Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://www.tandfonline.com/loi/tjor20

A hybrid genetic algorithm for an identical parallel-machine problem with maintenance activity

Wen-Chiung Lee, Jen-Ya Wang & Lin-Yo Lee

To cite this article: Wen-Chiung Lee, Jen-Ya Wang & Lin-Yo Lee (2015) A hybrid genetic algorithm for an identical parallel-machine problem with maintenance activity, Journal of the Operational Research Society, 66:11, 1906-1918, DOI: 10.1057/jors.2015.19

To link to this article: https://doi.org/10.1057/jors.2015.19

Journal of the Operational Research Society (2015) 66, 1906–1918

www.palgrave-journals.com/jors/

A hybrid genetic algorithm for an identical parallel-machine problem with maintenance activity

Wen-Chiung Lee[1], Jen-Ya Wang[2*] and Lin-Yo Lee[1]

[1]*Feng Chia University, Taichung, Taiwan (R.O.C.); and* [2]*Hungkuang University, Taichung, Taiwan (R.O.C.)*

The scheduling of maintenance activities has been extensively studied, with most studies focusing on single-machine problems. In real-world applications, however, multiple machines or assembly lines process numerous jobs simultaneously. In this paper, we study a parallel-machine scheduling problem in which the objective is to minimize the total tardiness given that there is a maintenance activity on each machine. We develop a branch-and-bound algorithm to solve the problem with a small problem size. In addition, we propose a hybrid genetic algorithm to obtain the approximate solutions when the number of jobs is large. The performance of the proposed algorithms is evaluated based mainly on computational results.

Journal of the Operational Research Society (2015) **66**(11), 1906–1918. doi:10.1057/jors.2015.19
Published online 15 April 2015

Keywords: scheduling; total tardiness; parallel machine; makespan; maintenance activity

1. Introduction

In traditional scheduling, it is assumed that all the machines are available for processing jobs at all times. This assumption is applicable in most circumstances. However, since machines need preventive maintenance or part replacements, an important issue for many manufacturing industries is finding ways to schedule maintenance activities and jobs while still meeting production schedules.

In recent years, maintenance activities have been the focus of increasing attention in the field of job scheduling. Adiri *et al* (1991) and Lee and Liman (1992) were among the first researchers to introduce maintenance activities into production scheduling. Since their initial work, many researchers have devoted efforts to this field. Schmidt (2000) and Ma *et al* (2010) provided a comprehensive survey on scheduling with limited machine availability. For more research into scheduling with maintenance activity, please refer to Qi *et al* (1999), Yang *et al* (2011), Lee and Kim (2012), Xu and Xiong (2012), Yang (2013) Yu *et al* (2013), Gustavsson *et al* (2014), and Luo *et al* (2015).

Single-machine job scheduling has been explored for decades, and the related academic achievements are of good practical use to industry. Pinedo (2008) pointed out that it is important to consider a bank of machines in parallel. From a theoretical standpoint, such a problem is a generalization of the single machine and a special case of the flexible flowshop. From a practical standpoint, it is important because in the real world, resources often function in parallel.

Single-machine scheduling problems can be solved optimally by many mature algorithms when the problem size is small. Recently, Sbihi and Varnier (2008) studied a single-machine maximum tardiness problem with several maintenance periods. They proposed branch-and-bound algorithms for the situations wherein the maintenance periods are periodically fixed or the maximum allowed continuous working time of the machine is determined. Chen (2009) considered a single-machine scheduling problem to minimize the number of tardy jobs subject to periodic maintenance. He proposed a branch-and-bound algorithm to find the optimal schedule. Yang (2010) introduced a model of joint start-time dependent learning and position dependent aging effects to find jointly the optimal maintenance position and the optimal sequence such that the makespan, the total completion time, or the total absolute deviation of completion times are minimized. He showed that all the problems could be optimally solved by polynomial-time algorithms. Yang *et al* (2011) studied a single-machine problem with a single maintenance activity where the objective was to minimize the total completion time. They showed that the shortest processing time algorithm was optimal for the resumable case. They also developed a dynamic programming algorithm and a branch-and-bound algorithm to generate an optimal solution for the non-resumable case. Yang (2012) considered scheduling jointly the deterioration and learning effects as well as multi-maintenance activities on a single machine. The objectives were to determine the optimal maintenance frequencies, the optimal maintenance locations, and the optimal job schedule such that the makespan and the total

*Correspondence: Jen-Ya Wang, Department of Computer Science and Information Management, Hungkuang University, No. 1018, Sec. 6, Taiwan Boulevard, Shalu, Taichung 43302, Taiwan (R.O.C.).
E-mail: jywang@sunrise.hk.edu.tw

completion time were minimized, respectively. He showed that both problems could be solved in polynomial time.

Many heuristics are useful for obtaining near-optimal schedules on a single machine when the problem size is large. Ghodratnama *et al* (2010) investigated a multi-objective single-machine problem to minimize the sum of the weighted jobs completion, to minimize the sum of the weighted delay times, and to maximize the sum of the job values in makespan with the consideration of maintenance periods, deterioration of jobs, and learning effect. They proposed solving this problem with simulated annealing. Lee and Kim (2012) investigated a single-machine problem that required periodic maintenance with the objective of minimizing the number of tardy jobs. They presented a two-phase heuristic algorithm in which an initial solution is obtained first with a method modified from Moore's algorithm for the problem without maintenance, and then the solution is improved in the second phase. Joo and Kim (2013) provided a genetic algorithm for the single-machine problem with time-dependent deterioration and multiple rate-modifying activities.

To the best of our knowledge, maintenance activity is relatively unexplored on parallel machines. Wang and Wei (2011) considered scheduling problems with identical parallel machines and deteriorating maintenance activities. They showed that minimizing the total absolute differences in completion times and the total absolute differences in waiting times remain polynomially solvable under the proposed model. Cheng *et al* (2011) studied the problem of unrelated parallel-machine scheduling with deteriorating maintenance activities. They showed that minimizing the total completion time or the total machine load could be optimally solved in polynomial time. Shen *et al* (2013) considered a parallel-machine problem in which the machines may not be available simultaneously at time zero. They presented the polynomial time algorithms to minimize the total completion time and total absolute difference in completion times and to minimize the total waiting time and total absolute difference in waiting times.

Other important issues in today's job scheduling are parallel environments, tardiness, and maintenance activities. A parallel environment allows mass production, but it makes job scheduling difficult. Moreover, as we perform job scheduling, it is important to remember that less tardiness means more customer satisfaction. However, most machines cannot process jobs endlessly; they need to be deactivated periodically for maintenance. Motivated by this observation, we consider a parallel-machine scheduling problem to minimize the total tardiness given that there is a maintenance activity for each machine, and that the maintenance must be performed within a pre-specified time interval. For more recent developments in scheduling with total tardiness, readers can refer to Du and Leung (1990), Azizoglu and Kirca (1998), Shim and Kim (2007), Madhushini *et al* (2009), Sbihi and Varnier (2008), Schaller and Valente (2012), and Yin *et al* (2014).

The rest of the paper is organized as follows. The formulation of our problem is described in Section 2. In Section 3, a branch-and-bound algorithm with several elimination rules and a lower bound is developed. In Section 4, two genetic algorithms are proposed to solve this problem. In Section 5, computational experiments are conducted to evaluate the performance of the algorithms. In Section 6, a real problem instance is tested for solution quality and execution speed. The conclusion is given in Section 7.

2. Problem description

This study is motivated by the following two real-world applications. The first application is common in the logistics industry (Chen, 2014). Consider a company that owns many vans (ie, machines) used to deliver goods with different due dates (ie, jobs). The purchase dates of these vans are different, so their maintenance tasks are staggered. For example, the oil and air filters need to be changed every 6 months, and the brake fluid and tires, every 12. Clearly, not all the vans can be maintained within 1 working day. A good job schedule can help the company maintain customer satisfaction (ie, total tardiness) while still meeting all the maintenance needs of the vans.

