Complex & Intelligent Systems (2022) 8:199–212
https://doi.org/10.1007/s40747-021-00355-4

ORIGINAL ARTICLE

A two-agent one-machine multitasking scheduling problem solving by exact and metaheuristics

Chin-Chia Wu[1] · Ameni Azzouz[2] · Jia-Yang Chen[1] · Jianyou Xu[3] · Wei-Lun Shen[1] · Lingfa Lu[4] · Lamjed Ben Said[2] · Win-Chin Lin[1]

Received: 19 October 2020 / Accepted: 25 March 2021 / Published online: 15 April 2021

Abstract
This paper studies a single-machine multitasking scheduling problem together with two-agent consideration. The objective is to look for an optimal schedule to minimize the total tardiness of one agent subject to the total completion time of another agent has an upper bound. For this problem, a branch-and-bound method equipped with several dominant properties and a lower bound is exploited to search optimal solutions for small size jobs. Three metaheuristics, cloud simulated annealing algorithm, genetic algorithm, and simulated annealing algorithm, each with three improvement ways, are proposed to find the near-optimal solutions for large size jobs. The computational studies, experiments, are provided to evaluate the capabilities for the proposed algorithms. Finally, statistical analysis methods are applied to compare the performances of these algorithms.

Keywords Cloud simulated annealing algorithm · Genetic algorithm · Multitasking · Scheduling · Simulated annealing algorithm · Two agent

Introduction

Multitasking means that there are a set of tasks processed at the same time or within a limited time interval for a person or a machine (a computer, a team, a center). Multitasking phenomenon has been discussed in many disciplines, for example, behavioral psychology [39], cognitive science [6], community pharmacy [25], experimental psychology [33, 34], healthcare operations management [20], managerial behavior [42], chemical engineering [23], computer science (embedded systems, distribution system and parallel computing), Noguera and Badia [31], Appasami and Suresh Joseph [4], Kardos et al. [19], Lai et al. [24], and so on.

There are two models of multitasking [9, 38,46]. One is the concurrent mode, i.e. to perform two or more tasks at the same time, for example, media multitasking [40] and [32]. The other is the sequential mode, i.e. to perform tasks by interleaving and switching from one task to other tasks [29]. The two types of multitasking are also seen in embedded

✉ Win-Chin Lin
linwc@fcu.edu.tw

Chin-Chia Wu
cchwu@fcu.edu.tw

Ameni Azzouz
ameni.azzouz@isg.rnu.tn

Jia-Yang Chen
m0523982@mail.fcu.edu.tw

Jianyou Xu
xujianyou@mail.neu.edu.cn

Wei-Lun Shen
m0705918@mail.fcu.edu.tw

Lingfa Lu
lulingfa@zzu.edu.cn

Lamjed Ben Said
lamjed.bensaid@isg.rnu.tn

1 Department of Statistics, Feng Chia University, No. 100, Wenhwa Road, Seatwen Dist., Taichung 40724, Taiwan, ROC

2 MART Lab., Insitut Supèrieur de Gestion de Tunis, University of Tunis, 2000 Tunis, Tunisia

3 College of Information Science and Engineering, Northeastern University, Shenyang 110819, People's Republic of China

4 School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, Henan, People's Republic of China

computing system design employed in system-on-chip platforms [31].

In job scheduling domain or manufacturing field, several possible applications of multitasking were observed in, for example, administration processing [12], manufacturing [50]. In this paper, we study the sequential model of multitasking for job scheduling problem, i.e. a 'machine' (a team, a service center, and so on) processes a sequence of jobs (tasks) by interleaving and switching from one job to another during an interval of time. The operations for classical scheduling problem are to finish one job at one time then process another job, until all the required jobs are finished. The operations for multitasking scheduling problem (MSP) are also to perform a predetermined job (called a primary job) on a job sequence at one time, however, before the primary job to be fully finished, due to a certain reason, operations switch to process all the unfinished jobs (or called waiting jobs) by interleaving to finish a proportion of each waiting job, afterwards, the primary job is finished completely. Hall et al. [12,13], launched the multitasking research in the scheduling problem area, carried on by Sum and Ho [43], Zhu et al. [57, 58], Liu et al. [27], Ho and Sum [14], Li and Chong [26], and Ji et al. [18]. More recently, Zhu et al. [60] proposed optimal solution algorithms for multitasking scheduling with multiple rate-modifying activities and analyzed some special cases of them. Xiong et al. [48] applied an exact branch-and-price algorithm to solve an unrelated parallel-machine multitasking scheduling problem.

Hall et al. [12] constructed a mathematical model incorporated the multitasking into the traditional single-machine scheduling problem, and provided a summary of five motivations for multitasking. The objective criteria investigated were maximum lateness, the total weighted completion time, and total number of late jobs. Hall et al. [13] studied two models to accommodate two features that are possible happened on multitasking scheduling environment for human workers. One model employed "alternate period processing" for a disruption (and have to perform multitasks) owing to workers' tiredness situation and the other model employed "shared processing" for a distraction (and have to perform multitasks) owing to routine tasks situation. The studied criteria were the maximum lateness, the total completion time, and the number of late jobs. They discussed the complexity of models and then developed algorithms to resolve the considered problems.

Following the considered problem by Hall et al. [13], Sum and Ho [43] derived the expected total (weighted) completion time and provided statistical analysis on the performance of multitasking for a human worker under the assumptions that the switching cost is constant and the amount of each interruption is proportional to the remaining processing time of the waiting job. They gave an advice "that multitasking should be avoided in a work place". Ho and Sum [14] investigated the MSP with asymmetric switch cost functions and demonstrated that and the problem is unary NP-hard for minimization of the total completion time or a common due date assignment problem, and the problem is binary NP-hard for the makespan. They also showed that if a cost function has a special structure then the study problem can be solved in polynomial time for three common adopted criteria. Zhu et al. [57,58], explored the MSP with a (or multiply) rate-modifying activity (activities), respectively. They formulated models and developed algorithms to solve the MSP for criteria including the makespan, the total completion time, the lateness, and a multi-criteria that combines several criterions related to a common due date assignment. Zhu [57] put forward several optimal algorithms to solve the MSP with a deterioration effect to minimize the makespan and the total absolute differences in completion times. Liu et al. [27] added a common due date required for jobs into the multitasking scheduling formulation investigated by Hall et al. [12]. The objective function was a linear combination of the total earliness, the total tardiness, and a common due date assignment. They provided some theoretical results and developed an algorithm to solve the problem with linear interruption functions. Moreover, they suggested several future research topics for the MSP. Li and Chong [26] developed a single-machine multitasking scheduling model to management the progress on the processing jobs which can be split into subtasks to be processed, and also considered to check the progress at different milestones (time points). The main concern is to monitor or to control the amount of jobs been processed. Ji et al. [18], motivated from the service industry (internet service, medical service), investigated the MSP under parallel-machine setting with machine-dependent slack due window scenario. The objective was to minimize a total cost function which is a combination of criteria of the earliness, the tardiness and the time related to slack due window assignment. The main contribution of them was to represent the interruption function as a more general form that can reflect real service or manufacturing situations.

On the other hand, two-agent or multi-agent scheduling problem has been studied in the recent about 20 years. For example, Baker and Smith [5] and Agnetis et al. [1] are among the pioneers on this topic. In a two-agent single machine scheduling problem, different agents share one processing resource (machine) and each agent demands to optimize its own optimality criterion concerning to its own set of jobs. As the agents have its own optimality criterion, in general, they have to compete for sharing a collaborative resource to make the best their own criterion. Baker and Smith [5] investigated the problems of minimizing an aggregate of different optimality criteria adopted by two agents, in which the optimality criteria considered were the total weighted completion time, the maximum lateness, and the makespan. Depending on the optimality criterion of each agent and on the processing environment (single machine or job shop), Agnetis et al. [1] addressed a constrained optimization problem, in which one agent have to optimize its optimality criterion with respect to its own jobs subject to a constraint on the other agent's objective function. Since then, study on the multi-agent job sequencing problem has been developed in the area of scheduling. For more discussing on two-agent or multi-agent scheduling research, the reader may refer, Mor and Mosheiov [30], Gerstl and Mosheiov [10], Kovalyovet al. [22], Yin et al. [51, 52, 53, 54], Li et al. [28], Zhang and Wang [56], Ahmadi-Darani et al. [3] a review paper by Perez-Gonzalez and Framinan [35], and the book by Agnetis et al. [2]. More recently, Yang et al. [49] introduced due date assignment into a two-agent scheduling problem to minimize the completion times of given jobs. Wang et al. [45] studied several multitasking scheduling models with multiple agents in which the objective functions were included the maximum of a regular function (associated with each task), the total completion time, and the weighted number of late jobs.

