Computers & Operations Research 61 (2015) 81–88

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Iterated local search based on multi-type perturbation for single-machine earliness/tardiness scheduling

Tao Qin [a], Bo Peng [a], Una Benlic [b], T.C.E. Cheng [c], Yang Wang [a], Zhipeng Lü [a,c,*]

[a] SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, PR China
[b] Computing Science and Mathematics School of Natural Sciences, University of Stirling, Stirling, UK
[c] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

ARTICLE INFO

Available online 16 March 2015

Keywords:
Single machine
Iterated local search
Multi-type perturbation
Tabu search

ABSTRACT

We propose an iterated local search based on a multi-type perturbation (ILS-MP) approach for single-machine scheduling to minimize the sum of linear earliness and quadratic tardiness penalties. The multi-type perturbation mechanism in ILS-MP probabilistically combines three types of perturbation strategies, namely tabu-based perturbation, construction-based perturbation, and random perturbation. Despite its simplicity, experimental results on a wide set of commonly used benchmark instances show that ILS-MP performs favourably in comparison with the current best approaches in the literature.

1. Introduction

Due to its theoretical challenge and practical relevance, single-machine earliness/tardiness scheduling has attracted considerable research attention of the scheduling community. In particular, it is related to the popular just-in-time (JIT) production philosophy and the practice of supply chain management. JIT production advocates that goods are produced only when they are needed, so both early and tardy deliveries are undesirable, which incur earliness and tardiness penalties, respectively. Thus, the ideal production schedule is one under which all the jobs are completed exactly on their due dates. Earliness/tardiness scheduling is also compatible with the practice of supply chain management, which emphasizes the coordination of material flows from suppliers to customers with a view to improving overall supply chain performance and providing better customer service. In general, the results and insights gained from single-machine scheduling research are applicable to more complex production environments such as the flow shop and job shop settings.

In this paper we consider the single-machine scheduling problem to minimize the sum of linear earliness and quadratic tardiness penalties (SMSP-LEQT), which is formally defined as follows: a set $\{J_1, J_2, \ldots J_n\}$ of independent jobs has to be scheduled on a single machine, which is continuously available from time zero onwards without idle time. The machine can process at most one job at a time and no preemption is allowed. Each job $J_j, j = 1, 2, \ldots, n$ has a processing time p_j and should be ideally completed on its due date d_j. The earliness and tardiness of job J_j are defined as $E_j = \max\{0, d_j - C_j\}$ and $T_j = \max\{0, C_j - d_j\}$, respectively, where C_j is the completion time of J_j. For SMSP-LEQT, we wish to find a schedule that minimizes $\sum_{j=1}^{n}(E_j + T_j^2)$ [15].

Similar single-machine scheduling problems with related objective functions have been studied in the literature. These include minimization of the sum of linear earliness and tardiness penalties, i.e., $\sum_{j=1}^{n}(E_j + T_j)$ [4,11,16], and minimization of quadratic lateness, i.e., $\sum_{j=1}^{n} L_j^2$ [8,12,14,17,18], where the lateness L_j of J_j is defined as $L_j = C_j - d_j$.

A generalization of the NP-hard single-machine scheduling problem to minimize the weighted tardiness [9], SMSP-LEQT is NP-hard. Both heuristic and exact methods have been applied to tackle this problem. Valente [20] presents a lower bounding procedure based on relaxation of the jobs' completion times and a branch-and-bound algorithm that adopts the proposed lower bound, together with an insertion-based dominance test. Schaller [16] introduces several branch-and-bound procedures for the problem, which are developed by incorporating a tighter lower bound and examining two dominance conditions. Among the heuristic approaches are the dispatching heuristics by Valente [19], which consider linear early/quadratic tardy dispatching rules, as well as a greedy-type procedure. Valente [21] proposes several heuristic algorithms based on the beam search technique. These include the classical beam search procedure, with both priority and total cost evaluation functions, as well as the filtered and recovering variants. Valente and Gonçalves [22] suggest several genetic algorithms based on a random key alphabet, which differ in the generation of the initial population and in the use of local search. Valente and Schaller [23] present some other heuristics for

* Corresponding author at: SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, PR China.
E-mail addresses: ube@cs.stir.ac.uk (U. Benlic), Edwin.Cheng@polyu.edu.hk (T.C.E. Cheng), zhipeng.lv@hust.edu.cn (Z. Lü).

http://dx.doi.org/10.1016/j.cor.2015.03.005
0305-0548/

the considered problem, for both versions with and without idle time.

In this paper we propose a heuristic algorithm for SMSP-LEQT that follows the general framework of Iterated Local Search (ILS) [10] and adopts multi-type perturbation from Breakout Local Search (BLS) [1–3]. In essence, the basic idea of our ILS approach based on multi-type perturbation (ILS-MP) is to use local search to find local optima and apply a multi-type perturbation mechanism that probabilistically selects among three types of perturbation strategies that introduce different degrees of diversification to the search space. While one of the perturbation types is based on random moves, the other two are based on the principles that underpin the Tabu Search (TS) [5–7] and Greedy Randomized Adaptive Search Procedure (also known as GRASP [13]) metaheuristics.

We summarize the merits of the proposed ILS-MP algorithm as follows:

- Compared with the state-of-the-art algorithms in the literature for solving SMSP-LEQT, such as MA_IN, ILS-MP is conceptually simple and easy to implement. Indeed, while the reference algorithms often apply hybrid or evolutionary methods, ILS-MP is based on a simple local search working in conjunction with three dedicated perturbation operators.
- ILS-MP achieves better performance compared with the reference algorithms in the literature in terms of both solution quality and computational efficiency in many cases.