The second application is to aluminium extrusion (Huang, 2014). Consider a factory with dozens of identical machines responsible for manufacturing aluminium products (ie, jobs). Some of the jobs are from the sales department, and some are from the R&D department, so they have different due dates. After a period of operation, each machine needs maintenance. For example, oil filters must be changed every month, and sliding blocks, linear bearings, and motors require lubrication every 3 months. Since these maintenance activities cannot be conducted within 1 day, they require scheduling. Exploring such a scheduling problem is also helpful to reducing total tardiness.

The problem is formally defined as follows. There are m identical parallel machines, each of which requires maintenance. The maintenance activity on machine i takes time q_i and must be performed within a specified maintenance window, that is, $[B_i, E_i]$, for $i = 1, 2, \ldots, m$, where B_i and E_i are two non-negative integers for $i = 1, 2, \ldots, m$. There are also $n - m$ real jobs numbered $m+1, m+2, \ldots, n$ which can be processed by any machine and whose processing times are p_j and due dates are d_j for $j = m+1, m+2, \ldots, n$. For simplicity, let $p_i = q_i$ for $i = 1, 2, \ldots, m$. Both represent the processing time of a maintenance activity (ie, a particular job). For a schedule S, we let $C_j(S)$ be the completion time and $T_j(S) = \max\{0, C_j(S) - d_j\}$ be the tardiness of job j. In this study, we consider a parallel-machine problem to minimize the total tardiness given that the maintenance jobs must be performed within the specified intervals. The objective is to find a schedule S that minimizes the total tardiness, that is, $F(S) = \sum_{i=1}^{m} \left(\sum_{\text{job } j \text{ is allocated to machine } i} T_j \right)$, subject to $C_i(S) \in [B_i + q_i, E_i]$, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Using the three-field notation, the problem is denoted as Pm/Maintenance/$\sum T_j$.

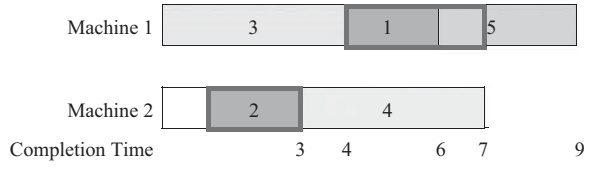

Figure 1 An example of this problem.

Figure 1 illustrates an example of $m=2$ and $n=5$. There are two machines with maintenance windows $B_1=4$, $E_1=7$, $B_2=2$, $E_2=3$, and processing times $q_1=q_2=2$. We also have $p_3=3$, $p_4=4$, $p_5=3$, $d_3=4$, $d_4=7$, and $d_5=9$. A feasible schedule can be obtained as follows. First, we allocate each maintenance activity to each corresponding maintenance window in turn (shaded area). Second, jobs 3 and 5 are allocated to machine 1, and job 4 is allocated to machine 2 immediately after the maintenance activity. Note that there is idle time on machine 2, since all jobs are non-preemptive. Since each job is accomplished before its due date, the total tardiness is 0.

3. Branch-and-bound algorithm for generating optimal schedules

Du and Leung (1990) showed that the total tardiness problem is NP-hard even for the single-machine case, so our problem is at least NP-hard. In this section, we develop a branch-and-bound algorithm to search for the optimal solution. We use the notation $<k>$ to denote the maintenance activity on machine k and the delimiter '0' to separate jobs from machines. For instance, the job sequence $(5, 6, <1>, 9, 0, 8, <2>, 10, 4, 0, 7, <3>, 11)$ means that jobs 5 and 6 are processed before the maintenance activity; job 9 is processed after the maintenance activity on machine 1; job 8 is before the maintenance activity; jobs 10 and 4 are processed after the maintenance activity on machine 2; job 7 is before the maintenance activity; and job 11 is processed after the maintenance activity on machine 3. In this paper, we assign jobs, maintenance activities, and delimiters in a forward way, starting from the first position. A depth-first search process is adapted in our algorithm. In the following, we will provide several properties to facilitate the search process.

3.1. Dominance properties

Azizoglu and Kirca (1998) proposed an upper bound for the sum of processing times of jobs processed on each machine. With the consideration of the maintenance activity, we modify the upper bound for the sum of processing times of jobs processed on machine k when partial sequence α is determined. Suppose that $S=(\alpha,\beta)$ is a schedule in which the last job in partial sequence α is neither a delimiter nor a maintenance activity. Furthermore, suppose that the last job in α is processed on machine k and its completion time is t. Among jobs in α, let α_i be the set of real jobs scheduled on machine i, n_{α_i} be the number of jobs in α_i, and $\Omega_i(\alpha)=\cup_{j=1}^{i}\alpha_j$ for $i=1, 2, \ldots, k$ and Ω be the set of the $n-m$ real jobs. Property 1 shows that there exists an upper bound of the objective cost, that is, total tardiness, for each machine.

Property 1 If job j is processed on machine k and its completion time exceeds

$$U_k=\frac{1}{m-k+1}\left[\sum_{j\in\Omega\setminus\Omega_{k-1}(\alpha)}p_j+(2m-2k+1)\max_{j\in\Omega\setminus\Omega_{k-1}(\alpha)}p_j\right]+q_k$$

then $S=(\alpha,\beta)$ is a dominated sequence.

Furthermore, if the maintenance activity on machine k has not been scheduled yet, then the schedule might not be feasible, which is stated as follows.

Property 2 If the maintenance activity on machine k has not been scheduled yet and $E_k-t<q_k$, then $S=(\alpha,\beta)$ is not a feasible sequence.

In addition, we develop an upper bound for the number of jobs processed on machine k. Among jobs in β, let $p^{\beta}_{(1)}\leqslant p^{\beta}_{(2)}\leqslant\ldots$ denote the processing times of the real jobs when they are in a non-decreasing order and n^{β}_k be a number such that $t+q_k+\sum_{j=1}^{n^{\beta}_k}p^{\beta}_{(j)}\leqslant U_k$ and $t+q_k+\sum_{j=1}^{n^{\beta}_k+1}p^{\beta}_{(j)}>U_k$. Property 3 shows that there also exists an upper bound for the number of jobs allocated to machine k.

Property 3 If the number of jobs from β to be processed on machine k is greater than n^{β}_k, then $S=(\alpha,\beta)$ is a dominated sequence.

Suppose that $S=(\pi_1, j, \pi_2, i, \pi_3)$ and $S'=(\pi_1, i, \pi_2, j, \pi_3)$, where π_1, π_2, and π_3 are partial sequences. Furthermore, let $|\pi_2|$ denote the number of jobs in partial sequence π_2. Shim and Kim (2007) provided the following result. Property 4 checks whether we can reduce the total tardiness by interchanging two jobs, for which a small job i is originally allocated after a large job j in S'. If so, the new schedule S dominates S'.

Property 4 If jobs i and j are processed on the same machine, $p_i\leqslant p_j$ and $(p_j-p_i)\,|\pi_2|+T_j(S)+T_i(S)\leqslant T_i(S')+T_j(S')$, then S dominates S'.

Property 5 checks whether we can sustain the total tardiness after interchanging two adjacent jobs. In Case (i), both jobs are tardy in S', so we can move a small job forward. Case (ii) shows that we can move a large job backward if both jobs are tardy in all situations. Case (iii) implies that a job i, already tardy in S', can be moved backward if there is no additional tardiness of job j. Case (iv) implies that a job j, already tardy in S', can be moved backward if there is no additional tardiness of job j. In Cases (v) and (vi), it worth moving a tardy job i in S' forward to sustain the current total tardiness. In Case (vii), for two jobs

that are not tardy in any situations, the job with the earlier due date is forced to be scheduled first.

Property 5 If two adjacent jobs i and j are processed on the same machine, and if any one of the following conditions holds, then $S=(\pi, i, j, \pi')$ dominates $S'=(\pi, j, i, \pi')$.

