Future Generation Computer Systems 95 (2019) 79–88

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Pseudo transformation mechanism between resource allocation and bin-packing in batching environments

Xinle Liang [a], Shengchao Zhou [b,*], Huaping Chen [a], Rui Xu [c]

[a] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China
[b] School of Information Science and Engineering, Central South University, Changsha 410083, China
[c] School of Business, Hohai University, Nanjing 210098, China

HIGHLIGHTS

- A transformation mechanism is proposed for batch-scheduling with resource allocation.
- An effective lower bound on the optimal objective value is presented.
- A heuristic is developed by utilizing a well-known approach for bin-packing problems.
- Computational experiments show that our approach outperforms previous algorithms.

ARTICLE INFO

Article history:
Received 25 February 2018
Received in revised form 11 November 2018
Accepted 2 January 2019
Available online 6 January 2019

Keywords:
Batch-scheduling
Resource allocation
Bin-packing problems
Pseudo transformation

ABSTRACT

Job planning with resource allocations constitutes a classical subfield of scheduling. This research is devoted to connecting a batch-scheduling problem with resource allocations to a bin-packing problem (BPP). A mechanism of transforming the batch-scheduling problem into BPP is proposed. Based on the transformation mechanism, a heuristic is proposed by utilizing an effective approach for BPP. In order to evaluate the efficiency of the proposed heuristic, extensive experiments are carried out on the performance comparisons against several available methods. The results show that the proposed heuristic can be a strong alternative for the problem under study, which, in turn, demonstrates the effectiveness of the proposed mechanism.

1. Introduction

Scheduling with resource allocations originates from various real-life systems [1–4], where the processing or setup times are controllable by resource allocations such as money, energy, fuel or additional manpower. Since more and more people from industrial or academic areas are concerned about improving the resource efficiency, deep investigation into this subfield has high practical significance.

Since its appearance in [5,6], lots of researches have been devoted to this subfield of scheduling. The following lists two criteria (other criteria can be found in [7]) that can be used to classify researches in the subfield. On the one hand, according to the machine settings, the relevant researches can be categorized as follows: single machine scheduling [8–13], parallel machine scheduling [14], flow-shop scheduling [15,16], and batch-scheduling [17–22]. On the other hand, researches in this subfield can also be classified into the following categories: scheduling without or with resource constraints. The former problem setting receives more attention in relevant researches [17,18]. The latter setting assumes that the allocations for each operation should not exceed the maximal amount. Such constraints are called technological constraints hereafter.

The problem under study combines batch-scheduling [23,24] with resource constraints. Such a combination leads to the optimizations in both criteria, for which the scheduling and allocation decisions should be determined simultaneously to achieve efficient system performance.

In literature, there exist lots of researches dealing with similar problem settings. Cheng and Kovalyov [12] studied the problem of scheduling a batching machine with deadlines, where the processing and setup times were controllable. Cheng et al. [17] presented several interesting characteristics for optimal solutions of a single batching machine in which the resource allocation was the same for all jobs. Ng et al. [18] investigated the batch-scheduling with resource constraints where the total allocations for the processing and setup operations were upper bounded. Similar settings were employed further in [1,25]. Other interesting applications can be seen in [26,27]. The consideration that the resources for setup or processing operations are upper bounded are more practical and may arouse more complex applications.

* Corresponding author.
E-mail address: zhousc@mail.ustc.edu.cn (S. Zhou).

https://doi.org/10.1016/j.future.2019.01.006

In order to deal with the variants in this subfield of scheduling, almost all the researches tried to figure out the scheduled job set as well as the resource allocation set. Garey et al. [28] studied a special case of the general multiprocessor scheduling problem with resource constraint. They showed that the problem can be transformed into generalized BPP (bin-packing problem) that has been examined extensively in literature [29–31]. Investigating the connections between different problems can present theoretical basis for developing common approaches for different research groups. The transformation provides relevant community with theoretical basis for applying many mature approaches in the field of BPP.

Inspired by the above research, we aim to study the connections between batch-scheduling and classical optimization problems. In this context, it is proved that batch-scheduling with resource constraints can be solved by dealing with multiple BPPs. In order to validate the efficiency of such transformation, we present a heuristic for a variant of batch-scheduling with resource constraints. The experimental study shows that the proposed heuristic outperforms GAMS and some existing approaches in solution quality. To the best of our knowledge, this is the first research work devoted to investigating the connections between scheduling with resource allocations and classical optimization problems.

The remainder of the paper is organized as follows. In Section 2, several relevant preliminaries are introduced. In Section 3, the problem under study is formulated. In Section 4, a lower bound and the characteristics of optimal solutions are proposed. In Section 5, the mechanism for the transformation into BPP is presented and a heuristic is proposed for validation. In Section 6, an experimental study is carried out to evaluate the algorithm proposed. Conclusions on the study and directions for future research are provided in Section 7.

2. Preliminaries

Batch-scheduling problems with resource allocation are specified by the following two factors: the resource consumption functions and the resource allocation policies within each batch.

2.1. Resource consumption functions

One of the most widely employed models is the linear resource consumption function [11–14,17], which has the form of

$$p_i(u_i) = \bar{p}_i - a_i u_i, \quad i = 1, \dots, n, \quad 0 \le u_i \le \bar{u}_i < \bar{p}_i / a_i \tag{1}$$

where n is the number of non-preemptive jobs, u_i is the amount of resource allocated to job i, $p_i(u_i)$ is the processing time of job i which is a function of u_i, $\bar{u}_i$ is the upper bound on the amount of resource that can be allocated to job i, a_i is the positive compression rate of job i and $\bar{p}_i$ is the incompressible processing time for job i. The compression rate a_i represents the linear decrease of p_i when the resource allocation u_i increases by one unit (within the predefined constraint $\bar{u}_i$) and the incompressible processing time $\bar{p}_i$ denotes the initial processing time with no resource allocation.

However, in many resource allocation problems, especially those related to physical and economic systems, the linear resource consumption function fails to reflect the law of diminishing marginal returns, which states that productivity increases at a decreasing rate with respect to the amount of resource employed. To avoid this, some studies, including [32–36], applied a specific convex-decreasing resource consumption function of the form

$$p_i(u_i) = (\frac{w_i}{u_i})^k, \quad i = 1, 2, \dots, n, \quad u_i > 0, \tag{2}$$

where w_i is a job-dependent parameter (the workload of job i) and $k > 0$ is a constant positive parameter that is identical for all jobs. Similarly, the resource consumption function for the setup operations is as follows,

$$S_j(V_j) = (\frac{S}{V_j})^k, \quad V_j > 0, \tag{3}$$

where S denotes the workload of the setup operation of a single machine which is independent of the batches and V_j signifies the resources that are allocated to the setup operation of the jth batch. Eqs. (2) and (3) share the same exponent k when resource-dependent processing and setup times are considered simultaneously.

Monma et al. [37] studied several significant applications in the resource allocation problems with precedence constraints and non-renewable resources, and then showed that $k = 1$ case corresponds to many actual government or industrial projects and that $k = 0.5$ case arises from VLSI (Very Large Scale Integration) circuit design. They further pointed out that the product of the silicon area (resource) and the square of the time spent (time squared) equals a constant value (the workload) for an individual job.

2.2. Resource allocation policies

The other factor that affects the total system performance is the resource allocation policy among the jobs within each batch. The policies existing in literature can be categorized as follows.

