Flow Shop Scheduling Based on A Novel Cooling Temperature Driven Simulated Annealing Algorithm

Ruey-Maw Chen*
Dept. Of Computer Science and Information Engineering, National Chin-Yi University of Technology
No.57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan, R.O.C.
raymond@ncut.edu.tw

Frode Eika Sandnes
Oslo and Akershus University College of Applied Sciences, P.O. Box 4, St. Olavs plass, N-0130 Oslo, Norway
frodes@hio.no

ABSTRACT: The permutation flow shop problem (PFSP) has been applied to many types of problems. The PFSP is an *NP-hard* permutation sequencing scheduling problem. A local search with simulated annealing scheme involving two phases is proposed in this investigation for solving PFSP. First, for lowering computation complexity, a simple insertion local search is applied to generate the solution of the PFSP. Second, two non-decreasing cooling temperature driven simulated annealing (SA) named steady SA and reheating SA are employed to maintain successive exploration or exploitation in the solution space. The steady SA maintains the same temperature and keeps the same search behavior and thereby allows the neighbors of the worse solutions to be explored, consequently increasing the chances of finding better solutions, while the reheating SA increases the temperature and increases the exploration ability. The most important feature of the proposed method is its simple implementation and low computation time complexity. Experimental results are compared with other state-of-the-art algorithms and reveal that the proposed simple insertion with steady SA (SI-SSA) method is able to efficiently yield the best permutation schedules.

KEYWORDS: Permutation flow shop problem (PFSP), scheduling, local search, simulated annealing (SA), simple insertion with steady simulated annealing (SI-SSA).

1 Introduction

There are many classes of real-world scheduling problems, such as job-shop scheduling, open-shop scheduling, flow shop problem (FSP), task assignment scheduling, real-time scheduling, etc. Generally, scheduling problems involve the allocation of resources (such as machines or processors) to execute a set of activities (such as processes or tasks) satisfying given constraints and optimizing given criteria. Processes or tasks usually have time constraints, such as ready time, execution time, precedence, and deadline. Scheduling algorithms must determine a schedule for a set of processes that satisfies the prerequisite constraints; FSP is one of these and is currently the focus of much research since it can be used for finding near optimal solutions to many real-world optimization problems. FSP can be defined as the problem of assigning a set of independent jobs to run on a set of machines.

* Corresponding author: Ruey-Maw Chen
E-mail: raymond@mail.ncut.edu.tw
Tel.: +886 4 2392 4505; fax: +886 4 2391 7426.

Each job requires a given fixed, non-negative processing time on every machine. In this study, we focus on the permutation flow shop problem (PFSP), a special case of FSP, where the processing order of jobs always is the same on every machine. That is, all jobs follow the same machine order in the shop starting from the first machine and finishing on the last machine. PFSP 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 cyclic scheduling of a no-wait production line, where multiple parts enter and leave the line during a cycle. For example, a multi-degree cyclic scheduling of a permutation flowshop with two robots was investigated by Che and Chu [1]. MPEG-4 macroblock decoding is an application of a permutation flowshop problem for synchronization in a co-processor system while implementing tasks with low turnaround time [2]. Maintenance and production are two functions in various industries which act on the same resources. In another study [3], a complex joint production and maintenance scheduling problem in permutation flowshop was also investigated. The PFSP has been confirmed to be *NP-hard* (Taillard) [4]. Its solution search space comprises *n!* permutations. Hence, finding the optimal solution to PFSP problems with exact algorithms is not feasible in reasonable time. Instead, many approximation algorithms and heuristics have been studied to reveal near optimal solution with less effort, such as the slope-index-based heuristic [5], the CDS heuristic [6], the NEH algorithm [7], etc. However, all these schemes require a substantial amount of computational effort to find solutions that usually are far from optimal. To efficiently obtain high quality solutions, many metaheuristics have been introduced for solving PFSP, in particular genetic algorithms (GA) [8,9,10], simulated annealing (SA) [11], tabu search (TS) [12], ant colony optimization (ACO) [13,14], artificial bee colony (ABC) [15], particle swarm optimization (PSO)-[16-17], etc. Furthermore, metaheuristics are often combined with local search, for example GA mutation [18], ACO with pheromone mutation [19], construction phase in iterated greedy (IG) heuristic [20], and so forth. Other approaches include linear programming relaxation to handle specific job-lists in a bidirectional flow-shop [21] and priority rules embedded in the heuristics for solving the sequence dependent setup time flow-shop problem [22].

Many of these meta-heuristics provide quite acceptable and close to optimal solutions. However, they are often either very complex to implement or suffer from excessive computational complexity. In some cases, the complexity of the algorithms means that independent implementations are unlikely to capture the intended effectiveness and efficiency. Moreover, other approaches exploit PFSP-specific features such that the schemes do not generalize to other flowshop variants. Consequently, in 2007 Ruiz and Stützle proposed the iterated greedy (IG) [20] to provide a simple iterated greedy local search based on the NEHT heuristic [7] to simplify implementation and reduce computational complexity. Still, 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. However, the complexity of NEHT is still $O(n^2m)$ which is time consuming for large instances. After IG phases, a simulated annealing-like acceptance criterion with a constant temperature based on Osman & Potts (1989) is applied. The constant temperature follows the suggestion of Osman & Potts [23] and depends on the particular instances in the OR-Library to be solved.

