Int J Adv Manuf Technol (2013) 64:369–388
DOI 10.1007/s00170-012-4031-5

ORIGINAL ARTICLE

A simulated annealing/local search to minimize the makespan and total tardiness on a hybrid flowshop

S. M. Mousavi · M. Zandieh · M. Yazdani

Received: 2 July 2009 / Accepted: 28 February 2012 / Published online: 30 June 2012

Abstract Scheduling is a major issue faced every day in manufacturing systems as well as in the service industry, so it is essential to develop effective and efficient advanced manufacturing and scheduling technologies and approaches. Also, it can be said that bi-criteria scheduling problems are classified in two general categories respecting the approach used to solve the problem. In one category, the aim is to determine a schedule that minimizes a convex combination of two objectives and in the other category is to find a good approximation of the set of efficient solutions. The aim of this paper is to determine a schedule for hybrid flowshop problem that minimizes a convex combination of the makespan and total tardiness. For the optimization problem, a meta-heuristic procedure is proposed based on the simulated annealing/local search (SA/LS) along with some basic improvement procedures. The performance of the proposed algorithm, SA/LS, is compared with a genetic algorithm which had been presented in the literature for hybrid flowshop with the objective of minimizing a convex combination of the makespan and the number of tardy jobs. Several computational tests are used to evaluate the effectiveness and efficiency of the proposed algorithm against the other algorithm provided in the literature. From the results obtained, it can be seen that the proposed algorithm in comparison with the other algorithm is more effective and efficient.

Keywords Bi-objective scheduling · Hybrid flowshop · Simulated annealing · Makespan · Total tardiness

1 Introduction

The single-objective scheduling problem has been widely studied in the literature. However, the analysis of the performance of a schedule often involves more than one aspect and requires a multi-objective treatment, because often each particular decision maker wants to minimize a given criterion. For example, in a company, the commercial manager is interested in satisfying his customers which can be achieved by minimizing the tardiness. On the other hand, the production manager wishes to optimize the use of the machines by minimizing the maximum completion time (makespan or $C_{\max}$) or the work in process by minimizing the maximum flow time. Also it can be said from a general point of view that these objectives are each valid. Since these objectives are in conflict of interests, a solution may perform well for one objective, but giving bad results for others. For this reason, scheduling problems have often a multi-objective nature. In the following, we briefly review the related research on the area of multi-objective scheduling problem.

The bi-criteria problem has been considered for the weighted sum of the makespan and the maximum tardiness by Daniels and Chambers [1]. They present an algorithm identifying the exact set of so-called efficient schedules for special cases of two-machine flowshops. Ishibuchi and

S. M. Mousavi
Department of Technical and Engineering,
Faculty of Industrial Engineering, Islamic Azad University,
Noshahr, Iran

M. Zandieh (✉)
Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C.,
Tehran, Iran
e-mail: m_zandieh@sbu.ac.ir

M. Yazdani
Department of Industrial Engineering, Faculty of Industrial and Mechanical Engineering, Qazvin branch, Islamic Azad University,
Qazvin, Iran

Murata [2] extend the multi-objective genetic algorithm by incorporating a local search procedure to the offspring (after the mutation operation). The local search tries to maximize the weighted sum of objective functions with variable weights. Sayin and Karabati [3] deal with the scheduling problem in a two-machine flowshop environment by minimizing makespan and sum of completion times simultaneously. For solving this problem, they developed a branch-and-bound (B&B) procedure that iteratively solves restricted single-objective scheduling problems until the set of efficient solutions is completely enumerated. Chakravarthy and Rajendran [4] propose a heuristic for the same problem, which outperforms that of Daniels and Chambers [1]. Allahverdi [5] addresses the two-machine flowshop scheduling problem with all three criteria (tri-criteria) where the objective is to minimize a weighted sum of mean flow time, makespan, and maximum lateness. He presents a B&B algorithm for the problem and also develops several heuristics to solve large size problems. Lee and Wu [6] develop a B&B procedure for minimizing the weighted sum of total completion time and the total tardiness problem for the two-machine flowshop and the problems with up to 18 jobs can be solved. Chang et al. [7] propose a gradual priority weighting (GPW) approach based on a genetic algorithm. The main characteristic of GPW-based genetic algorithm (GPWGA) is the searching process that starts along the direction of the first selected objective axis (or function coordinate), and the search process progresses in the manner that, gradually, the weight for the first objective function decreases, and the weight for the second objective function increases. Toktas et al. [8] consider a two-machine flowshop scheduling by minimizing the makespan and maximizing the maximum earliness simultaneously. They develop a B&B procedure that generates all efficient solutions with respect to the two criteria, and also a heuristic procedure that generates the approximately efficient solutions.

Ponnambalam et al. [9] propose a TSPGA multi-objective algorithm for flowshop scheduling in which a weighted sum of multiple objectives (i.e., minimizing makespan, mean flow time, and machine idle time) is used. The weights are randomly generated for each generation to enable a multi-directional search. The proposed algorithm is evaluated by applying it to the benchmark problems available in the OR-Library. Ravindran et al. [10] propose three heuristic algorithms for solving the flowshop scheduling problem by makespan and total flow. Loukil et al. [11] propose multi-objective simulated annealing algorithm to tackle the multi-objective production scheduling problems (one machine, parallel machines, and permutation flowshops). They consider seven possible objective functions (the mean weighted completion time, the mean weighted tardiness, the mean weighted earliness, the maximum completion time, the maximum tardiness, the maximum earliness, and the number of tardy jobs). They claim that the proposed multi-objective simulated annealing algorithm is able to solve any subset of seven possible objective functions. A new multi-objective particle swarm is designed by Rahimi-Vahed et al. [12] to search locally Pareto-optimal frontier for the multi-objective mixed-model assembly line-sequencing problem. Jungwattanakit et al. [13] consider the bi-objective hybrid flowshop with unrelated machines and setup times. They formulate the problem by a 0–1 mixed integer programming and propose a genetic algorithm for the problem. This algorithm is used to find a schedule that minimizes a convex combination of makespan and the number of tardy jobs.

In this section, we are going to justify the necessity of this particular study. The papers on bi-criteria scheduling problems are classified in two general categories respecting the approach used to solve the problem. In one category "C1", the approach is through a convex combination of two objectives and, in the other category "C2" is through a set of non-dominated solutions (efficient solutions or Pareto front). As just reviewed, in many studies, the algorithms are developed to search Pareto-optimal solutions for the scheduling problems. These algorithms have not the capability to solve a convex combination of scheduling problems. Also, there are rather fewer studies that considered this special kind of scheduling problems. Hence, our objective is to find a good quality schedule as a convex combination on bi-criteria scheduling problems.

The remainder of the paper is organized as follows. The scheduling problem considered in this paper is described in Section 2. The proposed algorithm is presented in Section 3. The computational results and numerical comparisons are reported in Section 4. Finally, conclusions and future study are given in Section 5.

2 Problem definition

2.1 Hybrid flowshop scheduling problem

One of the most applied and recognized scheduling problems is the hybrid flowshop which has various applications in real-world industries. The hybrid flowshop scheduling problem involves the sequence of jobs in a flowshop which at least one stage has to consist of several machines (more than one machine).

In this section, we are going to describe the general and particular features of this study. All data in this problem (i.e., processing times of operations, setup times, due date, number of stages, number of machines, and number of jobs) are known and constant (and deterministic) when scheduling

Table 1 Factor levels

Factor	Levels
Number of jobs	15; 20; 25; 30
Number of stages	5; 10
Number of machines	Unif (1, 4); Unif (1, 10)
Processing times	Unif (50, 70); Unif (20, 100)
Setup times	Unif (12, 24)

is undertaken. Machines are available at all times, with no breakdowns or scheduled or unscheduled maintenance. All jobs are available at zero time. Jobs are always processed without error. Job processing cannot be interrupted (no preemption is allowed) and jobs have no associated priority values. There is no travel time between stages; an operation of a job cannot be performed until its preceding operations are completed. Each machine can process only one job at the same time, a job cannot be processed on more than one machine at the same time, there is no restriction on queue length at any machine. Machines in parallel are identical in capability and processing rate.

One of the most widely used assumptions in scheduling configurations is the consideration of setup times. The setup times considered in this problem are classified into two types: (1) sequence-independent setup time and (2) sequence-dependent setup times. A sequence-independent setup time deals with the setup time that depends on the machine to which a job is assigned. It is assumed to occur only when a job is assigned to a machine at the first position at some stage. The particular assumption of this study points that the sequence-dependent setup times is required when switching occurs between two different jobs. After the completing of one job and before the beginning of the process for the next job, some sort of setup has to be performed. The time required to do the setup depends on both the prior and the current job to be processed; which means the setup times are sequence dependent. For further study on setup times literature in scheduling problem you can refer to the complete survey which is presented by Allahverdi et al. [14].

2.2 Scheduling objectives (objective functions)

The scheduling objective in such industries may vary. Analysts consider the possible objectives such as: mean weighted completion time, mean weighted tardiness, mean weighted earliness, the makespan, the maximum tardiness, the maximum earliness, number of tardy jobs, total completion time, total tardiness, total setup time, etc. However, we have tried to work with two of the most common objectives in industries. The maximum completion time criterion has been used in many studies and also has been chosen as one of the objectives in this paper. Decreasing the completion times is an effective method in reducing job

Table 2 Design of test problems

Problem	$n \times g$	Number of machine	$[MinC_{\max}; \overline{C}_{\max}; MaxC_{\max}]$	(τ, R)	$\overline{d}$	$[\overline{d} - R\overline{d},\ \overline{d}]$	$[\overline{d}, \overline{d} + (\overline{C}_{\max} - \overline{d})R]$
T1	15×5	{2-1-4-3-2}	[1,394.28; 1,422.48; 1,450.68]	(0.2, 0.2)	1,138	[910.5, 1,138]	[1,138, 1,195.5]
T2				(0.2, 0.8)	1,138	[227.5, 1,138]	[1,138, 1,365.5]
T3				(0.5, 0.5)	711	[355.5, 711]	[711, 1,066.5]
T4				(0.8, 0.2)	284.5	[227.5, 284.5]	[284.5, 512.5]
T5				(0.8, 0.8)	284.5	[57, 284.5]	[284.5, 1,195]
T6	20×5	{4-1-2-3-2}	[1,793.18; 1,825.16; 1,857.14]	(0.2, 0.2)	1,460	[1,168, 1,460]	[1,460, 1,533]
T7				(0.2, 0.8)	1,460	[292, 1,460]	[1,460, 2,920]
T8				(0.5, 0.5)	912.5	[456, 912.5]	[912.5, 1,368.5]
T9				(0.8, 0.2)	365	[292, 365]	[365, 657]
T10				(0.8, 0.8)	365	[73, 365]	[365, 1,533]
T11	25×10	{5-4-6-5-3-4-3-1-2-2}	[2,530.34; 2,611.07; 2,691.80]	(0.2, 0.2)	2,089	[1,671, 2,089]	[2,089, 2,193]
T12				(0.2, 0.8)	2089	[418, 2,089]	[2,089, 2,507]
T13				(0.5, 0.5)	1,305.5	[653, 1,305.5]	[1,305.5, 1,958.5]
T14				(0.8, 0.2)	522	[417.5, 522]	[522, 939.5]
T15				(0.8, 0.8)	522	[104.5, 522]	[522, 2,193.5]
T16	30×10	{6-3-2-4-5-1-6-3-2-4}	[2,844.8; 2,929.20; 3,013.60]	(0.2, 0.2)	2,343	[1,874.5, 2,343]	[2,343, 2,460.5]
T17				(0.2, 0.8)	2,343	[468.5, 2,343]	[2,343, 2,811.5]
T18				(0.5, 0.5)	1,464.5	[732, 1,464.5]	[1,464.5, 2,196.5]
T19				(0.8, 0.2)	586	[469, 586]	[586, 1,055]
T20				(0.8, 0.8)	586	[117, 586]	[586, 2,460.5]

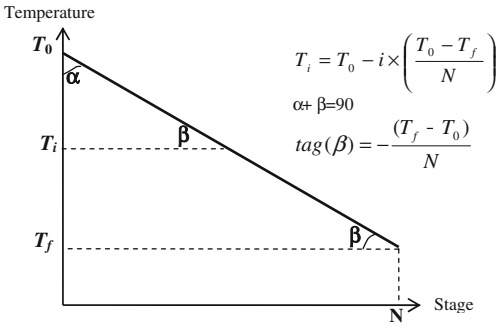

Fig. 1 Linear cooling schedule

lateness and tardiness. Therefore, minimizing the makespan is one of the most important criteria for manufacturing and service organizations.