To the best of authors' knowledge, there is no research on the machine multitasking scheduling problem (MSP) together with two (or more) agents. This study tries to bridge the gap, and also responds to one of the further research problems—"Second, the classical scheduling literature contains a large number of problems that remain to be studied in the presence of multitasking." mentioned in Hall et al. [12]. Moreover, one of the future research directions in Agnetis et al. [1], in a multi-agent scheduling problems environment, is to "Extension to different resource usage modes (concurrent usage, preemption, etc.) and/or different system structure (e.g. parallel machines)." Therefore, we study the alliance of these two emerging topics in the scheduling problem area. The objective is to minimize the total tardiness of first agent's jobs subject to an upper bound constraint on the total completion time of the second agent's jobs.

The rest of this paper is organized as follows. The problem statement is given in "Problem statement". In "Properties and a lower bound", we derive several dominance properties and a lower bound for the study problem. In "Metaheuristics GA, SA, and CSA", three commonly used metaheuristics [cloud simulated annealing algorithm (CSA), genetic algorithm (GA), simulated annealing algorithm (SA)] are adapted as tools to search (approximate) optimal solutions. "Data simulation analysis" is analysis of experimental simulation data, and conclusion and suggestions are in "Conclusion and suggestions".

Problem statement

The proposed multitasking scheduling problem (MSP) in this study is formulated as follows. Consider a set of n jobs, $\left\{J_1, J_2, \ldots, J_n\right\}$, to be processed by a machine. For convenient explanations, sometimes we use the job i or i to represent J_i. The machine can operate at most one job at a time. Suppose that there are two competing agents denoted, respectively, by agents A and B. Besides, let $X \in \{A, B\}$ be the agent code. The jobs belong to agent X are presented as X-jobs. Let J_A and J_B denote the sets of jobs from agent A (or A-jobs) and jobs from agent B (or B-jobs), respectively, meanwhile let N_a and N_b denote the number of A-jobs in J_A and B-jobs in J_B. That is $J_A = \left\{J_1^A, J_2^A, \ldots, J_{N_a}^A\right\}$, $J_B = \left\{J_1^B, J_2^B, \ldots, J_{N_b}^B\right\}$, and $J_A \cup J_B = \left\{J_1, J_2, \ldots, J_n\right\}$. Both of jobs in the set $\{A, B\}$ are available at the begin of operation (time zero). Both of the processing time (t_j^X) and due date (d_j^X) for a job J_j^X are nonnegative integers. For a job schedules σ, let $C_j^X(\sigma)$, or C_j^X, be the completion time of job J_j, and $T_j^X = \max\{C_j^X(\sigma) - d_j^X, 0\}$ be the tardiness of job j as in the job sequence σ.

The serial (sequential) multitasking environment means that when a job is being processed, it is fractured by other unfinished jobs that are waiting for processing. Every job is assigned at any time point is represented as the primary job, while the jobs that are ready and not fully processed are denoted as waiting jobs. Under the multitasking situation, as a primary job is being processed, it is interrupted by all the waiting jobs. Two more assumptions are required to model the process on which the waiting jobs interrupt the primary jobs. One is called "interruption time" that is the time during which a waiting job interrupts the primary job, which is independent of the characteristics of the primary job. Let $t_i^{X\prime}$ be the not yet finished processing time of a waiting job i at the beginning of a time point when job j is scheduled as a primary job. In this study, the amount of time for which job i interrupts job j is given by $g(t_i^{X\prime}) = Dt_i^{X\prime}$ as in Hall et al. (12), where D is a ratio of interruption (time) cost with $0 < D < 1$. The other assumption is called "switching time" that is the amount of time prepared to deal with the primary job is scheduled, which depends only on the number of waiting jobs. Let S_j be the set of waiting jobs when j is assigned to be the primary job and $f(\left|S_j\right|)$ be the amount of switching time for processing the interrupting jobs. In this study, the switching time is adopted as $f\left(\left|S_j\right|\right) = |S_j|$, which depends only on the number of waiting jobs. The objective function of this study is to minimize the total tardiness of all the jobs of the agent A subject to an upper bound, say Q, required on the total completion time of jobs of agent B. The three-field notation for this problem in scheduling domain can be denoted as $1/\text{'mt'}/\sum T_j^A : \sum C_j^B \le Q$, where 'mt' represents multitasking.

Properties and a lower bound

The proposed problem is NP-hard, since the classical problem without agent concept or multitasking phenomenon is NP-hard (Pinedo 2014). To efficiently deal with this problem, a branch-and-bound (B&B) algorithm is devised to find the optimal solutions for small-size instances and three metaheuristics are then utilized to quickly find the approximate solutions for relatively large-size instances. It is noted that the dominance properties act a considerable role in a B&B algorithm for searching an exact solution (schedule) because many lower nodes of the instances can be cut down by applying these dominant properties. To help speeding up the search process of the B&B, several dominance properties are established in this section. Let $\sigma = (\pi_1, i, j, \pi_2)$ and $\sigma' = (\pi_1, j, i, \pi_2)$ be two job sequences (schedules) in which π_1 and π_2 are two parts of subsequences with k jobs and (n-k-2) jobs, respectively. The switching cost function is $f\left(\left|\left\{i, j, \pi_2\right\}\right|\right) = n - k, k{=}n{-}1, n{-}2, \ldots, 1$, and s denote the staring time of job i in $\sigma = (\pi_1, i, j, \pi_2)$ and job j in $\sigma' = (\pi_1, j, i, \pi_2)$. D denotes the ratio of interruption cost, where $t_i^{X\prime} = D \times t_i^X$. To show that σ dominates σ', the following suffices: $\max\left\{C_i^A(\sigma) - d_i^A, 0\right\} + \max\left\{C_j^A(\sigma) - d_j^A, 0\right\}$, $< \max\left\{C_j^A\left(\sigma'\right) - d_j^A, 0\right\} + \max\left\{C_i^A\left(\sigma'\right) - d_i^A, 0\right\}$ and $C_j^A(\sigma) < C_i^A\left(\sigma'\right)$, where $C_j^X(\sigma) = s + (1-D)^{k-1} t_j^X + f(k) + \sum_{v \in \pi_2 \bigcup \{j\}} D(1-D)^{k-1} t_v^X$. The proofs of following properties are left out because they can be proved by applying a pairwise interchange method.

Property 1 *If* $s + (1-D)^{k-1} t_i^A + f(k) + \sum_{v \in \pi_2 \cup \{j^A\}} D(1-D)^{k-1} t_v^X < d_i^A < s + (1-D)^{k-1} t_j^A + f(k) + \sum_{v \in \pi_2 \cup \{i^A\}} D(1-D)^{k-1} t_v^X + (1-D)^k t_i^A + f(k+1) + \sum_{v \in \pi_2} D(1-D)^k t_v^X$ *and* $\max\{s + (1-D)^{k-1} t_j^A + f(k) + \sum_{v \in \pi_2 \cup \{i^A\}} D(1-D)^{k-1} t_v^X, s + (1-D)^{k-1} t_i^A + f(k) + \sum_{v \in \pi_2 \cup \{j^A\}} D(1-D)^{k-1} t_v^X + (1-D)^k t_j^A + f(k+1) + \sum_{v \in \pi_2} D(1-D)^k t_v^X\} < d_j^A$, *then* σ *dominates* σ'.

Property 2 *If* $s + (1-D)^{k-1} t_i^A + f(k) + \sum_{v \in \pi_2 \cup \{j^A\}} D(1-D)^{k-1} t_v^X > d_i^A$ *and* $s + (1-D)^{k-1} t_i^X + f(k) + \sum_{v \in \pi_2 \cup \{j^A\}} D(1-D)^{k-1} t_v^X + (1-D)^k t_j^A + f(k+1) + \sum_{v \in \pi_2} D(1-D)^k t_v^X \leq d_j^A$, *then* σ *dominates* σ'.

Property 3 *If* $s + (1-D)^{k-1} t_i^A + f(k) + \sum_{v \in \pi_2 \cup \{j^A\}} D(1-D)^{k-1} t_v^X > d_i^A$, $s + (1-D)^{k-1} t_j^A + f(k) + \sum_{v \in \pi_2 \cup \{i^A\}} D(1-D)^{k-1} t_v^X > d_j^A$, $d_i^A < d_j^A$, *and*, $d_j^A - d_i^A < t_j^A - t_i^A$, *then* σ *dominates* σ'.