The first policy assumes that all processes of resource allocations can be manually controllable, i.e., the resource allocation set can be any combination of the resource allocations as long as the summation of resources equals the available resources [34,35,38,39]. Although such policy is the most efficient one (theoretically speaking), lots of applications fail to follow it due to the machine or resource restrictions.

The second policy is that resources are evenly allocated within each batch [18,25,40]. Biskup and Jahnke [40] gave two examples where the resource allocation is the same for all the jobs in each batch. In the steel production industry, a furnace can be heated to a specific temperature every day before the processing of an order for ingots (jobs) starts. It is not beneficial to change the temperature for every single job. A higher temperature reduces the processing times but incurs a cost for using the furnace. A similar situation might occur for a machine at which specific tools have to be changed after a fixed period of time, like a week. Each time a new tool is installed, a decision about the tool characteristics that is the productive power, needs to be made. For example, a drilling machine might run with a diamond drill, a high- or a low-quality steel drill. If the diamond drill is set up, the processing of the jobs can be carried out faster than with a steel drill by incurring higher costs.

Another example given in [40] is an assembly line in which the speed depends on the number of workers and tools available. It is generally not possible or advantageous to change the speed during the day. Since the available resource allocation for each batch determines a common processing speed for all the jobs contained, it would be more practical to consider that more resources should be guaranteed if a longer processing time is required. Relying on such cases, another resource allocation policy is proposed in which the amount of resources that is allocated to each job is in direct proportion to its processing time.

For a brief representation, the short notations ERA (even resource allocation), MCRA (manually controllable resource allocation) and PRA (proportional resource allocation) are employed to denote the three resource allocation policies. For MCRA, Oron [26] proved that in a batching environment each job gets such resource allocation that its processing time equals to the batch processing time in any optimal solution. For a pair of jobs (i, j) in the same

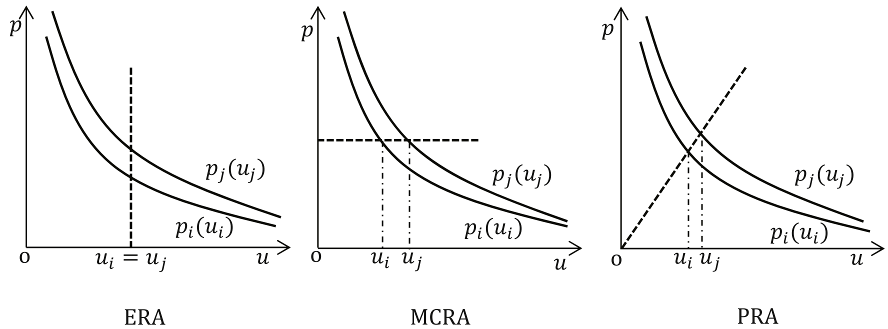


Fig. 1. Resource allocation policies. u_i and u_j represent the actual resource consumption of jobs i and j.

Table 1
Notations.

Notation	Description
n	Number of jobs
J	Job set, $J=\{J_1, J_2 \ldots, J_n\}$
w_i	Workload of job J_i
S	Workload of machine setup operation
p_i	Processing time of job J_i
u_i	Resource allocation for job J_i
m	Number of batches
B_j	The jth batch, $j=1, 2, \ldots, m$
O_j	The component set of B_j
$\|O_j\|$	The number of jobs in O_j
π	Batch sequence, $\pi=\{B_1, B_2, \ldots, B_m\}$
β	Number capacity for each batch
v	Resource constraint for each setup
u	Resource constraint for each batch
B_S_j	Setup operation of B_j
γ	Setup sequence, $\gamma=\{B\_S_1, B\_S_2, \ldots, B\_S_m\}$
P_j	Processing time of B_j
S_j	Setup time of B_S_j
U_j	Resources allocation for B_j
V_j	Resources allocation for B_S_j
C	Makespan constraint

batch, Fig. 1 summarizes the different constraints among ERA, MCRA and PRA in relevant optimal solutions.

Since optimal resource allocation decisions differ among the resource allocation policies (ERA, MCRA and PRA), the solutions for ERA and MCRA employed in the above-cited literature do not fit for the problems with PRA.

3. Problem formulation

Before the problem setting is proposed, a detailed description of the notations that have been employed or will be used is presented in Table 1. Additionally, each notation will be explained in detail upon its first appearance in the following.

In the problem studied, considering the corresponding practical backgrounds, the convex-decreasing resource function and PRA are employed. Under PRA, for each J_i in O_j, the resources allocated to J_i is given by

$$u_i = \frac{p_i U_j}{\sum_{J_l \in O_j} p_l} \tag{4}$$

Accordingly, by substituting the above equation into Eq. (2), P_j can be given by

$$P_j(U_j) = \frac{(\sum_{J_l \in O_j} w_l^{\frac{k}{k+1}})^{k+1}}{U_j^k}. \tag{5}$$

The batch-scheduling problem with resource constraints is modeled with six basic objects: *jobs, batches, setups, constraints, decision variables* and *objective*.

1. *Jobs.* A job set J is available for processing at time zero. Each job $J_i \in J$ has a workload w_i, a processing time p_i and a resource allocation u_i.
2. *Batches.* The jobs are to be processed in batches. The arrangement of these jobs forms a set of batches π. Each $B_j \in \pi$ has a resource allocation U_j, a processing time P_j, and a component job set O_j.
3. *Setups.* A setup operation is required before any batch processing operation is started. Each setup operation $B\_S_j \in \gamma$ has a resource allocation V_j and a setup time S_j, where γ represents the set of setups.
4. *Constraints.* Each job can be allocated to only one batch, and the number capacity for each batch is β:

$$\begin{cases} \sum_{j=1}^{m} x_{ij} = 1, & i = 1, 2, \ldots, n, \\ \sum_{i=1}^{n} x_{ij} \le \beta, & j = 1, 2, \ldots, m. \end{cases} \tag{6}$$

For each batch, the resource allocations for processing and setup operations should not break the resource constraints:

$$\begin{cases} U_j \le u, & j = 1, 2, \ldots, m, \\ V_j \le v, & j = 1, 2, \ldots, m. \end{cases} \tag{7}$$

The time constraints on processing and setup operations are given by Eqs. (3) and (5)

$$\begin{cases} P_j = \dfrac{(\sum_{i=1}^{n} x_{ij} w_i^{\frac{k}{k+1}})^{k+1}}{U_j^k}, & j = 1, 2, \ldots, m, \\ S_j = (\dfrac{S}{V_j})^k, & j = 1, 2, \ldots, m. \\ \sum_{j=1}^{m} (P_j + S_j) \le C. \end{cases} \tag{8}$$

5. *Decision variables.* The decision variables include: m, the number of batches; x_{ij} (x_{ij} equals "1" if J_i is allocated to O_j and "0" otherwise); and additionally U_j, V_j, the resource allocation for each B_j and B_S_j.
6. *Objective.* The objective of the problem considered is to minimize the resources invested:

$$\textbf{Min} \quad \sum_{j=1}^{m} (U_j + V_j) \tag{9}$$

This problem is a continuous optimization problem. Following the three-field notation introduced by [41] for scheduling problems, the problem under consideration can be represented by $1|s\text{-}batch, conv, w_i, U_j \le u,\ V_j \le v, C_{max} \le C\ |\sum(U_j+V_j)$, where *s-batch* denotes serial-batching machine [42,43] and *conv* specifies the convex resource function. For more brief representation, it is denoted by RCM (Resource Consumption Minimization).