This study proposes a simple insertion with the steady simulated annealing (SI-SSA) scheme to reduce computational complexity and simplify implementation; as such, this method still generalizes to other flow-shop variants. SI-SSA includes two steps, a simple insertion local search and SA with a novel temperature cooling schedule. The insertion local search is easy to integrate into trajectory meta-heuristics, such as simulated annealing, tabu search, and others. Intrinsically, simulated annealing is a memory-less operation. Additionally, the acceptance criterion of the hill climbing in simulated annealing is modified by adjusting the temperature schedule to reduce the turbulence of the acceptance probability for PFSP based on energy deviation instead of energy difference. Furthermore, a threshold for excluding undesired solutions is also incorporated. The acceptance criterion is the key factor of simulated annealing, which enables it to escape from local minima. As for the cooling in the simulated annealing approach, two non-decreasing temperature control mechanisms are employed to provide an opportunity for continuous exploration or exploitation; they are named reheating SA and steady SA, respectively. The reheating SA increases exploration search ability and the steady SA en-

hances exploitation search ability. Analysis of the search behavior corresponding to these two cooling schemes is also provided.

This article is organized as follows: Section 2 introduces the problem definition. Section 3 introduces simulated annealing. Section 4 presents the details of the SI-SSA scheme for solving PFSP. In Section 5, the effectiveness and efficiency of SI-SSA is demonstrated and the results are analyzed and compared to those of other state-of-the-art schemes. Finally, section 6 makes the conclusions.

2 Problem definition

A well-known scheduling problem with a background in industrial manufacture is the flow shop problem (FSP) [4]. In this study, the permutation flow shop problem (PFSP), in which the jobs' sequence on every machine is the same, is addressed. The PFSP can be defined as follows:

- There are n independent jobs ($N = \{1,...,n\}$) and m independent machines ($M = \{1,...,m\}$) in the production system. All n jobs have to be run on m machines in the same order. Assuming that the set-up times of all jobs are included in the jobs' processing time.
- Each job j ($j \in N$) must be processed on m machines, i.e., each job consists of m sub-jobs, $o_{j,k}$ ($k=1,\ldots, m$). Meanwhile, each job j requires different processing times $p_{j,i}$ on different machines i ($i \in M$). Moreover, any executing job is not preemptive.
- PFSP requires all jobs to be processed with the same processing order $\pi = \{\pi(1),\ldots, \pi(n)\}$ from the first machine to the last machine. The permutation π presents the solution to the PFSP and $\pi(r)$ indicates the r^{th} order processing job.
- Obtaining the minimum makespan is the commonly defined objective of the PFSP. The minimization of the *makespan* is highly affected by the permutation π.

For example, given 3 jobs, there are 3! possible permutations; the objective is to find the permutation π that yields the shortest *makespan*.

3 The Simulated Annealing Algorithm

The well-known simulated annealing (SA) trajectory meta-heuristic was first introduced by Kirkpatrick et al. [24]. An important characteristic of SA is to provide the capability of escaping from local optima. Restated, allowing a neighborhood moving step with a worse solution quality was designed. Movements to worse neighborhoods offer a mechanism of diversification search through hill climbing. However, the worse movement in SA is determined by an acceptance criterion which is relevant to a probability called acceptance probability. A key component in SA is its cooling scheduling, which controls the acceptance probability. Typically, the temperature decreases over time, and thus the acceptance probability decreases over time. Restated, higher acceptance probability in the early stage and lower acceptance probability in the latter stage have been implemented in SA. The rationale of this design is to provide the diversification search ability at the beginning and to give intensification search ability in later stages.

The conventional SA iterates until a termination condition is met. At iteration t, four steps are conducted:

- Neighborhood search: a candidate solution S_t' from the neighborhoods of S_t is generated. There are many local search schemes suggested to generate the candidate solution S_t'.
- Energy evaluation: the energy related to the solution S_t' is evaluated; wherein $E(S_t')$ is denoted as an energy function, and the variance of energy ΔE is also obtained.
- Acceptance criterion: the acceptance criterion is applied; if the variance of energy ΔE is less than zero, the S_t would be replaced by S_t'. Otherwise, S_t' with worse quality can be accepted with an acceptance probability P. The acceptance probability $P(S_t,S_t',T_t)$ is defined in Equation (1) below.

$$P(S_t,S_t',T_t) = e^{-\Delta E/T_t} = e^{-\frac{E(S_t')-E(S_t)}{T_t}} \qquad (1)$$

Here T_t is the cooling temperature of the t^{th} iteration; it and the acceptance probability decrease over time A worse solution is accepted when a randomly generated probability r is smaller than the determined acceptance probability ($r<P$).

– Cooling strategies: the *temperature decrement rule* follows an exponential cooling scheme with *cooling rate* α as listed in Equation (2):

$$T_{t+1} = T_t \times \alpha \quad , \alpha \in [0,1] \qquad (2)$$

4 Simple insertion with steady simulated annealing (SI-SSA)

This section outlines the details of the proposed SI-SSA approach. The procedures of SI-SSA are provided first. Then, all steps in the procedures are presented and described in the following paragraphs.