For any uncompleted node (*PS*, *US*) in which *PS* is already assigned partial sequence with k jobs and *US* is an undecided subset of jobs with (n-k) jobs, the following two properties hold.

Property 4 *Let* $C_{[l]}^B$ *be the completion time of the last job in PS. If* $C_{[l]}^B > Q$, *then node* (*PS, US*) *should be an infeasible schedule.*

Property 5 *Let* $C_{[k]}$ *be the completion time of the last job in PS. If there is any B job in US and* $C_{[k]} + (1-D)^{k-1} t_j^B + f(k) + \sum_{v \in US \backslash \{j^B\}} D(1-D)^{k-1} t_v^X > Q$, *then node* (*PS, US*) *should be eliminated.*

The performance of the B&B algorithm is also related to a good lower bound. In what follows, a simple lower bound will be presented. Assume that PS is a partial sequence in which the order of the first k jobs is determined and US be the unscheduled part with $(n-k)$ jobs. Furthermore, let $n - k = n_A + n_B$, where n_A denote the number of A-jobs while n_B denote the number of B-jobs in *US*. Let σdnote a complete sequence obtained from *PS*. The completion time for the $(k+1)$th job to the $(k+n_A)$th position in a $\sigma = $ (PS, US), if the n_A A-jobs schedule on the first n_A positions in US, is given by

$$C_{[k+1]}^A(\sigma) = s + (1-D)^k t_{[k+1]}^A + f(k+1) + \sum_{v \in \mathrm{US} \backslash \{j_{[k+1]}\}} D(1-D)^k t_v^X,$$

$$\begin{aligned} C_{[k+2]}^A(\sigma) = s &+ (1-D)^k t_{[k+1]}^A + f(k+1) \\ &+ \sum_{v \in \mathrm{US} \backslash \{j_{[k+1]}^A\}} D(1-D)^k t_v^X \\ &+ (1-D)^{k+1} t_{[k+2]}^A + f(k+2) \\ &+ \sum_{v \in \mathrm{US} \backslash \{j_{[k+1]}^A, j_{[k+2]}^A\}} D(1-D)^{k+1} t_v^X. \end{aligned}$$

In a similar way, we have

$$\begin{aligned} C_{[k+n_A]}^A(\sigma) = s + (1-D)^k t_{[k+1]}^A + f(k+1) &+ \sum_{v \in \mathrm{US} \backslash \{j_{[k+1]}^A\}} D(1-D)^k t_v^X \\ + (1-D)^{k+1} t_{[k+2]}^A + f(k+2) &+ \sum_{v \in \mathrm{US} \backslash \{j_{[k+1]}^A, j_{[k+2]}^A\}} D(1-D)^{k+1} t_v^X \\ \ldots \\ + (1-D)^{k+n_A} t_{[k+n_A]}^A + f(k+n_A) &+ \sum_{v \in \mathrm{US} \backslash \{j_{[k+1]}^A, j_{[k+2]}^A, \ldots, j_{[k+n_A]}^A\}} D(1-D)^{k+n_A-1} t_v^X. \end{aligned}$$

According to the above definitions, those n_A are arranged in a non-decreasing order to obtain the least amount of the completion times from n_A A-jobs in US. Next problem is to find the maximum due date (say d_{M}) from n_A A-jobs in US and assign d_{M} to be the due date for each A-job in US to yield the least amount of tardiness for the unscheduled part. It can be simplified as follows:

$$\mathrm{LB} = \sum_{i\in \mathrm{PS}} \max\left\{0, C^A_{[i]}(\sigma) - d^A_{[i]}\right\} + \sum_{i=1}^{n_A} \max\left\{0, C^A_{(k+i)}(\sigma) - d_{\mathrm{M}}\right\},$$

where $C^A_{(k+i)}(\sigma) = s + (1-D)^k t^A_{(k+1)} + f(k+1) + \sum_{v\in \mathrm{US}\backslash\left\{j^A_{[k+1]}\right\}} D(1-D)^k t^X_v$

$$+(1-D)^{k+1} t^A_{(k+2)} + f(k+2) + \sum_{v\in \mathrm{US}\backslash\left\{j^A_{[k+1]}, j^A_{[k+2]}\right\}} D(1-D)^{k+1} t^X_v$$

$$+ \cdots + (1-D)^{k+i} t^A_{(k+i)} + f(k+i) + \sum_{v\in \mathrm{US}\backslash\left\{j^A_{[k+1]}, j^A_{[k+2]},\cdots, j^A_{[k+i]}\right\}} D(1-D)^{k+i} t^X_v.$$

Metaheuristics GA, SA, and CSA

It is commonly known that the computation load can be much saved by utilizing a heuristic or metaheuristic (to quickly search a near optimal solution) to lay an (upper or lower) bound on a scheduling problem proceeding to execute a B&B (branch-and-bound) method. Thus, many research paid more attention on developing and analysis of heuristic or metaheuristic algorithms for finding optimal solutions for scheduling problems. Moreover, it is noted that the search for optimal solutions for large-size jobs is required much CPU time. However, an effective heuristic or metaheuristic can yield a good-quality approximate solution with a small margin of error and least amount of CPU time. In light of these observations, this study utilizes three metaheuristics, including the genetic algorithm (GA), the simulated annealing algorithm (SA), and the cloud simulated algorithm (CSA), to solve the study problem. These metaheuristics GA, SA, and CSA have been successfully used to deal with a wide variety of discrete combinatorial optimization problems. The main steps of GA are to start with a set of feasible population and iteratively to replace the current population by a new population throughout encode, reproduction mechanism, a crossover operator, a mutation operator, and decode (see [8, 11, 15, 41]). The details of the GA are given below.

```
procedure GA:
    Input:
    Nsize:= number of parents in a primary population, generated randomly
    Gsize := number of generation
    P:= probability of mutation
    Initialization:
        Generate Nsize parents randomly and calculate their value of fitness (objective
        function).
    Repeat
        - Generate a set, $P_N$, of Nsize parents
        - For each member in the $P_N$, decide the selection probability
        - Select two parents from the $P_N$, by the roulette wheel selection mechanic, to
          produce a set of (Nsize-1) new offspring, by applying linear order crossover
        - Keep the best schedule (parent), which has minimum of total completion time
        - Calculate the value of fitness for each member in the $P_N$
        - Reserve the best value of fitness found until now
        - Return
    Until {$i \geq Gsize$}
```

The SA algorithm by Kirkpatrick et al. [21] is the most commonly utilized metaheuristic algorithm for solving problems from combinatorial optimization. The major steps of SA are provided below.

```
Procedure SA:
Input:
      $\sigma_0$:= a schedule (randomly generated)
      $T_i$:= initial temperature
      $T_f$:= stop temperature
      $C_f$:= decay (cooling) factor
      $N_r$:= times of improvements
   Repeat {$T>T_f$}
      $k$=1
      Repeat
         Exchange randomly two jobs in $\sigma_0$ to produce a new schedule $\sigma_1$;
         Let $\Delta=TT(\sigma_1)$ - $TT(\sigma_0)$;
         If ($\Delta<0$), then set $\sigma_0=\sigma_1$; else $\sigma_0=\sigma_1$ via a probability $\exp(-\frac{\Delta}{T})$;
         $k=k+1$
      Until {$k\geq N_r$}
      Save the best schedule as yet
      $T = \lambda \cdot T_i$
   Until {$T<T_f$}
End procedure
```

The cloud simulated algorithm (CSA) put forward by Torabzadeh and Zandieh [44] is also used to resolve the study problem. The key parameters of CSA are including the start (initial) temperature T_i, stop temperature T_f, and the decay (cooling) factor λ. The major details of CSA are provided below.

```
Procedure CSA:
   Input:
      $\sigma_0$:= a schedule (randomly generated)
      $T_i$:=initial temperature
      $T_f$:= stop temperature
      $\lambda$ := decay (cooling) factor
      $N_r$:= times of improvements
Initialization:
   Let $H_e=T$, $E_n=T$, and $w=1-T$
   Set $E_n' = \max\{\varepsilon, E_n + H_e - \frac{r}{3}\}$, where $0 < \varepsilon < 10^{-8}$, $r \in$ Unif(0,1)
   Set $T' = E_n'\sqrt{-2\ln(w)}$
   Repeat
         $k$=1
      Repeat
         Exchange randomly two jobs in $\sigma_0$ to produce a new schedule $\sigma_1$;
         Let $\Delta=TT(\sigma_1)$ - $TT(\sigma_0)$;
         If ($\Delta<0$), then set $\sigma_0=\sigma_1$; else $\sigma_0=\sigma_1$ via a probability $\exp(-\frac{\Delta}{T})$;
         $k=k+1$
      Until {$j \geq N_r$}
        Save the best schedule until now
        $T = T_i \times \lambda^j$
   Until{$T<T_f$}
End procedure
```