4. Characteristics

4.1. A lower bound

In order to analyze the problem properties and to evaluate the approaches, this section is devoted to presenting an effective lower bound by employing KKT conditions [44,45]. Several constraints in the above formulation are relaxed for conducting KKT conditions:

- On one hand, the resource constraints are eliminated, i.e., ignoring Eq. (7);
- On the other hand, it is assumed that the resource allocations among jobs are manually controllable.

Since all setup operations share the same status, it is obvious that in any optimal solutions, each setup is allocated the same resources, denoted by V_s in the following parts. The proposed lower bound is based on the following model,

$$\begin{cases} \textbf{Min} & \lceil \frac{n}{\beta} \rceil V_s + \sum_{i=1}^{n} u_i \\ \text{s.t.} & C - \lceil \frac{n}{\beta} \rceil (\frac{S}{V_s})^k - \sum_{i=1}^{n} (\frac{w_i}{u_i})^k \geq 0. \end{cases} \tag{10}$$

Any resource allocation decision that satisfies the KKT conditions of the above model is optimal. By solving the necessary and sufficient KKT conditions we yield an effective lower bound

$$LB = (\frac{1}{C})^{\frac{1}{k}} (\lceil \frac{n}{\beta} \rceil S^{\frac{k}{k+1}} + \sum_{i=1}^{n} w_i^{\frac{k}{k+1}})^{\frac{k+1}{k}} \tag{11}$$

4.2. Characteristics of optimal solutions

This subsection reveals the characteristics of resource utilities in the optimal solutions. Utility refers to the satisfaction that each choice provides to the decision maker. The resource utilities are the deviations of P_j and S_j, which are denoted by e in the following.

Lemma 1. *Given any optimal solution of RCM, each B_j or B_S_j has identical resource utility, respectively.*

Proof. The proof can be easily obtained by employing Lagrange analysis, which is neglected for the sake of simplicity. □

Based on the above lemma, the optimal solutions can be restricted into an infinite set ρ, where $\rho = \bigcup \rho(e_i)$ $(e_i \in \Re^+)$ and $\rho(e_i)$ is a set of feasible schedules completed at C whose processing operations have the same resource utility e_i. Lemma 1 can also be drawn by employing marginal analysis, which can derive the following lemmas as a by-product.

Lemma 2. *Given any optimal solution in $\rho(e)$, the relationship between V_s and e can be expressed as follows*

$$V_s = \begin{cases} \upsilon, & e < \frac{kS^k}{\upsilon^{k+1}}, \\ (\frac{kS^k}{e})^{\frac{1}{k+1}}, & e \geq \frac{kS^k}{\upsilon^{k+1}}, \end{cases} \tag{12}$$

and the total processing time can be calculated by

$$\sum_{j=1}^{m} P_j = (\frac{e}{k})^{\frac{k}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}} \tag{13}$$

Proof. In every infinitesimally-divisible step in making resource allocation decisions, an operation with the highest resource utility is allocated resources under the resource constraints Eq. (7). As resources are invested, the resource utilities decrease. However, no resource is allowed for the setup operations when the resource reaches υ with the resource utility kS^k/υ^{k+1}. Therefore, in optimal resource allocation decisions, if $e < kS^k/\upsilon^{k+1}$, υ amounts of resources should be allocated to each setup operation to achieve the highest utilities; and if $e \geq kS^k/\upsilon^{k+1}$ and $e = kS^k/\upsilon_0^{k+1} (0 < \upsilon_0 \leq \upsilon)$, υ_0 unit is invested to the setup operations, which completes our proof. □

Lemma 3. *Given any schedule subset $\rho(e)$, the optimal schedules, if exist, have the fewest batches.*

Proof. According to Lemma 2, for any schedule in $\rho(e)$, the total processing time $\sum P_j$, the total setup time $\sum S_j$ are fixed. Hence, the total resource consumption is given by

$$\begin{aligned} \sum (U_j + V_j) &= mV_s + \sum_{j=1}^{m} U_j \\ &= (\frac{k}{e})^{\frac{1}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}} + S(\frac{1}{\sum_{j=1}^{m} S_j})^{\frac{1}{k}} m^{\frac{k}{k+1}} \end{aligned} \tag{14}$$

which implies that there exists a positive correlation between m and the objective value. Thus, given any subset $\rho(e)$, the optimal schedules, if exist, have the fewest batches. □

Lemma 3 implies that, if $\rho(e)$ is already known, the optimal solution lies in those solutions with the fewest batches. This observation may provide an effective way to build the connection between RCM with BPP.

5. Transforming RCM into BPPs

Based on the preliminary idea of transforming RCM into BPP, this section is devoted to proposing the mechanism for such transformation:

1. In the first part, the mechanism for transforming RCM into BPPs is proposed.
2. Based on such mechanism, a heuristic is proposed for RCM.

5.1. Transformation mechanism

The properties and conditions in Section 3 are rearranged to form the constraints for the optimal solutions in this subsection.

Theorem 1. *The constraints for optimal solutions of RCM can be summarized into two types*

(1) equivalent constraints: V_s and e can be rewritten as the functions of m respectively as follows:

$$V_s = \begin{cases} (\frac{mS^k + S^{\frac{k^2}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}}}{C})^{\frac{1}{k}}, & \text{if } cond(m) = 1, \\ \upsilon, & otherwise. \end{cases} \tag{15}$$

and

$$e = \begin{cases} \frac{kC^{\frac{k+1}{k}}}{(m + S^{-\frac{k}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}})^{\frac{k+1}{k}} S}, & \text{if } cond(m) = 1, \\ (\frac{C - m(\frac{S}{\upsilon})^k}{\sum_{l=1}^{n} w_l^{\frac{k}{k+1}}})^{\frac{k+1}{k}} k, & otherwise. \end{cases} \tag{16}$$

where

$$cond(m) = \begin{cases} 0, & \text{if } m > \frac{C\upsilon^k - S^{\frac{k^2}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}}}{S^k}, \\ 1, & otherwise. \end{cases} \tag{17}$$

(2) batching constraints: for any batch B_j, the following constraints hold

$$\begin{cases} \sum_{J_l \in O_j} w_l^{\frac{k}{k+1}} \le u(\frac{e}{k})^{\frac{1}{k+1}}, \\ |O_j| \le \beta. \end{cases} \tag{18}$$

Proof. (1) According to Lemma 2, for each batch in any solution in the set $\rho(e)$, the total processing and setup time can be given by

$$\begin{cases} \sum_{j=1}^{m} P_j = (\frac{e}{k})^{\frac{k}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}}, \\ \sum_{j=1}^{m} S_j = m(\frac{e}{k})^{\frac{1}{k+1}} S^{\frac{k^2}{k+1}}. \end{cases} \tag{19}$$

Associating Eq. (12) with Eq. (19) yields the expression Eqs. (15)–(17).

(2) According to Lemma 3, since each processing operation has resource constraint u, the following expression holds:

$$(\frac{k}{e})^{\frac{1}{k+1}} \sum_{l=1}^{n} w_l^{\frac{k}{k+1}} \le u. \tag{20}$$

Therefore the batching constraints for RCM are two-fold: the summation of $w_i^{k/k+1}$ should not exceed $u(e/k)^{1/(k+1)}$; the maximal number of jobs that can be packed into each batch is β, which can be expressed as Eq. (18). □

As can be concluded from Eq. (18), for a unique value of e, the batching constraint is an off-line one-dimensional BPP. There is a set of n items with weights of $W=\{W_1, W_2, \ldots, W_n\}$, $W_i=w_i^{k/(k+1)}$, which are to be assigned to a bin with the capacity of $c=u(e/k)^{1/(k+1)}$ such that the total weight of items in each bin does not exceed c and the maximal number of items assigned to each bin is β. The objective is to minimize the number of bins used (the optimal solution for BPP is denoted by SOB hereafter). The values of e determine the capacities of BPPs and thus for different e, their relevant BPPs are non-identical.