4.1 *The simple insertion with steady simulated annealing algorithm (SI-SSA)*

The pseudo-code of the proposed simple insertion with steady simulated annealing (SI-SSA) for PFSP is summarized below in Figure 1.

4.2 *Insertion local search*

Many local search strategies for PFSP have been studied and combined with metaheuristics, especially for trajectory metaheuristics. The local search is one way to implement the neighborhood search and the suggested insertion local search is easy to implement. Suppose an existing job processing order is denoted by the permutation π. The insertion operation removes the job at the i^{th} position in π and then inserts it in the j^{th} position, where $i \neq j$, and i, j are randomly generated [25].

Once a permutation π is obtained as a PFSP solution, in the case of $i<j$, the $\pi=\{\pi(1),\ldots, \pi(i\text{-}1), \boldsymbol{\pi(i)}, \pi(i\text{+}1),\ldots, \pi(j\text{-}1), \boldsymbol{\pi(j)}, \pi(j\text{+}1),\ldots, \pi(n)\}$ is the permutation before insertion local search is applied, the new permutation $\pi'=\{\pi(1),\ldots, \pi(i\text{-}1), \pi(i\text{+}1),\ldots, \pi(j\text{-}1), \boldsymbol{\pi(j)}, \boldsymbol{\pi(i)}, \pi(j\text{+}1),\ldots, \pi(n)\}$ can be obtained after applying insertion. And in the case of $i>j$, the permutation before insertion is $\pi=\{\pi(1),\ldots, \pi(j\text{-}1), \boldsymbol{\pi(j)}, \pi(j\text{+}1),\ldots, \pi(i\text{-}1), \boldsymbol{\pi(i)}, \pi(i\text{+}1),\ldots, \pi(n)\}$; after insertion, the new permutation is $\pi'=\{\pi(1),\ldots, \pi(j\text{-}1), \boldsymbol{\pi(i)}, \boldsymbol{\pi(j)}, \pi(j\text{+}1),\ldots, \pi(i\text{-}1), \pi(i\text{+}1),\ldots, \pi(n)\}$. Figure 2 shows an example of a permutation π before insertion local search is applied and two permutations π' after insertion local search is performed from π. The operation of this simple insertion local search is listed in step 1.1 of Figure 1. The insertion operation schematic diagram is shown as follows in Figure 2.

4.3 *Non-decreasing temperature control*

In several modified adaptive simulated annealing strategies, the temperature is not always decreasing, but controlled by certain schemes, such as in [19] which uses a constant temperature. Some studies also employ reheating such as Azizi and Zolfaghari [26] in which the temperature was not associated with the iteration. In this study, two non-decreasing temperature controlling schemes for stage t+1 are applied; the first temperature control scheme applied in SA is a heating temperature control defined in Equation (3). This scheme is called reheating SA herein as indicated in the right part of the step 1.5 in the Figure 1.

$$T_{t+1} = \begin{cases} T_t \times \alpha & , \quad S_t = S_t' \\ T_t \times (1+\beta) & ,otherwise \end{cases} \qquad (3)$$

The rationale of this control is inspired by [26]. In [26], a reheating mechanism was proposed, but the temperature control was not associated with the iteration. This simple reheating mechanism associated with iteration to increase the search ability may be preferable for the following reasons. The ac-

ceptance probability is increased due to heating and thus the exploration area is expanded, i.e., expanding the search pace. Notably, too wide a search range resembles a random walk; therefore, a small β value is suggested for preventing a fully random search. Restated, movement toward too worse a solution is prohibited. The other temperature controlling scheme used in SA is a named steady temperature control (see Equation (4) below) and called steady SA in this work as indicated in the left part of the step 1.5 in the Figure 1.

$$T_{t+1} = \begin{cases} T_t \times \alpha & , \quad S_t = S_t' \\ T_t & ,otherwise \end{cases} \qquad (4)$$

The intent of this design is that once a candidate solution is not accepted (going uphill), that is, $S_t < S_t'$, then the temperature is maintained and hence the P is kept at a higher value. Restated, subsequent exploration ability is preserved for finding a lower makespan solution from high makespan solution neighbors. Otherwise, the temperature is decreased accordingly for further exploitation. These two non-decreasing temperature control mechanisms focus on facilitating enough exploration ability during the solution search. The acceptance probability used in this study is based on relative energy change $rel\_\Delta E=\{E(S_t')-E(S_t)\}/E(S_t)$ (Chen & Hsieh) [27] as indicated in Equation (5).

$$P(S_t, S_t', T_t) = e^{-rel\_\Delta E/T_t} \qquad (5)$$

5 Experimental results

To evaluate the effectiveness of the proposed simple insertion simulated annealing algorithm; the following simulation instances from the Taillard benchmark [28] were tested. Taillard generated a set of FSP scheduling problems with different combinations of jobs (n) and machines (m), $n \in \{20, 50, 100, 200, 500\}$ and $m \in \{5, 10, 20\}$, and there are 10 instances for each problem size. The processing time of each job is distributed uniformly in the interval [1, 99]. This test suite is available from http://mistic.heig-vd.ch/taillard/ and http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop2.txt. Before comparing the performance of the proposed SI-SSA with other schemes, the efficiency of the proposed non-decreasing temperature control mechanisms (steady SA and reheating SA) were first evaluated. The comparison criterion is computed as the relative percentage deviation (*RPD*) as follows.