Note that $\mathrm{TT}(\sigma_1)$ and $\mathrm{TT}(\sigma_0)$ denote the total tardiness of σ_1 and σ_0, respectively, while $\Delta = (TT(\sigma_1) - TT(\sigma_0))/TT(\sigma_0)$. To further improve the quality of the initial solutions (seeds) used in GA, SA, and CSA, three local searches were applied to improve each of the initial solutions in GA, SA, and CSA. During generating an initial solution process, those B-jobs are first generated and then A-jobs were generated and appended after B-jobs to form as a feasible initial solution. To get a good quality of solution, this initial solution is separately employed three local search methods to improve it. Those operations are the pairwise interchange (PI), extraction and backward-shifted reinsertion (EBSR), and extraction and forward-shifted reinsertion (EFSR), refer Della Croce et al. [7]. The initial solutions improved by the PI, EBSR, and EFSR are recorded as GA_p, GA_b, GA_f in GA, as SA_p, SA_b, SA_f in SA, and as CSA_p, CSA_b, CSA_f in CSA, respectively. Thus, three variants of each SA, GA and CSA are utilized in this study. Additionally, following the scheme of Wu et al. [47], the proposed properties, a lower bound, and the best solution among all proposed approximate solutions are utilized in a branch-and-bound method (B&B).

For further testing, all parameters in proposed nine algorithms, we examined parameters including the initial temperature (T_i), cooling factor (C_f), and number of improvement (N_r) in three SAs, three parameters including the initial temperature (T_i), annealing index (λ) and number of improvement (N_r) in three CSAs, and the number of parents (N_size), genetic generation (G_size) and mutation rate (P) in three GAs, respectively. When fixed $n = 12$, we randomly generated 100 instances and utilized all nine algorithms and a B&B method equipped with the dominance properties derived in "Properties and a lower bound" to solve them, and calculated the average error percentage (AEP) for the differences between the results yielded from nine algorithms and those yielded from the B&B method for determining appropriate parameters.

In our pretests, we found that the larger values of tardiness factor (τ), due date range (ρ), number of B-jobs (N_B) are important factors that affected the algorithms to find a feasible schedule. The smaller value of interruption factor D and a control variable, Qlevel (refer "Data simulation analysis"), for bound above total completion time of B-jobs also had effects on finding a feasible solution. In view of this observation, the initial tested instances are generated by setting a design at $n = 12$, $N_B = 10, D = 0.001, \rho = 0.75, \tau = 0.50$ and Qlevel = 1.6. After parameters tuning process, the (T_i, C_f, N_r) is adopted as (0.85, 0.4, 20) in three SAs, the (T_i, λ, N_r) as (0.65, 0.3, 30) in three CSAs, the (N_size, G_size, P) as (32, 140, 0.09) in three GAs, in the later experimental tests.

Data simulation analysis

The processing times of A-jobs, t_j^A, or B-jobs, t_j^B, on a machine are generated randomly from uniformly distributed on integers between 1 and 100. The due date for each

A-job is randomly generated from uniform distributions $T_A \times U\left(1-\tau-\frac{\rho}{2}, 1-\tau+\frac{\rho}{2}\right)$, where $T_A = \sum_{j\in J_A} t_j^A$, the control parameters of the due dates are tardiness factor τ and range of due dates ρ. In particular, τ is set to 0.25 and 0.50, and ρ is set to 0.25, 0.50, and 0.75. Regarding the setting of the upper bound for the B-jobs, we designate $Q = \text{Qlevel} \times \sum_{j\in J_B} t_j^B$, in which the control parameters are Qlevel. Another parameter relevant to two-agent scenario is the number of B-jobs, N_B. In this study for small number of jobs, fixed $n=12$, three types of Qlevel is set at 1.6, 1.7, and 1.8, while N_B is set to be 2, 4, 6, 8 and 10, and the levels of the interruption factor D are set at 0.1, 0.01, and 0.001. In total, there are 270 ($D \times \tau \times \rho \times \text{Qlevel} \times N_B = 3\times 2\times 3\times 3\times 5$) cases of test combinations and 100 instances are generated for each case. Note that if the number of nodes in B&B algorithm exceeds 10^8, it is considered as a failure and turns to next instance, otherwise, it is recorded as a feasible case. The performances of the branch-and-bound method and the proposed nine algorithms over different parameters are summarized in Tables 1, 2, and 3.

As shown in Table 2, it can be seen that B&B algorithm consumes fewer nodes as the interruption factor, D, is at higher value ($D=0.1$). As for the impact of parameters τ and ρ, Table 2 shows that, on average, B&B method takes fewer nodes or run less CPU time at the big value of τ or at a small value of ρ. Regarding the impact of N_B over different size, it can be seen in Table 2 that the average numbers of nodes have significant changes, this is possibly, because based on the fact that B-jobs have an upper limit constraint, the size of the searching space directly shrinks as the big number of N_B, say at 10. In other words, a smaller number of A-jobs reduce the search difficulty, so the B&B algorithm can complete the search process in fewer nodes. In contrast, as the values of Qlevel increases, the average number of nodes is gradually decreased, but the range of fluctuation affected by Qlevel is less than that affected by N_b.

Regarding the performance of all proposed nine algorithms, as show in Table 3, Figs. 1, 2, and 3, all nine algorithms performed better at a big value of D or τ than those at a small value of D or τ. For the value of D at 0.1, it not only has better AEP performance, but also has a smaller variance on AEP than those of D at 0.01 and 0.001. All nine algorithms performed better at a small value of ρ than those at a big value of ρ.

Over all, GA_f, GA_b, and GA_p performed worse on average AEP than the rest of the algorithms did.

Regarding the effect of parameters Qlevel and N_b on the AEP, it can be seen from the box plots in Fig. 4 that for different levels of Qlevel the dispersions on the AEP is not too large, meaning that the search ability for (near-) optimal solutions were about the same for all 9 algorithms. Meanwhile, as shown in Fig. 5, the impact on AEP for different N_b, there are two trends occurred in three GAs, where median and IQR of AEPs increase significantly as N_b increases from 2 to 8, but AEP declines sharply as N_b is up to 10. The other trend occurred in three SAs and three CSAs, where the median of AEPs is about 1–3%, and the variances is in general less than that of GAs. This is due to the characteristic that GA is more sensitive to the initial solutions. As the value of N_b is close to 6, the search space for A-jobs is wider than the other values of N_b. Therefore, that only using simple pairwise, forward or backward two-point interchange methods cannot provide good initial solutions, resulting in poor GA performance on the AEPs.

As shown in Table 3 and Fig. 6, GA_f, GA_b, and GA_p generally have larger IQR and three CSAs perform better on average AEP than other six algorithms do.

To examine whether the difference between nine algorithms is statistically significant or not, we used the analysis of variance (ANOVA) on AEPs and found that the observations from the experiments are not followed normal distributions. Thus, we performed the Kruskal–Wallis test (a nonparametric statistical method) to examine the statistical differences. The third column of Table 4 showed the mean rank of AEPs for 9 algorithms. The p value was <0.001 (and the value of test statistic, Chi-Square distribution, was 198.6 with 8 degrees of freedom) of the Kruskal–Wallis (K–W) test, thus the differences among the performances of the nine algorithms are statistically significant.

For further multiple comparison the performances among the 9 algorithms, the DSCF method (the Dwass–Steel–Critchlow–Fligner procedure, Holland [15]) was employed and the algorithms were grouped (run on SAS 9.4). As shown in Table 5, it can be seen that the three initial solutions used in GA, SA or CSA have no significant difference on AEP, but AEPs of GA and SA are significantly different with that of CSA. The CSAs did the best performance in term of the mean of AEP.

For the large number of jobs, fixed $n=60$, three levels of D is set at 0.1, 0.01, and 0.001, three types of Qlevel is set at 1.6, 1.7, and 1.8, while N_b is set at 10, 20, 30, 40, and 50. In total, there are 270 test combinations and 100 instances are generated for each combination. The results were summarized in Table 6.

