-1424, 2009.
[4] K. C. Ying, Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic, *International Journal of Advanced Manufacturing Technology*, vol.38, no.3-4, pp.348-354, 2008.
[5] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, vol.1, no.1, pp.61-68, 1954.
[6] J. N. D. Gupta and Jr. E. F. Stafford, Flowshop scheduling research after five decades, *European Journal of Operational Research*, vol.169, no.3, pp.699-711, 2006.
[7] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. R. Kan, Optimization and approximation in deterministic sequencing and scheduling, *Annals of Discrete Mathematics*, vol.5, no.2, pp.287-326, 1979.
[8] J. M. Framinan and R. Leisten, An efficient constructive heuristic for flowtime minimisation in permutation flow shops, *OMEGA, International Journal of Management Science*, vol.31, no.4, pp.311-317, 2003.
[9] M. R. Garey, D. S. Johnson and R. Sethi, The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research*, vol.1, no.2, pp.117-129, 1976.
[10] E. Ignall and L. Schrage, Application of the branch and bound technique to some flowshop scheduling problem, *Operations Research*, vol.13, no.3, pp.400-412, 1965.
[11] S. P. Bansal, Minimizing the sum of completion times of n-jobs over m-machines in a flowshop – A branch and bound approach, *AIIE Transactions*, vol.9, no.3, pp.306-311, 1977.
[12] F. D. Croce, V. Narayan and R. Tadei, The two-machine total completion time flow shop problem, *European Journal of Operational Research*, vol.90, no.2, pp.227-237, 1996.
[13] F. D. Croce, M. Ghirardi and R. Tadei, An improved branch-and-bound algorithm for the two machine total completion time flow shop problem, *European Journal of Operational Research*, vol.139, no.2, pp.293-301, 2002.
[14] E. F. Stafford, On the development of a mixed integer linear programming model for the flowshop sequencing problem, *Journal of the Operational Research Society*, vol.39, no.12, pp.1163-1174, 1988.
[15] K. C. Ying and C. J. Liao, An ant colony system for permutation flow-shop sequencing, *Computers & Operations Research*, vol.31, no.5, pp.791-801, 2004.
[16] K. C. Ying, Z. J. Lee and S. W. Lin, Makespan minimization for scheduling unrelated parallel machines with setup times, *Journal of Intelligent Manufacturing*, 2011.
[17] J. N. D. Gupta, Heuristic algorithms for multistage flowshop scheduling problem, *AIIE Transactions*, vol.4, no.1, pp.11-18, 1972.
[18] L. F. Gerlders and N. Sambandam, Four simple heuristics for scheduling a flow-shop, *International Journal of Production Research*, vol.16, no.3, pp.221-231, 1978.
[19] S. Miyazaki, N. Nishiyama and F. Hashimoto, An adjacent pairwise approach to the mean flowtime scheduling problem, *Journal of the Operations Research Society of Japan*, vol.21, no.1, pp.287-299, 1978.
[20] J. C. Ho and Y. L. Chang, A new heuristic for the n-job, m-machine flow-shop problem, *European Journal of Operational Research*, vol.52, no.2, pp.194-202, 1991.
[21] C. Rajendran and D. Chaudhuri, A flowshop scheduling algorithm to minimize total flowtime, *Journal of the Operations Research Society of Japan*, vol.34, no.1, pp.28-46, 1991.
[22] C. Rajendran and D. Chaudhuri, An efficient heuristic approach to the scheduling of jobs in a flowshop, *European Journal of Operational Research*, vol.61, no.3, pp.318-325, 1992.
[23] C. Rajendran, Heuristic algorithm for scheduling in a flowshop to minimize total flowtime, *International Journal of Production Economics*, vol.29, no.1, pp.65-73, 1993.
[24] J. C. Ho, Flowshop sequencing with mean flowtime objective, *European Journal of Operational Research*, vol.81, no.3, pp.571-578, 1995.
[25] C. Wang, C. Chu and J. M. Proth, Heuristic approaches for $n/m/P/\sum C_i$ scheduling problems, *European Journal of Operational Research*, vol.96, no.3, pp.636-644, 1997.
[26] C. Rajendran and H. Ziegler, An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs, *European Journal of Operational Research*, vol.103, no.1, pp.129-138, 1997.
[27] 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.