$$RPD = \frac{Min_{sol} - Best_{sol}}{Best_{sol}} \times 100\% \qquad (6)$$

where Min_{sol} represents the shortest *makespan* of the best solution obtained from the best trial of a specific algorithm and $Best_{sol}$ is the *makespan* of the optimal solution or known upper bound provided by Taillard's instances. To be more objective, the average relative percentage deviation (*ARPD*) is applied as defined in (Ruiz & Stützle 2007) [20].

$$ARPD = \sum_{i=1}^{T} \left(\frac{Min_{sol_i} - Best_{sol}}{Best_{sol}} \times 100\% \right) \Big/ T \qquad (7)$$

where Min_{sol_i} is the *makespan* of a solution given by any of the T repetitions of the compared algorithms. The parameter settings of this simulation are: T_0=1, α=0.99 and β=0.001. There are two comparison bases commonly used, namely the Taillard benchmark upper bound data published in 2004 and 2006. The efficiency comparison between the proposed steady and reheating SAs with the conventional SA is based on both benchmark data as displayed in Tables 1 and 2. Experiments were on the basis of 100000 solutions generated. Five test runs (T=5) were conducted to calculate *ARPD* for fair comparison as defined in [20].

According to the simulation results, the suggested SI-SSA with steady SA yielded the smallest RPDs (0.531% and 0.836%) compared to the other two temperature controlling schemes. Restated, the performance of the proposed SI-SSA with steady SA is better than the other two SAs. Moreover,

the makespan and temperature evolution of these three SAs (conventional SA, reheating SA and steady SA) are displayed in Figure 3; the tests were conducted on tai20_20_1 (20/20 case instance 1) with 100,000 iterations, Figure 3 only shows temperature changes for the first 35,500 iterations.
The conventional SA temperature quickly drops to 0, and therefore the acceptance probability of escaping from the local optimal for searching the neighborhoods is zero, that is, it quickly becomes unable to search the neighborhood to find better solutions. Although the reheating SA does not quickly cool; the temperature increases which cause the acceptance probability of escaping from local minimum remain high, and this does bias the search behavior toward a wider range. On the other hand, the steady SA maintains the same temperature, keeping the same search behavior and allowing the exploration of the neighbors with worse solutions, and therefore increases the possibility of finding better solutions. Figure 3 displays the simulation results on Tai_20_20_1 (20/20 case, instance 1), Figure 3(a)-1 shows that the conventional SA algorithm is trapped on local optimal (*makespan*=2349) since acceptance probability is almost zero all the time due to the temperature quickly dropping to zero; Figure 3(b)-1 indicates that the reheating SA algorithm randomly walks about the solution space due to its high acceptance probability owing to the increased temperature, hence it is unable to find good solutions (*makespan*=2370). Nevertheless, Figure 3(c)-1 demonstrates that the steady SA algorithm provides an adequate exploration search ability in the neighborhood since the acceptance probability is maintained so as to have high opportunity to obtain the best solution (*makespan*=2318).
Therefore, the corresponding search behavior of the temperature changing process is compliant with the scheduling results of Tables 1 and 2.

To compare with the top state-of-the-art algorithms listed in [20] (Ruiz & Stützle), the termination condition was based on computation time. The comparison was made using Taillard's 2004 benchmark. Each problem instance was repeated for five independent trials (*T=5*) rather than using the best trial; the average of the five trials was chosen and all the 10 instances for every problem case (*n/m*; *n* jobs/*m* machines) were calculated. The termination condition (*on the basis of computation time*) in [20] was "$n\times(m/2)\times60$", and the algorithm was running on an Athlon XP 1600+ (1400 MHz) system. The proposed scheme was running on a core i7 (3.4 GHz) PC. Therefore, the computation time of this work was "$n\times(m/2)\times60\times(1.4/3.4)$". The simulation results are listed in Table 3. The SI-SSA has the smallest average *ARPD* 0.27%, it ranks 1st compared to the other algorithms. Moreover, the test on the latest upper bound in Taillard's 2006 benchmark was also conducted and the simulation results are displayed in Table 4. The yielded *ARPD*s by the SI-SSA are less than 1.19 % (50/20 case in Table 3) and 1.74% (50/20 case in Table 4) for Taillard 2004 and 2006 benchmark upper bounds, respectively. Simulation results indicate that the performance of the proposed SI-SSA is excellent and competitive on the basis of its generated solution quality.

Since the upper bound changes over time, the percentage increases of the *makespan* above the minimal lower bound (Incr_lb%) are also provided in addition to the average deviation (*ARPD*). As displayed in Table 5, the average Incr_lb% is only 4.98%. Furthermore, the stability of the proposed scheme was also inspected. The *ARPD* intervals (Δ*ARPD*=Max *ARPD* - Min *ARPD*) are shown in Table 5. The maximum *ARPD* interval is 0.57%. Hence, the SI-SSA scheme is considered stable.

