Soft Computing (2019) 23:8677–8688
https://doi.org/10.1007/s00500-018-3466-5

METHODOLOGIES AND APPLICATION

Artificial bee colony algorithms for the order scheduling with release dates

Win-Chin Lin[1] · Jianyou Xu[2] · Danyu Bai[3] · I-Hong Chung[1] · Shang-Chia Liu[4] · Chin-Chia Wu[1]

Published online: 28 August 2018

Abstract
The order scheduling models have kept growing attention in the research community. However, as studying research regarding order scheduling models with release dates is relatively limited; this study addresses an order scheduling problem with release dates where the objective function is to minimize the weighted number of tardy orders of all the given orders. To solve this intractable problem, this study first proposes some dominance properties and a lower bound used in a branch-and-bound method for finding an optimal solution. This paper then utilizes four basic bee colony algorithms, and four hybrid bee colony algorithms for searching the optimal solution and approximate solution, and performs one-way analysis of variance and Fisher's least significant difference tests to determine and evaluate the performances of all eight proposed algorithms.

Keywords Order scheduling · Artificial bee colony algorithm · Branch-and-bound algorithm

1 Introduction

Recently, order scheduling (OS) problem has become an important issue in many service and manufacturing environments, in which a product development team develops modules independently for several products, and the product design is deemed to be completed when all the modules have been designed (seen Ahmadi et al. 2005; Wang and Cheng 2007).

For relevant OS literature studies, Wagneur and Sriskandarajah (1993) claimed that the maximum completion time minimization and the maximum tardiness minimization problems are solved in polynomial time, that the number of late jobs minimization problem is in binary NP-hard in the two-machine case, and that the total completion time minimization and the total tardiness minimization problems are unary NP-hard for $m \geqslant 2$. Meanwhile, Leung et al. (2005a) gave a counterexample and showed that the number of late jobs minimization problem in the two-machine case is still an open issue. Leung et al. (2005b) showed that the OS model on three- or more machine settings is strongly NP-hard, and derived a heuristic based on the shortest processing time rule on the machine with the largest current load and a heuristic based on the earliest completion time for approximate solutions. For the two-machine case, Sung and Yoon (1998) provided the shortest maximum processing time (SMPT) algorithm and the shortest total processing time (STPT) algorithms to solve it. Yang (2005a, b) considered *OS* problems associated with the completion time of the orders and analyzed the complexity of several problems with different types of objectives, job restrictions, and machine environments. Ahmadi et al. (2005) showed that the weighted sum of customer order delivery times minimization is proved as unary NP-hard, characterized the optimal schedule, solved several special cases of the problem, derived tight lower bounds, and proposed several heuristic solutions.

Regarding the literature research for minimizing the total weighted order completion time, Wang and Cheng (2007) claimed that the problem is unary NP-hard, and proposed a heuristic and discussed its worst-case behavior. Yoon and Sung (2005) built a branch-and-bound method to solve

Communicated by V. Loia.

✉ Chin-Chia Wu
cchwu@fcu.edu.tw

[1] Department of Statistics, Feng Chia University, Taichung 40724, Taiwan

[2] College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

[3] College of Economics and Management, Nanjing Forestry University, Nanjing 210037, China

[4] Department of Business Administration, Fujen Catholic University, New Taipei City, Taiwan

the problem, while Leung et al. (2006a, 2007a, 2008a) derived heuristic algorithms for finding approximate solutions.