As can be concluded, e is dependent on m uniquely and independently, which implies that there exists only one corresponding BPP for each m. Following Lemma 3, if m is equal to SOB, then the current solution is optimal. A feasible approach can start by assigning a specified m to Eqs. (15)–(17), and then verifying whether it is identical to the optimal solution of the corresponding BPP. Since $\lceil n/\beta \rceil \le m \le n$, RCM can be tackled by solving at most $n-\lceil n/\beta \rceil+1$ BPPs.

5.2. A heuristic for RCM

A heuristic called RMB-BPP (Resource Minimization Based on BPP) is proposed in this subsection. In the RMB-BPP algorithm, variable m ranges from $\lceil n/\beta \rceil$ to n. Each m is verified in terms of its equality to SOB of the corresponding BPP. For the case where a first tie occurs, the objective value is calculated and then RMB-BPP is ended. Due to the NP-completeness of BPP [46], the computational effort for gaining the global optima is exponential. Fig. 2 illustrates the detailed process of how different approaches find the corresponding solution.

As can be seen in Fig. 2, since the optimal solution for the BPP problem cannot be guaranteed, the solution quality of RCM greatly depends on the performance of BPP solving-approaches. In order to guarantee both an efficient solution and reasonable computational time, effective approaches for BPP should be employed.

In order to solve the multiple BPPs, the SAWMBS heuristic proposed by [47–49] is employed. If the maximum number of items that fit in one bin is β, it has a time complexity of $O(n^{\beta+1})$.

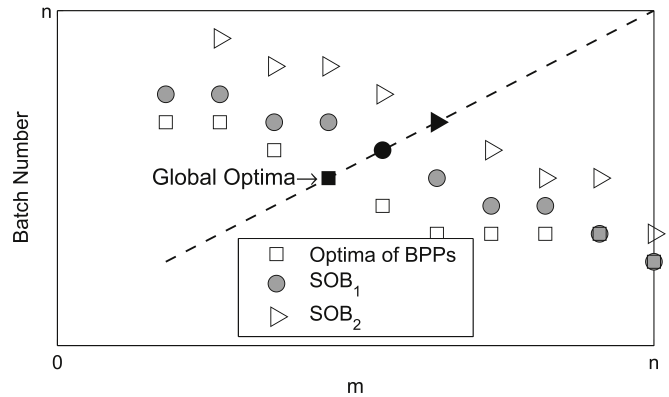


Fig. 2. An illustration of the process of solution validation. The marked global optimum is the target solution. The dotted line is the validation line. The approaches for BPP determine the solution quality. The markers located on the line illustrate the results obtained by different approaches.

With all the analysis presented above, the RMB-BPP heuristic is presented in Algorithm 1. Note that each operation in the loop can be completed in $O(1)$ time except for the SAWMBS algorithm. Since the loops are executed by $n-\lceil n/\beta \rceil$ times at most, RMB-BPP has a time complexity of $O((n-\lceil n/\beta \rceil)n^{\beta+1})$.

Algorithm 1: RMB-BPP

Input: Job set J, bin number capacity β, resource constraints u, v
Output: Resource consumption U^*
Initial value: unpacked bin set $L \leftarrow W$, initial bin $A^* = \emptyset$;
begin
  $m \leftarrow \lceil n/\beta \rceil$
  while $m \le n$ **do**
    $SOB \leftarrow 0$
    Calculate V_s according to Eq. (15)
    Calculate e according to Eq. (16) `/* Calculate` V_s `and` e `*/;`
    $c \leftarrow u(e/k)^{1/(k+1)}$ `/* Obtain the capacity of the corresponding BPP */;`
    while $|L|>0$ **do**
      $A^* \leftarrow$ Pack a new bin by SAWMBS;
      $L \leftarrow L \backslash A^*$ `/* Remove the packed bin */;`
      $SOB \leftarrow SOB + 1$
    if $m \ne SOB$ **then**
      $m \leftarrow m + 1$
      continue `/* Skip to next loop */;`
    $U^* \leftarrow mV_s + (k/e)^{1/(k+1)} \sum_{l=1}^{n} w_l^{k/(k+1)}$ `/* Output the result when the first tie happens */;`
    break `/* Skip out of the loop */;`

6. Experimental study

The experimental study is devoted to the following issues:

1. Comparing RMB-BPP with GAMS for small-scale instances in terms of solution quality;
2. Comparing RMB-BPP with other comparative approaches in literature on small- and large-scale instances.

6.1. Dataset, experiment protocols and comparative approaches

To the best of our knowledge, there is no available dataset corresponding to the PRA policy. In our experimental study, the instances were generated based on the methods of [27,39]: the job

Table 2
Performance comparisons of RMB-BPP with GAMS.

	n	RD(I)								
		0	5%	10%	15%	20%	25%	30%	35%	40%
RMB-BPP	8	0.30	0.33	0.33	0.03	0.00	0.00	0.00	0.00	0.00
	20	0.10	0.40	0.33	0.17	0.00	0.00	0.00	0.00	0.00
	30	0.07	0.33	0.20	0.30	0.10	0.00	0.00	0.00	0.00
	40	0.00	0.13	0.23	0.23	0.27	0.13	0.00	0.00	0.00
GAMS	8	0.30	0.33	0.33	0.03	0.00	0.00	0.00	0.00	0.00
	20	0.13	0.17	0.13	0.30	0.27	0.00	0.00	0.00	0.00
	30	0.00	0.00	0.30	0.07	0.27	0.20	0.00	0.00	0.00
	40	0.00	0.00	0.07	0.23	0.13	0.17	0.07	0.17	0.17

number n was given before each performance test; the machine capacity β was generated from a discrete uniform distribution ranging between $\lceil n/15 \rceil$ and $\lceil n/3 \rceil$ and the workload of the setup operations S was obtained from a continuous uniform distribution $[0.5, 1.5]$; the resource constraint parameters v and u were generated from continuous uniform distributions, i.e., from 1.0 to 2.0 and from 2.0 to 4.0, respectively; k was selected randomly from the set $\{0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$; the workload parameter w_i was generated from a discrete uniform distribution ranging from 1 to 7; the makespan C was also generated from a discrete uniform distribution ranging from $4n$ to $8n$.

In the experimental study, the performance of the RMB-BPP algorithm is compared with available optimization software and approaches in literature. In the first part, the performances of RMB-BPP and GAMS on small instances are compared, and in the second one, RMB-BPP, BFF-KKT (Batch First Fit principle with KKT conditions) and SGA-KKT (Simple Genetic Algorithm with KKT conditions) are evaluated on both small- and large-scale problems. BFF is one of the simplest heuristics developed by Uzsoy [50] for scheduling a batch processing machine with nonidentical job sizes. SGA is a stochastic search algorithm which uses the mechanics of natural selection and natural genetics [51,52]. The fitness is set to $\sum(U_j + V_j)_{max} - \sum(U_j + V_j)$ where $\sum(U_j + V_j)_{max}$ is the maximum possible value of $\sum(U_j + V_j)$. The parameters of SGA-KKT are taken as follows: the population size is 1000, the crossover probability is 0.5, and the mutation probability is 0.02. SGA-KKT is stopped after 1000 generations. Both BFF-KKT and SGA-KKT are based on the following strategy: batching first and allocating resources afterwards. Both approaches apply the KKT conditions for resource allocation.