6 **Conclusions**

A strategy named simple insertion simulated annealing with steady SA (SI-SSA) is suggested. In SISA, a simple insertion local search is applied to generate the solution of the PFSP to lower computation complexity. Meanwhile, two non-decreasing cooling temperature driven simulated annealing (SA) named steady SA and reheating SA are employed to maintain successive exploration or exploitation in the solution space to increase the chance of finding better solutions. The advantages of the proposed SI-SSA algorithm can be summarized as follows.

– This scheme is easy to implement since only a simple insertion local search is applied. Simple insertion local search used for constructing PFSP solutions reduces the time-complexity to *O(n)* and only one parameter (α) is required for simulated annealing.

– The steady SA maintains a constant temperature, keeping the same search behavior and thus allows the exploration of the neighbors with worse solutions, thereby increasing the chances of finding better solutions so as to escape from local minima.
– The SI-SSA scheme outperforms many complex meta-heuristics, the averaged *ARPD* is only 0.27%, ranking it 1st as shown in Table 1.
– The maximum *ARPD*s are less than 1.19% and 1.74% (for the 50/20 case) on the basis of Taillard's 2004 and the latest 2006 upper bounds as shown in Tables 1 and 2.
– The maximum *RPD* interval (Δ*RPD*) is only 0.57% as indicated in Table 3, indicating that SI-SSA is a stable algorithm for solving PFSP class problems. Meanwhile, the average Incr_lb% is only 4.98%.

Acknowledgements

This work was partly supported by the Ministry of Science and Technology, Taiwan, under Contract MOST 103-2221-E-167-009.

Conflict of interests

The authors declare that this manuscript has no any conflict of interests with other people or institutions.

References

[1]. Che, A. and Chu, C. "Multi-degree cyclic scheduling of two robots in a no-wait flowshop", *IEEE Transactions on Automation Science and Engineering*, 2(2), pp. 173 – 183 (2005).
[2]. Boutellier, J., Bhattacharyya, S.S. and Silven, O. "Low-Overhead Run-Time Scheduling for Fine-Grained Acceleration of Signal Processing Systems", *2007 IEEE Workshop on Signal Processing Systems*, Shanghai, China, pp. 457 – 462 (2007).
[3]. Benbouzid-Sitayeb, F., Varnier, C. and Zerhouni, N. "Proposition of New Genetic Operator for Solving Joint Production and Maintenance Scheduling: Application to the Flow Shop Problem", *2006 International Conference on Service Systems and Service Management*, 1, Troyes, France, pp. 607 – 613 (2006).
[4]. Taillard, E. "Some efficient heuristic methods for the flow shop sequencing problem", *European Journal of Operational Research*, 47, pp. 65-74 (1990).
[5]. Palmer, D.S. "Sequencing jobs through a multistage process in the minimum total time: A quick method of obtaining a near-optimum", *Operational Research Quarterly*, 16, pp. 101–107 (1965).
[6]. Campbell, H.G., Dudek, R.A. and Smith, M.L. "A heuristic algorithm for the n-job, m-machine sequencing problem", *Management Science*, 16, pp. B630–B637 (1970).
[7]. Nawaz, M., Enscore, E.E. and Ham, I. "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem", *OMEGA, International Journal of Management Science*, 11, pp. 91–95 (1983).
[8]. Tseng, L.Y. and Lin, Y.T. "A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem", *Int. J. Production Economics*, 127, pp. 121-128 (2010).
[9]. Xu, X., Xu, Z.H. and Gu, X.S. "An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization", *Expert Systems with Applications*, 38, pp. 7970-7979 (2011).
[10]. Chen, S.H., Chang, P.C., Cheng, T.C.E. and Zhang, Q.F. "A Self-guided Genetic Algorithm for permutation flowshop scheduling problems", *Computers & Operations Research*, 39, pp. 1450-1457 (2012).
[11]. Javadian, N., Mozdgir, A., Kouhi, E.G., Qajar, D. and Shiraqai, M.E. "Solving assembly flowshop scheduling problem with parallel machines using Variable Neighborhood Search", *International Conference on Computers & Industrial Engineering, CIE 2009*, Troyes, France, pp. 102-107 (2009).
[12]. Liu, S., Cui, J.H. and Li, Y. "Heuristic-Tabu Algorithm for Hybrid Flowshop Scheduling with Limited Waiting Time", *International Symposium on Computational Intelligence and Design, ISCID '08*, 2, Wuhan, China, pp. 233-237 (2008).
[13]. Ahmadizar, F. "A new ant colony algorithm for makespan minimization in permutation flow shops" *Computers and Industrial Engineering*, 63(2), pp. 355-361 (2012).