(i) $t+p_i \geqslant d_i$, $t+p_j \geqslant d_j$, and $p_i < p_j$;
(ii) $d_i \leqslant t+p_i < d_j \leqslant t+p_i+p_j$ and $t+p_j+d_j$;
(iii) $t+p_i \geqslant d_i$ and $t+p_i+p_j \leqslant d_j$;
(iv) $t+p_i \leqslant d_i \leqslant t+p_j$ and $t+p_j \geqslant d_j$;
(v) $t+p_i \leqslant d_i$, $t+p_j \leqslant d_j \leqslant t+p_i+p_j$, and $d_i < d_j$;
(vi) $t+p_i \leqslant d_i$, $t+p_i+p_j \leqslant d_j$, and $t+p_i+p_j > d_i$;
(vii) $t+p_j \leqslant d_j$, $t+p_j+p_i \leqslant d_i$, $t+p_i \leqslant d_i$, $t+p_i+p_j \leqslant d_j$, and $d_i < d_j$.

Property 6 checks whether we can sustain the total tardiness after interchanging two adjacent jobs, one of which is a maintenance activity. The property checks whether the interchange will lead to any violation of the maintenance activity. Note that the total tardiness will not be worse if a real job is moved forward, whether it is already tardy or not.

Property 6 $S=(\alpha, j, <k>, \beta)$ dominates $S'=(\alpha, <k>, j, \beta)$ if $\max\{t+p_j, B_k\}+q_k \leqslant E_k$.

Azizoglu and Kirca (1998) provided the following property, which is stated without proof.

Property 7 If jobs i and j are processed on machine k, $d_i \leqslant d_j$ and $p_i = p_j$, then job i must be processed before job j in an optimal sequence.

3.2. Lower bound

The main idea behind the proposed lower bound is like the u-tube principle of physics. Consider all the available machines as connected pipes and their initial times (ie, $t=0$) as the bottoms of the pipes. For each available machine k, we first schedule its maintenance activity. Then we schedule the remaining jobs one by one as we fill the connected pipes with water. The resultant water levels can be viewed as completion times of the remaining jobs and used to estimate the optimal cost.

Before introducing the proposed lower bound, we define some symbols first. Let $S=(\alpha, \beta)$ be an incomplete schedule, $L_{\max}$ denote the highest water level in the remaining available connected pipes, and L_k denote the lowest water level of all remaining pipes for some pipe k at the beginning of each loop. Note that α is a determined subsequence. If there are some K delimiters in α, it means that the former K pipes cannot be assigned any new jobs. Before starting to calculate the lower bound, we transform all jobs j in β into new jobs (j) such that the processing times and due dates of these new jobs are *agreeable*; that is, $p_{(i)} \leqslant p_{(j)}$ implies that $d_{(i)} \leqslant d_{(j)}$. Then we check each unmaintained machine k, if any, and assign its maintenance activity to $[E_k-q_k, E_k]$, which is viewed as an *occupied* area in pipe k. The major steps of the proposed lower bound are stated as follows:

(1) for each job (j) do Steps (2) ~ (11);
(2) while $(p_{(j)} > 0)$ do Steps (3) ~ (10);
(3) choose an available pipe k with the lowest water level L_k;
(4) if $(L_k < L_{\max})$ then do Steps (5) ~ (6);
(5) fill a proper amount of water a into pipe k such that its final water level is:
L_k+a, if $a \leqslant p_{(j)}$ and, $L_k+a \leqslant L_{\max}$, //no interference of occupied area
$L_{\max}$, if $L_{\max} \leqslant E_k-q_k$ and $L_k+p_{(j)} \geqslant L_{\max}$, //no interference of occupied area
$E_k-q_{(k)}$, if $L_{\max} \geqslant E_k-q_k$ and $L_k+p_{(j)} \geqslant E_k-q_k$, //interference of occupied area
$L_{\max}$, if $(L_{\max} \geqslant E_k$ and $L_{\max}-q_k-L_k \leqslant p_{(j)})$; //interference of occupied area
(6) set $p_{(j)} = p_{(j)} - a$;
(7) else do Steps (8) ~ (9);
(8) fill $p_{(j)}$ into all the available pipes such that they have the same final water level (except the occupied areas);
(9) set $p_{(j)} = 0$;
(10) determine $L_{\max}$, $C_{(j)}(\beta)$, and all L_k's;
(11) set $T_{(j)} = \max\{0, C_{(j)}(\beta) - d_{(j)}\}$;
(12) return the lower bound $LB(S) = \sum_{j \in \alpha} T_j + \sum_{(j)} T_{(j)}$.

4. Genetic algorithm for generating near-optimal schedules

In this section, we utilize a genetic algorithm (GA) to obtain near-optimal solutions. Moreover, we use a local search to help the proposed GA with accelerating the convergence speed and improving the solution quality simultaneously.

4.1. Representation and population size

We encode each solution (ie, job schedule) using a sequence of integers from 1 to $n-m$ and $(2m-1)$ 0's in a random order, where the $2k$th 0 means a separator for separating jobs from machines k and $k+1$, and the $(2k-1)$st 0 is the maintenance activity on machine k. For example, for $n=8$ and $m=3$, a chromosome $\pi=(4, <0>, 0, 6, <0>, 8, 0, 7, <0>, 5)$ represents a sequence in which job 4 is scheduled on machine 1, followed by the maintenance activity of machine 1; job 6 is scheduled on machine 2, followed by the maintenance activity of machine 2 and job 8; and finally job 7 is scheduled on machine 3 first, followed by the maintenance activity of machine 3 and lastly job 5. The population size P_S is set to 500 in this study.

4.2. Fitness and selection

Because of the constraint that the maintenance activity on each machine must be performed within a specified period of time, a penalty of $100nv$ will be imposed on the original objective function, where v is the number of violations in a solution and n is the number of jobs. That is, the modified objective function

for chromosome π_i is

$$f_i = \sum_{j \in \pi_i} T_j + 100nv$$

Moreover, we use the standard roulette wheel selection operation to select the chromosome. The probability of selecting chromosome π_i is

$$q_i = \frac{g_i}{\sum_{k=1}^{P_S} g_k} \text{ for } i = 1, \ \ldots \ , \quad P_s$$

where $g_i = f_{\max} - f_i + 1$ is the ith fitness value for chromosome π_i and $f_{\max} = \max\{f_i\}$.

4.3. Crossover

In this study, we use a partially matched crossover (PMX) operator (Goldberg and Lingle, 1985; Lopez and O'Neill, 2009), which is described as follows. First, two parent chromosomes are randomly selected according to the selection operation. Second, two distinct random positions x and y are generated to form a substring for each parent. These substrings are called the mapping sections. For the case of $x < y$, an example of $x = 2$ and $y = 5$ is illustrated in Figure 2. Then the mapping relationship is determined as shown in Figure 2c. Third, these two substrings (ie, shaded areas) are exchanged to obtain two new offspring. Finally, these two offspring are legalized according to the mapping relationship if there is a duplicate job outside the substrings. Similarly, we can exchange the unshaded areas for the case of $x > y$. In this study, the crossover rate r_c is 0.8.

4.5. Mutation

Three kinds of mutation operations are used in the genetic algorithm. At the beginning, two distinct positions x and y are randomly selected. Operation 1 is performed if $x < y$. Jobs within positions x and y circularly shift left. For example, we have a schedule (1, 2, 3, 4, 5, 6,7,8) and two random positions $x = 2$ and $y = 5$. The resultant schedule is (1, 3, 4, 5, 2, 6, 7, 8). Operation 2 is performed if $x > y$. Jobs not within positions x and y circularly shift left. For example, the schedule (1, 2, 3, 4, 5, 6, 7, 8) with $x = 5$ and $y = 2$ is mutated to (2, 5, 3, 4, 6, 7, 8, 1). Operation 3 is just the swapping of two jobs at positions x and y. For each randomly selected schedule to mutate, a probability z is randomly drawn. We perform operation 1 if $z < 0.45$, operation 2 if $0.45 \leqslant z < 0.9$, and operation 3 otherwise. In this study, the mutation rate r_m is 0.1.