It can be observed from Table 6 that the three initial solutions used in GA, SA or CSA have no significant difference on the RPD. The CSA seems have differences from both GA and SA; the CSAs did the best performance in term of the average of RPD.

As shown in Table 6 and Fig. 7, it can be seen that when the level of interruption factor is at high level (i.e. $D=0.1$), all nine algorithms have smaller RPDs and smaller dispersions. It means that for a bigger value of D, proposed algorithms not only have a better RPD performance, but also have a lower

Table 1 Performances of B&B over different parameters ($n = 12$)

D	Qlevel	N_b	Node		CPU times	
			Mean	Max	Mean	Max
0.1	1.6	2	1,262,030.02	5,867,625	18.01	68.8
		4	1,265,873.77	6,288,987	19.15	88.05
		6	19,391.90	1,794,245	0.42	27.11
		8	287.64	4343	0.01	0.11
		10	51.24	85	0	0.02
	1.7	2	1,341,512.94	5,867,649	19.53	69.73
		4	1,109,542.67	7,408,941	17.29	103.44
		6	13,696.56	802,057	0.34	14.29
		8	287.77	4344	0.01	0.08
		10	51.31	85	0	0.02
	1.8	2	1,468,471.79	5,430,421	21.73	67.22
		4	831,246.54	9,578,716	13.53	133.43
		6	13,696.62	802,057	0.34	14.4
		8	287.81	4345	0.01	0.09
		10	51.36	85	0	0.02
0.01	1.6	2	1,222,165.50	3,408,420	16.42	43.23
		4	1,572,122.14	8,613,239	20.04	94.15
		6	5,256,577.99	24,942,703	56.24	262.69
		8	1,292,546.22	23,157,617	10.75	192.35
		10	13.22	86	0	0.02
	1.7	2	1,414,446.01	4,334,821	18.66	56.11
		4	2,071,946.32	10,658,221	25.93	135.61
		6	3,567,067.93	39,031,541	37.28	419.32
		8	295,887.48	23,816,151	2.37	162.15
		10	13.22	86	0	0.02
	1.8	2	1,617,922.93	5,174,566	21.32	70.81
		4	2,710,071.00	13,718,005	33.79	163.49
		6	2,270,214.58	23,029,503	22.9	226.69
		8	1517.08	70,007	0.03	0.77
		10	13.22	86	0	0.02
0.001	1.6	2	1,304,847.14	4,140,059	16.14	46.14
		4	1,570,536.46	10,411,040	18.27	94.99
		6	6,221,690.88	24,838,378	60.49	222.75
		8	2,712,333.58	54,472,393	19.9	325.34
		10	32.95	154	0	0.02
	1.7	2	1,491,661.00	4,844,055	17.07	46.2
		4	2,173,290.81	11,929,326	23.3	101.52
		6	5,144,872.28	32,920,768	45.9	270.27
		8	1,626,005.25	32,465,999	11.42	205.28
		10	32.99	154	0	0.02
	1.8	2	1,717,811.47	6,207,522	19.14	62.03
		4	2,819,090.68	15,858,631	28.96	137.3
		6	4,557,010.50	53,091,766	38.32	403.41
		8	164,245.73	23,330,099	1.15	137.69
		10	33.01	154	0	0.02
		Mean	1,380,499.99	11,073,767.44	15.03	99.27

Table 2 Performances of B&B over four parameters ($n = 12$)

	Node	CPU_time	FS
D			
0.001	2,100,232.981	20.003	100
0.01	1,552,834.989	17.714	100
0.1	488,431.997	7.358	100
Qlevel			
1.6	1,580,033.377	17.055	100
1.7	1,350,020.969	14.606	100
1.8	1,211,445.621	13.414	100
τ			
0.25	1,763,968.097	17.713	100
0.5	997,031.881	12.337	100
ρ			
0.25	1,316,927.039	12.860	100
0.5	1,365,103.422	15.548	100
0.75	1,459,469.507	16.667	100
N_b			
2	1,426,763.200	18.669	100
4	1,791,524.488	22.249	100
6	3,007,135.470	29.136	100
8	677,044.286	5.070	100
10	32.502	0.001	100

degree of variation on the values of RPD. As D becomes smaller, the average RPDs and variances of RPDs became larger for the nine algorithms, in particular, for the three GAs.

Regarding the effects of τ, and ρ on the performance of each algorithm, as shown in Table 6, all nine algorithms performed at $\tau = 0.5$ better than they did at $\tau = 0.25$. As ρ increased the RPD of all three GAs increased, as seen in Table 6, but there is no such pattern for three SAs and three CSAs.

Regarding the impacts of Qlevel, and N_b on the performance of each algorithm, it can be observed in Table 6 that the differences of RPD between three GAs or three SAs or three CSAs are very slight. It means that these nine algorithms can effectively solve instances, not affecting by the values of Qlevel. The RPDs of the nine algorithms increased as N_b increased, as shown in Table 6.

To examine whether the difference between the nine algorithms is statistically significant for the large size jobs, we performed the K–W test to examine the statistical differences among the nine algorithms. The fifth column of Table 4 showed the mean rank of the nine algorithms. The p value was < 0.001 (and the value of test statistic, Chi-square distribution, was 813.9 with 8 degrees of freedom) of the Kruskal–Wallis test, thus there are significant differences among the performances of the nine algorithms.

Table 3 Performances of AEP of algorithms for parameters as $n = 12$

	GA_p	GA_f	GA_b	SA_p	SA_f	SA_b	CSA_p	CSA_f	CSA_b
D									
0.001	6.462	6.551	6.344	3.163	3.061	2.789	1.953	2.375	1.849
0.01	5.416	5.086	5.111	2.577	2.422	2.602	1.787	1.809	1.686
0.1	1.615	1.692	1.693	1.221	1.252	1.226	0.891	0.835	0.819
τ									
0.25	6.476	6.309	6.129	2.923	2.853	2.807	1.982	2.220	1.944
0.5	2.519	2.577	2.636	1.718	1.637	1.605	1.105	1.126	0.958
ρ									
0.25	4.024	3.854	3.961	1.703	1.744	1.610	1.183	1.175	1.039
0.5	4.170	4.238	4.051	2.303	2.141	2.190	1.578	1.577	1.460
0.75	5.299	5.237	5.136	2.954	2.850	2.816	1.869	2.267	1.854
Qlevel									
1.6	4.057	4.114	3.991	2.325	2.162	2.140	1.405	1.643	1.409
1.7	4.326	4.158	4.351	2.303	2.232	2.196	1.570	1.637	1.459
1.8	5.110	5.058	4.806	2.334	2.340	2.281	1.655	1.739	1.486
N_b									
2	1.516	1.497	1.562	1.805	1.803	1.830	1.307	1.375	1.300
4	5.197	5.122	5.281	1.771	1.729	1.800	1.238	1.240	1.209
6	7.406	7.226	7.406	2.059	2.110	2.089	1.628	1.545	1.486
8	6.319	6.699	5.901	3.165	3.167	3.078	2.143	2.447	2.195
10	2.050	1.673	1.764	2.801	2.415	2.233	1.400	1.758	1.065
Mean	4.498	4.443	4.383	2.320	2.245	2.206	1.543	1.673	1.451