[14]. Chen, R.M., Lo, S.T., Wu, C.L. and Lin, T.H. "An effective ant colony optimization-based algorithm for flow shop scheduling", *2008 IEEE Conference on Soft Computing in Industrial Applications, SMCia '08*, Muroran, Japan, pp. 101-106 (2008).
[15]. Tasgetiren, M.F., Pan, Q.K., Suganthan, P.N. and Chen, H.L. "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops", *Information Sciences*, 181, pp. 3459–3475 (2011).
[16]. Tang, L. and Wang, X. "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(6), pp. 1-12 (2009).
[17]. Wang, X. and Tang, L. "A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking", *Applied Soft Computing*, 12, pp. 652–662 (2012).
[18]. Murata, T., Ishibuchi, H. and Gen, M. "Neighborhood structures for genetic local search algorithms", *International Conference on Knowledge-Based Intelligent Electronic Systems, KES '98*, 2, Adelaide, South Australia, pp. 256-263 (1998).
[19]. Song, X.M., Wang, K. and Xiao, Y. "An Improved Ant Colony Optimization and Its Applications in Flow-Shop Problems", *International Conference on Computational Intelligence and Software Engineering, CiSE 2009*, Wuhan, China, pp. 1-4 (2009).
[20]. Ruiz, R. and Stützle, T. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem", *European Journal of Operational Research*, 177, pp. 2033-2049 (2007).
[21]. Zhao, Z.Y.J., Lau, H.C. and Ge, S.S. "Integrated Resource Allocation and Scheduling in a Bidirectional Flowshop With Multimachine and COS Constraints", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(2), pp. 190-200 (2009).
[22]. Dong, X., Huang, H.K. and Chen, P. "Study on heuristics for the permutation flowshop with sequence dependent setup times", *IEEE International Conference on Information Reuse & Integration, IRI '09*, Las Vegas, Nevada, USA, pp. 417-421 (2009).
[23]. Osman, I. and Potts, C. "Simulated annealing for permutation flow-shop scheduling", *OMEGA, International Journal of Management Science*, 17(6), pp. 551-557 (1989).
[24]. Kirkpatrick, S., Gelatt C.D. and Vecchi, M.P. "Optimization by Simulated Annealing", *Science*, 220 (4598), pp. 671 – 680 (1983).
[25]. Stützle, T. "An ant approach to the flow shop problem",. *The Sixth European Congress on Intelligent Techniques and Soft Computing (EUFIT'98),* 3, Verlag Mainz, Aachen, Germany, pp. 1560–1564 (1998).
[26]. Azizi, N. and Zolfaghari, S. "Adaptive temperature control for simulated annealing: a comparative study", *Computers & Operations Research*, 31, pp. 2439-2451 (2004).
[27]. Chen, R.M. and Hsieh F.R. "An Exchange Local Search Heuristic Based Scheme for Permutation Flow Shop Problems", *Applied Mathematics & Information Sciences*, 8, pp. 209-215 (2014).
[28]. Taillard, E. "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, 64, pp. 278-285 (1993).

Figure captions

Figure 1. The pseudo-code of the proposed simple insertion with steady simulated annealing (SI-SSA)
Figure 2. Insertion operations for the cases of: (a) $i<j$ and (b) $i>j$
Figure 3. Makespan and SA temperature evolutions for the cases of: (a) conventional SA; (b) reheating SA; and (c) steady SA

Table captions

Table 1. Comparison between steady SA, reheating SA and conventional SA–ARPD (Taillard 2004)
Table 2. Comparison between steady SA, reheating SA and conventional SA-ARPD (Taillard 2006)
Table 3. Comparison of different algorithms in [20] on the basis of Taillard 2004 upper bound data-*ARPD*(*%*)
Table 4. Simulation results on the basis of Taillard 2006 upper bound data-*ARPD*(%)
Table 5. *RPD(%)* interval and Incr_lb on the basis of Taillard 2006 data

Figures

1. Loop
1.1. Simple **insertion** local search determines the neighbor of S_t as S_t' at iteration t.
1.2. Calculate the energy $E(S_t')$ of S_t'
1.3. Calculate $\Delta E = E(S_t') - E(S_t)$, $rel\_\Delta E = \Delta E / E(S_t)$
1.4. if $\Delta E \leq 0$
then $S_t := S_t'$,
else if a random variable p, $p \leq e^{-rel\_\Delta E/T_t} \mid p \in U(0,1)$ then $S_t := S_t'$
1.5. **steady temperature controlling scheme**

if $S_t = S_t'$
$T_{t+1} = T_t \times \alpha$
else
$T_{t+1} = T_t$

reheating temperature controlling scheme

if $S_t = S_t'$
$T_{t+1} = T_t \times \alpha$
else
$T_{t+1} = T_t(1+\beta)$

1.6. t:=t+1
2. Until the End condition is satisfied, return S_t

Figure 1. The pseudo-code of the proposed simple insertion with steady simulated annealing (SI-SSA)