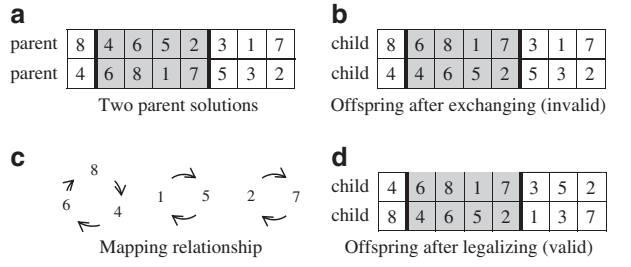

Figure 2 An example of the PMX crossover operation. (a) Two parent solutions; (b) mapping relationship; (c) offspring after exchanging (invalid); (d) offspring after legalizing (valid).

4.6. Local search

The local search includes two passes: an intra-machine exchange and an inter-machine exchange. In the first pass, for each single machine, every two adjacent jobs (eg, jobs i and j at positions k and $k+1$) are exchanged to test if any improvement can be made. In the second pass, for every two adjacent machines (eg, machines numbered i and $i+1$), pairwise jobs at the same position k are exchanged to test if any improvement can be made for $k = 1, 2, \ldots$, and so on. When the two passes are completed, the original solution is replaced with the trial with the largest improvement. In this study, the local search rate r_{ls} is 0.15.

5. Computational experiments

The experimental results are divided into two parts: one for small problem instances and the other for large problem instances. Before formal experiments, a pilot experiment is conducted to determine the parameter setting for both genetic algorithms GA and GA + LS (ie, GA with local search). In the first part, three problem sizes ($n = 12, 16, 20$) are considered to test the capabilities of the branch-and-bound algorithm (B&B), GA, and GA + LS. In the second part, two large problem sizes ($n = 50, 100$) are used to test the convergence speed and solution quality of GA and GA + LS. All the proposed algorithms are implemented in PASCAL and executed in a Windows 7 environment on an Intel Xeon E3 1230 @ 3.20 GHz with 8 GB RAM. For each setting, 50 random trials are conducted and recorded.

Table 1 lists the parameters used in this section. Parameter n denotes the number of jobs, and the processing time p_j is randomly drawn from $\{1, 2, \ldots, 100\}$. The due date d_j follows a uniform distribution over the integers between $T(1 - \tau - R/2)$ and $T(1 - \tau + R/2)$, where R is the factor of the due date range, τ is the factor of the tardiness, and T is the sum of the processing times of all jobs. Parameters B_i and E_i are two random integers with $B_i \in \{1, 2, \ldots \lfloor T/m \rfloor\}$ and $0 \leqslant E_i - B_i - q_i \leqslant 100$. For both genetic algorithms, the crossover rate, mutation rate, local search rate, and population size are denoted by r_c, r_m, r_{ls}, and P_S, respectively. Both genetic algorithms have two stopping criteria: (1) run time is more than $n + 10$ s; (2) no improvement is made during the recent $10n + 100$ generations. Moreover, observation of a pilot experiment suggests that $r_c = 0.8$, $r_m = 0.1$, and $r_{ls} = 0.15$ can achieve high solution quality within acceptable run time. The population size is fixed at $P_S = 500$.

Table 1 Parameters used in the experiments

Parameter	*Range*	*Meaning*
n	12, 16, 20, 50, 100	Number of jobs (including maintenance activities)
m	2, 3, 5	Number of servers
p_j	1, 2, …, 100	Processing time of job j
q_i	1, 2, …, 100	Processing time of maintenance activity i, ie, $p_i = q_i$ for $i = 1, 2, \ldots, m$
d_j	$[T(1-\tau-R/2)/m, T(1-\tau+R/2)/m]$	Due date of job j
τ	0.25, 0.5, 0.75	Tardiness factor
R	0.25, 0.5, 0.75	Due date factor
$[B_i, E_i]$	$1 \leqslant B_i \leqslant \lfloor T/m \rfloor$, $0 \leqslant E_i - B_i - q_i \leqslant 100$	Time interval for maintenance activity on machine i
r_c	0.8	Crossover rate for genetic algorithm
r_m	0.1	Mutation rate for genetic algorithm
r_{ls}	0.15	Local search rate for genetic algorithm
P_S	500	Population size for genetic algorithm
S_T	10, 30, 60	Stop criterion for genetic algorithm
S_G	100, 300, 600	Stop criterion for genetic algorithm

Table 2 shows the effects of m, τ, and R on the performances of B&B, GA, and GA + LS when $n = 12$. For B&B, the NA column means the times node numbers exceed a hundred million among the 50 trials and their run times are measured in seconds, where NA means 'Not Available'. As m increases, the problem becomes easy to solve for B&B; the maximal mean of nodes are 2513, 3742, 1712 for $m = 2, 3, 5$, respectively. When m is small, the most difficult instance occurs at $\tau = 0.25$. However, when m is large, the most difficult instance occurs at $\tau = 0.75$. It means that earlier due dates will complicate the problem even if we have more machines. For both genetic algorithms, the relative error percentage (REP) is defined as $(f_{GA+LS} - f_{B\&B})/f_{B\&B} \times 100\%$ and $(f_{GA} - f_{B\&B})/f_{B\&B} \times 100\%$, where f means the cost obtained by the proposed algorithm GA, GA + LS, or B&B. Moreover, the NA column means the times when B&B has a cost 0 where GA or GA + LS cannot achieve the same result among the 50 trials. As shown in this table, GA + LS always has lower REP and NA values. This implies that the proposed local search indeed improves the solution quality.

Table 3 shows the effects of m, τ, and R on the performances of B&B, GA, and GA + LS when $n = 16$. The meanings of the columns are the same as those in Table 2. Note that the maximal mean of nodes are 242 548, 492 928, 942 515 for $m = 2, 3, 5$, respectively. A different rule is observed: the larger m is, the more nodes B&B has. It is totally different from that of $n = 12$. On the other hand, GA + LS still has lower REP and NA values. Without a local search, the maximal mean REP of GA rises to 133.95%. The outperformance comes at the cost of run time. In general, an average run time of less than 1 s is acceptable for $n = 16$.

Table 4 shows the effects of m, τ, and R on the performances of B&B, GA, and GA + LS for $n = 20$. The meanings of the columns are the same as those in Table 2, except for NS. The column label 'NS' means the instances are 'Not Solvable' within a hundred million nodes. A similar rule is that the larger m is, the more nodes B&B has. It is similar to that of $n = 16$. The difficult setting for B&B is that $m = 5$, $\tau = 0.75$, and $R = 0.25$, and the difficult setting for GA + LS is that $m = 5$, $\tau = 0.25$, and $R = 0.25$. Without a local search, the maximal mean REP of GA rises to 172.22%. Moreover, when $m = 5$, $\tau = 0.25$, and $R = 0.25$ the average REP of GA + LS rises to 5.83%. For the worst trial, its maximal REP is 100% (ie, $f_{B\&B} = 2$ and $f_{GA+LS} = 4$) because GA + LS terminates prematurely. Note that GA + LS takes at most 3 s on average. If a larger tolerance is allowed (eg, 500 generations of no improvement), the REP is very likely to be improved further.

Table 5 shows the solution qualities and run times of both genetic algorithms when $n \geqslant 50$. To evaluate their efficiencies, we define the relative deviation percentage (RDP) as $(f_{GA+LS} - f_{min})/f_{min} \times 100\%$ and $(f_{GA} - f_{min})/f_{min} \times 100\%$, where $f_{min} = \min(f_{GA+LS}, f_{GA})$. The NA columns indicate the times when $f_{GA+LS} > 0$ or $f_{GA} > 0$ where $f_{min} = 0$. In most situations, GA + LS achieves 0% RDP and converges within 11 s. This result means the solution quality and convergence speed are stable. Note that GA + LS spends at most 60 s to reach high-quality solutions, whereas GA cannot achieve the same results and stops after 600 fruitless generations, even if slack time remains. The result shows that the proposed local search can efficiently scrutinize a small neighbourhood. This design significantly increases the probability of obtaining optimal solutions.