The rest of the paper is organized as follows: in Section 2 we describe in detail the proposed ILS-MP for SMSP-LEQT. We provide extensive computational evaluations and comparisons of ILS-MP with the current state-of-art approaches for SMSP-LEQT in Section 3. Finally, we conclude the paper in Section 4.

2. Iterated local search based on multi-type perturbation

2.1. General procedure

Following the general framework of ILS, our ILS-MP alternates iteratively between a local search phase (to reach local optima) and a dedicated perturbation phase (to discover new promising search spaces). Specifically, starting with an initial solution S_0, ILS-MP applies the steepest descent procedure (see Section 2.3) to reach a local optimum S. As explained in Section 2.3, each iteration of the steepest descent performs an exhaustive search in one of the randomly selected neighbourhoods (an insert neighbourhood or a swap neighbourhood), and selects the best-improving neighbouring solution. If such a solution does not exist in the current neighbourhood, local optimality is reached. At this point, ILS-MP triggers its multi-type perturbation mechanism, which first probabilistically selects among three different types of perturbation moves (tabu-based, construction-based, or random), and then applies a given number of moves of the selected type to the current local optimum S. This perturbed solution becomes a new starting point for the next phase of the descent. The best overall solution S_{best} is kept as the result. The proposed algorithm terminates once the number of consecutive local search phases with no improvement of S_{best} reaches a certain threshold *stop_iter*.

The general framework of the ILS-MP algorithm is outlined in Algorithm 1. The following sections detail the main algorithmic components of ILS-MP.

Algorithm 1. ILS-MP for SMSP-LEQT.

```
1:  Generate an initial solution S' /* See Section 2.2 */
2:  S_best = S'
3:  iter_no_improv ← 0
4:  while iter_no_improv < stop_iter do
5:    S ← LocalSearch(S') /* See Section 2.3 */
6:    if S is better than S_best then
7:      S_best ← S
8:      iter_no_improv ← 0
9:    else
10:     iter_no_improv ← iter_no_improv + 1
11:   end if
12:   if (rand(0, 0.1, ..., 1) < P) /*P ∈ [0, 0.01, ..., 1] is a coeff.*/
      then
13:     S' = TabuBasedPerturbation(S, L1)    /* See Section 2.5 */
14:   else if (rand(0, 0.01, ..., 1) < (1 − P) · Q) /*Q ∈ [0, 0.01, ..., 1]
      is a coeff.*/ then
15:     S' = ConstructionBasedPerturbation(S, L2) /* See Section
      2.5 */
16:   else
17:     S' = RandomPerturbation(S, L3)    /* See Section 2.5 */
18:   end if
19: end while
20: return S_best
```

2.2. Initial solution

Given an unordered set of n independent jobs $S=\{J_1,J_2,\ldots,J_n\}$, ILS-MP uses a greedy heuristic procedure to generate a starting point for the search, i.e., an ordered set S' of n jobs. Initially, $S'=\varnothing$. Starting from scratch, we randomly select a job J_i from S, let $S=S-\{J_i\}$, and insert J_i into the best position in S' according to the evaluation strategy (see Section 2.4). Then, $S'=S'\cup\{J_i\}$. The above procedure is repeated until $S=\varnothing$.

Note that we carried out additional experiments by using the NEH heuristic to generate the initial solution. However, the results show that there is no significant difference in terms of the best results found. The reason might lie in the fact that ILS-MP makes use of several diversification mechanisms to diversify the search when it gets stuck in a local optimum trap. So the performance of ILS-MP does not rely much on the quality of the initial solution. This is also one of the advantages of ILS-MP.

2.3. Neighbourhood relations and their exploitation

To move from one solution to another in the search space, the local search procedure of ILS-MP employs two different neighbourhood relations:

Insert: A job $J_i\in S$ $(1\le i\le n)$ is removed from its current position in the schedule S and inserted immediately before (forwards) or after (backwards) another job J_k $(k\neq i)$, thus producing a total of $n-1$ possible positions (i.e., solutions). Examples of insertion forwards and backwards are provided in Figs. 1 and 2, respectively.

Swap: Two jobs swap their positions, which generates a total of $n(n-1)/2$ possible solutions. An example of a swap move is given in Fig. 3.

To reduce the size of the neighbourhood, the local search in ILS-MP algorithm applies distance restrictions for both insert and swap moves. The distance for an insert move is the absolute

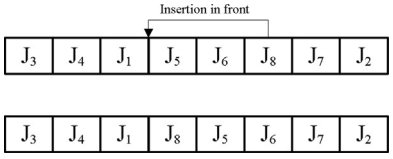

Fig. 1. Insertion forwards. J_8 is inserted immediately before J_5.

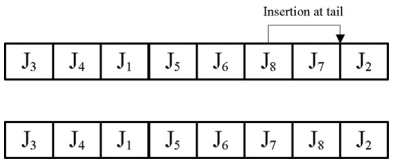

Fig. 2. Insertion backwards. J_8 is inserted immediately after J_7.

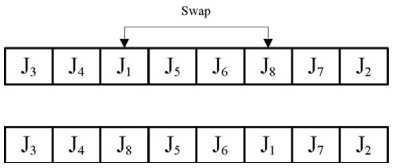

Fig. 3. J_8 and J_1 swap positions.

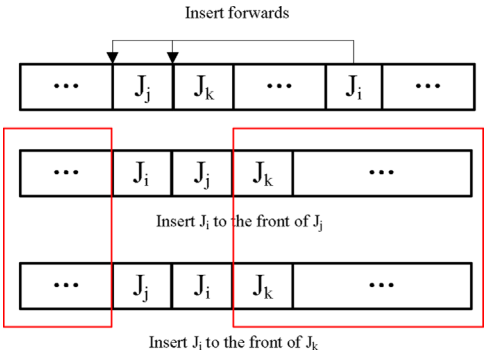

Fig. 4. Insert evaluation (forwards).