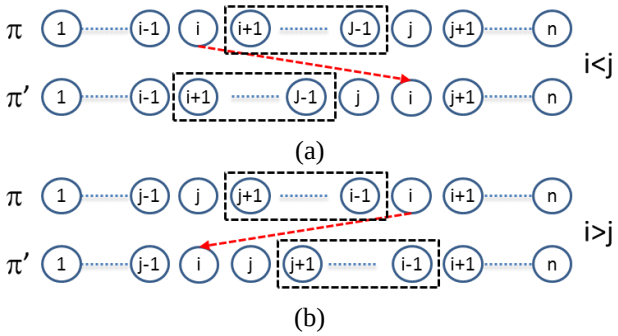
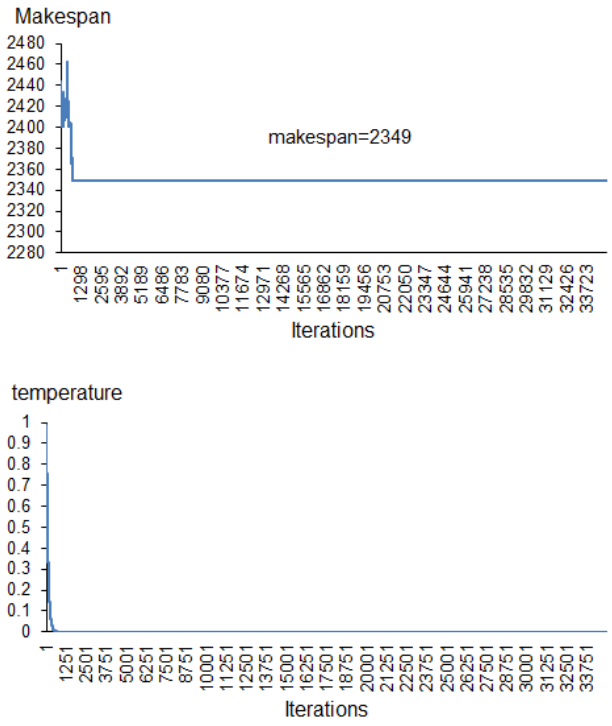
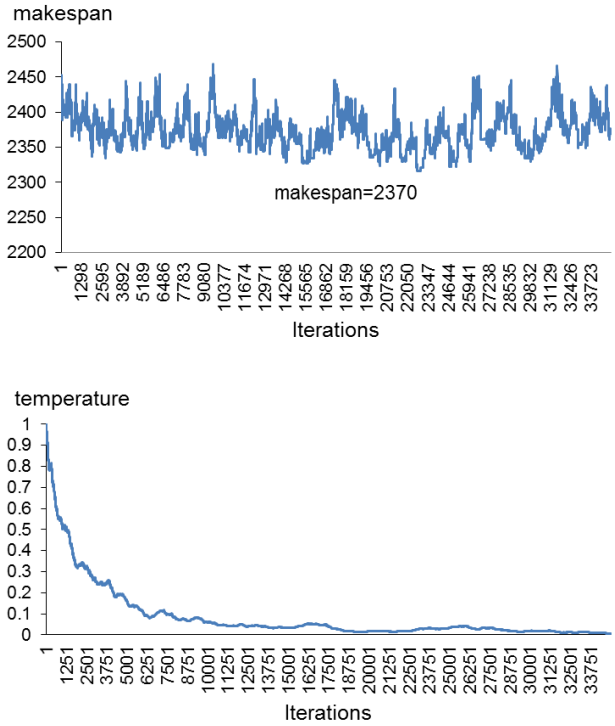


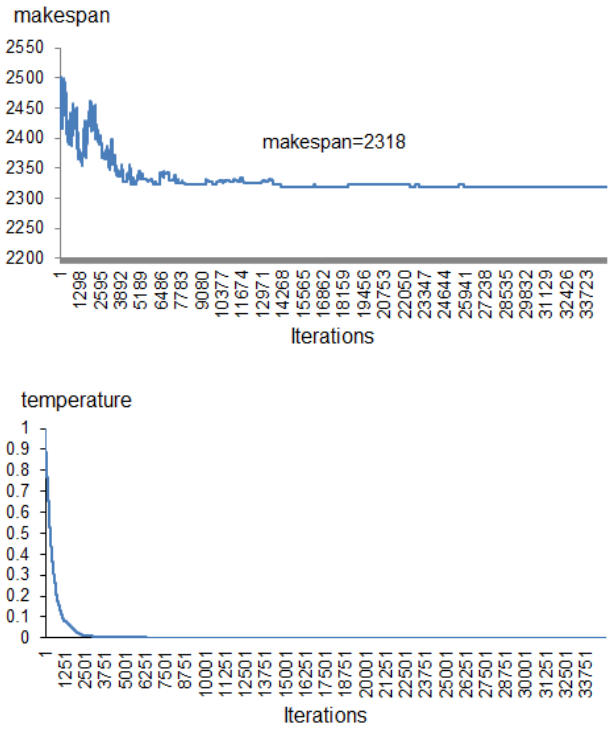
Figure 2. Insertion operations for the cases of: (a) $i<j$ and (b) $i>j$.



(a) SA



(b) reheating SA



(c) steady SA

Figure 3. Makespan and SA temperature evolutions for the cases of: (a) conventional SA; (b) reheating SA; and (c) steady SA.

Tables

Table 1. Comparison between steady SA, reheating SA and conventional SA –ARPD (Taillard 2004)

n/m	SA	reheating SA	steady SA
20/5	0.095	0.202	0.089
20/10	0.601	0.459	0.247
20/20	0.548	0.660	0.159
50/5	0.038	0.021	0.025
50/10	0.432	1.853	0.376
50/20	1.511	4.411	1.453
100/5	-0.016	-0.011	-0.014
100/10	0.433	0.479	0.414
100/20	1.110	2.069	1.041
200/10	0.296	0.370	0.349
200/20	1.138	2.785	1.122
500/20	1.029	1.531	1.108
Average	0.601	1.236	**0.531**