Nowadays, many companies are concerned with meeting customers' demand in terms of due dates; therefore, scheduling problems with due-date-related measures have more practical meaning than before. Date-related criteria are important objectives in just-in-time environment. Therefore, another criterion namely total tardiness is considered in this problem.

2.3 Scheduling of jobs

In this scheduling, the objective is assigning jobs to the machines at the corresponding stages and to determine the processing sequences on them in order to minimize a convex combination of makespan and the total tardiness. Presented below, is explanatory example of a hypothetical problem to demonstrate how (1) a job sequence is represented as a single solution, (2) the jobs are assigned to machines, and (3) to determine the processing sequences on machines in order to calculate the criteria. Parameters and variables are described as follows:

Parameters:

- n Number of jobs to be scheduled
- g Number of serial stages
- m^t Number of parallel machines at stage t (t=1, 2, 3, …, g)
- s_{ij}^t Setup time between job j and job i at stage t if j is immediately after i
- s_{0j}^t Independent setup time for job j is used when job j at stage t is the first job for scheduling
- p_j^t Processing time for job j at stage t
- d_j Due date of job j

Variables:

- C_j^t Completion time of job j at stage t
- $C_{\max}$ The makespan
- T_j Tardiness of job j
- TT The sum of tardiness of each job
- X_{ijk}^t 1 if job j is scheduled immediately after job i on machine k at stage t, and 0 otherwise
- X_{0jk}^t 1 if job j is sequenced as the first job on machine k at stage t
- X_{i0k}^t 1 if job i is sequenced as the last job on machine k at stage t

Suppose that hybrid flowshop scheduling problem includes n=5 jobs, g=2 stages which there are m^1=2 identical machines at stage 1 and m^2=3 identical machines at stage 2. Now, a job sequence is randomly determined according to chromosome-coding scheme. In this kind of representation, a single row array is formed with a size equal to the number of the jobs which are going to be scheduled. The numeric value of the first element from the array shows which job has been firstly scheduled. Following the same order, the second value shows the secondly scheduled job, and so on. For example, one solution of a hypothetical

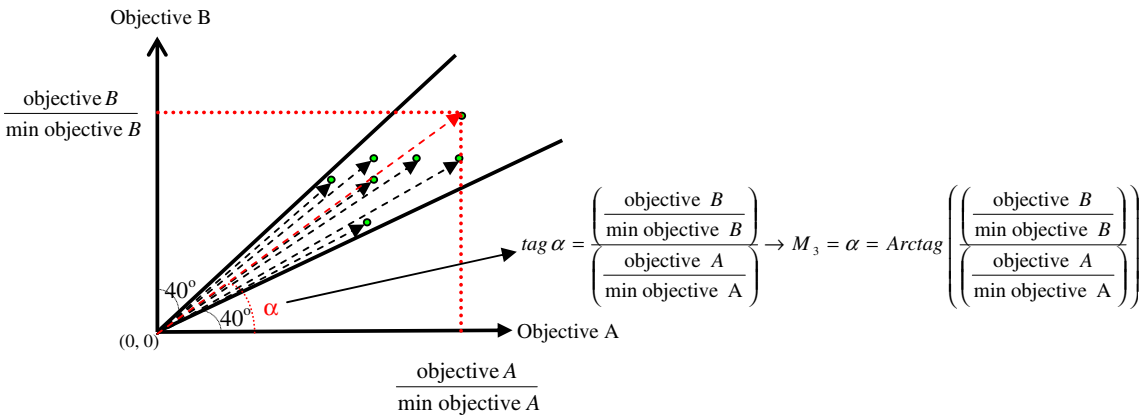

Fig. 2 Definition a range of trade-off between two objectives as an angle

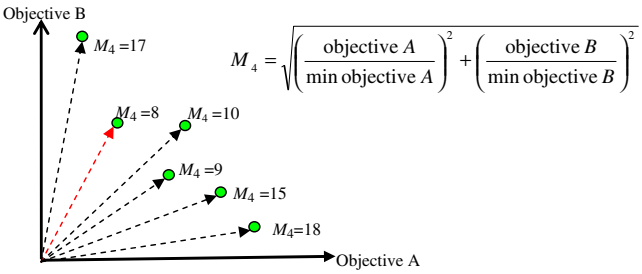

Fig. 3 An example of a hypothetical problem for distance

problem with five jobs as [1–5] denotes that job 3 is process first, and then job 1, job 2, job 5, and job 4 are processed successively.

Now, the jobs are assigned to the machines at stage 1, using the determined job sequence. Therefore, job 3 is processed on machine 1 ($X^1_{031} = 1$) and job 1 is processed on machine 2 at stage 1 ($X^1_{012} = 1$). Thence, job 2 is assigned to the machine with the shortest completion time among the machines at stage 1 ($X^1_{122} = 1$ or $X^1_{321} = 1$). This process continues like this, until all jobs assign to the first stage machines.

In progress, the jobs are delivered to stage 2 for the completion of succeeding operations. The job sequence for the next stage (from stage 2 to stage g) is determined according to their completion times at the previous stage. In second stage, the jobs in places 1 to 3 of the obtained sequence assign to the three machines at stage two, because the machines in parallel are identical in capability and processing rate. The job in place 4 assigns to the machine that its completion time is smaller of other machines; and so on. Finally, the

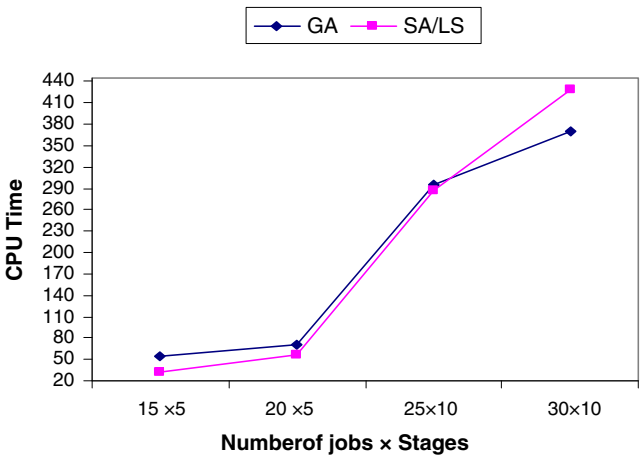

Fig. 4 The computational times of algorithms

makespan and total tardiness time are computed as follows:

1. Makespan ($C_{\max}$): the length of time required to finish processing all jobs, i.e., $C_{\max} = \text{Max}\{C^g_1, C^g_2, ..., C^g_n\}$.
2. Total tardiness ($\text{TT} = \sum_{j=1} T_j$): the sum of tardiness of each job and the tardiness of each job (T_j) equals to $\text{Max}\{(C^g_j - d_j), 0\}$.

2.4 Total objective function

The first decision that has to be made when dealing with a multi-objective optimization problem is on how to combine the objectives. This decision is important, because it can affect on the success of the search. The most straightforward approach to deal with multi-objective problems is to combine them into a single scalar value. The most common way of doing it is setting weights to each criterion and adding those all together by using an aggregating function.

Therefore, the total objective function is constituted of the linear combination of objective functions. For a solution x, the objective function in the study is represented as follows:

$$f(x) = \lambda_1 \times f_1(x) + \lambda_2 \times f_2(x)$$

$f_1(x)$= makespan ; $f_2(x)$= total tardiness ; $\lambda_1+\lambda_2=1$Where $\lambda_i \geq 0$ are the weighting coefficients representing the relative importance of the k objective functions of our problem. It is usually assumed that $\sum_{i=1}^{k} \lambda_i = 1$. The idea behind the λ_1 values is to balance both objectives. In this problem, for a low value of λ_1, the total tardiness problem will dominate the makespan problem, whereas for a large value of λ_1, the makespan problem will dominate the total tardiness problem.

The normalized objectives can be obtained by the relation:

$$f'_1(x) = \frac{\min f_1}{f_1(x)}$$

$$f'_2(x) = \begin{cases} \frac{\min f_2}{f_2(x)} & \text{if } \min f_2 > 0 \\ \frac{1+\min f_2}{1+f_2(x)} & \text{if } \min f_2 = 0 \end{cases}$$

Where "min f_1" and "min f_2" are the minimum of the makespan and the total tardiness values, respectively. If the minimum total tardiness of the jobs equals zero then one is added to the denominator and numerator. In following, an example is provided to clarify:

Suppose objectives of two sequences x_1 and x_2 (1) $f_1(x_1)$=10, $f_2(x_1)$=20 (2) $f_1(x_2)$=10, $f_2(x_2)$=100, and "min

f_2"=0. The normalized objective by the relation "$\frac{\min f_2}{f_2(x)}$" is equal to zero for both sequences. The result concludes that the second objective has no effect on the total objective function. In order to solve this problem the value "1" is added to the denominator and numerator. Hence, the total objective function changed into Eq. 1:

$$
\begin{aligned}
&f(x) = [\lambda_1 \times f'_1(x) + \lambda_2 \times f'_2(x)]^{-1} \\
&\text{Where} \quad \lambda_1 + \lambda_2 = 1 \\
&f'_1(x) = \frac{\min f_1}{f_1(x)} \\
&f'_2(x) = \left\{ \begin{array}{ll} \frac{\min f_2}{f_2(x)} & \text{if } \min f_2 > 0 \\ \frac{1+\min f_2}{1+f_2(x)} & \text{if } \min f_2 = 0 \end{array} \right\}
\end{aligned}
\tag{1}
$$

For further study on the mathematical model (constraints related to problem) can refer to Jungwattanakit et al. [13] and Kurz and Askin [15].