Figure 3 compares the three proposed algorithms for different problem sizes. Each bar represents the average run time for a corresponding algorithm and a particular problem size (ie, n); the endpoints of a line segment represent two consecutive REPs for $n \leqslant 20$ or RDPs for $n \geqslant 50$. Although B&B always generates the optimal schedules, it cannot be applied to large instances for $n > 20$ due to its time complexity. On the other hand, GA converges very quickly, but its solution quality is poor. Its REP is up to 35% and its RDP is up to 25%. By contrast, GA + LS always generates near-optimal schedules at reasonable time costs. It implies that GA + LS is a good choice for solving this problem for $n > 20$.

Table 2 The performance of B&B, GA, and GA + LS ($n = 12$)

m	τ	R	*B&B*			
			Nodes		*Run time*	
			Mean	*Maximum*	*Mean*	*Maximum*
2	0.25	0.25	2512.86	16 902	0.02	0.19
		0.50	1103.98	17 454	0.01	0.09
		0.75	840.94	12 838	0.01	0.09
	0.50	0.25	1847.42	9060	0.02	0.06
		0.50	2270.10	14 164	0.02	0.13
		0.75	1874.06	13 215	0.02	0.11
	0.75	0.25	821.74	3060	0.01	0.05
		0.50	1099.04	4881	0.01	0.05
3	0.25	0.25	1649.86	10 056	0.02	0.08
		0.50	856.96	6499	0.01	0.08
		0.75	352.22	5390	0.00	0.05
	0.50	0.25	3742.16	11 372	0.03	0.09
		0.50	2435.14	8343	0.02	0.08
		0.75	1855.42	5722	0.02	0.05
	0.75	0.25	2788.14	7139	0.03	0.06
		0.50	2386.04	5883	0.03	0.06
5	0.25	0.25	205.66	5152	0.00	0.03
		0.50	132.56	2312	0.00	0.02
		0.75	155.26	4456	0.00	0.03
	0.50	0.25	883.30	5299	0.01	0.05
		0.50	626.16	3571	0.01	0.03
		0.75	342.46	2876	0.00	0.02
	0.75	0.25	1712.36	6061	0.01	0.05
		0.50	1442.24	5610	0.01	0.05

m	τ	R	*GA*					*GA + LS*			
			Run time		*REP*			*Run time*		*REP*	
			Mean	*Maximum*	*Mean*	*Maximum*	*NA*	*Mean*	*Maximum*	*Mean*	*Maximum*
2	0.25	0.25	0.06	0.16	5.76	75.00	1	0.18	0.27	0.00	0.00
		0.50	0.05	0.12	6.90	43.75	0	0.13	0.28	0.00	0.00
		0.75	0.04	0.16	4.28	86.21	2	0.09	0.27	0.00	0.00
	0.50	0.25	0.07	0.13	2.15	10.77	0	0.21	0.28	0.00	0.00
		0.50	0.08	0.13	2.58	13.83	0	0.23	0.33	0.00	0.00
		0.75	0.08	0.17	6.63	62.71	0	0.23	0.28	0.00	0.00
	0.75	0.25	0.07	0.11	1.28	7.20	0	0.22	0.25	0.00	0.00
		0.50	0.08	0.13	2.18	16.57	0	0.23	0.30	0.00	0.00
3	0.25	0.25	0.06	0.19	17.49	325.00	2	0.14	0.27	0.00	0.00
		0.50	0.05	0.14	14.52	160.00	4	0.12	0.31	0.00	0.00
		0.75	0.05	0.16	17.39	200.00	6	0.09	0.27	0.00	0.00
	0.50	0.25	0.08	0.14	5.83	33.33	0	0.21	0.27	0.00	0.00
		0.50	0.08	0.22	6.81	41.18	0	0.21	0.30	0.00	0.00
		0.75	0.08	0.16	17.05	200.00	3	0.20	0.28	0.00	0.00
	0.75	0.25	0.08	0.16	3.37	22.82	0	0.22	0.27	0.00	0.00
		0.50	0.08	0.14	4.87	37.96	0	0.22	0.28	0.00	0.00
5	0.25	0.25	0.03	0.11	3.36	80.00	1	0.07	0.27	0.00	0.00
		0.50	0.03	0.13	66.93	3100.00	3	0.07	0.23	0.00	0.00
		0.75	0.02	0.14	0.53	11.43	2	0.05	0.23	0.00	0.00
	0.50	0.25	0.05	0.13	13.46	400.00	2	0.13	0.25	0.00	0.00
		0.50	0.05	0.17	29.01	600.00	2	0.13	0.22	0.00	0.00
		0.75	0.05	0.20	2.93	66.67	2	0.10	0.23	0.00	0.00
	0.75	0.25	0.06	0.14	4.53	74.07	0	0.18	0.23	0.00	0.00
		0.50	0.06	0.12	5.66	65.00	1	0.17	0.25	0.00	0.00

Table 3 The performance of B&B, GA, and GA + LS ($n = 16$)

m	τ	R	*B&B*			
			Nodes		*Run time*	
			Mean	*Maximum*	*Mean*	*Maximum*
2	0.25	0.25	174 198.18	2 184 772	2.05	32.40
		0.50	231 397.56	3 024 515	2.31	33.10
		0.75	21 586.32	232 986	0.22	2.03
	0.50	0.25	124 883.78	574 045	1.54	7.18
		0.50	193 179.44	2 521 859	2.20	25.77
		0.75	242 548.48	1 105 619	3.04	22.87
	0.75	0.25	28 989.02	131 298	0.44	2.70
		0.50	100 218.36	413 208	1.43	5.90
3	0.25	0.25	492 927.98	5 009 633	5.13	44.98
		0.50	109 187.68	765 022	1.21	9.33
		0.75	35 746.68	347 253	0.37	3.46
	0.50	0.25	389 246.90	2 017 178	4.42	23.65
		0.50	252 983.12	1 297 338	2.88	13.87
		0.75	338 075.62	2 114 976	3.62	24.68
	0.75	0.25	219 396.28	1 028 681	2.95	10.45
		0.50	287 993.22	2 146 505	3.48	21.79
5	0.25	0.25	13 010.46	224 098	0.14	2.40
		0.50	9197.84	111 860	0.10	1.12
		0.75	11 203.34	188 058	0.10	1.56
	0.50	0.25	406 634.64	2 268 194	3.75	18.92
		0.50	167 650.92	1 144 552	1.80	11.25
		0.75	86 129.62	458 199	0.91	4.35
	0.75	0.25	942 514.68	3 686 284	8.18	26.97
		0.50	644 530.94	3 151 546	6.01	24.04