Table 2. Comparison between steady SA, reheating SA and conventional SA-ARPD (Taillard 2006)

n/m	SA	reheating SA	steady SA

20/5	0.144	0.251	0.138
20/10	0.608	0.467	0.255
20/20	0.553	0.664	0.164
50/5	0.038	0.021	0.025
50/10	0.914	2.343	0.858
50/20	2.088	5.004	2.030
100/5	0.030	0.034	0.032
100/10	0.479	0.525	0.460
100/20	2.147	3.115	2.077
200/10	0.321	0.395	0.374
200/20	2.192	3.856	2.176
500/20	1.362	1.865	1.441
Average	0.906	1.545	**0.836**

Table 3. Comparison of different algorithms in [20] on the basis of Taillard 2004 upper bound data-*ARPD*(*%*).

n / m	**This work**	IG_ RSLS	HGA_ RMA	IG_ RS	PACO	M_M MAS	SEASA*	ILS	GA_ RMA	GA_ REEV	GA_ AA	SA_ OP	GA_ MIT	NEHT	GA_ CHEN	SPRIT
20/5	0.09	**0.04**	**0.04**	**0.04**	0.21	**0.04**	0.24	0.49	0.26	0.62	0.94	1.09	0.8	3.35	4.15	4.33
20/10	0.18	**0.06**	0.13	0.25	0.37	0.15	0.88	0.59	0.73	2.04	1.54	2.63	2.14	5.02	5.18	6.07
20/20	0.17	**0.03**	0.09	0.21	0.24	0.06	0.87	0.36	0.43	1.32	1.43	2.38	1.75	3.73	4.26	4.44
50/5	0.02	**0.00**	0.02	0.04	0.01	0.03	0.19	0.2	0.07	0.21	0.36	0.52	0.3	0.84	2.03	2.19
50/10	**0.13**	0.56	0.72	1.06	0.85	1.4	0.76	1.48	1.71	2.06	3.72	3.51	3.55	5.12	6.54	6.04
50/20	1.19	**0.94**	1.28	1.82	1.59	2.18	2.19	2.2	2.74	3.56	4.69	4.52	5.09	6.26	7.74	7.63
100/5	**0.00**	0.01	0.02	0.05	0.03	0.04	0.09	0.18	0.07	0.17	0.32	0.3	0.27	0.46	1.35	1.06
100/10	**0.15**	0.2	0.29	0.39	0.27	0.47	0.65	0.68	0.62	0.85	1.72	1.48	1.63	2.13	3.8	3.01
100/20	**0.34**	1.3	1.66	2.04	2.09	2.59	1.52	2.55	2.75	3.41	4.91	4.63	4.87	5.23	8.15	6.74
200/10	0.14	**0.12**	0.2	0.34	0.27	0.23	0.66	0.56	0.43	0.55	1.27	1.01	1.14	1.43	2.76	2.07
200/20	**0.36**	1.26	1.48	1.99	1.92	2.26	1.43	2.24	2.31	2.84	4.21	3.81	4.18	4.41	7.24	4.97
500/20	**0.47**	0.78	0.96	1.13	1.09	1.15	1.81	1.25	1.4	1.66	2.23	2.52	3.34	2.24	4.72	12.58
Average	**0.27**	0.44	0.574	0.78	0.75	0.88	0.94	1.06	1.13	1.61	2.28	2.37	2.42	3.35	4.83	5.09

* SEASA (Chen et al. 2014)

Table 4. Simulation results on the basis of Taillard 2006 upper bound data-*ARPD*(%)

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
ARPD	0.13	0.18	0.16	0.02	0.59	**1.74**	0.02	0.21	1.38	0.16	1.41	0.81	**0.57**

Table 5. *RPD(%)* interval and Incr_lb on the basis of Taillard 2006 data

m/m	Min *RPD*	*ARPD*	Max *RPD*	Δ*RPD*	Incr_lb%

20/5	0.08	0.13	0.14	0.06	2.56
20/10	0.05	0.18	0.37	0.32	9.58
20/20	0.08	0.16	0.27	0.20	20.76
50/5	0.00	0.02	0.04	0.04	0.83
50/10	0.50	0.59	0.67	0.17	2.76
50/20	1.44	1.74	2.01	**0.57**	10.79
100/5	0.02	0.02	0.03	0.00	1.10
100/10	0.12	0.21	0.33	0.21	1.01
100/20	1.22	1.38	1.56	0.34	5.34
200/10	0.08	0.16	0.24	0.16	0.83
200/20	1.27	1.41	1.50	0.24	3.00
500/20	0.75	0.81	0.87	0.12	1.23
Average	0.47	0.57	0.67	0.20	4.98

Biorgaphies

Ruey-Maw Chen, he received the B. S., the M. S. and the PhD degree in engineering science from National Cheng Kung University of Taiwan ROC 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). Dr. Chen 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, scheduling in the Cloud, and computer networks.

Frode Eika Sandnes received a B.Sc. in computer science from the University of Newcastle upon Tyne, U.K., and a Ph.D. in computer science from the University of Reading, U.K. He is currently pro-rector for Research and internationalization at Oslo and Akershus University College of Applied Sciences in Norway and a Professor in the Institute of Computer Science, Faculty of Technology, Art and Design. His research interests include human computer interaction. Dr. Sandnes has been instrumental in the establishment of the first master specialization in Norway that addresses assistive technologies. He is an editorial member of several journals.