The GAMS results were provided by applying GAMS IDE (Ver. 23.2.1) with the formulations in Section 3. The other approaches were implemented on MATLAB R2013b (8.2.0.701). All the approaches were executed on a 2.2 GHz, Core II processor, 2 GB RAM PC with Win7 OS.

6.2. Performance comparisons against GAMS

GAMS is a modeling system for mathematical programming and optimization, which offers a stable of commercial solvers. During the executions, it was found that when n is equal to or greater than 50, GAMS cannot provide an effective solution in reasonable time (2000 s for 50 jobs). Therefore, the performance comparisons of RMB-BPP against GAMS are conducted on relatively small-scale instances, including 8, 20, 30, and 40 jobs.

For each n, 30 instances were generated at random. The results of RMB-BPP and GAMS are compared with the lower bound computed by Eq. (11). The parameters that were kept record of are the relative deviation:

$$RD(I) = \frac{A_i(I) - LB(I)}{LB(I)}, \tag{21}$$

where $A_i(I)$ represents the result gained by method A_i (A_1 = RMB-BPP, A_2 = GAMS) and $LB(I)$ denotes the lower bound of instance I.

Among the 30 instances of each n, for both RMB-BPP and GAMS, the percentages of RD were kept track of, which are less than or equal to 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%. The results are shown in Table 2.

According to Table 2, in general, RMB-BPP shows better performance in that RMB-BPP can achieve smaller RD values (for the 8 case, ties happen). In order to better illustrate the results, the radar chart of each instance configuration is presented in Fig. 3, and the comparisons of the median and average values are given in Fig. 4.

As can be seen from Figs. 3 and 4, most of the values gained are lower than 0.2, which demonstrates that both RMB-BPP and GAMS are effective. Except for the instances of 8 jobs, RMB-BPP achieved better performance than GAMS. An interesting observation is that, the gap between RMB-BPP and GAMS increases as the problem scales from 8 to 40, which implies that RMB-BPP has better scalability than GAMS.

6.3. Comparisons against BFF-KKT and SGA-KKT

The comparisons among RMB-BPP, BFF-KKT and SGA-KKT are carried out on the instances with 20, 40, 80, 100, 150, 200 jobs. For each case, 10 instances are generated. RMB-BPP and BFF-KKT are executed on each instance for once and SGA-KKT for 5 times. The results of each instance are compared with the corresponding LBs.

Fig. 5 shows the convergence curves of SGA-KKT on the first instance of each case. To make brief comparisons, the results of RMB-BPP and BFF-KKT are also presented.

As can be seen from Fig. 5, generally speaking, RMB-BPP can achieve better performance than BFF-KKT and SGA-KKT. BFF-KKT has the most unsatisfying behavior. In some cases (for example, instance of 80 jobs), the result by SGA-KKT is very close to RMB-BPP. However, SGA-KKT spent much more time than RMB-BPP to provide the final results. Moreover, being a stochastic search meta-heuristic, the solution quality of SGA-KKT cannot be guaranteed theoretically.

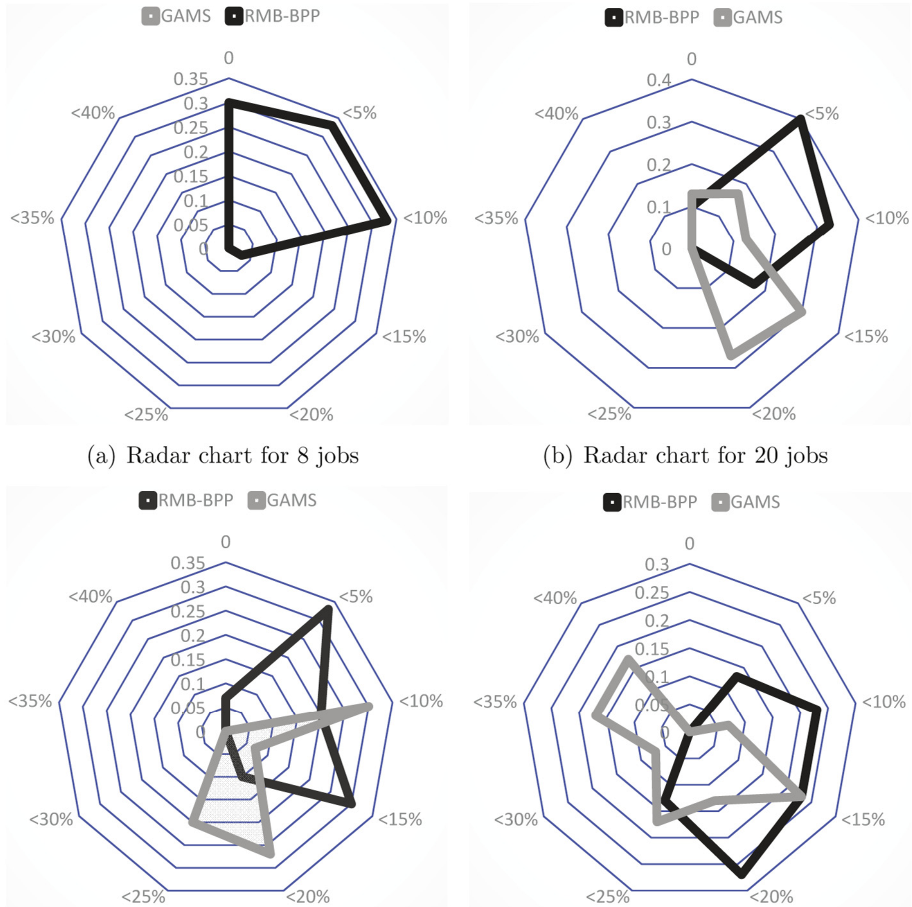
The box plots of the results on all the instances are presented in Fig. 6. As can be seen from Fig. 6, most of the values of RMB-BPP are between 0.1 and 0.2, which demonstrates that it can be a strong alternative for solving RCM. Under special cases, SGA-KKT can achieve better (or not worse) performance than RMB-BPP. However, for each case, the median value of SGA-KKT is greater than RMB-BPP, which in turn proves that RMB-BPP achieves more efficient solutions. BFF-KKT presents the worst behavior for all the instances.

To statistically compare the performance of the algorithms, a Wilcoxon signed rank test is performed with a significance level of 0.05. As the results depicted in Table 3 show, the difference between RMB-BPP and SGA-KKT is statistically significant on all the instances. The difference between RMB-BPP and BFF-KKT is statistically significant on all the instances.

Table 3
Wilcoxon signed rank test for RMB-BPP, SGA-KKT and BFF-KKT.