3 The proposed simulated annealing/local search

3.1 Introduction to simulated annealing

Simulated annealing is so named because of its analogy to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration (i.e., its minimum lattice energy state), and thus is free of crystal defects. If the cooling schedule is sufficiently slow, the final configuration results in a solid with such superior structural integrity. Simulated annealing establishes the connection between this type of thermodynamic behavior and the search for global minima for a discrete optimization problem. Furthermore, it provides an algorithmic means for exploiting such a connection. Simulated annealing is introduced to combinational optimization by Kirkpatrick [16] in 1982.

Table 3 Solutions obtained by SA/LS and genetic algorithm (GA) for problem T1 to T5 (table×10^3)

	T1		T2		T3		T4		T5	
	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T
λ_1=0.25										
SA/LS	1.3737	0.3981	1.4261	0.0946	1.3829	0.8898	1.3668	6.7197	1.3644	6.0083
	1.3828	0.4208	1.4193	0.0921	1.3706	0.9031	1.3542	6.7284	1.3747	6.0694
	1.3749	0.4318	1.4193	0.0921	1.3802	0.8652	1.3715	6.6888	1.3685	6.0018
GA	1.3886	0.4511	1.4111	0.1141	1.3802	0.8652	1.3958	6.7852	1.3588	6.1509
	1.3737	0.4368	1.4171	0.1142	1.3778	0.9116	1.3746	6.7014	1.3795	6.0796
	1.3852	0.4385	1.4369	0.1188	1.3866	0.9805	1.3851	6.8629	1.3944	6.1372
	1.3564[a]	0.3981[b]	1.3560[a]	0.0921[b]	1.3571[a]	0.8652[b]	1.3527[a]	6.6868[b]	1.3556[a]	5.9905[b]
λ_1=0.5										
SA/LS	1.3718	0.4004	1.4343	0.1158	1.3676	0.9040	1.3546	6.7843	1.3524	6.1578
	1.3569	0.4276	1.4193	0.0921	1.3877	0.9257	1.3522	6.7834	1.3700	6.1116
	1.3675	0.4120	1.4193	0.0921	1.3944	0.9336	1.3592	6.6863	1.3676	6.0684
GA	1.3683	0.4402	1.4160	0.1973	1.3884	0.9994	1.3750	6.7302	1.3764	6.0973
	1.3713	0.4231	1.4277	0.1233	1.3875	1.0387	1.3763	6.8552	1.3835	6.0990
	1.3773	0.5116	1.4167	0.1197	1.3947	0.9824	1.3798	6.7542	1.3787	6.1062
	1.3509[a]	0.4004[b]	1.3699[a]	0.0921[b]	1.3617[a]	0.9040[b]	1.3520[a]	6.6863[b]	1.3524[a]	6.0498[b]
λ_1=0.75										
SA/LS	1.3667	0.4496	1.4343	0.1158	1.3706	0.9031	1.3563	6.7057	1.3591	6.1488
	1.3792	0.4486	1.4050	0.1080	1.3801	0.9778	1.3617	6.9423	1.3583	6.0840
	1.3711	0.4165	1.4277	0.0962	1.3737	0.8724	1.3504	6.8544	1.3622	6.2258
GA	1.3698	0.4591	1.4069	0.2299	1.3753	0.9655	1.3594	6.7560	1.3663	6.1901
	1.3701	0.4919	1.4266	0.1342	1.3643	1.0221	1.3731	6.7592	1.3624	6.1870
	1.3892	0.4539	1.4043	0.2615	1.3986	1.0640	1.3872	6.8626	1.3833	6.1830
	1.3520[a]	0.4165[b]	1.3652[a]	0.0962[b]	1.3577[a]	0.8652[b]	1.3472[a]	6.6999[b]	1.3517[a]	6.0401[b]

[a] min C_{max}

[b] min T.T

Simulated annealing is a neighborhood searching approach designed to obtain a global optimum solution for combinatorial optimization problems. It starts with an initial solution and iteratively moves towards the other existing solutions, while remembering the best solution found so far. In order to reduce the probability of getting trapped in local optima, simulated annealing accepts the moves even toward the inferior neighboring solutions under the control of randomized scheme. More precisely, if a move from current solution S to another inferior neighboring solution S^* results in a change $\Delta E = f(S^*) - f(S)$ in the objective function value, the move is still accepted if $R < \exp^{\left(\frac{-\Delta E}{T}\right)}$, where T is a control parameter, called temperature, and R is a uniform random number between interval (0, 1). Initially, the temperature T is high enough permitting many deteriorative moves to be accepted, and then it is lowered at a low speed of rate to the point which the inferior moves are approximately rejected. This algorithm sequentially and slowly investigates the possible neighbors in each temperature in order to find the best solution.

3.2 Principles of the proposed algorithm

Although the proposed simulated annealing algorithm is structurally similar to the original simulated annealing, there are some differences made by authors. The first difference is in objective function (i.e., step 2 of pseudo code). The normalized objectives are computed by dividing the minimum value of the objectives to the very value of the objectives. The original concept of the objective function in simulated annealing is the smaller the better. Hence, it contradicts the original idea of objective function by considering the normalized objectives. A transformation should be made to reverse the maximization to minimization. Thus, the objective function of a solution is given by Eq. 1. Note that the minimum value of each objective is updated at each

Table 4 Solutions obtained by SA/LS and genetic algorithm (GA) for problem T6 to T10 (table×10^3)

	T6		T7		T8		T9		T10	
	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T
λ_1=0.25										
SA/LS	1.7744	0.6693	1.8155	0.5843	1.7860	2.2728	1.781	11.505	1.7788	8.2337
	1.7661	0.6384	1.8129	0.5843	1.7863	2.2764	1.785	11.617	1.7869	8.4242
	1.7798	0.6700	1.8172	0.5843	1.7917	2.2223	1.776	11.569	1.7630	8.3433
GA	1.7813	0.7341	1.8203	0.6322	1.7988	2.5592	1.791	11.570	1.8034	8.4707
	1.7674	0.7175	1.8097	0.6246	1.7545	2.2987	1.789	11.513	1.7902	8.5691
	1.7695	0.7550	1.8289	0.5843	1.7903	2.5248	1.804	11.639	1.7751	8.3356
	1.7509[a]	0.6384[b]	1.7693[a]	0.5843[b]	1.7528[a]	2.2223[b]	1.737[a]	11.488[b]	1.7548[a]	8.2337[b]
λ_1=0.5										
SA/LS	1.7605	0.6626	1.8214	0.5843	1.7845	2.3914	1.761	11.735	1.7669	8.2615
	1.7819	0.6644	1.8101	0.5843	1.7983	2.3962	1.775	11.726	1.7835	8.3097
	1.7717	0.7207	1.8099	0.5843	1.7777	2.2411	1.756	11.647	1.7766	8.3276
GA	1.7743	0.7778	1.7975	0.7030	1.8053	2.4182	1.779	11.765	1.7984	8.4528
	1.7777	0.8144	1.7901	0.6094	1.7546	2.3801	1.784	11.711	1.8000	8.5514
	1.7909	1.0335	1.8371	0.6322	1.8025	2.6337	1.782	11.770	1.7953	8.5459
	1.7450[a]	0.6578[b]	1.7648[a]	0.5843[b]	1.7546[a]	2.2387[b]	1.739[a]	11.616[b]	1.7539[a]	8.2615[b]
λ_1=0.75										
SA/LS	1.7565	0.7117	1.7788	0.6094	1.7547	2.3878	1.742	11.893	1.7666	8.6658
	1.7673	0.6879	1.7810	0.6094	1.7809	2.2535	1.754	11.863	1.7552	8.4936
	1.7590	0.7622	1.7824	0.6094	1.7691	2.3827	1.754	11.818	1.7829	8.5878
GA	1.7821	0.8871	1.8038	0.8285	1.7935	2.3820	1.787	11.726	1.8008	8.5984
	1.7792	0.7360	1.8336	0.6729	1.7700	2.5197	1.788	11.755	1.7902	8.5589
	1.7800	0.7504	1.7933	0.7595	1.8014	2.5225	1.788	11.657	1.7964	8.5025
	1.7481[a]	0.6821[b]	1.7567[a]	0.5843[b]	1.7495[a]	2.2535[b]	1.741[a]	11.441[b]	1.7552[a]	8.3810[b]

[a] min C_{max}

[b] min T.T

Table 5 Solutions obtained by SA/LS and genetic algorithm (GA) for problem T11 to T15 (table×10^3)

	T11		T12		T13		T14		T15	
	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T	C_{max}	T.T
λ_1=0.25										
SA/LS	2.4760	1.6761	2.5279	8.3255	2.525	10.398	2.436	20.971	2.5518	5.7175
	2.4629	1.4872	2.5159	8.2181	2.536	10.147	2.436	20.971	2.5518	5.7175
	2.4842	1.6824	2.5281	8.3017	2.536	10.147	2.436	20.971	2.5518	5.7175
GA	2.5510	1.7694	2.5818	9.2808	2.564	10.198	2.521	21.213	2.5856	5.7343
	2.4969	1.6216	2.5940	8.6647	2.565	10.715	2.522	21.260	2.5918	5.9560
	2.5627	1.7467	2.5320	9.0372	2.548	10.544	2.501	21.252	2.5956	6.1994
	2.4343[a]	1.4840[b]	2.4674[a]	8.2181[b]	2.450[a]	10.147[b]	2.436[a]	20.923[b]	2.4648[a]	5.7175[b]
λ_1=0.5										
SA/LS	2.5185	1.7051	2.5280	8.3529	2.502	10.695	2.442	21.302	2.5041	5.7886
	2.5185	1.7051	2.5280	8.3529	2.501	10.187	2.456	21.488	2.4876	5.5056
	2.5185	1.7051	2.5280	8.3529	2.501	10.187	2.437	21.130	2.4876	5.5056
GA	2.5202	1.8303	2.5253	8.7814	2.525	10.465	2.487	21.287	2.5700	5.6643
	2.4973	1.7275	2.5392	9.0126	2.537	10.601	2.492	21.116	2.5397	5.8925
	2.5013	1.8858	2.5287	8.6221	2.568	10.600	2.540	21.301	2.5437	6.0939
	2.4492[a]	1.7051[b]	2.4667[a]	8.2471[b]	2.467[a]	10.187[b]	2.436[a]	21.116[b]	2.4664	5.5056[b]
λ_1=0.75										
SA/LS	2.4630	1.7242	2.5212	8.7949	2.449	11.453	2.440	21.726	2.4866	5.8117
	2.4630	1.7242	2.5212	8.7949	2.485	10.841	2.445	21.425	2.4475	6.4423
	2.4630	1.7242	2.5212	8.7949	2.485	10.841	2.443	21.027	2.4537	5.7153
GA	2.5207	1.8302	2.5521	9.0378	2.528	10.952	2.469	21.305	2.4943	5.9981
	2.4869	1.9609	2.5728	9.1551	2.518	10.791	2.477	21.325	2.4874	5.6878
	2.4994	1.8301	2.5748	8.7834	2.547	10.873	2.475	21.484	2.5136	5.9263
	2.4520[a]	1.7010[b]	2.4649[a]	8.7834[b]	2.449[a]	10.714[b]	2.437[a]	21.018[b]	2.4475[a]	5.6259[b]

[a] min C_{max}

[b] min T.T

successive iteration procedure during the search. Therefore, the final values of these minimums are obtained in the end of run.