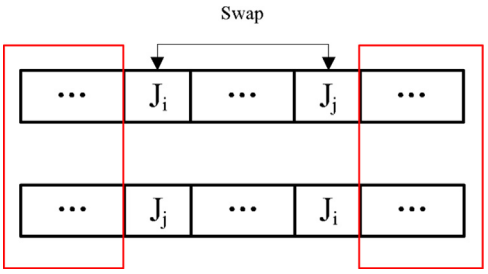

Fig. 5. Swap evaluation.

difference (i.e., the number of jobs) between the current job position and the new job position. The distance for a swap move is defined as the number of jobs between the positions of the first and the second job to be swapped. We thus introduce a distance threshold for the insert and the swap moves, denoted as *threshold* (*insert*) and *threshold*(*swap*), respectively. Any move with a distance beyond this threshold is not considered. This is due to the fact that moving a job to a position that is far from its current position may not be meaningful, and swapping two jobs that are a long distance apart would not significantly improve solution quality. These move restrictions help us to save a large amount of CPU time without sacrificing much of solution quality.

To exploit the above defined neighbourhoods, ILS-MP employs a simple best-improvement local search. In each iteration, this procedure first randomly chooses between the insert and swap neighbourhoods with a given probability λ, and then determines the best-improving solution from the selected neighbourhood. The local search phase terminates once a locally optimal solution is reached, i.e., no improving solution exists in the chosen neighbourhood of the current solution.

2.4. Fast neighbourhood evaluation strategy

In order to efficiently evaluate the costs of the above defined insert and swap moves, we employ a fast evaluation strategy to calculate the incremental value of each move.

For example, we move job J_i immediately before (forwards) J_j or J_k $(j,k<i)$ in Fig. 4. Comparing the two moves, the positions of the jobs marked in a box are the same, so we just need to consider the difference between J_i and J_k as the changed objective value with the two neighbourhood moves. Given that the objective value is f when job J_i is inserted immediately before J_j while becoming f' after being inserted immediately before J_k, then we can calculate f' using the formula $f'=f+(P'_i-P_i)+(P'_j-P_j)$, where P_i and P'_i denote the penalty values of job J_i for the two different moves described in Fig. 4 (likewise for P_j and P'_j). Afterwards, making use of the formula iteratively, we can quickly calculate the other new penalty values when J_i is moved immediately before other jobs. Likewise, the evaluation results of different backwards moves of job i immediately after other jobs can be iteratively calculated in a similar manner.

Furthermore, considering the swap move involving J_i and J_j as shown in Fig. 5, we can easily find that only the jobs whose positions range from i to j have changed. Hence the total changed penalty values of the jobs between i and j need to be re-calculated. Assuming that the objective value without the swap move is f, then the new objective value after the swap move can be calculated by the formula $f'=f+\sum_{l=i}^{j}(P'_l-P_l)$, where P_l stands for the original penalty of J_i while P'_l presents its new penalty after the swap move.

2.5. Multi-type perturbation phase

As mentioned above, ILS-MP probabilistically selects among three types of perturbation moves, namely tabu-based, construction-based, and random.

Tabu-based perturbation is basically a standard tabu search procedure [5–7] that consists of a certain number of iterations. Each of these iterations works essentially like the local search procedure, with the exception that tabu moves are added (along with an aspiration criterion) and we move to the best non-tabu neighbour even if it is worse than the current solution. It may be effectively used for the purpose of perturbation since the classical tabu search ensures a certain degree of diversification in each iteration by means of a tabu list. This perturbation directs the search towards promising search spaces by using a selection rule that favours moves that minimize cost degradation, under the constraint that the move is prohibited by the tabu list. The proposed perturbation probabilistically explores both the insert and swap neighbourhoods, i.e., in the same way as during the local search phase (see Section 2.3). Each time a job is moved from its current position i to position j, it is forbidden to be placed back to i for γ iterations, where γ is determined according to the values *threshold*(*insert*) (in case of an insert move) and *threshold*(*swap*) (in case of a swap move) as follows:

$$\gamma_{insert}=\alpha_1\cdot threshold(insert)+random(\alpha_2\cdot threshold(insert)),$$
$$\gamma_{swap}=\alpha_1\cdot threshold(swap)+random(\alpha_2\cdot threshold(swap)), \quad (1)$$

where α_1 and α_2 are coefficients, and function $random(\alpha_2\cdot threshold(insert\ or\ swap))$ means to generate a random number between 0 and $\alpha_2\cdot threshold(insert\ or\ swap)$. The tabu status of a move is neglected only if the move leads to a new solution better than the best solution found so far.

Construction-based perturbation begins by removing a randomly chosen job J_r from the current schedule S_c. Then the procedure

sorts all the possible insertion positions for J_r in an increasing order of the objective function (penalty) values. Finally, J_r is inserted into one of the possible positions in an adaptive and random way, i.e., according to the penalty scores assigned to the positions – the smaller its penalty value is, the higher the chance a position is chosen. For this purpose, the kth best position is selected with probability $VF_i(j) = (1/k)\sum_{k=1} 1/k$, where j represents the index of the considered position for J_r and k is the rank assigned to position j in the sorted order.

Random perturbation consists of performing a randomly selected insert or swap move.