m	τ	R	*GA*					*GA + LS*			
			Run time		*REP*			*Run time*		*REP*	
			Mean	*Maximum*	*Mean*	*Maximum*	*NA*	*Mean*	*Maximum*	*Mean*	*Maximum*
2	0.25	0.25	0.08	0.17	10.00	83.33	0	0.34	0.87	0.08	3.74
		0.50	0.08	0.16	29.38	666.67	1	0.31	0.56	0.00	0.00
		0.75	0.06	0.23	59.22	640.00	8	0.17	0.50	0.00	0.00
	0.50	0.25	0.10	0.22	5.51	20.48	0	0.46	0.83	0.05	0.72
		0.50	0.11	0.19	9.69	46.84	0	0.47	0.64	0.05	1.49
		0.75	0.11	0.33	13.37	46.88	0	0.50	0.59	0.00	0.00
	0.75	0.25	0.12	0.23	2.63	25.46	0	0.49	0.89	0.00	0.00
		0.50	0.12	0.20	3.51	11.10	0	0.52	0.81	0.00	0.07
3	0.25	0.25	0.09	0.16	21.47	120.00	2	0.38	1.09	0.17	2.00
		0.50	0.09	0.17	88.21	1300.00	5	0.32	0.76	0.10	2.56
		0.75	0.07	0.17	71.80	960.00	15	0.16	0.94	0.22	11.11
	0.50	0.25	0.12	0.22	8.19	22.89	0	0.46	0.89	0.01	0.44
		0.50	0.12	0.23	16.92	87.50	0	0.51	0.81	0.05	1.39
		0.75	0.12	0.27	42.29	900.00	1	0.49	0.86	0.09	1.85
	0.75	0.25	0.12	0.27	5.78	16.07	0	0.54	1.06	0.03	0.48
		0.50	0.13	0.25	5.81	15.65	0	0.49	0.97	0.05	1.14
5	0.25	0.25	0.07	0.20	71.32	650.00	3	0.23	0.78	1.14	25.00
		0.50	0.08	0.27	110.62	1450.00	9	0.22	0.75	0.62	30.77
		0.75	0.08	0.25	36.65	650.00	12	0.14	0.77	0.00	0.00
	0.50	0.25	0.11	0.23	12.66	66.67	0	0.43	0.70	0.20	6.52
		0.50	0.12	0.23	73.70	780.00	2	0.45	0.87	0.39	7.58
		0.75	0.12	0.19	133.95	1300.00	1	0.47	1.05	0.19	9.33
	0.75	0.25	0.12	0.28	7.38	48.44	0	0.41	0.64	0.06	3.07
		0.50	0.11	0.20	11.38	43.31	0	0.42	0.66	0.32	6.00

Table 4 The performance of B&B, GA, and GA + LS ($n = 20$)

m	τ	R	B&B				
			Nodes		*Run time*		
			Mean	*Maximum*	*Mean*	*Maximum*	*NS*
2	0.25	0.25	21 277 508.6	92 661 286	230.2	1215.7	5
		0.50	8 097 579.2	90 659 236	90.5	913.7	2
		0.75	2 956 453.0	43 676 756	32.5	475.4	0
	0.50	0.25	19 492 329.6	96 623 440	256.7	1216.5	1
		0.50	16 716 744.6	87 244 844	241.6	1316.4	4
		0.75	17 021 575.8	80 434 652	248.2	1123.9	3
	0.75	0.25	2 947 143.2	18 142 582	53.8	366.2	0
		0.50	7 484 929.3	49 647 122	126.1	811.7	0
3	0.25	0.25	38 714 820.7	90 280 042	527.1	1291.7	15
		0.50	9 909 567.6	73 436 634	131.3	953.5	3
		0.75	1 895 627.8	72 210 738	25.6	1001.4	0
	0.50	0.25	35 834 409.8	97 932 286	551.2	1714.0	5
		0.50	33 113 844.3	95 437 582	460.4	1276.4	7
		0.75	24 901 382.2	81 492 327	371.1	1439.2	11
	0.75	0.25	18 003 463.9	89 955 893	312.7	1255.1	0
		0.50	26 766 759.6	87 316 047	424.9	1309.6	3
5	0.25	0.25	5 831 472.9	49 732 129	96.9	862.4	4
		0.50	1 223 904.3	37 862 364	18.0	486.3	0
		0.75	498 853.0	4 480 038	7.7	102.2	0
	0.50	0.25	40 578 823.5	96 125 147	573.9	1228.5	17
		0.50	33 121 494.7	94 366 229	451.3	1427.8	6
		0.75	16 625 316.6	99 097 263	214.1	1121.1	5
	0.75	0.25	45 951 724.6	98 953 413	730.3	1623.2	12
		0.50	42 947 082.8	93 046 809	673.9	1369.7	9

m	τ	R	GA					GA + LS				
			Run Time		*REP*			*Run Time*		*REP*		
			Mean	*Max*	*Mean*	*Max*	*NA*	*Mean*	*Max*	*Mean*	*Max*	*NA*
2	0.25	0.25	0.25	0.42	14.01	96.67	5	1.68	3.15	0.02	0.65	5
		0.50	0.26	0.61	89.86	1300.00	4	1.52	2.59	0.00	0.00	2
		0.75	0.19	0.53	53.42	1080.00	11	0.48	3.25	0.00	0.00	0
	0.50	0.25	0.35	0.58	5.40	36.61	1	2.31	3.65	0.03	0.50	1
		0.50	0.35	0.61	12.28	97.25	4	2.48	3.78	0.01	0.35	4
		0.75	0.39	0.95	21.29	133.33	3	3.05	5.83	0.03	0.78	3
	0.75	0.25	0.41	0.73	2.94	12.13	0	2.77	4.99	0.00	0.14	0
		0.50	0.42	0.83	3.30	11.09	0	3.09	6.16	0.00	0.16	0
3	0.25	0.25	0.31	0.70	17.48	145.95	15	1.97	5.34	0.33	3.70	15
		0.50	0.29	0.70	58.67	800.00	9	1.36	3.37	1.83	33.33	4
		0.75	0.20	0.55	12.88	190.91	15	0.33	3.70	0.23	9.68	0
	0.50	0.25	0.38	0.73	8.17	56.84	5	2.68	7.11	0.19	1.64	5
		0.50	0.42	0.69	14.24	49.27	7	2.73	6.05	0.57	8.49	7
		0.75	0.41	0.73	31.85	194.92	11	3.14	6.47	0.38	2.40	11
	0.75	0.25	0.41	0.75	4.23	14.63	0	3.17	7.74	0.06	0.97	0
		0.50	0.43	0.83	5.74	17.50	3	3.00	5.79	0.09	1.58	3
5	0.25	0.25	0.41	1.05	172.22	2600.00	12	2.22	5.40	5.83	100.00	4
		0.50	0.28	0.67	65.30	600.00	12	1.00	4.37	3.18	35.29	1
		0.75	0.29	0.77	89.74	1400.00	15	0.80	4.09	0.31	10.53	0
	0.50	0.25	0.41	0.81	13.90	30.84	17	2.74	5.49	0.85	6.90	17
		0.50	0.44	0.86	26.65	192.86	6	2.72	4.88	2.42	50.00	6
		0.75	0.43	0.75	66.42	514.29	8	2.77	5.38	2.73	28.57	6
	0.75	0.25	0.44	0.87	7.42	39.40	12	3.07	6.85	0.40	2.88	12
		0.50	0.46	1.12	10.06	35.71	9	2.87	5.63	0.36	2.22	9

Table 5 The performance of GA and GA + LS ($n \geqslant 50$)