The second difference is in the step 4 of pseudo code. Here, the obtained schedule of previous step (step 3) imports into a neighborhood search mechanism to produce new solutions from current solution with the aim of improving a solution. One of key characteristics of proposed simulated annealing is neighborhood search structure which is randomly selected among several alternatives at each iteration procedure. This feature provides a means in order to guide the search to another promising region through various moves. After applying the mentioned mechanism, objectives are calculated for each neighborhood solution. Then the solution which has the lowest value for f_{SA} (Eq. 1) is accepted among neighborhood solutions.

The last difference between two algorithms is in the end of pseudo code. In this section of the algorithm, in order to reinforce the performance, two simple forms of local search are applied on the best solution of archive.

3.3 SA/LS pseudo code

Initialization

- The representation of a solution

We apply a scheme using integers which show the number of job(s).

- Draw a seed sequence x

The seed sequence is generated in a random way from the search space.

- Evaluate $f_1(x)$ and $f_2(x)$

Where $f_1(x)$ is the makespan, $f_2(x)$ is the total tardiness of seed sequence which is generated.

- Input parameters

Initial temperature (T_0); final temperature(T_f); number of stages for reach of T_0 to T_f (N); maximum number of

iterations in each temperature (max_it); the weighting coefficients ($\lambda \in \{0.25,\ 0.5,\ 0.75\}$)

- Set

$T = T_0;\ it = 1;\ q = 1;\ \text{Archive}(q) = \{x\};$

While $T > T_f$

For $it = 1 : \max\_it$

Step 1: *y*=Move *x*;

To apply the simulated annealing method we must define the neighborhood $N(x)$ of a solution x. We use the simplest way to create a neighbor: a new order of the n jobs is created by exchange of two jobs chosen randomly in the order corresponding to x. For this new order, the corresponding schedule y, $\left(f_1(y), f_2(y)\right)$ are calculated, so that the comparison between x and y can be made as described in the simulated annealing method.

Step 2: Calculate f_{SA} and ΔE as follows;

$$\min\ f_1 = \min\left(f_1(x), f_1(y)\right)$$
$$\min\ f_2 = \min\left(f_2(x), f_2(y)\right)$$
$$f_{SA}(x) = \left[\lambda_1 \times f_1'(x) + \lambda_2 \times f_2'(x)\right]^{-1}$$
$$f_{SA}(y) = \left[\lambda_1 \times f_1'(y) + \lambda_2 \times f_2'(y)\right]^{-1}$$
$$\Delta E = f_{SA}(y) - f_{SA}(x)$$

Step 3: Decision making;

If $\Delta E < 0$ **Then**
$q = q + 1;\ Archive(q) = \{y\};\ x = y;$
Else If $random < \exp(-\Delta E / T)$ **Then**
$q = q + 1;\ Archive(q) = \{y\};\ x = y;$
End If

Step 4: Improvement of solution which obtained in step 3 as follows;

i) Choose randomly integer number (k), in the range [1–4].

- If $k = 1$; perform the **Swap move** [18] on x and generate $(n-1)$ sequences.
- If $k = 2$; perform the **Random insertion scheme** [19] on x and generate $(n-1)$ sequences.
- If $k = 3$; perform the **Inversion move** [20] on x and generate $(n-1)$ sequences.
- If $k = 4$; perform the **Shift move** [18] on x and generate $(n-1)$ sequences.

ii) Evaluate objectives of the new solutions in the neighborhood of x (which is named z_i).

$$\min\ f_1 = \min\left(f_1(x), \min\ f_1(z_i)\right);\ i = 1, 2, ..., (n-1)$$
$$\min\ f_2 = \min\left(f_2(x), \min\ f_2(z_i)\right);\ i = 1, 2, ..., (n-1)$$
$$z_n = x$$
$$f_{SA}(z_i) = \left[\lambda_1 \times f_1'(z_i) + \lambda_2 \times f_2'(z_i)\right]^{-1};\ i = 1, 2, ..., n$$

iii) Accept solution with minimum $f_{SA}(z_i)$ and update solution x with obtained solution of neighborhood $(x = z_i)$. Also, transfer accepted solution with archive: $q = q + 1;\ Archive(q) = \{z_i\}$

End For

$T =$ **Temperature reduction by a linear schedule**

End While

Local search: note that an archive is created, maintained during successive iterations to preserve solution identified during the search. The best solution in the archive (solution corresponding with minimum f_{SA}) is now subjected to two local search schemes, namely, neighborhood swapping [17] and RIPS [18]. Then, solution with minimum f_{SA} is selected.

$$f_{SA} = [\lambda_1 \times f'_1 + \lambda_2 \times f'_2]^{-1}$$

4 Performance analysis of the proposed algorithm

4.1 Generation of test problem

An experiment was conducted to test the performance of the algorithm. One difficulty faced by researchers in scheduling is to compare their methodologies with those of other researchers. If the standard set of test problems is accessible, the performances of different algorithms can be compared on exactly the same set of test problems.

Data required for a problem consist of the range of processing times, range of setup times, number of stages, number of jobs, range in number of machines per stage, and range of due date. Processing times are distributed uniformly over two ranges with a mean of 60: [50–70] and [20–100]. The setup times are uniformly distributed from 12 to 24 which are 20% to 40% of the mean of the processing time. The performance of the meta-heuristics is compared on a set of test problems with 15 jobs×5 stages, 20 jobs×5 stages, 25 jobs×10 stages, and 30 jobs×10 stages. Numbers of machines are distributed uniformly over two ranges of [1–4] and [1–10]. Level range design of each factor is illustrated as shown in Table 1.

The initial test problems are designed based on the information above (Table 1). After several trials on each problem designed for hybrid flowshop, the minimum and maximum values of $C_{\max}$ can be obtained approximately. $\overline{C}_{\max}$ is known as average of bound values. Table 2 shows the obtained results from the experiments. The above results allow producing final problems set including due dates.

Table 6 Solutions obtained by SA/LS and genetic algorithm (GA) for problem T16 to T20 (table×10^3)

	T16		T17		T18		T19		T20	
	$C_{\max}$	T.T	$C_{\max}$	T.T	$C_{\max}$	T.T	$C_{\max}$	T.T	$C_{\max}$	T.T
λ_1=0.25										
SA/LS	2.9482	3.2668	2.9503	8.5884	2.9687	9.1137	2.853	27.698	2.9052	7.5764
	3.0383	3.3664	2.9482	8.6408	2.9557	9.8490	2.947	28.447	2.9799	8.5628
	2.9459	3.5751	2.9419	8.4290	2.9872	9.7147	2.929	28.508	2.9654	8.8081
GA	2.9573	3.3842	2.9098	9.5724	3.056	10.044	2.950	28.563	2.9242	9.0903
	2.9463	3.8518	2.9451	9.0006	3.0533	9.7761	2.944	28.272	2.9831	8.6045
	2.9814	3.6976	2.9664	9.4036	2.966	10.683	2.997	28.233	3.0300	8.8869
	2.8670[a]	3.2582[b]	2.8713[a]	8.4290[b]	2.8632[a]	9.1137[b]	2.853[a]	27.607[b]	2.8895[a]	7.5576[b]
λ_1=0.5										
SA/LS	2.9092	3.2417	2.9084	8.5277	2.9720	8.5640	2.919	28.305	2.9350	7.5024
	2.9238	3.7224	2.9329	8.2318	2.9456	9.9449	2.919	28.709	2.9141	7.9450
	2.9588	3.4787	2.9208	8.9974	2.8910	10.410	2.924	28.741	2.8928	8.3890
GA	2.9651	3.9248	2.9932	9.5431	2.986	10.061	3.041	29.607	2.9948	8.6901
	3.0023	3.6114	2.9180	10.288	2.916	10.197	3.013	28.371	3.0064	8.2599
	2.9416	3.3860	2.9708	9.1786	3.029	10.912	3.034	28.608	3.0359	9.1690
	2.8624[a]	3.2386[b]	2.8648[a]	8.2318[b]	2.863[a]	8.564[b]	2.872[a]	28.285[b]	2.8782[a]	7.4030[b]
λ_1=0.75										
SA/LS	2.9563	3.1270	2.8832	8.3245	2.881	10.680	2.854	30.749	2.9132	9.0324
	2.8973	3.9292	2.8915	9.7114	2.885	10.653	2.853	28.863	2.9312	8.9634
	2.9298	3.2077	2.9103	9.8877	2.892	9.6963	2.856	29.392	2.9056	8.3218
GA	2.9327	3.6519	2.9217	9.2354	2.956	10.162	2.978	28.982	2.9529	9.0169
	3.0053	3.8788	3.003	10.457	3.038	11.320	2.976	28.467	3.0714	8.9795
	2.9267	3.6704	2.9144	9.5001	2.953	11.031	2.996	28.706	3.0462	9.2298
	2.8682[a]	3.1270[b]	2.8580[a]	8.2833[b]	2.863[a]	9.5635[b]	2.852[a]	28.340[b]	2.8761[a]	8.9275[b]

[a] min $C_{\max}$

[b] min T.T

Due dates can be generated from a composite uniform distribution based on R and τ; with probability τ the due date is uniformly distributed over the interval $[\overline{d} - R\overline{d},\ \overline{d}]$ and with probability $(1-\tau)$ over the interval $[\overline{d}, \overline{d} + (C_{\max} - \overline{d})R]$, where τ and R are two parameters called the tardiness factor $(\tau = 1 - \overline{d}/C_{\max})$ and the due date range $(R = (d_{\max} - d_{\min})/C_{\max})$, respectively. Values of τ close to 1, indicate that the due dates are tight, and values close to 0 indicate that the due dates are loose. A high value of R indicates a wide range of due dates, whereas a low value indicates a narrow range of due dates [19]. The values of τ are taken as 0.2, 0.5, and 0.8 and the values of R are taken as 0.2, 0.5, and 0.8. For each problem structure, data based on five different τ and R combinations are used (see Table 2): (0.2, 0.2), (0.2, 0.8), (0.5, 0.5), (0.8, 0.2), (0.8, 0.8).

Note that $d_{\max}$, $d_{\min}$, and $\overline{d}$ are respectively the maximum, the minimum, and the average due date. The average due date $(\overline{d})$ with specific τ can be obtained by the relation $\tau = 1 - \overline{d}/\overline{C}_{\max}$. Following that, the bound values of due dates can be obtained with a specific R by using both the relations $(\overline{d} - R\overline{d})$ and $(\overline{d} + (\overline{C}_{\max} - \overline{d})R)$ that the results are shown in Table 2.

4.2 Parameter setting

Algorithm parameter values vary depending on different problem types when applying algorithm to achieve efficient solutions, so appropriate value selection has significant impact on the efficiency of algorithm. In this paper, the existing parameters of the proposed algorithm (T_0, T_f, and N) are determined through relation "$p = \exp\left(\frac{-\Delta E}{T}\right) = \text{theprobabilityofacceptinganinferiorsolution}$ ". These parameters are obtained after conducting several trials on different instances. But, before setting the parameters' values, it is better to indicate some concepts in order to understand the method.