It is evident that these three perturbation types introduce increasingly larger degrees of diversification to the search space. Tabu-based perturbation is the weakest (i.e., introducing the smallest degree of diversification to the search space) because it takes solution quality into consideration by avoiding too much deterioration of the current solution quality. On the other hand, random perturbation is the strongest (i.e., introducing the largest degree of diversification to the search space).

ILS-MP selects among the three perturbation types in a probabilistic way. The weakest tabu-based perturbation is selected with a probability P, while the construction-based and random perturbations are selected with probabilities $(1-P)\cdot Q$ and $(1-P)\cdot(1-Q)$, respectively. Once the type of move is selected, we apply L moves of the corresponding perturbation to the current local optimum S_c. The value L depends on the chosen perturbation ($L = L_1$ for tabu-based perturbation, $L = L_2$ for construction-based perturbation, and $L = L_3$ for random perturbation).

3. Experimental results

3.1. Benchmark instances and experimental protocol

We conducted extensive experimental evaluations and comparisons of the proposed ILS-MP algorithm on a wide range of commonly used benchmark instances of SMSP-LEQT with 10, 15, 20, 25, 30, 40, 50, 75, and 100 jobs. We randomly generated the instances in the following way. Each job J_j was assigned an integer processing time p_j from one of the two uniform distributions, $[45, 55]$ and $[1, 100]$, so as to obtain a low (L) and a high (H) variability, respectively, for the processing times. For each job J_j, an integer due date d_j was generated from the uniform distribution $[P(1-T-R/2), P(1-T+R/2)]$, where P is the sum of the processing times of all the jobs, T is the tardiness factor, which took values 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0, and R is the range of the due dates, which was set to 0.2, 0.4, 0.6, and 0.8. A subset of 50 instances was randomly generated for each combination of problem size (i.e., number of jobs n), processing time variability (Var), T, and R. This makes a total of 1200 instances for each combination of problem size and processing variability. The complete benchmark set is available at http://fep.up.pt/docentes/jvalente/benchmarks.html. This website also provides the optimal objective function values (when available), as well as the objective values obtained by several state-of-art approaches for SMSP-LEQT (MA_IN [22], RBS [21], and EQTP [19]).

We coded ILS-MP in CodeBlocks and ran it on an Intel (R) Celeron(R) G530-2.40 GHz personal computer, while the other reference algorithms, i.e., EQTP, RBS, GA, GA_IN, MA, and MA_IN, were all run on a Pentium IV-2.8 GHz personal computer [19,21,22]. It is noted that both machines are comparable in speed although our machine is a little bit slower. Hence, we just report the actual computing time without time conversion. For each instance, we performed 10 independent runs of ILS-MP. The parameter settings of ILS-MP used in the experiments are given in Table 1. In fact, we determined the adequate values of the various parameters as follows: we first used a set of preliminary values for the parameters. Then, we tuned the value of one parameter at a time until the results could not be improved. We repeated this process until the tuning of any parameter did not yield better performance. After determining the adequate values of the parameters for one typical instance, we used the same parameter values to test all the benchmark instances.

To evaluate the performance of the proposed approach, we performed two comparisons that follow the same experimental procedure and presentation of results as in [22]. The first one, provided in Section 3.2, compares our ILS-MP algorithm with the current best performing heuristics for SMSP-LEQT. The second comparison, presented in Section 3.3, is with the optimal results.

3.2. Comparison with existing heuristics

In this section we discuss the performance of ILS-MP for SMSP-LEQT in comparison with existing algorithms, namely the beam search heuristic (RBS) [21] and four population-based heuristic procedures [22] (i.e., two genetic algorithms GA & GA_IN and two memetic approaches MA & MA_IN), which differ in the generation of the initial solution and the use of local search. MA_IN is the current best performing approach (in terms of solution quality) for SMSP-LEQT.

Table 2 shows the comparison results of GA, GA_IN, MA, MA_IN, and ILS-MP with RBS. For this comparison, the evaluation criterion is the best, average, and worse relative percentage improvement over the RBS, calculated as $\%\text{imp} = (\text{OFV}_{RBS} - \text{OFV})/\text{OFV}_{RBS} \times 100$, where OFV_{RBS} and OFV are the objective function values of RBS and the given heuristic, respectively. For each instance, the provided results are based on 10 independent executions of the corresponding algorithm. The best results are provided in bold. From Table 2, we observe that ILS-MP clearly exhibits better performance than RBS, and provides in many cases equal or slightly better results than MA and MA_IN. Indeed, the relative average improvement of ILS-MP over RBS is about 0.78%

Table 1
Parameter values.

Parameter	Description	Value
stop_iter	Number of iter. without improvement (stopping condition)	50
threshold(*insert*)	Distance threshold for insert move	$n/2$
threshold(*swap*)	Distance threshold for swap move	$n/3$
λ	Prob. of applying insert over swap move	0.5
L_1	Number of moves for tabu-based perturbation	10
L_2	Number of moves for construction-based perturb.	4
L_3	Number of moves for random perturb.	$n/3$
α_1	Coefficient for tabu-based perturb.	0.5
α_2	Coefficient for tabu-based perturb.	0.9
P	Prob. of applying tabu-based perturb.	0.75
Q	Prob. of applying construction-based over random perturb.	0.5

Table 2
Comparison with RBS – relative improvement.