[28] J. Y. Liu and C. R. Reeves, Constructive and composite heuristic solutions to the $P//\sum C_i$ scheduling problem, *European Journal of Operational Research*, vol.132, no.2, pp.439-452, 2001.
[29] X. P. Li and C. Wu, An efficient constructive heuristic for permutation flow shops to minimize total flowtime, *Chinese Journal of Electronics*, vol.14, no.2, pp.203-208, 2005.
[30] X. Li, Q. Wang and C. Wu, Efficient composite heuristics for total flowtime minimization in permutation flow shops, *OMEGA, International Journal of Management Science*, vol.37, no.1, pp.155-164, 2009.
[31] S. W. Lin, S. Y. Chou and K. C. Ying, A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date, *European Journal Operational Research*, vol.177, no.2, pp.1294-1301, 2007.
[32] K. C. Ying and H. M. Cheng, Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic, *Expert Systems with Applications*, vol.37, no.4, pp.2848-2852, 2010.
[33] A. Allahverdi and T. Aldowaisan, New heuristics to minimize total completion time in m-machine flowshops, *International Journal of Production Economics*, vol.77, no.1, pp.71-83, 2002.
[34] J. M. Framinan, R. Leisten and R. Ruiz-Usano, Comparison of heuristics for flowtime minimisation in permutation flowshops, *Computers & Operations Research*, vol.32, no.5, pp.1237-1254, 2005.
[35] P.-W. Tsai, J.-S. Pan, B.-Y. Liao and S.-C. Chu, Enhanced artificial bee colony optimization, *International Journal of Innovative Computing, Information and Control.* vol.5, no.12(B), pp.5081-5092, 2009.
[36] K. C. Ying, S. W. Lin and Z. J. Lee, Hybrid-directional planning: Improving improvement heuristics for scheduling resource-constrained projects, *International Journal of Advanced Manufacturing Technology*, vol.41, no.3-4, pp.358-366, 2009.
[37] K. C. Ying and C. J. Liao, An ant colony system approach for scheduling problem, *Production Planning & Control*, vol.14, no.1, pp.68-75, 2003.
[38] K. C. Ying, An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks, *Journal of the Operational Research Society*, vol.60, no.12, pp.810-817, 2009.
[39] S. W. Lin, Z. J. Lee, K. C. Ying and C. Y. Lee, Applying hybrid meta-heuristics for capacitated vehicle routing problems, *Expert Systems with Applications*, vol.36, no.2, pp.1505-1512, 2009.
[40] T. Yamada and C. R. Reeves, Solving the C_{sum} permutation flowshop scheduling problem by genetic local search, *Proc. of IEEE International Conference on Evolutionary Computation*, pp.230-234, 1998.
[41] L. Y. Tseng and Y. T. Lin, A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, *European Journal of Operational Research*, vol.198, no.1, pp.84-92, 2009.
[42] Y. Zhang, X. Li and Q. Wang, Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization, *European Journal of Operational Research*, vol.196, no.3, pp.869-876, 2009.
[43] 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.2426-438, 2004.
[44] C. Rajendran and H. Ziegler, Two ant-colony algorithms for minimizing total flowtime in permutation flowshops, *Computers and Industrial Engineering*, vol.48, no.4, pp.789-797, 2005.
[45] M. F. Tasgetiren, Y. C. Liang, M. Sevkli and G. Gencyilmaz, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, *European Journal of Operational Research*, vol.177, no.3, pp.1930-1947, 2007.
[46] B. Jarboui, S. Ibrahim, P. Siarry and A. Rebai, A combinatorial particle swarm optimization for solving permutation flowshop problems, *Computers & Industrial Engineering*, vol.54, no.3, pp.526-538, 2008.
[47] A. El-Bouri, S. Balakrishnan and N. Popplewell, A neural network to enhance local search in the permutation flowshop, *Computers and Industrial Engineering*, vol.49, no.1, pp.182-196, 2005.
[48] X. Dong, H. Huang and P. Chen, An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion, *Computers & Operations Research*, vol.36, no.5, pp.1664-1669, 2009.
[49] B. Jarboui, M. Eddaly and P. Siarry, An estimation of distribution algorithm for the total flowtime in permutation flowshop scheduling problems, *Computers & Operations Research*, vol.36, no.9, pp.2638-2646, 2009.

[50] T. Zheng and M. Yamashiro, Solving flow shop scheduling problems by quantum differential evolutionary algorithm, *International Journal of Advanced Manufacturing Technology*, vol.5, no.5-8, pp.643-662, 2010.

[51] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equation of state calculation by fast computing machine, *Journal of Chemical Physics*, vol.21, no.6, pp.1087-1092, 1953.

[52] R. Martí, Multi-start methods, in *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger (eds.), Dordrecht, The Netherlands, Kluwer Academic Publishers, 2003.

[53] M. Nawaz, E. E. Enscore and I. Ham, A heuristic algorithm for m-machine n-job flow-shop sequencing problems, *OMEGA, International Journal of Management Science*, vol.11, no.1, pp.91-95, 1983.

[54] E. Taillard, Benchmarks for basic scheduling problems, *European Journal of Operational Research*, vol.64, no.2, pp.278-285, 1993.