When ΔE has a negative value, algorithm achieved a superior solution. The acceptance of it is not depending on the temperature value. Consequently, where "ΔE" has a positive value, algorithm has to deal with an inferior solution. It is known that if $R<p$ (R = random (0,1)) then the algorithm accepts an inferior solution. This means the probability of accepting non-improving solutions depends on a temperature parameter, which is typically non-increasing with each iteration of the algorithm. Thus, we know

Table 7 Results of metrics for λ_1=0.25 ($M_2 \times 10^3$)

Test problem	Simulated annealing/local search				Genetic algorithm			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
T1	0.5725	0.6420	44.6369	1.4233	0.5951	0.6710	47.2922	1.4932
T2	0.5751	0.4239	43.6934	1.4476	0.6267	0.4384	49.9702	1.6179
T3	0.5728	0.9940	44.5165	1.4263	0.5728	0.9940	44.5165	1.4263
T4	0.5726	5.3595	44.6132	1.4243	0.5733	5.3697	44.6024	1.4272
T5	0.5727	4.8435	44.7827	1.4223	0.5765	4.9046	44.9223	1.4372
AVE	0.57314	2.45258		1.42876	0.58888	2.47554		1.48036
T6	0.5721	0.9203	44.7524	1.4204	0.6006	0.9800	48.0717	1.5107
T7	0.5734	0.8915	44.3027	1.4317	0.5741	0.8955	44.0510	1.4382
T8	0.5732	2.1147	44.3712	1.4300	0.5798	2.1627	45.9404	1.4394
T9	0.5738	9.0740	44.3258	1.4333	0.5743	9.0820	44.2173	1.4371
T10	0.5725	6.6200	44.6109	1.4239	0.5754	6.6955	45.0229	1.4311
AVE	0.5730	3.9241		1.42786	0.58084	3.96314		1.4513
T11	0.5729	1.7311	44.7271	1.4241	0.5952	1.8404	46.8117	1.4987
T12	0.5730	6.7925	44.4424	1.4282	0.5885	7.1470	45.0826	1.4889
T13	0.5742	8.2443	44.0118	1.4392	0.5763	8.2895	43.8410	1.4510
T14	0.5720	16.3372	45.0656	1.4158	0.5776	16.5400	44.4118	1.4488
T15	0.5742	4.9261	44.0065	1.4394	0.5760	4.9471	43.7138	1.4513
AVE	0.57306	7.60624		1.42934	0.58272	7.7528		1.46774
T16	0.5743	3.1871	44.2755	1.4362	0.5833	3.2775	45.1986	1.4638
T17	0.5736	7.1789	44.2225	1.4338	0.5850	7.4867	45.6159	1.4664
T18	0.5743	7.5774	43.9636	1.4405	0.5940	8.0954	45.1684	1.5126
T19	0.5722	21.4867	45.0943	1.4165	0.5809	21.9240	44.2318	1.4661
T20	0.5725	6.4086	44.9159	1.4198	0.6057	7.1991	47.7988	1.5369
AVE	0.57338	9.16774		1.42936	0.58278	9.59654		1.48916

that the probability in which algorithm accepts an inferior solution in initial temperatures (i.e., inT_0) is high and in finishing temperatures (i.e., in T_f) is low. Therefore, "p = the probability of accepting an inferior solution" has a higher value in the initial temperature (i.e., p=0.95) and on the contrary it has a smaller value in the final temperature (i.e., p=0.05).

The maximum ΔE is selected among ΔE positive values obtained through several trials on every designed problem for hybrid flowshop (here, maximum ΔE=0.1496). Then, by accepting an inferior solution with the acceptance ratio (p) of 0.95 (high probability), the initial temperature (T_0) is found through this equation:

$$T_0 = \frac{-\Delta E}{Ln(p)} \left(\text{here;} T_0 = \frac{-0.1496}{Ln(0.95)} = 2.916 \right)$$

It means that the algorithm accepts the worst inferior solution in the initial temperature (T_0) with the acceptance ratio (p) of 0.95.

The temperature T is lowered so the inferior moves are approximately rejected. Here, by accepting an inferior solution, with the acceptance ratio (p) of 0.05 (low probability), the final temperature (T_f) is found by this equation:

$$T_f = \frac{-\Delta E}{Ln(p)} \left(\text{here;} T_f = \frac{-0.1496}{Ln(0.05)} = 0.05 \right)$$

It means that the algorithm accepts the worst inferior solution in the final temperature (T_f) with the acceptance ratio (p) of 0.05.

The temperature is reduced after a predetermined number of iterations at a given temperature, using the relationship equation below:

$$T_i = T_0 - i \times \left(\frac{T_0 - T_f}{N} \right)$$

The number of stages for reaching of T_0 to T_f in a linear schedule (N) is obtained as follows: Based on Fig. 1, (0, T_0) and (N, T_f) are two points of a straight line where N is uncertain. It is known that sum of α and β is equal to 90°. Angle of the slope for α will be appropriate if α is bigger than 45 (α>45). Also, angle of the slope for β will be appropriate if β is smaller than 45 (β<45) since at a lower

Table 8 Results of metrics for λ_1=0.5 ($M_2 \times 10^3$)

Test problem	Simulated annealing/local search				Genetic algorithm			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
T1	0.6701	0.8861	44.5602	1.4252	0.6823	0.8972	46.1501	1.4653
T2	0.6745	0.7557	43.9853	1.4399	0.7309	0.7682	51.4904	1.6609
T3	0.6676	1.1358	44.8761	1.4173	0.6905	1.1886	46.6956	1.4933
T4	0.6678	4.0228	44.8478	1.4180	0.6719	4.0526	44.7042	1.4309
T5	0.6698	3.7180	44.7678	1.4243	0.6723	3.7369	44.7201	1.4323
AVE	0.66996	2.10368		1.42494	0.68958	2.1287		1.49654
T6	0.6703	1.2115	44.9549	1.4257	0.7069	1.2761	49.3071	1.5595
T7	0.6723	1.1971	44.2772	1.4324	0.6792	1.1998	45.7971	1.4549
T8	0.6698	2.0094	44.6560	1.4243	0.6801	2.0674	46.7535	1.4596
T9	0.6694	6.7015	44.7977	1.4230	0.6742	6.7475	44.5015	1.4383
T10	0.6683	5.0142	44.7884	1.4195	0.6774	5.1256	44.9380	1.4485
AVE	0.67002	3.22674		1.42498	0.68356	3.28328		1.47216
T11	0.6728	2.1118	44.2008	1.4344	0.6739	2.1124	44.8167	1.4374
T12	0.6750	5.4405	44.6620	1.4409	0.6821	5.5754	45.5627	1.4642
T13	0.6697	6.3440	44.6079	1.4240	0.6779	6.4950	45.1056	1.4501
T14	0.6669	11.7835	45.0072	1.4150	0.6717	11.8040	44.3489	1.4306
T15	0.6686	3.9966	44.7548	1.4203	0.6822	4.1171	44.6354	1.4643
AVE	0.6706	5.93528		1.42346	0.67756	6.01998		1.44932
T16	0.6692	3.0755	44.7315	1.4222	0.6814	3.1638	45.6618	1.4618
T17	0.6700	5.7180	44.5673	1.4250	0.6912	6.0747	46.0661	1.4946
T18	0.6749	5.7680	43.9298	1.4414	0.7117	6.5235	48.4021	1.5710
T19	0.6704	15.6120	44.5552	1.4263	0.6779	15.6920	43.7144	1.4514
T20	0.6740	5.2187	44.8222	1.4377	0.7009	5.6332	46.8879	1.5284
AVE	0.6717	7.07844		1.43052	0.69262	7.41744		1.50144

Table 9 Results of metrics for λ_1=0.75 ($M_2 \times 10^3$)

Test problem	Simulated annealing/local search				Genetic algorithm			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
T1	0.8067	1.1325	44.5981	1.4242	0.8217	1.1421	47.4122	1.4972
T2	0.8216	1.0948	43.7180	1.4469	0.8719	1.1035	53.1638	1.7430
T3	0.8070	1.2484	44.9018	1.4284	0.8234	1.2729	47.7690	1.5071
T4	0.8034	2.6936	44.8319	1.4196	0.8057	2.7085	44.9806	1.4265
T5	0.8035	2.5397	45.0679	1.4228	0.8076	2.5686	45.4625	1.4371
AVE	0.80844	1.7418		1.42838	0.82606	1.75912		1.52218
T6	0.8066	1.4974	44.9296	1.4280	0.8206	1.5184	46.6726	1.4833
T7	0.8128	1.4864	45.8467	1.4536	0.8496	1.5348	51.8557	1.6528
T8	0.8086	1.8990	44.4904	1.4270	0.8209	1.9406	45.8770	1.4725
T9	0.8064	4.2797	46.0933	1.4428	0.8159	4.2553	44.7727	1.4467
T10	0.8021	3.4398	45.3823	1.4237	0.8135	3.4729	44.7476	1.4411
AVE	0.8073	2.52046		1.43502	0.8241	2.5444		1.49928
T11	0.8043	2.2783	45.2599	1.4270	0.8209	2.3321	46.5465	1.4821
T12	0.8111	4.0896	44.3906	1.4314	0.8210	4.1270	43.7508	1.4461
T13	0.8089	4.5740	44.9195	1.4330	0.8146	4.5862	44.4092	1.4393
T14	0.8012	7.0890	44.9418	1.4163	0.8085	7.1780	45.0148	1.4332
T15	0.8037	3.2691	45.3792	1.4273	0.8096	3.2875	44.8502	1.4335
AVE	0.80584	4.2600		1.4270	0.81472	4.30216		1.44684
T16	0.8146	2.9990	44.1334	1.4361	0.8350	3.1125	48.7971	1.5522
T17	0.8050	4.2435	44.8906	1.4240	0.8279	4.5001	47.4823	1.5127
T18	0.8071	4.5930	45.1055	1.4312	0.8253	4.7575	45.8231	1.4816
T19	0.8031	9.3555	45.5138	1.4276	0.8213	9.3487	43.9091	1.4484
T20	0.8081	4.4430	44.9675	1.4316	0.8143	4.4689	44.5305	1.4402
AVE	0.80758	5.1268		1.4301	0.82476	5.23754		1.48702

speed the temperature is dropping. Now, N for certain β (i.e., β=3) will be obtained by equation below:

$$N = \frac{(T_f - T_0)}{-\text{tag}(\beta)} \left(\text{here;} N = \frac{(0.05 - 2.916)}{-\text{tag}(3)} = 55 \right)$$

4.3 Performance measures

In order to compare the obtaining solutions of each algorithm, some of the quality measures are explained below. The use of performance measures (or metrics) allows a researcher to