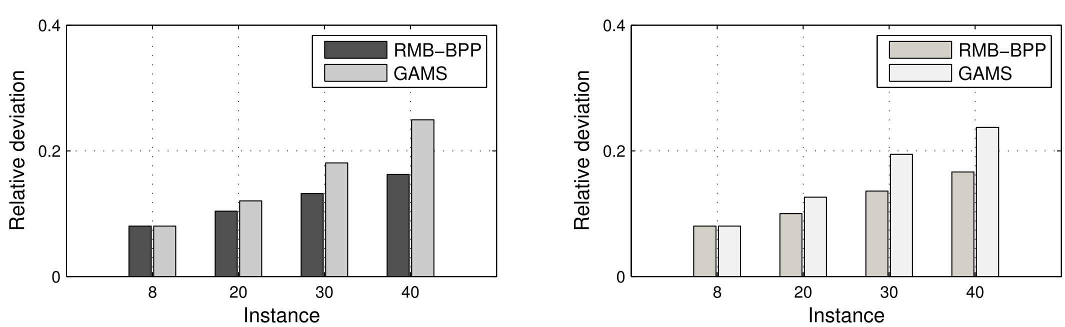
(RMB-BPP)-(SGA-KKT)		(RMB-BPP)-(BFF-KKT)	
n	p-value	n	p-value
20	0.0021	20	0.0004
40	0.0036	40	0.0003
80	0.0032	80	0.0020
100	0.0220	100	0.0024
150	0.0014	150	0.0009
200	0.0018	200	0.0013
All	0.0000	All	0.0000

Based on the above experiments, the following conclusions can be drawn:



(a) Radar chart for 8 jobs
(b) Radar chart for 20 jobs
(c) Radar chart for 30 jobs
(d) Radar chart for 40 jobs

Fig. 3. Radar charts for RDs of RMB-BPP and GAMS. The closer a result is to 0, the better it is.



(a) Average values for different job numbers
(b) Median values for different job numbers

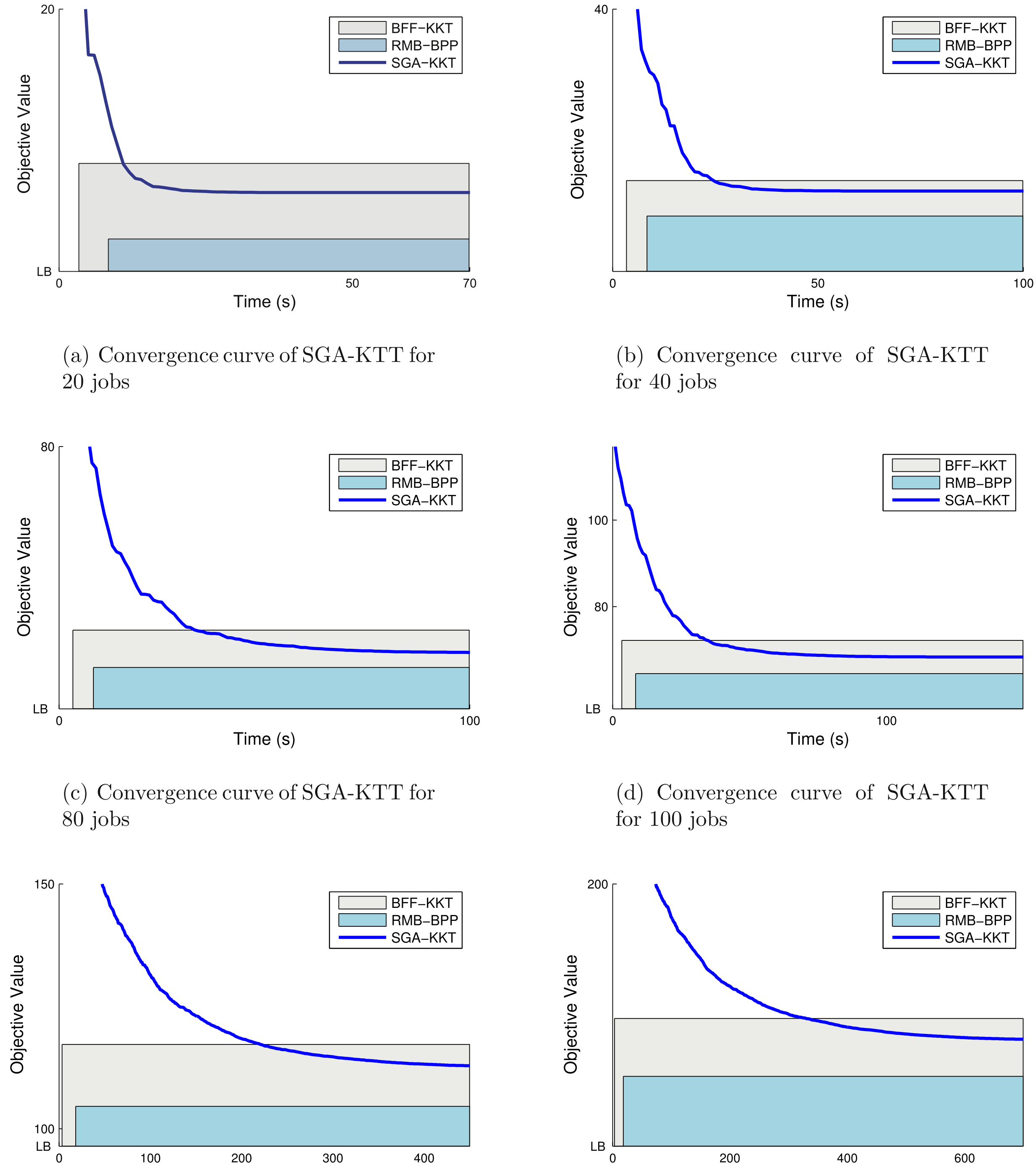
Fig. 4. Comparison results of RMB-BPP against GAMS. The results on both the average and median values for different job numbers (each with 30 instances) are presented.

1. Both RMB-BPP and GAMS show competitive performances on small-scale instances.
2. The computational time of GAMS is huge, which makes it improper for large-scale instances.
3. RMB-BPP outperforms SGA-KKT and BFF-KKT in solution quality, especially for large-scale instances.

7. Conclusions

In this paper, the mechanism for transforming RCM into BPP is studied. To construct the mechanism, the characteristics of the optimal solutions are analyzed. Following the analysis, it is shown that the problem can be tackled by finite BPPs. Based on a mature approach for BPP, the heuristic algorithm, RMB-BPP is provided. In order to demonstrate its effectiveness and efficiency, extensive experiments are conducted. The results of the experiments show that RMB-BPP is effective in both small- and large-scale instances, which provides the best results among all the compared approaches. The conclusions provide theoretical basis and a new view for tackling problems in this subfield of scheduling.

The paper presents a new view of allocating resources by bin-packing. Concluding the precursory research [28] and this work,



(a) Convergence curve of SGA-KTT for 20 jobs

(b) Convergence curve of SGA-KTT for 40 jobs

(c) Convergence curve of SGA-KTT for 80 jobs

(d) Convergence curve of SGA-KTT for 100 jobs

(e) Convergence curve of SGA-KTT for 150 jobs

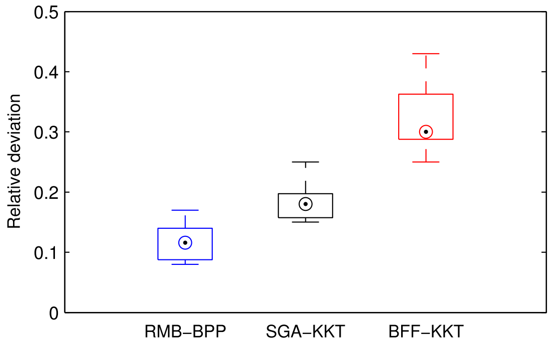
(f) Convergence curve of SGA-KTT for 200 jobs

Fig. 5. Convergence curves of SGA-KKT on the first instance of each case. For comparison, the results of RMB-BPP and BFF-KKT are also presented. Both BFF-KTT and RMB-BPP start from the time when they present the final solution. The curve of SGA-KKT is the one with the best result in 5 executions. The value of each Y axis starts from the corresponding LB.