n	m	τ	R	*GA*					*GA + LS*				
				Run time		*RDP*			*Run time*		*RDP*		
				Mean	*Maximum*	*Mean*	*Maximum*	*NA*	*Mean*	*Maximum*	*Mean*	*Maximum*	*NA*
50	5	0.25	0.25	3.39	6.19	25.61	63.79	0	53.02	60.17	0.00	0.00	0
			0.50	3.90	7.40	168.71	927.27	9	45.36	60.15	0.00	0.00	0
			0.75	3.02	5.99	663.74	11 800.00	30	7.18	60.19	0.00	0.00	0
		0.50	0.25	4.35	6.35	7.51	19.64	0	59.74	60.19	0.00	0.00	0
			0.50	4.41	9.44	15.40	40.93	0	60.05	60.22	0.00	0.00	0
			0.75	4.74	11.06	27.37	90.93	0	60.12	60.22	0.00	0.00	0
		0.75	0.25	5.00	8.81	3.29	6.16	0	60.11	60.22	0.00	0.00	0
			0.50	5.23	9.02	4.19	7.62	0	60.11	60.23	0.00	0.00	0
	10	0.25	0.25	4.07	7.60	28.45	128.26	1	58.73	60.14	0.10	4.95	0
			0.50	3.69	6.55	182.10	900.00	16	44.35	60.15	0.00	0.00	0
			0.75	2.90	6.96	226.62	2800.00	19	18.63	60.14	0.00	0.00	0
		0.50	0.25	4.92	7.58	4.63	21.21	0	59.69	60.15	0.38	5.49	0
			0.50	4.76	10.50	9.51	48.62	0	60.08	60.15	0.26	3.63	0
			0.75	4.53	8.10	17.57	73.96	0	60.10	60.17	0.22	7.44	0
		0.75	0.25	4.47	8.30	3.20	13.29	0	60.04	60.19	0.21	3.18	0
			0.50	4.78	9.75	4.46	13.35	0	60.08	60.17	0.12	3.13	0
100	5	0.25	0.25	13.79	23.29	33.28	111.59	0	60.46	60.97	0.00	0.00	0
			0.50	16.61	29.23	467.75	6816.67	5	56.03	60.93	0.00	0.00	0
			0.75	11.84	22.06	0.00	0.00	33	2.23	20.64	0.00	0.00	0
		0.50	0.25	22.17	39.83	7.66	18.03	0	60.63	61.45	0.00	0.00	0
			0.50	24.69	43.82	14.61	34.64	0	60.65	61.40	0.00	0.00	0
			0.75	28.01	51.09	24.35	74.29	0	60.92	61.98	0.00	0.00	0
		0.75	0.25	29.27	42.96	2.56	7.57	0	60.94	62.09	0.01	0.26	0
			0.50	31.29	45.68	2.82	8.70	0	61.20	62.31	0.02	0.61	0
	10	0.25	0.25	16.59	30.73	18.93	71.97	0	60.45	61.03	0.01	0.62	0
			0.50	18.27	29.72	158.96	1000.00	0	60.48	61.01	0.08	4.13	0
			0.75	16.52	30.98	793.65	6300.00	30	35.44	61.25	0.07	3.23	2
		0.50	0.25	21.19	34.26	3.91	10.05	0	60.60	61.25	0.15	2.88	0
			0.50	23.29	40.26	7.01	25.01	0	60.66	61.51	0.26	5.76	0
			0.75	23.93	47.44	7.89	29.27	0	60.81	61.71	0.51	9.94	0
		0.75	0.25	25.98	37.55	1.41	9.15	0	60.80	61.78	0.25	1.97	0
			0.50	26.64	47.64	1.45	4.99	0	60.92	62.09	0.24	2.50	0

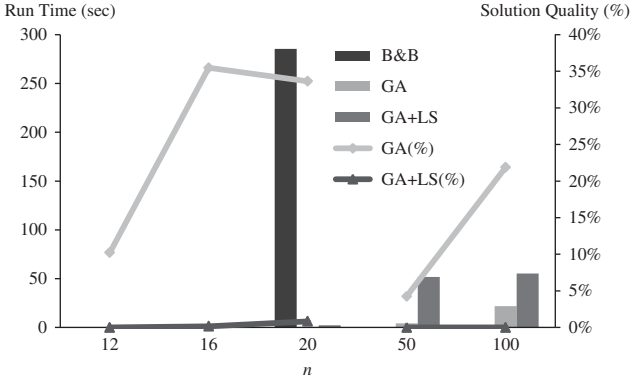

Figure 3 The effect of n on run time and solution quality.

6. A real instance

In this section, we examine a real instance in the aluminium industry. This instance concerns aluminium extrusion. In the company of Huang (2014), the main facilities are extrusion machines which extrude 7-inch hot aluminium billets and the other auxiliary facilities include furnaces, pullers, stretchers, age ovens, and so on. The main products of the company include golf club parts, parts processors, building structures, a series of windows and doors, and so on. The basic steps involved in aluminium extrusion are heating, extrusion, pulling, stretching, stacking, and aging.

Tables 6 and 7 record the raw data of the real instance. First, Table 6 shows the current statuses of three available extrusion machines during the first week of November 2014. For each machine, hydraulic oil is changed every 5000 h, oil filters are cleaned every 3 months, and air filters are changed every 6 months. During this time period, some of the extrusion machines need maintenance and some do not. Second, Table 7 lists the orders required within the first week of November 2014. Note that some jobs have the same title but may need different alloys, treatments, or processing times. Consequently, the jobs with the same title cannot be merged into a large job. Similarly, a single job cannot be divided into several small jobs.

Table 6 The statuses of three machines

Machine	*Model type*	*Maintenance item*	$[A_i, B_i]$	q_i
1	Extrusion 2100 ton	Hydraulic oil changing	3 November 2014 ~ 5 November 2014	2 days
2	Extrusion 2100 ton	Air filter changing and oil filter cleaning	7 November 2014 ~ 7 November 2014	4 h
3	Extrusion 2100 ton			

Table 7 The orders in the instance

Job title	p_j	d_j
Club head (150 pieces)	4 h	3 November 2014
Bike quick release (530 pieces)	4 h	3 November 2014
Bike quick release (100 pieces)	4 h	3 November 2014
Soundproof window (25 pieces)	2 h	3 November 2014
Airtight window handle (90 pieces)	4 h	3 November 2014
Window (51 pieces)	2 h	4 November 2014
Sofa leg (100 pieces)	2 h	4 November 2014
CNC lathe (10 pieces)	3 days	4 November 2014
Window (10 pieces)	2 h	5 November 2014
Club head (125 pieces)	4 h	5 November 2014
Cooling fin (100 pieces)	4 h	5 November 2014
Airtight window handle (120 pieces)	2 h	5 November 2014
Soundproof window (16 pieces)	2 h	5 November 2014
Soundproof window (24 pieces)	4 h	5 November 2014
Club head (100 pieces)	4 h	6 November 2014
Club head (70 pieces)	4 h	6 November 2014
Table leg (40 pieces)	2 h	6 November 2014
Bike handle bar (65 pieces)	8 h	7 November 2014
Bike quick release (400 pieces)	4 h	7 November 2014
Soundproof Door (10 pieces)	4 h	7 November 2014
Window (40 pieces)	2 h	7 November 2014
Soundproof door (20 pieces)	6 h	7 November 2014
Window (16 pieces)	2 h	7 November 2014
Door (80 pieces)	6 h	7 November 2014
Soundproof door (11 pieces)	4 h	7 November 2014
Forging part (100 parts)	4 h	7 November 2014
Club head (150 pieces)	4 h	7 November 2014
Cooling fin (100 pieces)	6 h	7 November 2014

Table 8 The final job list

j	*Job title*	p_j	d_j
1	Maintenance activity on machine 1	$q_1 = 16$	$[A_1, B_1] = [0, 24]$
2	Maintenance activity on machine 2	$q_2 = 4$	$[A_2, B_2] = [32, 40]$
3	Maintenance activity on machine 3	$q_3 = 0$	$[A_3, B_3] = [0, 40]$
4	Club head (150 pieces)	4	8
5	Bike quick release (530 pieces)	4	8
6	Bike quick release (100 pieces)	4	8
7	Soundproof window (25 pieces)	2	8
8	Airtight window handle (90 pieces)	4	8
9	Window (51 pieces)	2	16
10	Sofa leg (100 pieces)	2	16
11	CNC lathe (10 pieces)	24	24
12	Window (10 pieces)	2	24
13	Club head (125 pieces)	4	24
14	Cooling fin (100 pieces)	4	24
15	Airtight window handle (120 pieces)	2	24
16	Soundproof window (16 pieces)	2	24
17	Soundproof window (24 pieces)	4	24
18	Club head (100 pieces)	4	32
19	Club head (70 pieces)	4	32
20	Table leg (40 pieces)	2	32
21	Bike handle bar (65 pieces)	8	40
22	Bike quick release (400 pieces)	4	40
23	Soundproof door (10 pieces)	4	40
24	Window (40 pieces)	2	40
25	Soundproof door (20 pieces)	6	40
26	Window (16 pieces)	2	40
27	Door (80 pieces)	6	40
28	Soundproof door (11 pieces)	4	40
29	Forging part (100 parts)	4	40
30	Club head (150 pieces)	4	40
31	Cooling fin (100 pieces)	6	40

Before proceeding scheduling, some preparation is needed. Because these machines are not functioning 24 h each day and 7 days per week, the due dates and processing times need to be transformed to hours. The final job list is shown in Table 8. For convenience, we let the first three jobs be the maintenance activities. Since the third machine does not need maintenance in this time period, assume its processing time is 0 and maintenance window is $[0, 40]$. In this instance, there are three machines and 31 jobs and the total processing time is up to 144 h. The total capacity of the three machines in this week is just 120 h, so there must be some tardiness.