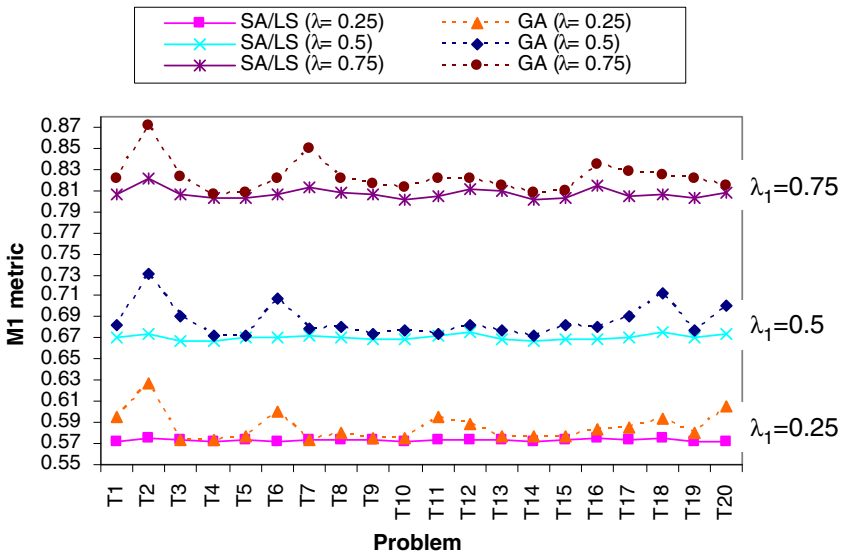

Fig. 5 Values of M1 metric for problems T1 to T20 and for various λ_1

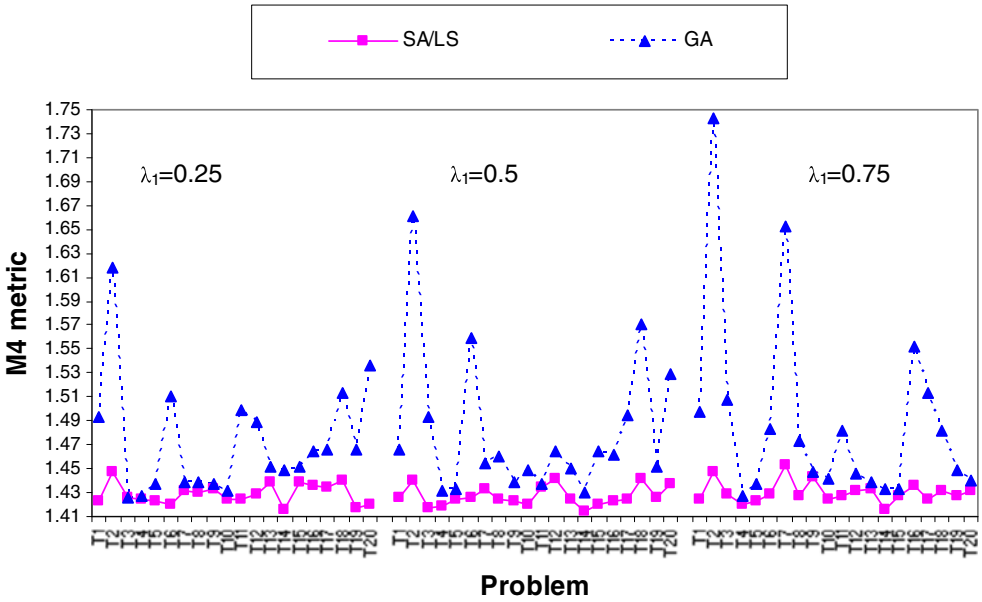

Fig. 6 Values of M4 metric for problems T1 to T20 and for various λ_1

assess the performance of their algorithms (in a quantitative way). In this paper, four metrics are used to evaluate the quality of solutions. The metrics applied in this paper are described as follows:

The first and the second metrics allow us to measure the performance of the algorithms by considering objective functions that applied in each of two algorithms. These metrics (objective functions) were applied to minimize a convex combination of the makespan and the total tardiness. So the metrics are given as follows:

$$M_1 = \left[\lambda_1 \times \frac{\min f_1}{f_1(x)} + \lambda_2 \times \frac{1+\min f_2}{1+f_2(x)}\right]^{-1} \text{ that } \quad \lambda_1 + \lambda_2 = 1$$

$$M_2 = [\lambda_1 \times f_1(x) + \lambda_2 \times f_2(x)] \quad \text{that} \quad \lambda_1 + \lambda_2 = 1$$

Where "M_1" and "M_2" are applied in SA/LS and genetic algorithm proposed by Jungwattanakit et al. [13] as the objective function (with and without normalizing), respectively. It is well-known that lower values of "M_1" and "M_2" represent better solution.

Generally, decision makers require schedules with respect to the trade-off between the objectives. Therefore the obtained solution should represent a trade-off between the various objectives. If there is a solution just along an individual axis (in one corner of the solution space), it will not be appropriate, because this solution is proper from just one objective view (similar to a single-objective problem). Hence, the trade-off surface between two objectives is required which is defined as an angle (see Fig. 2). Thus, the third metric measures the angle of slope in relation to the origin. Angles between 30 and 60 (45±15) degrees are acceptable. Angles between 30 and 40, 40 and 50, and 50 to 60° are acceptable when $\lambda_1 = 0.75, 0.5,$ and0.25, respectively. The definition of this metric is as follows:

$$M_3 = \text{Arctag}\left(\frac{f_2(x)/\min f_2}{f_1(x)/\min f_1}\right)$$

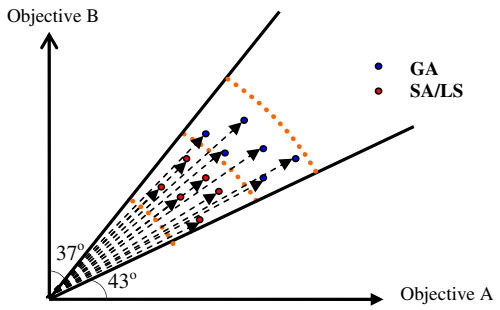

Fig. 7 Performances of the SA/LS and genetic algorithm by considering M_3 and M_4 metrics simultaneously

The fourth metric measures the distance between the obtained solutions and the origin, but before expressing this performance measure, some concepts should be indicated. First, we assume that the origin is (0, 0). Second, the normalized objectives are computed by dividing the minimum value of the objectives into the vary value of the objectives. Third, if a solution converges toward the side of an origin or ideal solution, we can claim that the performance of the algorithm is proper (see Fig. 3). So, the M_4 try to determine the convergence of solutions by calculating the distance between the obtained solutions and the origin/ideal solution. Lower values of this metric represent a better solution. The metric defines as follows:

$$M_4 = \sqrt{\left(\frac{f_1(x)}{\min f_1}\right)^2 + \left(\frac{f_2(x)}{\min f_2}\right)^2}$$

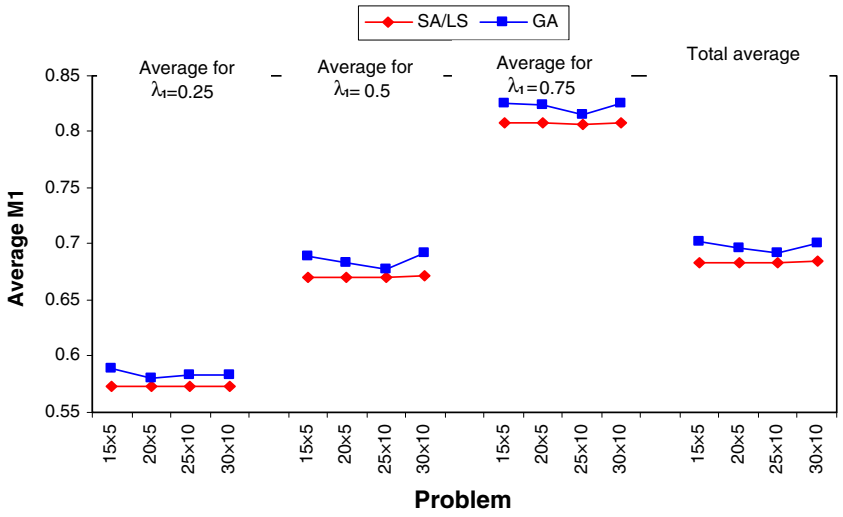

Fig. 8 Mean values of M_1 for both of algorithms by increasing the number of jobs and stages

4.4 Results from comparisons of SA/LS and genetic algorithm

This section of paper shows the performance of our algorithm against the other algorithm provided in the literature. The performance of the proposed SA/LS is compared with a specific genetic algorithm proposed by Jungwattanakit et al. [13]. The algorithms have been coded by Matlab7 which is a special mathematical computation language and ran on a PC-800 with 512 MB memory. It is notice that three replications for each problem size have been performed since there are some random conditions when applying the algorithm. Also, a weight vector is needed to aggregate the k objectives into a scalar function and different weights vectors (λ_1) are used to define different directions of search. Therefore, instances tested with $\lambda_1 \in \{0.25,0.5,0.75\}$ in the objective functions.

The comparison between two algorithms is performed based on CPU time, four performance measures (metrics), convex combinations yielded from each algorithm after several runs, archived solutions (an archive is created, maintained during successive iterations to preserve solution identified during the search), number of solutions processed by each algorithm, and the effect of "max_it" on the time and the improvement of the solutions.

Time cost is an important factor when comparing different algorithms. Hence, Fig. 4 plots the computational times of two algorithms as regards problem size of three runs. It is presented in Fig. 4 that our method runs faster in the cases of

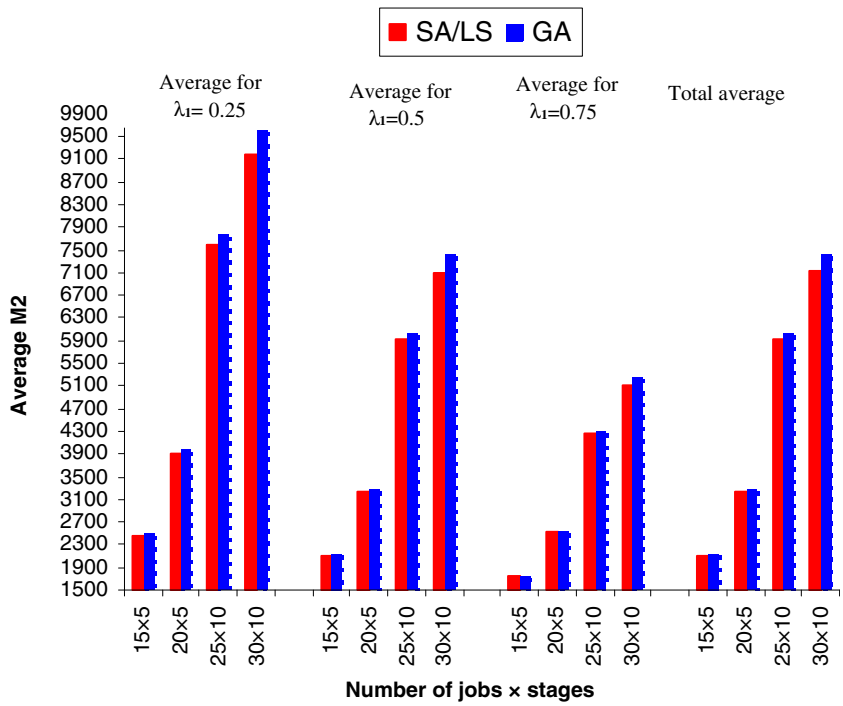

Fig. 9 Mean values of M_2 for both of algorithms by increasing the number of jobs and stages

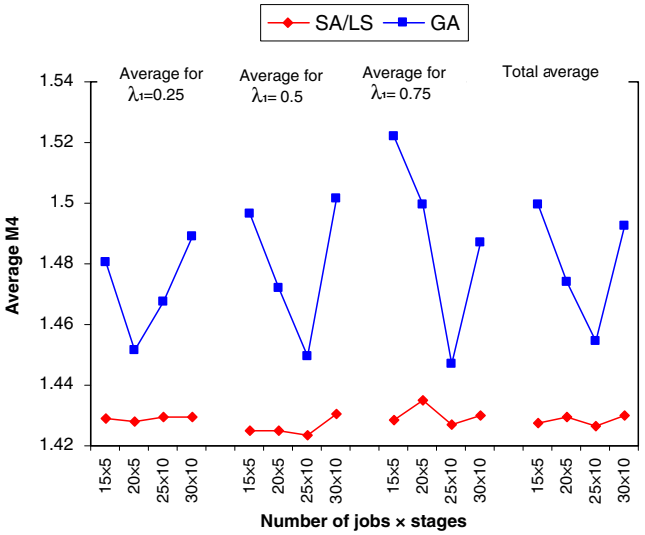

Fig. 10 Mean values of M_4 for both of algorithms by increasing the number of jobs and stages

15×5, 20×5, and 25×10, but in case of 30×10, the running time of SA/LS is higher than the genetic algorithm's one.