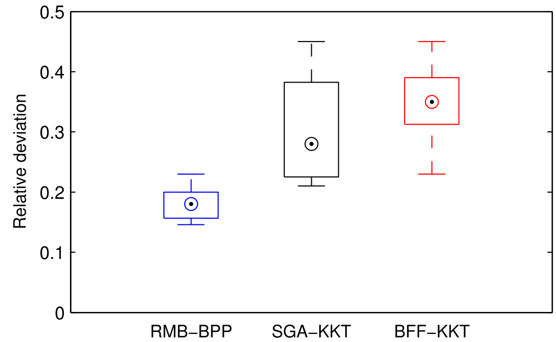
the resource constraints in scheduling problems are capacity-like, which explains the potential of the transformation into BPP variants. Note that in scheduling with resource allocations, there exist lots of capacity-like (upper-bounded) constraints. It is believed both interesting and challenging to investigate other variants, which may generate meaningful applications.

Acknowledgments

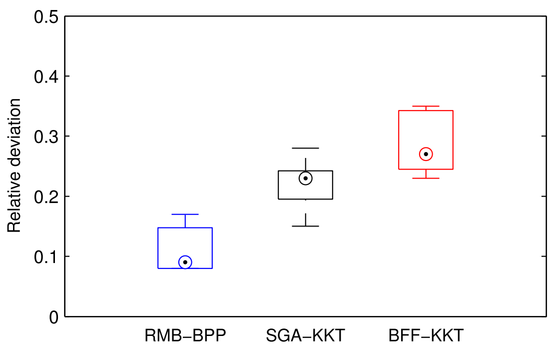
This work was supported by the National Natural Science Foundation of China (Grant no. 71671168) and the Natural Science Foundation of Hunan Province, China (Grant no. 2018JJ3891).



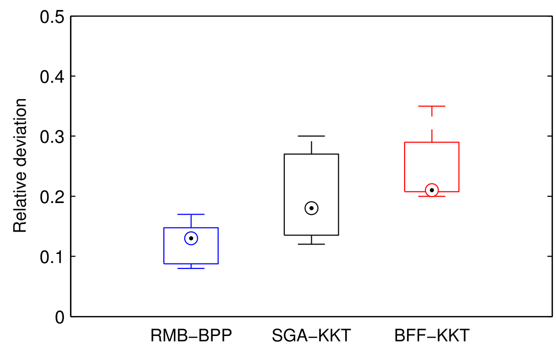
(a) Box plot for 20 jobs



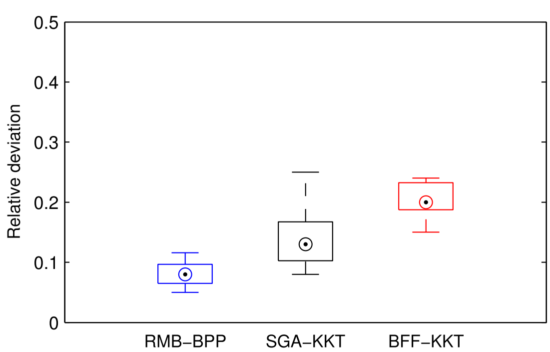
(b) Box plot for 40 jobs



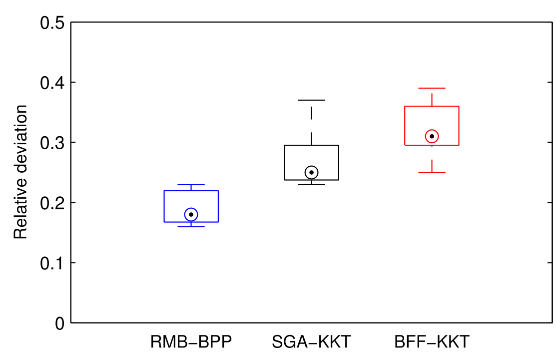
(c) Box plot for 80 jobs



(d) Box plot for 100 jobs



(e) Box plot for 150 jobs



(f) Box plot for 200 jobs

Fig. 6. The box plots of the relative deviations of the compared approaches. The best result in the 5 executions of SGA-KKT is used.

References

[1] C.T.D. Ng, T.C.E. Cheng, A. Janiak, M.Y. Kovalyov, Group scheduling with controllable setup and processing times: minimizing total weighted completion time, Ann. Oper. Res. 133 (1–4) (2005) 163–174.

[2] X. Liang, H. Chen, J.A. Lozano, A boltzmann-based estimation of distribution algorithm for a general resource scheduling model, IEEE Trans. Evol. Comput. 19 (6) (2015) 793–806.

[3] Y. Gu, W. Saad, M. Bennis, M. Debbah, Z. Han, Matching theory for future wireless networks: fundamentals and applications, IEEE Commun. Mag. 53 (5) (2015) 52–59.

[4] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, C.S. Hong, Caching in the sky: proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience, IEEE J. Sel. Areas Commun. 35 (5) (2017) 1046–1061.

[5] R. Vickson, Two single machine sequencing problems involving controllable job processing times, AIIE Trans 12 (3) (1980) 258–262.

[6] R. Vickson, Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine, Oper. Res. 28 (5) (1980) 1155–1167.

[7] D. Shabtay, G. Steiner, A survey of scheduling with controllable processing times, Discrete Appl. Math. 155 (13) (2007) 1643–1666.

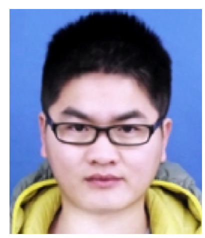
[8] X.-Y. Wang, J.-J. Wang, Single-machine due date assignment problem with deteriorating jobs and resource-dependent processing times, Int. J. Adv. Manuf. Technol. 67 (1–4) (2013) 255–260.

[9] Y. Yin, T.C.E. Cheng, C.-C. Wu, S.-R. Cheng, Single-machine common due-date scheduling with batch delivery costs and resource-dependent processing times, Int. J. Prod. Res. 51 (17) (2013) 5083–5099.

[10] S.-J. Yang, H.-T. Lee, J.-Y. Guo, Multiple common due dates assignment and scheduling problems with resource allocation and general position-dependent deterioration effect, Int. J. Adv. Manuf. Technol. 67 (1–4) (2013) 181–188.

[11] A. Janiak, One-machine scheduling with allocation of continuously-divisible resource and with no precedence constraints, Kybernetika 23 (4) (1987) 289–293.