Since the problem size is larger than 30, we employ the two proposed genetic algorithms. The final results achieved by the two algorithms are as follows. GA takes 0.67 s and generates the schedule $\pi_1 = (7, 4, <0>, 13, 17, 27, 11, 0, 14, 5, 30, 9, 26, 18, 25, 28, 22, <0>, 0, 6, 8, 20, 10, 15, 19, 16, <0>, 24, 12, 21, 23, 29, 31)$ with a total cost of 50. GA + LS takes 2.04 s and generates the schedule $\pi_2 = (6, 5, <0>, 19, 18, 24, 26, 23, 25, 0, 7, 8, 11, 30, <0>, 22, 21, 0, 16, 9, 4, 15, 10, 14, <0>, 17, 12, 13, 20, 28, 29, 31, 27)$ with a total cost of 36. Note that the zeros represent separators and maintenance activities.

Compared with GA, GA + LS achieves better solution quality, though it takes a little more run time. For the schedule π_2, there is no idle time and there are no small jobs processed across 2 days. It implies no overtime work. Moreover, it leads to a lower tardiness cost. This improvement in solution quality can be creditable to the proposed local search.

In sum, GA + LS has the following merits. First, it can easily take the maintenance activities into consideration. Second, it can generate near-optimal schedules. Third, the local search actually improves the solution quality. Finally, a real problem instance can be easily transformed into the theoretical model. It implies that GA + LS is qualified to solve the real problem instances in daily life.

7. Conclusion

In this study, a scheduling problem with maintenance activities is considered. The objective is to minimize the total tardiness of all regular jobs. A brand-and-bound algorithm (B&B) and a genetic algorithm (GA) equipped with a local search are designed to solve the problem. Note that the maintenance activities can be performed within predefined intervals only. This constraint makes the problem NP-hard. Therefore, a useful lower bound is also proposed for accelerating the convergence speed of B&B. As the computational results show, B&B can achieve optimality for most instances of 20 jobs. For the larger instances, the results show that GA + LS can provide high-quality schedules quickly. In the future, we may consider a situation in which only a few maintenance activities can be performed concurrently due to limited resources.

Acknowledgements—The authors are grateful to the editor and the referees, whose constructive comments have led to a substantial improvement in the presentation of the paper. We would also thank T.B. Huang for giving us the real-world application. The first author was supported under NSC 100-2221-E-035-029-MY3 and the second author was supported under NSC 102-2221-E-241 -018.

References

Adiri I, Frostig E and Rinnooy Kan AHG (1991). Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs. *Naval Research Logistics* **38**(2): 261–271.

Azizoglu M and Kirca O (1998). Tardiness minimization on parallel machines. *International Journal of Production Economics* **55**(2): 163–168.

Chen WJ (2009). Minimizing number of tardy jobs on a single machine subject to periodic maintenance. *Omega* **37**(3): 591–599.

Chen CC (2014). *TOYOTA: Let's Go Places*, http://www.toyota.com.tw/, accessed on 1 December 2014.

Cheng TCE, Hsu CJ and Yang DL (2011). Unrelated parallel-machine scheduling with deteriorating maintenance activities. *Computers & Industrial Engineering* **60**(4): 602–605.

Du J and Leung JYT (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research* **15**(3): 483–495.

Ghodratnama A, Rabbani M, Tavakkoli-Moghaddam R and Baboli A (2010). Solving a single-machine scheduling problem with maintenance, job deterioration and learning effect by simulated annealing. *Journal of Manufacturing Systems* **29**(1): 1–9.

Goldberg DE and Lingle R (1985). Alleles, loci and the traveling salesman problem. In: *Proceedings of an International Conference on Genetic Algorithms and Their Application*. Lawrence Erlbaum Associates: Hillsdale, NJ, pp 154-159.

Gustavsson E, Patriksson M, Stromberg AB, Wojciechowski A and Onnheim M (2014). Preventive maintenance scheduling of multi-component systems with interval costs. *Computers & Industrial Engineering* **76**: 390–400.

Huang TB (2014). Cheng Drong Aluminium Co., Ltd., http://www.cdal.com.tw/, accessed on 1 December 2014.

Joo CM and Kim BS (2013). Genetic algorithms for single machine scheduling with time-dependent deterioration and rate-modifying activities. *Expert Systems with Applications* **40**(8): 3036–3043.

Lee JY and Kim YD (2012). Minimizing the number of tardy jobs in a single-machine scheduling problem with periodic maintenance. *Computers & Operations Research* **39**(9): 2196–2205.

Lee CY and Liman SD (1992). Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica* **29**(4): 375–382.

Lopez EG and O'Neill M (2009). On the effects of locality in a permutation problem: The Sudoku puzzle. In: *IEEE Symposium on Computational Intelligence and Games*. 7–10 September, IEEE: Milano, Italy, pp 80–87.

Luo W, Cheng TCE and Ji M (2015). Single-machine scheduling with a variable maintenance activity. *Computers & Industrial Engineering* **79**: 168–174.

Ma Y, Chu C and Zuo C (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering* **58**(2): 199–211.

Madhushini N, Rajendran C and Deepa Y (2009). Branch-and-bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs. *Journal of the Operational Research Society* **60**(7): 991–1004.

Pinedo ML (2008). *Scheduling: Theory, Algorithms, and System*. 3rd edn, Prentice-Hall: New Jersey.

Qi X, Chen T and Tu F (1999). Scheduling the maintenance on a single machine. *Journal of the Operational Research Society* **50**(10): 1071–1078.

Sbihi M and Varnier C (2008). Single-machine scheduling with periodic and flexible periodic maintenance to minimize maximum tardiness. *Computers & Industrial Engineering* **55**(4): 830–840.

Schaller J and Valente JMS (2012). An evaluation of heuristics for scheduling a non-delay permutation flow shop with family setups to minimize total earliness and tardiness. *Journal of the Operational Research Society* **64**(6): 805–816.

Schmidt G (2000). Scheduling with limited machine availability. *European Journal of Operational Research* **121**(1): 1–15.

Shen L, Wang D and Wang XY (2013). Parallel-machine scheduling with non-simultaneous machine available time. *Applied Mathematical Modelling* **37**(7): 5227–5232.

Shim SO and Kim YD (2007). Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Operational Research* **177**(1): 135–146.

Wang JB and Wei CM (2011). Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties. *Applied Mathematics and Computation* **217**(20): 8093–8099.

Xu D and Xiong S (2012). Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. *Journal of the Operational Research Society* **63**(4): 567–567.

Yang SJ (2010). Single-machine scheduling problems with both start-time dependent learning and position dependent aging effects under deteriorating maintenance consideration. *Applied Mathematics and Computation* **217**(7): 3321–3329.

Yang SJ (2012). Single-machine scheduling problems simultaneously with deterioration and learning effects under deteriorating multi-maintenance activities consideration. *Computers & Industrial Engineering* **62**(1): 271–275.

Yang SJ (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling* **37**(5): 2995–3005.

Yang SL, Ma Y, Xu DL and Yang JB (2011). Minimizing total completion time on a single machine with a flexible maintenance activity. *Computers & Operations Research* **38**(4): 755–770.

Yin YQ, Wu WH, Wu WH and Wu CC (2014). A branch-and-bound algorithm for a single machine sequencing to minimize the total tardiness with arbitrary release dates and position-dependent learning effects. *Information Sciences* **256**: 91–108.

Yu X, Zhang Y, Xu D and Yin Y (2013). Single machine scheduling problem with two synergetic agents and piece-rate maintenance. *Applied Mathematical Modelling* **37**(3): 1390–1399.

Received 16 June 2014;
accepted 6 March 2015 after two revisions