n	Heur.	Low. var %imp			High var %imp		
		Best	*Avg.*	*Worst*	*Best*	*Avg.*	*Worst*
10	GA	0.02	−0.18	−1.58	0.36	−0.31	−2.42
	GA_IN	0.02	−0.01	−0.10	0.35	−0.17	−1.48
	MA	0.02	0.01	−0.09	0.37	0.35	0.28
	MA_IN	0.02	0.02	0.02	0.37	0.37	0.35
	ILS-MP	0.02	0.02	0.02	0.37	0.37	**0.37**
20	GA	0.11	−0.26	−1.74	0.36	−0.96	−3.95
	GA_IN	0.10	0.05	−0.03	0.51	0.17	−1.33
	MA	0.11	0.07	−0.13	0.66	0.50	0.02
	MA_IN	0.11	0.11	0.08	0.66	0.60	0.45
	ILS-MP	0.11	0.11	**0.11**	0.66	**0.65**	**0.60**
30	GA	0.22	−0.20	−1.42	0.35	−0.96	−4.07
	GA_IN	0.23	0.18	0.11	0.71	0.14	−0.67
	MA	0.25	0.21	0.03	0.98	0.76	0.23
	MA_IN	0.25	0.24	0.22	0.98	0.89	0.71
	ILS-MP	0.25	**0.25**	**0.24**	**0.99**	**0.96**	**0.88**
40	GA	0.43	−0.03	−1.49	0.59	−0.70	−3.40
	GA_IN	0.45	0.40	0.33	1.08	0.56	−0.18
	MA	0.48	0.44	0.37	1.45	1.23	0.81
	MA_IN	0.48	0.46	0.44	1.45	1.36	1.17
	ILS-MP	0.48	**0.48**	**0.46**	**1.48**	**1.44**	**1.34**
50	GA	0.42	−0.05	−1.63	0.68	−0.41	−2.48
	GA_IN	0.44	0.40	0.33	1.27	0.80	0.17
	MA	0.47	0.45	0.39	1.67	1.45	1.06
	MA_IN	0.47	0.46	0.44	1.69	1.59	1.42
	ILS-MP	0.47	**0.47**	0.44	**1.71**	**1.66**	**1.50**
75	GA	0.63	0.14	−1.15	1.00	−0.05	−2.01
	GA_IN	0.71	0.67	0.61	1.76	1.40	0.94
	MA	0.74	0.72	0.67	2.14	1.94	1.66
	MA_IN	0.74	0.73	**0.72**	2.18	2.09	1.96
	ILS-MP	**0.75**	0.73	0.64	**2.22**	**2.16**	1.96
100	GA	–	–	–	–	–	–
	GA_IN	0.76	0.73	0.68	2.31	2.01	1.62
	MA	0.80	0.78	0.76	2.73	2.57	2.53
	MA_IN	0.80	**0.79**	**0.77**	2.78	2.69	**2.58**
	ILS-MP	**0.81**	0.78	0.68	**2.83**	**2.77**	2.56

(2.77%) for the largest low (high) variability instances, while MA_IN achieves an average relative improvement over RBS of about 0.79% (2.69%) for the largest low (high) variability instances. Moreover, the results show that ILS-MP is sometimes able to slightly improve the best result obtained by MA_IN, especially for larger high variability instances.

The difference in performance between ILS-MP and MA_IN is more evident in Tables 3 and 4. Table 3 shows the mean relative improvement in the objective function value over MA_IN, while Table 4 provides the percentage number of times ILS-MP performs better ($<$), equal ($=$), or worse ($>$) than MA_IN. From Table 3, we observe that the best result attained with ILS-MP is generally slightly better than or equal to those obtained by MA_IN. However, MA_IN shows a slightly better average performance for low variability instances with $n \geq 75$. These observations are also confirmed by the results in Table 4.

Table 5 shows the relative improvement (calculated with the average objective function value) over RBS for instances with 50 jobs, considering different values for the tardiness factor T and the range of due dates R. We observe that the solution quality is quite the same with all the heuristics when most jobs are tardy ($T=1.0$). With a lower tardiness factor ($T \leq 0.4$), ILS-MP achieves a minor improvement over MA_IN in 17 out of 24 cases, and is slightly

Table 3
Comparison with MA_IN [22] – relative improvement.

n	Low. var			High var		
	Best	*Avg.*	*Worst*	*Best*	*Avg.*	*Worst*
10	0.000	0.000	0.003	0.000	0.003	0.019
15	0.000	0.004	0.020	0.000	0.023	0.095
20	0.000	0.008	0.031	0.002	0.047	0.144
25	0.000	0.010	0.030	0.004	0.081	0.188
30	0.000	0.009	0.023	0.009	0.071	0.172
40	0.001	0.011	0.009	0.016	0.088	0.180
50	0.002	0.005	−0.035	0.018	0.075	0.082
75	0.005	−0.002	−0.091	0.045	0.079	−0.029
100	0.007	−0.007	−0.119	0.068	0.092	−0.053

Table 4
Comparison with MA_IN – percentage of better, equal, and worse results.

Criteria	*n*	Low. var			High var		
		$<$	$=$	$>$	$<$	$=$	$>$
Best	10	0.0	100.0	0.0	0.0	100.0	0.0
	15	0.0	100.0	0.0	0.0	100.0	0.0
	20	0.0	100.0	0.0	1.0	99.0	0.0
	25	0.2	99.8	0.0	2.3	97.4	0.3
	30	0.4	99.6	0.0	4.3	95.5	0.3
	40	1.6	98.4	0.0	11.0	88.0	1.0
	50	4.5	95.5	0.0	18.0	79.5	2.5
	75	12.1	87.9	0.0	30.2	67.7	2.2
	100	16.5	83.5	0.0	37.1	61.3	1.7
Avg	10	0.5	99.5	0.0	1.3	98.7	0.1
	15	5.9	94.1	0.0	11.9	86.3	1.8
	20	14.1	85.8	0.1	24.1	72.1	3.8
	25	18.8	81.0	0.2	34.1	62.4	3.5
	30	21.8	77.8	0.3	40.0	57.0	3.0
	40	24.8	74.5	0.7	44.0	51.1	4.9
	50	25.3	72.6	2.1	46.0	47.8	6.3
	75	24.3	68.9	6.8	44.5	44.7	10.8
	100	24.1	67.6	8.3	47.9	39.5	12.6
Worst	10	0.8	99.2	0.0	1.8	98.1	0.1
	15	6.8	93.1	0.1	12.8	85.1	1.8
	20	14.8	85.0	0.3	23.6	72.3	4.1
	25	20.0	79.8	0.3	34.0	62.5	3.5
	30	23.9	75.7	0.4	40.3	56.2	3.6
	40	26.6	72.5	0.9	44.1	49.4	6.5
	50	27.4	70.3	2.3	47.0	45.7	7.3
	75	26.4	66.1	7.5	46.5	40.8	12.8
	100	26.2	63.3	10.5	48.3	36.0	15.7