[12] T. Cheng, M.Y. Kovalyov, Single machine batch scheduling with deadlines and resource dependent processing times, Oper. Res. Lett. 17 (5) (1995) 243–249.
[13] A. Janiak, M.Y. Kovalyov, Single machine scheduling subject to deadlines and resource dependent processing times, European J. Oper. Res. 94 (2) (1996) 284–291.
[14] B. Alidaee, A. Ahmadian, Two parallel machine sequencing problems involving controllable job processing times, European J. Oper. Res. 70 (3) (1993) 335–341.
[15] A. Rudek, R. Rudek, On flowshop scheduling problems with the aging effect and resource allocation, Int. J. Adv. Manuf. Technol. 62 (1–4) (2012) 135–145.
[16] X. Luo, W. Li, Y. Tu, D. Xue, J. Tang, Optimal resource allocation for hybrid flow shop in one-of-a-kind production, Int. J. Comput. Integr. Manuf. 23 (2) (2010) 146–154.
[17] T. Cheng, A. Janiak, M.Y. Kovalyov, Single machine batch scheduling with resource dependent setup and processing times, European J. Oper. Res. 135 (1) (2001) 177–183.
[18] C. Ng, T. Cheng, M.Y. Kovalyov, Single machine batch scheduling with jointly compressible setup and processing times, European J. Oper. Res. 153 (1) (2004) 211–219.
[19] B. Cheng, K. Li, B. Chen, Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization, J. Manuf. Syst. 29 (1) (2010) 29–34.
[20] X. Liang, H. Chen, R. Xu, Approximately optimal algorithms for scheduling a single machine with general convex resource functions, Int. J. Comput. Integr. Manuf. 28 (9) (2015) 920–935.
[21] B.-Y. Cheng, J.Y.-T. Leung, K. Li, S.-L. Yang, Single batch machine scheduling with deliveries, Nav. Res. Logist. 62 (6) (2015) 470–482.
[22] Z. Jia, X. Li, J.Y.-T. Leung, Minimizing makespan for arbitrary size jobs with release times on p-batch machines with arbitrary capacities, Future Gener. Comput. Syst. 67 (2017) 22–34.
[23] M. Mathirajan, A. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, Int. J. Adv. Manuf. Technol. 29 (9–10) (2006) 990–1001.
[24] C.A. Méndez, J. Cerdá, I.E. Grossmann, I. Harjunkoski, M. Fahl, State-of-the-art review of optimization methods for short-term scheduling of batch processes, Comput. Chem. Eng. 30 (6) (2006) 913–946.
[25] A. Janiak, M.Y. Kovalyov, M.-C. Portmann, Single machine group scheduling with resource dependent setup and processing times, European J. Oper. Res. 162 (1) (2005) 112–121.
[26] D. Oron, Scheduling a batching machine with convex resource consumption functions, Inform. Process. Lett. 111 (19) (2011) 962–967.
[27] B. Mor, G. Mosheiov, Batch scheduling of identical jobs with controllable processing times, Comput. Oper. Res. 41 (2014) 115–124.
[28] M.R. Garey, R.L. Graham, D.S. Johnson, A.C.-C. Yao, Resource constrained scheduling as generalized bin packing, J. Combin. Theory Ser. A 21 (3) (1976) 257–298.
[29] A. Grange, I. Kacem, S. Martin, Algorithms for the bin packing problem with overlapping items, Comput. Ind. Eng. 115 (2018) 331–341.
[30] A. Layeb, S.R. Boussalia, A novel quantum inspired cuckoo search algorithm for bin packing problem, Int. J. Inf. Technol. Comput. Sci. 4 (5) (2012) 58–67.
[31] H.I. Christensen, A. Khan, S. Pokutta, P. Tetali, Approximation and online algorithms for multidimensional bin packing: a survey, Comput. Sci. Rev. 24 (2017) 63–79.
[32] D. Shabtay, G. Steiner, Single machine batch scheduling to minimize total completion time and resource consumption costs, J. Sched. 10 (4–5) (2007) 255–261.
[33] B.-C. Choi, S.-H. Yoon, S.-J. Chung, Single machine scheduling problem with controllable processing times and resource dependent release times, European J. Oper. Res. 181 (2) (2007) 645–653.
[34] D. Shabtay, Single and two-resource allocation algorithms for minimizing the maximal lateness in a single machine, Comput. Oper. Res. 31 (8) (2004) 1303–1315.
[35] D. Shabtay, M. Kaspi, Parallel machine scheduling with a convex resource consumption function, European J. Oper. Res. 173 (1) (2006) 92–107.
[36] N.V. Shakhlevich, V.A. Strusevich, Single machine scheduling with controllable release and processing parameters, Discrete Appl. Math. 154 (15) (2006) 2178–2199.
[37] C.L. Monma, A. Schrijver, M.J. Todd, V.K. Wei, Convex resource allocation problems on directed acyclic graphs: duality, complexity, special cases, and extensions, Math. Oper. Res. 15 (4) (1990) 736–748.
[38] L. Yedidsion, D. Shabtay, M. Kaspi, A bicriteria approach to minimize maximal lateness and resource consumption for scheduling a single machine, J. Sched. 10 (6) (2007) 341–352.
[39] L. Yedidsion, D. Shabtay, E. Korach, M. Kaspi, A bicriteria approach to minimize number of tardy jobs and resource consumption in scheduling a single machine, Int. J. Prod. Econ. 119 (2) (2009) 298–307.
[40] D. Biskup, H. Jahnke, Common due date assignment for scheduling on a single machine with jointly reducible processing times, Int. J. Prod. Econ. (3) (2001) 317–322.
[41] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1979) 287–326.
[42] P. Brucker, Scheduling Algorithms, Springer, Berlin, 2004.
[43] J. Blazewicz, K.H. Ecker, G. Schmidt, J. Weglarz, Scheduling in computer and manufacturing systems, J. Oper. Res. Soc. 47 (4) (1996) 592–592.
[44] W. Karush, Minima of Functions of Several Variables with Inequalities as Side Constraints (Master's thesis), Dept. of Mathematics, Univ. of Chicago, 1939.
[45] H.W. Kuhn, A.W. Tucker, Nonlinear programming, in: Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability, 1951, pp. 481–492.
[46] R. Michael, D.S. Johnson, Computers and Intractability: A guide to the theory of NP-completeness, WH Freeman & Co., San Francisco, 1979.
[47] K. Fleszar, K.S. Hindi, New heuristics for one-dimensional bin-packing, Comput. Oper. Res. 29 (7) (2002) 821–839.
[48] M.R. Garey, D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, SIAM J. Comput. 4 (4) (1975) 397–411.
[49] J.N. Gupta, J.C. Ho, A new heuristic algorithm for the one-dimensional bin-packing problem, Prod. Plan. Control 10 (6) (1999) 598–603.
[50] R. Uzsoy, Scheduling a single batch processing machine with non-identical job sizes, Int. J. Prod. Res. 32 (7) (1994) 1615–1635.
[51] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, MIT Press, 1999.
[52] N. Samaan, C. Singh, Adequacy assessment of power system generation using a modified simple genetic algorithm, IEEE Trans. Power Syst. 17 (4) (2002) 974–981.



Xinle Liang received the bachelor's degree in information management and information system from Beijing Institute of Technology, Beijing, China, in 2011, and the Ph.D. degree in computer science and technology from University of Science and Technology of China, Hefei, China, in 2016. His current research interests include evolutionary algorithms, planning and scheduling. He has published papers in journals such as IEEE Transactions on Evolutionary Computation, IEEE Transactions on Services Computing, International Journal of Computer Integrated Manufacturing, and Expert Systems with Applications.

Shengchao Zhou received the bachelor's degree in information management and information system from Nanjing University of Science and Technology, Nanjing, China, in 2009, and the Ph.D. degree in management science and engineering from University of Science and Technology of China, Hefei, China, in 2016. He was a Visiting Scholar in the Department of Industrial and Systems Engineering at University of Tennessee, Knoxville, USA, between Aug. 2014 and Dec. 2015.

His research interests include evolutionary computation, modeling, scheduling and optimization in manufacturing systems, and differential evolution. He has published papers in journals such as Computers & Operations Research, Applied Mathematics and Computation, Applied Soft Computing, International Journal of Production Economics, and International Journal of Production Research.

Huaping Chen received the Ph.D. degree in computer science from University of Science and Technology of China, Hefei, China, in 1997.

He is a Professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include evolutionary algorithms, mixed integer programming, and scheduling.

Rui Xu received the bachelor's degree in information management and information system from Anhui Medical University, Hefei, China, in 2005, and the Ph.D. degree in management science and engineering from University of Science and Technology of China, Hefei, China, in 2011.

He is an Associate Professor with the Business School, Hohai University, Nanjing, China. His research interests include intelligent algorithms, and production scheduling.