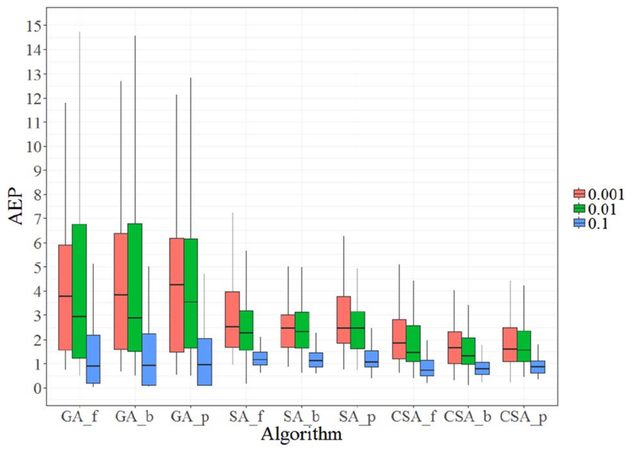

Fig. 1 Impact of D on the AEPs of nine algorithms

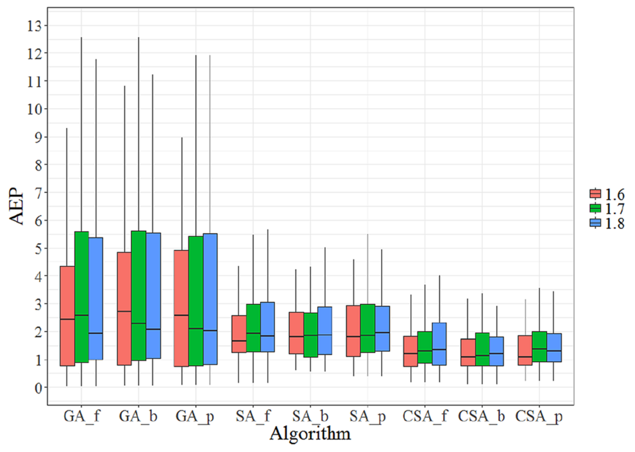

Fig. 4 AEP of nine algorithms over different levels of Qlevel

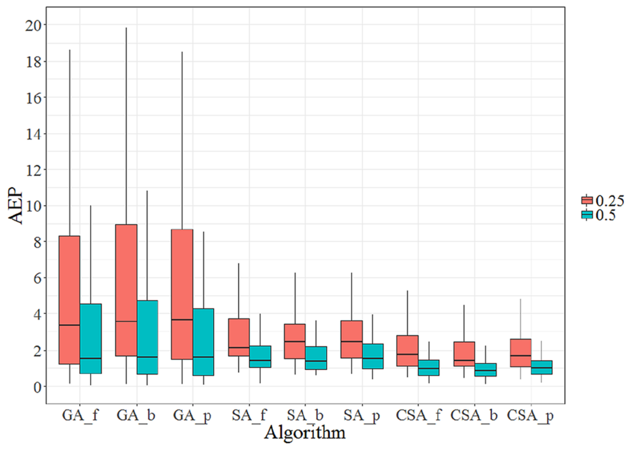

Fig. 2 Impact of τ on the AEPs of nine algorithms

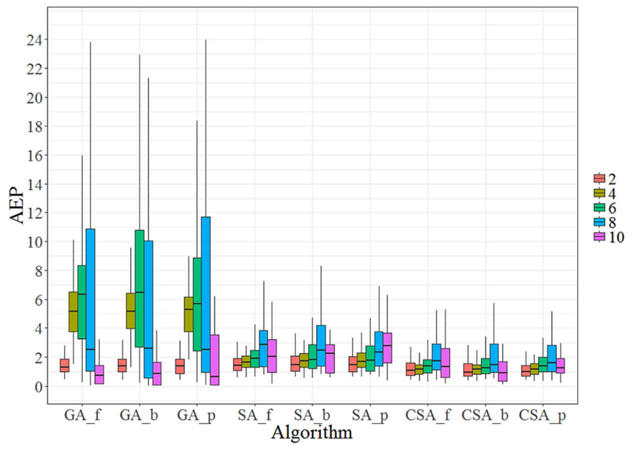

Fig. 5 AEP of nine algorithmsover different values of N_b

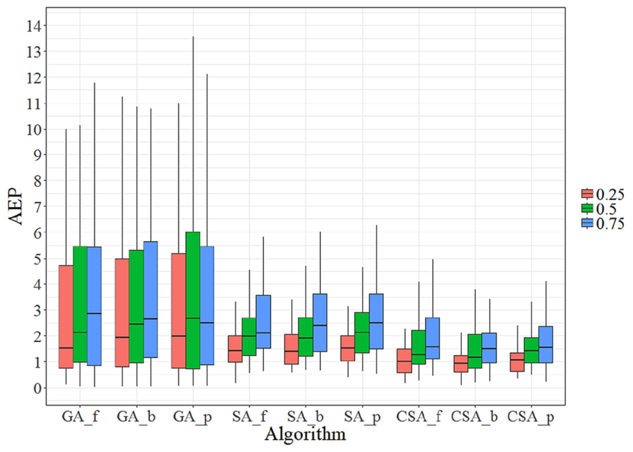

Fig. 3 Impact of ρ on the AEPs of nine algorithms

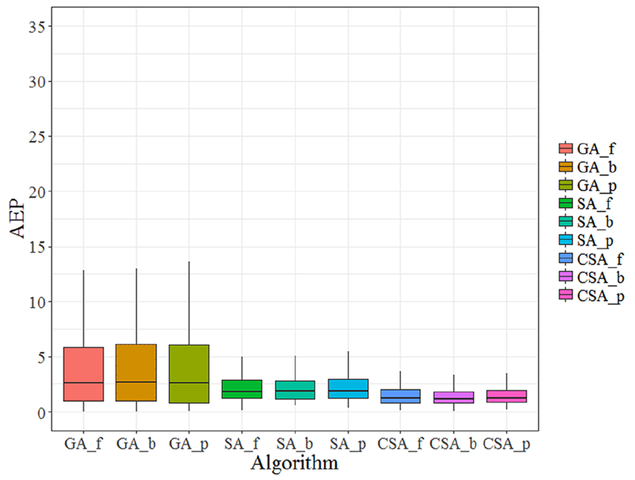

Fig. 6 Boxplots of AEP for nine algorithms

Table 4 Mean ranks of algorithms in K–W test for small and large size jobs

Algorithm	Small *n*		Large *n*	
	Number of obs	Mean rank	Number of obs	Mean rank
GA_p	270	1406.8	270	1763.4
GA_f	270	1397.5	270	1763.6
GA_b	270	1423.5	270	1761.7
SA_p	270	1313.8	270	1092.0
SA_f	270	1292.0	270	1092.3
SA_b	270	1279.6	270	1091.7
CSA_p	270	959.8	270	788.1
CSA_f	270	981.0	270	788.9
CSA_b	270	885.3	270	797.8
Total	2430	1215.5	2430	1215.5

*Average scores were used for ties

The boxplot of RPDs for nine algorithms in Fig. 8 shows that the RPD of CSA seems have difference from those of GA and SA, the CSA did the best performance.

To examine the statistical difference among these nine algorithms, we performed multiple comparisons and groupings for algorithms using DSCF test. The result shows that CSAs is the best one, SAs next, and GAs is the worst; the group difference among GA, SA and CSA is very clearly (also refer Fig. 8); CSA_f performed the best among all proposed algorithms.

Regarding the CPU times, as showed in Fig. 9, GAs costs the most, SAs next, and the CSAs the least, but all were less one second.

Conclusion and suggestions

In this paper, we study the sequential mode of multitasking for job scheduling together with two agents to minimize the total tardiness of A-agent's jobs subject to a restriction (upper bound) on the total completion time of the B-agent's jobs. The study issue belongs to the NP-hard set, thus, we derived five properties and a lower bound to embed in a B&B method to search optimal job sequences for small-sized jobs. Three variants of each of metaheuristics, GA, SA, and CSA, i.e., nine algorithms are employed to search optimal or approximate job sequences for large-sized jobs. Experimental results show that the CSAs performed best among these algorithms. For the future study, the complexity of multitasking scheduling problem with two (or more)-agents together with other criteria, for example the weighted number of lateness for one of agents, would be an interesting topic. Also, one can address the development of efficient optimization algorithm and heuristics for those hard problems.