outperformed by MA_IN in five cases. As in the previous comparison, we observe that the results obtained by genetic algorithms GA and GA_IN are somewhat worse than those generated with MA, MA_IN, and ILS-MP.

Finally, Table 6 reports the average running time (in seconds) over 10 runs for ILS-MP and the five reference heuristics. The results reported for these five approaches are obtained on a machine comparable with the one (see Section 3.1) we used to run ILS-MP. We observe that the proposed approach is comparable with the reference heuristics in terms of running time. Indeed, ILS-MP often requires less than 0.001 s for instances with $n \leq 20$ and less than one second for large instances with $n=100$.

As can be observed, all the heuristics generate solutions close to the optimal solutions when the variability is low. As such, there is little room for improvement for instances with low variability and more room for improving for instances with high variability (i.e., seemingly harder instances). This is evident in the comparison of the reference algorithms, i.e., there are larger differences

Table 5
Relative improvement over RBS for instances with 50 jobs.

Heur.	T	Low. var				High var			
		R=0.2	R=0.4	R=0.6	R=0.8	R=0.2	R=0.4	R=0.6	R=0.8
GA	0.0	−0.006	−0.013	−0.020	−0.019	−0.554	−0.700	−0.950	−1.130
	0.2	6.784	1.062	−0.037	0.131	11.251	−3.144	−4.203	−3.117
	0.4	−0.026	−0.112	−0.632	−7.333	0.405	−0.012	−0.493	−6.172
	0.6	−0.019	−0.055	−0.159	−0.451	0.202	0.016	−0.182	−0.622
	0.8	−0.014	−0.033	−0.049	−0.078	0.055	−0.056	−0.098	−0.127
	1.0	−0.008	−0.015	−0.021	−0.030	−0.030	−0.044	−0.052	−0.061
GA_IN	0.0	−0.001	−0.006	−0.011	−0.013	−0.090	−0.120	−0.153	−0.134
	0.2	6.801	1.522	0.177	0.258	11.361	2.408	1.955	1.615
	0.4	0.003	0.018	0.061	0.759	0.455	0.134	0.177	1.655
	0.6	0.000	−0.001	−0.002	−0.003	0.202	0.073	−0.024	−0.188
	0.8	−0.001	−0.001	0.000	−0.001	0.037	−0.024	−0.012	−0.006
	1.0	−0.001	0.000	0.000	0.000	−0.002	−0.010	−0.006	−0.003
MA	0.0	0.000	0.000	0.002	0.010	0.029	0.104	0.166	0.272
	0.2	6.867	1.649	0.292	0.430	12.032	4.003	3.389	3.887
	0.4	0.003	0.025	0.096	1.442	0.513	0.336	0.875	8.437
	0.6	0.002	0.001	0.001	0.000	0.257	0.153	0.121	0.064
	0.8	0.001	0.000	0.000	0.000	0.107	0.034	0.023	0.022
	1.0	0.000	0.000	0.000	0.000	0.001	0.002	0.004	0.003
MA_IN	0.0	0.000	0.001	0.004	0.013	0.054	0.130	0.214	0.322
	0.2	**6.875**	1.703	0.338	0.467	**12.145**	5.190	4.380	4.463
	0.4	0.007	**0.026**	0.097	1.538	**0.562**	**0.381**	0.893	8.634
	0.6	0.002	0.001	0.001	0.000	0.274	0.156	0.126	0.065
	0.8	0.001	0.000	0.000	0.000	0.107	0.034	0.023	0.022
	1.0	0.000	0.000	0.000	0.000	0.001	0.002	0.004	0.003
ILS-MP	0.0	0.000	**0.002**	**0.005**	**0.014**	**0.056**	**0.139**	**0.220**	**0.340**
	0.2	6.771	**1.714**	**0.374**	**0.514**	12.095	**5.460**	**4.598**	**4.811**
	0.4	0.007	0.018	**0.101**	**1.677**	0.527	0.373	**0.960**	**9.415**
	0.6	0.002	0.001	0.001	0.000	**0.285**	**0.160**	0.126	**0.066**
	0.8	0.001	0.000	0.000	0.000	**0.108**	**0.035**	0.023	0.022
	1.0	0.000	0.000	0.000	0.000	0.001	0.002	0.004	0.003

Table 6
Runtime (in seconds).