For some OS studies involving due dates, Lee (2013) considered the tardiness minimization as the objective function, derived some dominance properties and three lower bounds in a branch-and-bound algorithm for finding optimal solution, and proposed three heuristic algorithms for approximate solutions. Introducing the position-based learning idea to the Lee's model, Xu et al. (2016a) proposed a branch-and-bound algorithm incorporating certain dominance rules and three lower bounds and simulated annealing, particle swarm optimization, and modified earliest due dates of orders to solve the problem. Lin et al. (2017) addressed a two-agent order scheduling with ready times and utilized a branch-and-bound method, a particle swarm optimization, and opposite-based particle swarm optimization for the optimal solution and approximate solution. Additional OS works in other settings, readers may refer to Blocher et al. (1998), Erel and Ghosh (2007), Framinan and Perez-Gonzalez (2017), Lin and Kononov (2007), Hsu and Liu (2009), Wu et al. (2018), Xu et al. (2016b), Yang and Posner (2005), Zhao et al. (2016), and so on.

The issue of the release dates of orders in the OS models has, however, been rarely studied. In manufacturing semi-finished lenses, the complexity of relevant customer satisfaction criteria, along with meeting customer due dates with different priorities, poses a difficult challenge. Readers may refer to Ahmadi et al. (2005) for details. In addition, the occurrence of resources in parallel is common in many real applications. Given the unequal ready times of the jobs on parallel batch machines, a good policy is to wait and combine future job arrivals to increase the completeness of a batch (Monch et al. 2005). For the importance of the due dates in real applications, readers may refer to French (1982) and Armentano and Ronconi (1999) for the scheduled landing time of an aircraft and for several production examples involving due date settings.

Motivated by these observations and the relatively limited use of metaheuristic to solve order scheduling models, in this study we consider both release dates and number of tardy jobs in the OS model. The main contributions of this study are summarized as follows: A new OS model of considering release dates for each orders and the weighted number of tardy orders criterion is addressed. A branch-and-bound algorithm along with several properties and a lower bound is developed to find the optimal solution. Eight versions of artificial bee colony (ABC) algorithm with an improvement scheme are developed for finding the high-quality near-optimal solutions. Some extensive computational experiments and statistical tools are applied to evaluate the performances for all proposed algorithms.

The remainder of this study is summarized as follows. Section 2 presents the notation and problem formulation. Section 3 introduces several properties and a lower bound, artificial bee colony algorithm, and the branch-and-bound algorithm. Section 4 reports and evaluates observations of all the proposed algorithms, and Sect. 5 provides conclusions and suggestions.

2 Notation and problem statement

The notations used in this study are given in the following:

n: order number,
m: machine number,
M_k: machine code k, where $k=1, \ldots, m$,
σ, σ': schedules of n orders, where $\sigma = (\pi, i, j, \pi')$ and $\sigma' = (\pi, j, i, \pi')$ are two full sequences, and π and π' denote two partial sequences,
p_{vk}: component k processing time of order v processed on machine k, for $1 \le k \le m$.
d_v: due date for order v, for $1 \le v \le n$.
t_k: a starting time on M_k, for $1 \le k \le m$.
$C_v(\sigma)$: finished time of order v in σ.
[]: determined position in a given sequence;
$U_v(\sigma) = 0$, when $C_v(\sigma) < d_v$, while $U_v(\sigma) = 1$, when $C_v(\sigma) > d_i$.
$U_i(\sigma)$ and $U_j(\sigma)$: tardiness of order i and order j in σ;
$U_j(\sigma')$ and $U_i(\sigma')$: tardiness of order j and order i in σ';
N_b: number of employee bees;
N_{on}: number of onlooker bees;
ITRN: number of repeat cycles or the stopping criterion that the ABC algorithm terminates;
limit: number of improvements;
p_scout: the value for the employed bee abandoned;
em_bee$_i$: an employed bee with em_bee$_i \in R^n$, $i=1, 2, \ldots, N_b$;
on_bee$_i$: an onlooker bee with on_bee$_i \in R^n$, $i=1, 2, \ldots, N_{on}$;
em_bee$_{ij}$: an employed bee with em_bee$_{ij} \in R^n$, $i=1,2, \ldots, N_b$; $j=1,2, \ldots,$ ITRN;
on_bee$_{ij}$: an onlooker bee with on_bee$_{ij} \in R^n$, $i=1,2, \ldots, N_{on}$; $j=1,2, \ldots,$ ITRN;
v_{ij}: a new candidate bee with $v_{ij} \in R^n$, $i=1,2, \ldots, N_b$; $j=1,2, \ldots,$ ITRN;