Table 5 DSCF for nine algorithms

Pairwise comparison	DSCF			Pairwise comparison	DSCF		
Between algorithms	Statistic	*p* value	Sign	Between algorithms	Statistic	*p* value	Sign
GA_f–GA_b	0.2609	1.0000		GA_p–CSA_f	− 8.6887	< 0.001	***
GA_f–GA_p	0.1751	1.0000		GA_p–CSA_b	− 9.9715	< 0.001	***
GA_f–SA_f	− 3.8092	0.1501		GA_p–CSA_p	− 8.9660	< 0.001	***
GA_f–SA_b	− 4.0908	0.0904		SA_f–SA_b	− 0.4017	1.0000	
GA_f–SA_p	− 3.5342	0.2324		SA_f–SA_p	0.8160	0.9997	
GA_f–CSA_f	− 8.5400	< 0.001	***	SA_f–CSA_f	− 8.5240	< 0.001	***
GA_f–CSA_b	− 9.9407	< 0.001	***	SA_f–CSA_b	− 11.2715	< 0.001	***
GA_f–CSA_p	− 8.8661	< 0.001	***	SA_f–CSA_p	− 9.4839	< 0.001	***
GA_b–GA_p	0.0090	1.0000		SA_b–SA_p	1.1069	0.9973	
GA_b–SA_f	− 4.6735	0.0266	*	SA_b–CSA_f	− 8.1492	< 0.001	***
GA_b–SA_b	− 4.8233	0.0188	*	SA_b–CSA_b	− 11.0090	< 0.001	***
GA_b–SA_p	− 4.3069	0.0590		SA_b–CSA_p	− 9.1118	< 0.001	***
GA_b–CSA_f	− 9.3041	< 0.001	***	SA_p–CSA_f	− 8.9800	< 0.001	***
GA_b–CSA_b	− 10.7138	< 0.001	***	SA_p–CSA_b	− 11.6612	< 0.001	***
GA_b–CSA_p	− 9.7441	< 0.001	***	SA_p–CSA_p	− 9.8849	< 0.001	***
GA_p–SA_f	− 4.2503	0.0663		CSA_f–CSA_b	− 2.5337	0.6880	
GA_p–SA_b	− 4.3276	0.0565		CSA_f–CSA_p	− 0.2570	1.0000	
GA_p–SA_p	− 3.8927	0.1299		CSA_b–CSA_p	2.4499	0.7267	

Table 6 Performances of RPD of algorithms for parameters as $n=60$

	GA_p	GA_f	GA_b	SA_p	SA_f	SA_b	CSA_p	CSA_f	CSA_b
D									
0.001	50.741	50.752	50.843	10.715	10.505	10.765	6.885	6.655	6.931
0.01	39.848	39.543	40.272	9.355	9.098	8.946	5.874	5.790	6.031
0.1	10.279	10.279	10.037	4.010	4.012	4.067	2.582	2.551	2.607
Qlevel									
1.6	32.547	32.430	32.659	7.993	7.808	7.841	5.091	4.988	5.165
1.7	33.866	33.768	33.972	8.048	7.885	7.973	5.134	5.003	5.201
1.8	34.455	34.377	34.522	8.038	7.921	7.964	5.117	5.004	5.203
τ									
0.25	37.909	37.781	38.029	8.909	8.705	8.899	5.572	5.548	5.851
0.5	29.336	29.268	29.406	7.144	7.038	6.953	4.656	4.450	4.528
ρ									
0.25	32.717	32.509	32.952	7.563	7.340	7.572	5.092	4.986	5.096
0.5	33.443	33.301	33.486	8.054	8.248	7.893	4.872	5.065	5.056
0.75	34.708	34.765	34.714	8.462	8.026	8.313	5.377	4.945	5.417
N_b									
10	5.257	5.273	5.258	1.843	1.829	1.838	0.750	0.783	0.770
20	22.349	22.294	22.354	3.737	3.756	3.773	1.670	1.712	1.688
30	32.458	32.255	32.163	5.196	5.266	5.095	2.672	2.668	2.695
40	40.775	40.912	40.920	8.366	8.307	8.595	5.272	5.264	5.543
50	67.274	66.891	67.893	20.991	20.199	20.330	15.205	14.565	15.253
Mean	33.623	33.525	33.718	8.027	7.871	7.926	5.114	4.998	5.190

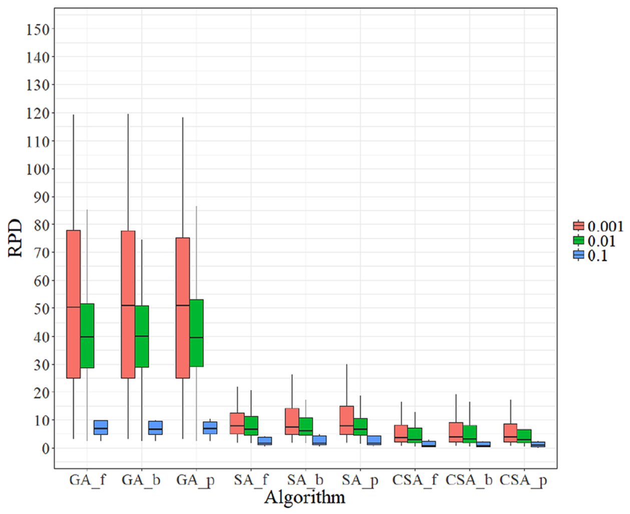

Fig. 7 RPD of nine algorithms over different levels of D

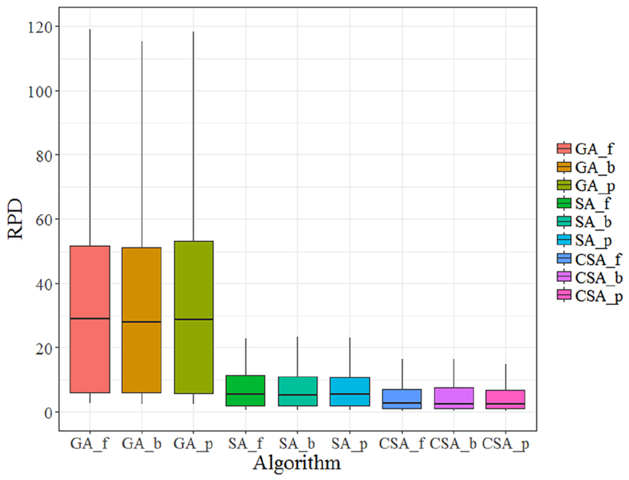

Fig. 8 Boxplots of RPD for nine algorithms

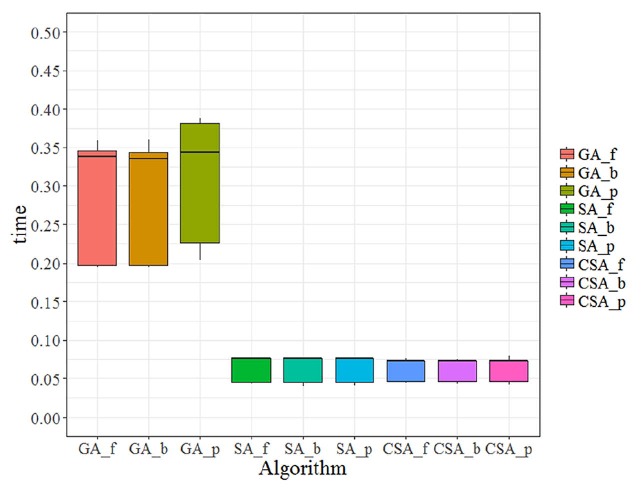

Fig. 9 CPU time for nine algorithms for large-size instances

Acknowledgements We thank the editor and two referees for their positive comments and useful suggestions. This study was supported in part by the Ministry of Science and Technology of Taiwan, MOST 109-2410-H-035-019.

Funding There are none.

Availability of data and materials The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Conflict of interest All of the authors declare that they have no conflict of interest.

Ethical approval This paper does not contain any studies with human participants or animals performed by any of the authors.