Var.	Heur.	n=10	n=20	n=30	n=40	n=50	n=75	n=100
L	RBS	0.001	0.004	0.009	0.019	0.033	0.100	0.227
	GA	0.012	0.174	0.221	0.492	0.932	3.028	–
	GA_IN	0.003	0.015	0.042	0.088	0.158	0.480	1.070
	MA	0.003	0.011	0.037	0.089	0.172	0.612	1.617
	MA_IN	0.004	0.011	0.033	0.079	0.153	0.545	1.445
	ILS-MP	0.000	0.001	0.002	0.006	0.013	0.068	0.205
H	RBS	0.001	0.004	0.009	0.020	0.035	0.109	0.250
	GA	0.013	0.008	0.242	0.549	1.051	3.476	–
	GA_IN	0.004	0.020	0.052	0.106	0.192	0.593	1.353
	MA	0.004	0.013	0.041	0.102	0.203	0.767	2.108
	MA_IN	0.004	0.011	0.036	0.089	0.172	0.649	1.790
	ILS-MP	0.000	0.001	0.004	0.021	0.052	0.263	0.843

between the heuristic results for the high variability instances, which is likely due to the extra room for improvement for such instances.

3.3. Comparison with optimal results

In this section we compare the optimal results with those obtained by ILS-MP and the reference heuristics. For this comparison, we only use instances with up to 20 jobs because the optimal objective values for larger instances are unknown.

Table 7 provides the average of the relative deviations (%dev) from the optimal values, calculated as $(H-O)/O \times 100$, where H and O are the heuristic and optimal objective values, respectively. We also provide the percentage number of times (%opt) each heuristic obtains an optimal solution. The results in Table 7 show that ILS-MP is able to find an optimal or near optimal solution with a relative deviation that is usually not more than 0.02%. The memetic procedures MA and MA_IN also perform extremely well, with a slightly worse deviation from the optimum (up to 0.06% and 0.16% for MA_IN and MA, respectively).

Table 8 shows the relative deviations from the optima for instances with 20 jobs, considering different values for the tardiness factor T and the range of due dates R. We observe that the deviations of the heuristic solutions from the optimal solutions are higher when most jobs are early and when a larger number of jobs are completed before their due dates ($T < 0.6$). When most jobs are tardy ($T \geq 0.6$), the considered heuristics are usually able to attain optimal or near-optimal solutions. The relative deviation from the optimum is larger for the instances with T=0.2 or 0.4. For ILS-MP, the relative deviation does not exceed 0.155% (0.002%) for the high (low) variability instances. For MA and MA_IN, the highest relative deviations for the high (low) variability instances are 0.325% (0.993%) and 0.058% (0.453%), respectively.

4. Conclusion

In this research we propose an iterated local search based on multi-type perturbation approach for single-machine scheduling to minimize the sum of linear earliness and quadratic tardiness penalties. ILS-MP applies a simple descent-based local search procedure to intensify the search in the current neighbourhood, and triggers a dedicated perturbation mechanism to escape from

Table 7
Comparison with optimal objective function values.

Heur.	T	$n=10$		$n=15$		$n=20$	
		%dev.	%opt	%dev.	%opt	%dev.	%opt
L	RBS	0.02	97.00	0.03	83.17	0.13	73.25
	GA	0.20	81.09	0.30	50.83	0.38	29.98
	GA_IN	0.03	79.59	0.05	56.53	0.06	44.23
	MA	0.01	99.56	0.02	97.31	0.04	91.75
	MA_IN	0.00	99.89	0.00	98.41	0.01	95.53
	ILS-MP	0.00	**100.00**	0.00	**100.00**	**0.00**	**99.50**
H	RBS	0.46	88.83	0.89	75.83	0.81	56.83
	GA	0.69	68.87	1.26	33.03	1.66	12.58
	GA_IN	0.55	60.18	0.89	28.88	0.86	17.48
	MA	0.02	98.98	0.09	93.80	0.16	84.37
	MA_IN	0.00	99.73	0.03	96.03	0.06	89.33
	ILS-MP	0.00	**99.80**	**0.01**	**98.20**	**0.01**	**94.30**

Table 8
Comparison with optimal solutions for instances with 20 jobs.

Heur.	T	Low. var				High var			
		$R=0.2$	$R=0.4$	$R=0.6$	$R=0.8$	$R=0.2$	$R=0.4$	$R=0.6$	$R=0.8$
RBS	0.0	0.000	0.000	0.003	0.007	0.040	0.115	0.279	0.147
	0.2	1.857	0.208	0.227	0.266	3.974	4.516	2.308	1.433
	0.4	0.012	0.031	0.077	0.505	0.682	0.335	0.718	3.790
	0.6	0.003	0.002	0.000	0.000	0.673	0.133	0.057	0.045
	0.8	0.000	0.000	0.000	0.000	0.043	0.059	0.001	0.008
	1.0	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000
GA	0.0	0.014	0.022	0.031	0.039	0.418	0.986	1.623	1.620
	0.2	0.073	0.256	0.397	0.458	1.013	5.955	7.442	6.676
	0.4	0.043	0.182	0.741	5.253	0.243	0.424	1.469	9.291
	0.6	0.031	0.106	0.236	0.737	0.148	0.256	0.451	0.853
	0.8	0.023	0.054	0.086	0.127	0.101	0.144	0.206	0.229
	1.0	0.013	0.025	0.037	0.058	0.050	0.082	0.086	0.107
GA_IN	0.0	0.010	0.009	0.019	0.022	0.191	0.317	0.515	0.529
	0.2	0.038	0.200	0.236	0.341	0.749	3.215	3.761	3.581
	0.4	0.011	0.020	0.057	0.460	0.240	0.390	1.058	4.114
	0.6	0.009	0.007	0.004	0.004	0.161	0.228	0.320	0.641
	0.8	0.004	0.003	0.002	0.001	0.094	0.175	0.134	0.135
	1.0	0.001	0.001	0.000	0.001	0.012	0.029	0.030	0.034
MA	0.0	0.001	0.004	0.004	0.008	0.025	0.075	0.129	0.114
	0.2	0.142	0.125	0.192	0.101	0.338	0.993	0.992	0.578
	0.4	0.029	0.045	0.013	0.325	0.108	0.086	0.114	0.267
	0.6	0.001	0.000	0.000	0.000	0.031	0.023	0.002	0.000
	0.8	0.000	0.000	0.000	0.000	0.002	0.001	0.000	0.000
	1.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MA_IN	0.0	0.000	0.000	0.001	0.000	0.015	0.027	0.026	0.030
	0.2	0.009	0.058	0.045	0.049	0.098	0.343	0.453	0.255
	0.4	0.000	0.001	0.003	0.038	0.026	0.015	0.036	0.145
	0.6	0.000	0.000	0.000	0.000	0.008	0.002	0.001	0.000
	0.8	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.001
	1.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
ILS-MP	0.0	0.000	0.000	**0.000**	0.000	**0.000**	**0.001**	**0.007**	**0.004**
	0.2	**0.000**	**0.000**	**0.001**	**0.002**	**0.006**	**0.155**	**0.078**	**0.070**
	0.4	0.000	**0.000**	**0.000**	**0.001**	**0.000**	**0.001**	**0.001**	**0.004**
	0.6	0.000	0.000	0.000	0.000	**0.000**	0.002	**0.000**	0.000
	0.8	0.000	0.000	0.000	0.000	**0.001**	0.000	0.000	**0.000**
	1.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