Tables 3, 4, 5 and 6 represent the convex combinations yielded by each algorithm after three replications. In this section, the algorithms are analyzed in terms of introduced metrics, and then, the problem size effects (numbers of stages and jobs), on the performance of them.

The comparison between the performance of the SA/LS and the genetic algorithm in terms of introduced metrics is given in Tables 7, 8, and 9.

Based on the results given in tables, the following observations can be made:

SA/LS algorithm is able to outperform the other algorithm in all test problems by considering "M_1", "M_2", and "M_4" metrics. To clarify, we plot the obtained values from "M_1" and "M_4" metrics in Figs. 5 and 6, respectively. These figures clearly show that the "SA/LS" has a better performance than the other algorithm, because the obtained values of metrics by

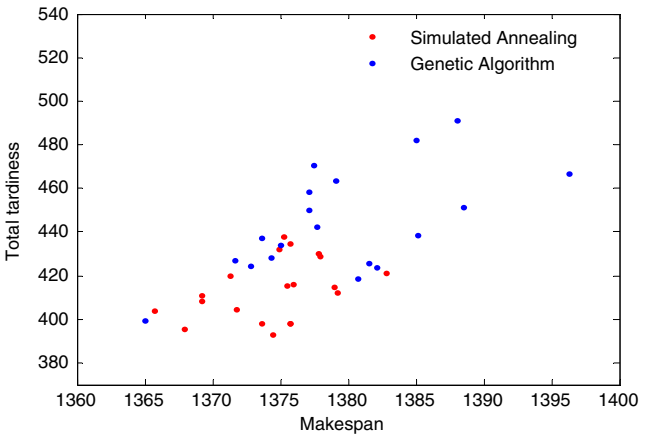

Fig. 11 The generated convex combinations by SA/LS and genetic algorithm for T1 problem (λ_1=0.25)

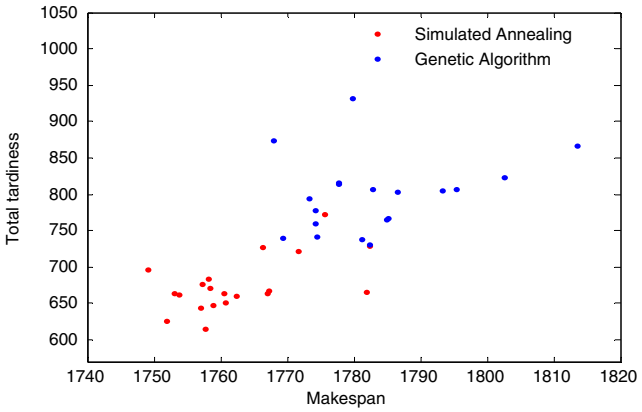

Fig. 12 The generated convex combinations by SA/LS and genetic algorithm for T6 problem (λ_1=0.5)

SA/LS are situated below the values obtained by the genetic algorithm's one for all the problem sizes and in each search direction (λ_1). It is note that smaller values of "M_1", "M_2", and "M_4" metrics indicate a better solution. The obtained values of "M_3" metric indicate that the solutions are generated in the trade-off surface by both of algorithms.

Based on the results provided in the tables, comparison of the SA/LS and the other algorithm through combination of"M_3" (angle) and "M_4" (distance) metrics are presented in Fig. 7. This figure shows the angle and the distance of solutions generated by SA/LS and the genetic algorithm simultaneously. It can be seen that "SA/LS" generates a more efficient convex combination than the other algorithm.

To analyze the behavior of different algorithms in the different situations, we plot the average of "M_1", "M_2", and "M_4" for the algorithms in different levels of the number of jobs and stages based on $\lambda_1 \in \{0.25,0.5,0.75\}$ in Figs. 8, 9, and 10. These figures show that in all cases, SA/LS works better than the other algorithm in comparison, since the average of metrics obtained by SA/LS are situated below the obtained values of the genetic algorithm for all problem sizes and each search direction. Thus, SA/LS shows better performance even by increasing in number

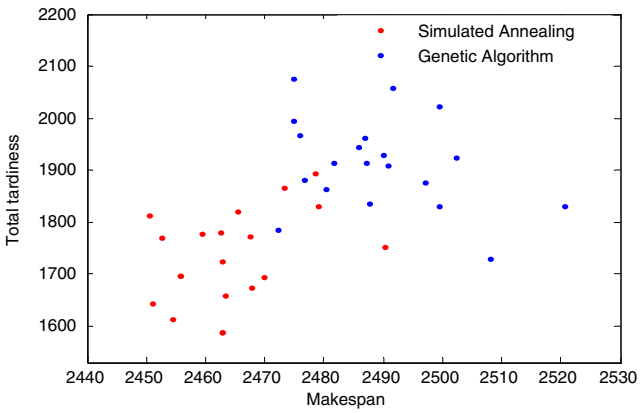

Fig. 13 The generated convex combinations by SA/LS and genetic algorithm for T11 problem (λ_1=0.75)

of jobs and stages. Also, figures show that changing in the value of λ_1 had no significant effect in the overall relative performance of the algorithms.

In this section, the performances of SA/LS and genetic algorithm are compared via diagrams: Figs. 11, 12, and 13. These figures show several convex combinations generated by SA/LS and the genetic algorithm through 20 runs. It can be found from these figures that the proposed algorithm yields the efficient solutions, in comparison with genetic algorithm. By considering "M_3" and "M_4" metrics (see Fig. 7, and compare with Figs. 11, 12, and 13) we conclude that generated points in SA/LS are better than the solutions of the genetic algorithm. Therefore, SA/LS is more effective in minimizing the makespan and total tardiness for the hybrid flowshop problem with sequence-dependent setup times than the genetic algorithm proposed by Jungwattanakit et al.[13].

In this section, we are going to demonstrate a comparison between the archived solutions of SA/LS and genetic algorithm.

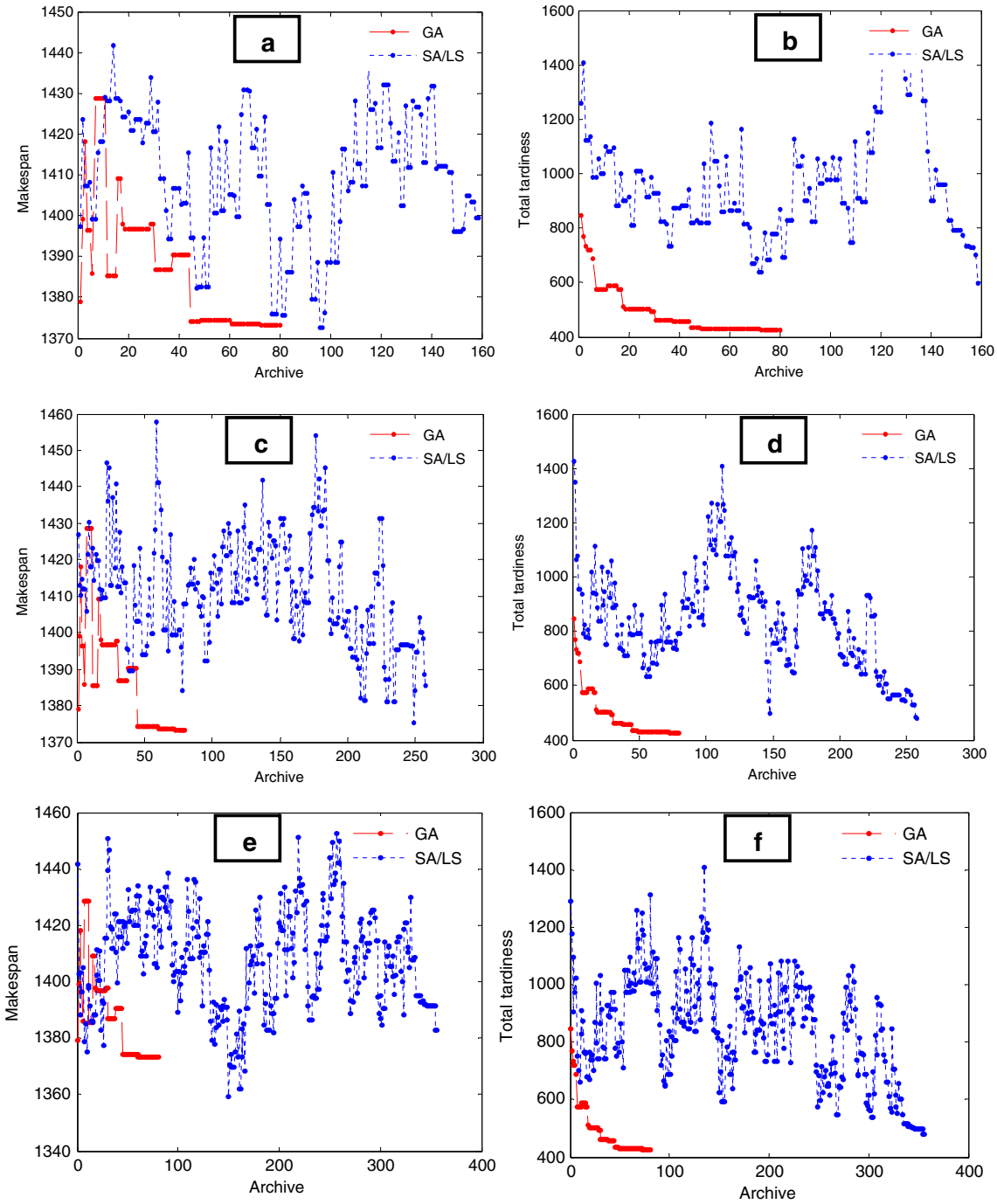

Fig. 14 Objective values of archived solutions on T1 problem (when max-it=1,2,3 and λ_1=0.25)

Here the problem T1 is under consideration where λ_1=0.25 (direction search).

In each experiment, the genetic algorithm processes on "S1" solutions where "S1 = (number of generation) × (population size)" regardless of kind of problem. The values of these parameters are: "population size=100" and "number of generation=80". Generally, to solve the problem "T1", genetic algorithm takes approximately about 8,000 solutions under process in 54.1265 s time.