References

1. Agnetis A, Mirchandani PB, Pacciarelli D, Pacifici A (2004) Scheduling problems with two competing agents. Oper Res 52:229–242
2. Agnetis A, Billaut J-C, Gawiejnowicz S, Pacciarelli D, Soukhal A (2014) Multiagent scheduling. Springer, Berlin
3. Ahmadi-Darani M-H, Moslehi G, Reisi-Nafchi M (2018) A two-agent scheduling problem in a two-machine flowshop. Int J Ind Eng Comput 9:289–306
4. Appasami G, Joseph KS (2011) Optimization of operating systems towards green computing. Int J Comb Optim Probl Inform 2(3):39–51
5. Baker KR, Smith JC (2003) A multiple criterion model for machine scheduling. J Sched 6:7–16
6. Charron S, Koechlin E (2010) Divided representation of concurrent goals in the human frontal lobes. Science 328(5976):360–363
7. Della Croce F, Narayan V, Tadei R (1996) The two-machine total completion time flowshop problem. Eur J Oper Res 90:227–237
8. Essafi I, Mati Y, Dauzère-Pérès S (2008) A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Comput Oper Res 35(8):2599–2616
9. Fischer R, Plessow F (2015) Efficient multitasking: parallel versus serial processing of multiple tasks. Front Psychol. https://doi.org/10.3389/fpsyg.2015.01366
10. Gerstl E, Mosheiov G (2012) Scheduling problems with two competing agents to minimize weighted earliness-tardiness. Comput Oper Res 40:109–116
11. Gao K, Huang Y, Sadollah A, Wang L (2020) A review of energy-efficient scheduling in intelligent production systems. Complex Intell Syst 6:237–249
12. Hall NG, Leung JY-T, Li CL (2015) The effects of multitasking on operations scheduling. Prod Oper Manag 24(8):1248–1265
13. Hall NG, Leung JY-T, Li C-L (2016) Multitasking via alternate and shared processing: algorithms and complexity. Discrete Appl Math 208:41–58
14. Ho KI-J, Sum J (2017) Scheduling jobs with multitasking and asymmetric switching costs. In: Proceedings of the IEEE international conference on systems, man, and cybernetics (SMC '17), Banff Center, Banff, Canada, October 5–8, 2017. http://www.smc2017.org/SMC2017_Papers/media/files/0184.pdf
15. Holland JH (1984) Genetic algorithms and adaptation. Adaptive control of ill-defined systems. Springer, pp 317–333
16. Hollander MD, Wolfe A, Chicken E (2014) Nonparametric statistical methods, 3rd edn. Wiley, Hoboken
17. Iyer SK, Saxena B (2004) Improved genetic algorithm for the permutation flowshop scheduling problem. Comput Oper Res 31(4):593–606
18. Ji M, Zhang W, Liao L, Cheng TCE, Tan Y (2019) Multitasking parallel-machine scheduling with machine-dependent slack due-window assignment. Int J Prod Res 57:1667–1684. https://doi.org/10.1080/00207543.2018.1497312
19. Kardos C, Kovács A, Váncza J (2020) A constraint model for assembly planning. J Manuf Syst 54:196–203
20. Kc DS (2014) Does multitasking improve performance? Evidence from the emergency department. Manuf Serv Oper Manag 16(2):168–183
21. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
22. Kovalyov MY, Oulamara A, Soukhal A (2015) Two-agent scheduling with agent specific batches on an unbounded serial batching machine. J Sched 18:423–434
23. Lagzi S, Fukasawa R, Ricardez-Sandoval L (2017) A multitasking continuous time formulation for short-term scheduling of operations in multipurpose plants. Comput Chem Eng 97:135–146. https://doi.org/10.1016/j.compchemeng.2016.11.012
24. Lai ZH, Tao W, Leu MC, Yin Z (2020) Smart augmented reality instructional system for mechanical assembly towards worker-centered intelligent manufacturing. J Manuf Syst 55:69–81
25. Lea VM, Corlett SA, Rodgers RM (2015) Describing interruptions, multi-tasking and task-switching in communitypharmacy: a qualitative study in England. Int J Clin Pharm 37:1086–1094

26. Li C-L, Chong W (2018) Task scheduling with progress control. IISE Trans 50:54–61
27. Liu M, Wang S, Zheng F, Chu C (2017) Algorithms for the joint multitasking scheduling and common due date assignment problem. Int J Prod Res 55(20):6052–6066
28. Li S, Cheng TCE, Ng CT, Yuan J (2017) Two-agent scheduling on a single sequential and compatible batching machine. Nav Res Logist 64:628–641
29. Logie RH, Law A, Trawley W, Nissan J (2010) Multitasking, working memory and remembering intensions. Psychologica Belgica 50(3 & 4):309–326
30. Mor B, Mosheiov G (2011) Single machine batch scheduling with two competing agents to minimize total flowtime. Eur J Oper Res 215:524–531
31. Noguera J, Badia RM (2004) Multitasking on reconfigurable architectures: micro architecture support and dynamic scheduling. ACM Trans Embed Comput Syst 3(2):385–406
32. Ophir E, Nass C, Wagner AD (2009) Cognitive control in media multitaskers. Psychol Cogn Sci 106(37):15583–15587
33. Pashler HE (1994) Dual-task interference in simple tasks: data and theory. Psychol Bull 116(2):220–244. https://doi.org/10.1037/0033-2909.116.2.220
34. Pashler HE (2000) Task switching and multitask performance. In: Control of cognitive processes: attention and performance XVIII, 2000, pp 277–307
35. Perez-Gonzalez P, Framinan JM (2014) A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: multi-agent scheduling problems. Eur J Oper Res 235(1):1–16
36. Pinedo M (2014) Scheduling: theory, algorithms, and systems, 3rd edn. Prentice Hall, Upper Saddle River
37. Reeves C (2003) Genetic algorithms. Handbook of metaheuristics. Springer, pp 55–82
38. Reissland J, Manzey D (2016) Serial or overlapping processing in multitasking as individual preference: effects of stimulus preview on task switching and concurrent dual-task performance. Acta Physiol (Oxf) 168:27–40. https://doi.org/10.1016/j.actpsy.2016.04.010
39. Rubinstein JS, Meyer DE, Evans JE (2001) Executive control of cognitive processes in task switching. J Exp Psychol Hum Percept Perform 27(4):763–797
40. Rosen C (2008) The myth of multitasking. New ATLANTIS 2008:105–110
41. Sayed GI, Hassanien AE (2018) A hybrid SA-MFO algorithm for function optimization and engineering design problems. Complex Intell Syst 4:195–212
42. Seshadri S, Shapira Z (2001) Managerial allocation of time and effort: the effects of interruptions. Manag Sci 47(5):647–662
43. Sum J, Ho K (2015) Analysis on the effect of multitasking. In; Proceedings of the IEEE international conference on systems, man, and cybernetics (SMC'15), pp 204–209, IEEE, Hong Kong, October 2015. https://doi.org/10.1109/SMC.2015.48
44. Torabzadeh E, Zandieh M (2010) Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. Adv Eng Softw 41:1238–1243
45. Wang D, Yu Y, Yin Y, Cheng TCE (2020) Multi-agent scheduling problems under multitasking. Int J Prod Res. https://doi.org/10.1080/00207543.2020.1748908
46. Wickens CD, McCareley JS (2008) Applied attention theory. Erlbaum, Mahwah
47. Wu CC, Lin WC, Zhang XG, Bai DY, Tsai YW, Ren T, Cheng SR (2020) Cloud theory-based simulated annealing for a single-machine past sequence setup scheduling with scenario-dependent processing times. Complex Intell Syst. https://doi.org/10.1007/s40747-020-00196-7
48. Xiong X, Zhou P, Yin Y, Cheng TCE, Li D (2019) An exact branch-and-price algorithm for multitasking scheduling on unrelated parallel machines. Nav Res Logist 66:502–516
49. Yang Y, Yin G, Wang C, Yin Y (2020) Due date assignment and two-agent scheduling under multitasking environment. J Comb Optim. https://doi.org/10.1007/s10878-020-00600-5
50. Yavari S (2018) Machine scheduling for multitask machining. Electronic Theses and Dissertations 2018, p 7409. https://scholar.uwindsor.ca/etd/7409.
51. Yin Y, Cheng TCE, Wang DJ, Wu C-C (2016) Just-in-time scheduling with two competing agents on unrelated parallel machines. Omega 63:41–47
52. Yin Y, Wang DJ, Wu C-C, Cheng TCE (2016) CON/SLK due date assignment and scheduling on a single machine with two agents. Nav Res Logist 63:416–429
53. Yin Y, Cheng TCE, Wang DJ, Wu C-C (2017) Two-agent flowshop scheduling to maximize the weighted number of just-in-time jobs. J Sched 20:313–335
54. Yin Y, Wang WY, Wang DJ, Cheng TCE (2017) Multi-agent single-machine scheduling and unrestricted due date assignment with a fixed machine unavailability interval. Comput Ind Eng 111:202–215
55. Zhang F, Cao J, Tan W, Khan SU, Li K, Zomaya AY (2014) Evolutionary scheduling of dynamic multitasking workloads for big data analytics in elastic cloud. IEEE Trans Emerg Top Comput 2(3):338–351
56. Zhang X, Wang Y (2017) Two-agent scheduling problems on a single-machine to minimize the total weighted late work. J Comb Optim 33(3):945–955
57. Zhu Z, Li J, Chu C (2017) Multitaskingschedulingproblemswith-deteriorationeffect. Math Probl Eng 2017:1–10
58. Zhu Z, Liu M, Chu C, Li L (2017) Multitasking scheduling with multiple rate-modifying activities. Int Trans Oper Res 00:1–21. https://doi.org/10.1111/itor.12393
59. Zhu Z, Zheng F, Chu C (2017) Multitasking scheduling problems with a rate-modifying activity. Int J Prod Res 55(1):296–312
60. Zhu ZG, Liu M, Chu CB, Li JL (2019) Multitasking scheduling with multiple rate-modifying activities. Int Trans Oper Res 26(5):1956–1976