local optima. This perturbation strategy probabilistically combines three types of perturbation strategies, namely tabu-based perturbation, which is based on tabu search principles; construction-based perturbation, which borrows some ideas from the GRASP metaheuristic; and classical random perturbation. We conduct extensive experiments to evaluate the performance of ILS-MP and compare it with the current state-of-the-art heuristics for SMSP-LEQT on a set of commonly used benchmark instances. Despite its simplicity, the results show that ILS-MP is effective and efficient in terms of both solution quality and computing time. In fact, in

many cases, ILS-MP attains equal or slightly better results than those obtained by the state-of-art approaches for SMSP-LEQT in the literature.

Acknowledgements

We thank two anonymous referees for their helpful comments on an earlier version of our paper. The research was supported in part by the National Natural Science Foundation of China under grant numbers 61370183 and 61100144, and the programme for New Century Excellent Talents in University (NCET 2013).

References

[1] Benlic U, Hao JK. Breakout local search for maximum clique problems. Comput Oper Res 2013;40(1):192–206.
[2] Benlic U, Hao JK. Breakout local search for the quadratic assignment problem. Appl Math Comput 2013;219(9):4800–15.
[3] Benlic U, Hao JK. Breakout local search for the vertex separator problem. In: IJCAI 2013, Proceedings of the 23rd international joint conference on artificial intelligence; 2013. p. 461–7.
[4] Garey MR, Tarjan RE, Wilfong GT. One-processor scheduling with symmetric earliness and tardiness penalties. Math Oper Res 1998;13(2):330–48.
[5] Glover F. Tabu search—Part I. ORSA J Comput 1989;1(3):190–260.
[6] Glover F. Tabu search—Part II. ORSA J Comput 1990;2(1):4–32.
[7] Glover F, Marti R. Tabu search. In: Metaheuristic procedures for training neutral networks; 2006. p. 53–69.
[8] Gupta SK, Sen T. Minimizing a quadratic function of job lateness on a single machine. Eng Costs Prod Econ 1983;7(3):187–94.
[9] Lenstra JK, Rinnooy Kan AHG, Brucker P. Complexity of machine scheduling problems. Ann Discrete Math 1977;1:343–62.
[10] Lourenco HR, Martin O, Stützle T. Iterated local search. In: Handbook of metaheuristics. Berlin, Heidelberg: Springer-Verlag; 2003.
[11] Kim YD, Yano CA. Minimizing mean tardiness and earliness in single machine scheduling problems with unequal due dates. Nav. Res Log 1994;41(7):913–33.
[12] Kodaganallur V, Sen AK, Mitra S. Application of graph search and genetic algorithms for the single machine scheduling problem with sequence-dependent setup times and quadratic penalty function of completion times. Comput Ind Eng 2014;67:10–9.
[13] Resende M, Ribeiro C. Greedy randomized adaptive search procedures. In: Handbook of metaheuristics, vol. 57; 2003. p. 219–49.
[14] Schaller J. Minimizing the sum of squares lateness on a single machine. Eur J Oper Res 2002;1(16):64–79.
[15] Schaller J. Single machine scheduling with early and quadratic tardy penalties. Comput Ind Eng 2004;46(3):511–32.
[16] Schaller J. A comparison of lower bounds for the single-machine early/tardy problem. Comput Oper Res 2007;34(8):2279–92.
[17] Sen T, Dileepan P, Lind MR. Minimizing a weighted quadratic function of job lateness in the single machine system. Int J Prod Econ 1995;42(3):237–43.
[18] Su LH, Chang PC. A heuristic to minimize a quadratic function of job lateness on a single machine. Int J Prod Econ 1998;55(2):169–75.
[19] Valente JMS. Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. Eur J Ind Eng 2007;1(4):431–48.
[20] Valente JMS. An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties. Asia-Pac J Oper Res 2008;25(2):169–86.
[21] Valente JMS. Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. Asia-Pac J Oper Res 2009;26(3):319–39.
[22] Valente JMS, Gonçalves JF. A genetic algorithm approach for the single machine scheduling problem with earliness and quadratic tardiness penalties. Comput Oper Res 2009;36:2707–15.
[23] Valente JMS, Schaller JE. Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. Eur J Ind Eng 2010;4(1):99–129.