In each experiment, the SA/LS processes on "S2" solutions where "$S2 = (n-1) \times (\max\_it) \times N$" that "number of stages for reach of T_0 to T_f (N)=55" and (n−1) depends on the problem size (n=15 in the problem T1). This algorithm also has another parameter defined as "maximum number of

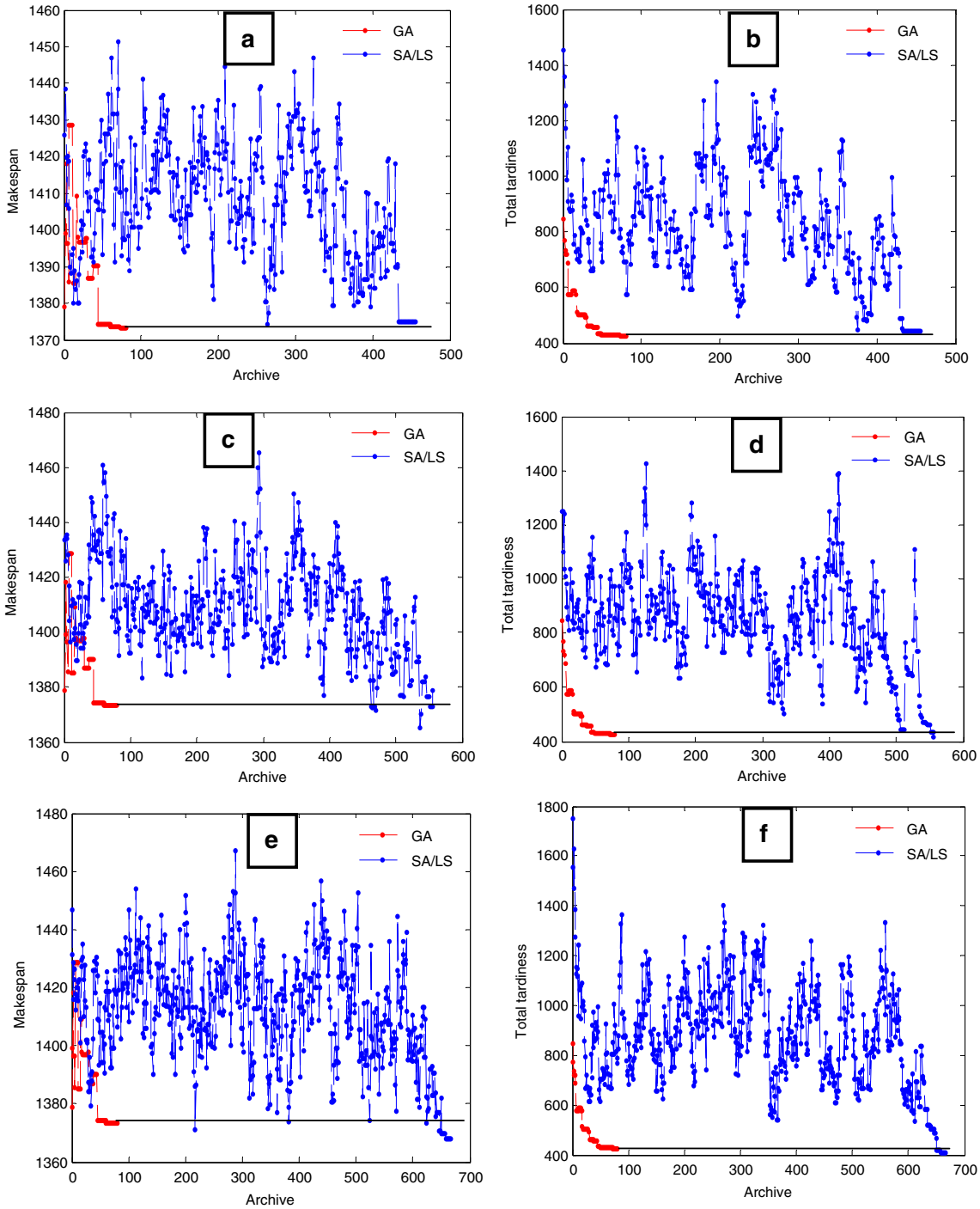

Fig. 15 Objective values of archived solutions on T1 problem (when max-it=4,5,6 and λ_1=0.25)

iterations in each temperature" which is denoted by "max_it". It is notice that the more the value of max_it the more the basic improvement procedure—designed by authors—is performed. Here the effect of max_it is shown on the performing time and the improvement of the solutions. Therefore, it is assumed that max_it is a variable in range [1–6].

The genetic algorithm and SA/LS (with max_it=1 to 6) is performed on problem T1 and the solutions are archived and shown in Figs. 14 and 15. In Fig. 14, plots ("*a*" and "*b*"), ("*c*" and "*d*"), and ("*e*" and "*f*") are related to max_it=1, 2, and 3, respectively. Also in Fig. 15 plots ("*a*" and "*b*"), ("*c*" and "*d*"), and ("*e*" and "*f*") are related to max_it=4, 5, and 6, respectively. It is notice that the plots "*a*", "*c*", and "*e*" in both Figs. 14 and 15 is related to makespan and the plots "*b*", "*d*", and "*f*" is related to total tardiness.

First, we consider plots "*a*" and "*b*" in Fig. 14 where max_it=1. SA/LS algorithm, to solve the problem "T1" (n= 15), takes approximately about 770 solutions under process in 8.4844 s time. It should be noticed that SA/LS algorithm concludes the process in a shorter time through searching less solutions. It can be seen that in plots "*a*" and "*b*" in Fig. 14, the algorithm SA/LS has tried to provide the proper solutions but it is not as strong as the genetic algorithm. Hence the results obtained by genetic algorithm are more effective.

First, the plots "*c*" and "*d*" in Fig. 14 are considered where max_it=2. In this case, our proposed algorithm tried 1,540 solutions in 16.0938 s. It can be seen that our proposed algorithm has found solutions that are almost near to the genetic algorithm's solutions but cannot compete with them, yet.

Plots "*e*" and "*f*" in Fig. 14 are considered where max_it= 3, result was 2,310 solutions in 22.1250 s. It is conceived that the solutions provided by our algorithm is more near to genetic algorithm's solutions but not as perfect as them.

In following, the plots "*a*" and "*b*" in Fig. 15 are considered where max_it is 4. Our algorithm tried 3,080 solutions in 28.9688 s. Based on the figure, it implies that the solutions provided by our algorithm are as good as genetic algorithm solutions.

As a result, the figures are showing that our proposed algorithm has produced solutions as effective as genetic algorithm solutions but by considering the time factor, the SA/LS algorithm is faster than genetic algorithm. Then by increasing in "max_it", it can be seen that our algorithm not only takes shorter time but also produced better solutions. The given plots, depict this claim: in plots "*c*" and "*d*" of Fig. 15 where max_it= 5, produced 3,850 solutions in 36.3594 s and in plots "*e*" and "*f*" 4,620 solutions in 42.9531 s. Therefore, as max_it has been grown, the performance of SA/LS has been improved.

5 Conclusion and future study

This paper is focused on the hybrid flowshop sequence-dependent setup times scheduling problems which belongs to NP-hard class. Our objective is to determine the schedule that minimizes a convex combination of the makespan and the total tardiness. The problem is configured as a bi-criteria model which is a sub-class of the multi-criteria models. For the optimization problem, simulated annealing is proposed. In order to enhance the performance of the simulated annealing, two simple forms of local search are applied on the best solution of archive. Therefore, combination of the simulated annealing and local search algorithms are used to solve the multi-objective problems. The performance evaluation of the SA/LS under study is carried out by using several test problems appropriately designed for this particular problem. The performance of SA/LS has been evaluated by comparing its solution with the obtained solution by the proposed genetic algorithm in the literature. For determining the analogy results; four metrics are used to evaluate the quality of the solutions. The first and the second metrics measure the performances of the algorithms by the functions applied in each algorithm. The third metric measures the angle of slope in relation to the origin. The fourth metric measures the distance of the obtained solutions and the origin. Comparison shows that the "SA/LS" provides better results than the proposed genetic algorithm in the literature by considering the results of three metrics out of four. The author's future study includes development of the other bi-objective algorithms of the simulated annealing by designing as much as possible functions to combine the various criteria into a single scalar value.

References

1. Daniels RL, Chambers RJ (1990) Multi-objective flow-shop scheduling. Nav Res Logist 37(6):981–995
2. Ishibuchi H, Murata T (1998) A multi-objective genetic local search algorithm and its application to flow shop scheduling. IEEE Transaction on Systems and Manufacturing Cybernetics 28 (3):392–403
3. Sayin S, Karabati S (1999) A bi-criteria approach to the two-machine flow shop scheduling problem. Eur J Oper Res 113 (2):435–449
4. Chakravarthy K, Rajendran C (1999) A heuristic for scheduling in a flow shop with the bi-criteria of makespan and maximum tardiness minimization. Production Planning and Control 10(7):707–714
5. Allahverdi A (2001) The tricriteria two-machine flow shop scheduling problem. International Transaction on Operational Research 8(4):403–425
6. Lee WC, Wu CC (2001) Minimizing the total flow time and the tardiness in a two-machine flow shop. Int J Syst Sci 32(3):365–373
7. Chang PC, Hsieh JC, Lin SG (2002) The development of gradual priority weighting approach for the multi-objective flowshop scheduling problem. Int J Prod Econ 79(3):171–183
8. Toktas B, Azizoglu M, Koksalan SK (2004) Two-machine flow shop scheduling with two criteria: maximum earliness and makespan. Eur J Oper Res 157(2):286–295
9. Ponnambalam SG, Jagannathan H, Kataria M, Gadicherla A (2004) A TSP-GA multi-objective algorithm for flow shop scheduling. Int J Adv Manuf Technol 23(11–12):909–915

10. Ravindran D, Noorul Haq A, Selvakuar SJ, Sivaraman R (2005) Flow shop scheduling with multiple objective of minimizing makespan and total flow time. Int J Adv Manuf Technol 25(9–10):1007–1012
11. Loukil T, Teghem J, Tuyttens D (2005) Solving multi-objective production scheduling problems using metaheuristics. Eur J Oper Res 161(1):42–61
12. Rahimi-Vahed AR, Mirghorbani SM, Rabbani M (2007) A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem. Soft Computing 11(10):997–1012
13. Jungwattanakit J, Reodecha M, Chaovalitwongse P, Werner F (2007) Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. Int J Adv Manuf Technol 37(3–4):354–370
14. Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187(3):985–1032
15. Kurz ME, Askin RG (2003) Scheduling flexible flow lines with sequence-dependent setup times. Eur J Oper Res 159(1):66–82
16. Kirkpatrik S (1983) Optimization by simulated annealing. Science 220(4598):671–680
17. Prandtstetter M, Raidl GR (2007) An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. Eur J Oper Res 191(3):1004–1022
18. Varadharajan TK, Rajendran C (2005) A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. Eur J Oper Res 167(3):772–795
19. Eren T, Güner E (2008) The tricriteria flowshop scheduling problem. Int J Adv Manuf Technol 36(11):1210–1220