$f_i = \frac{1}{1+\sum_{l=1}^{N_b} w_l U_l(\sigma)}$: the fitness value for employed bee i, where an employed bee as a schedule σ of the n orders in this study;
$p\_bee_i = \frac{f_i}{\sum_{l=1}^{N_b} f_l}$: a selection probability for an employed bee i;

The model can be formally stated as follows. Consider a set of n orders from n different clients and m components for each order. The m components are operated on different machines in parallel. A machine can only process a particular component. In this study, we assume that each order has its own release time. The interruption or preemption is not allowed. Let p_{vk} be the *component k* processing time of order v to be processed on M_k. Let w_i, d_i, and r_i denote the weighted, due date, and release date of order *i,* respectively. In this study, we aim to explore an optimal sequence to minimize the weighted number of tardy orders. Karp (1972) showed that the total weighted number of tardiness scheduling problem (or named as $1//\sum w_i U_i$) is NP-hard when each order has one component only. Accordingly, our proposed problem $1/r_i, OS/\sum w_i U_i$ is also NP-hard. Therefore, some dominant properties and a lower bound are derived to find an optimal solution for the branch-and-bound (B&B) method.

3 The branch-and-bound method

3.1 Dominance and lower bound

We build several properties and a lower bound to facilitate the B&B to quickly find an optimal schedule. To show that σ dominates σ', it suffices to prove $w_i U_i(\sigma) + w_j U_j(\sigma) \le w_j U_j(\sigma') + w_i U_i(\sigma')$ and $C_j(\sigma) < C_i(\sigma')$. The proofs are omitted because they can be discussed by the pairwise interchange method. For more details of the B&B, readers may refer to Wang et al. (2017a, b) and Cheng et al. (2017).

Property 1-1 If $r_j > r_i \ge \max_{1\le k\le m}\{t_k\}$, $\max_{1\le k\le m}\{p_{ik}\} < \max_{1\le k\le m}\{p_{jk}\}$, $r_i + \max_{1\le k\le m}\{p_{ik}\} > r_j$, $r_i + \max_{1\le k\le m}\{p_{ik}\} < d_i$, $r_j + \max_{1\le k\le m}\{p_{jk}\} + \max_{1\le k\le m}\{p_{ik}\} > d_i$, and $r_i + \max_{1\le k\le m}\{p_{ik}\} + \max_{1\le k\le m}\{p_{jk}\} < d_j$, then σ dominates σ'.

Property 1-2 If $r_j > r_i \ge \max_{1\le k\le m}\{t_k\}$, $\max_{1\le k\le m}\{p_{ik}\} < \max_{1\le k\le m}\{p_{jk}\}$, $r_i + \max_{1\le k\le m}\{p_{ik}\} > r_j$, $r_j + \max_{1\le k\le m}\{p_{jk}\} > d_j$, $r_j + \max_{1\le k\le m}\{p_{jk}\} + \max_{1\le k\le m}\{p_{ik}\} > d_i$, and $d_i > r_i + \max_{1\le k\le m}\{p_{ik}\}$, then σ dominates σ'.

Property 1-3 If $r_j > r_i \ge \max_{1\le k\le m}\{t_k\}$, $\max_{1\le k\le m}\{p_{ik}\} < \max_{1\le k\le m}\{p_{jk}\}$, $r_i + \max_{1\le k\le m}\{p_{ik}\} > r_j$, $r_j + \max_{1\le k\le m}\{p_{jk}\} > d_j$, and $d_i > r_i + \max_{1\le k\le m}\{p_{ik}\}$, then σ dominates σ'.

Property 1-4 If $r_j > r_i \ge \max_{1\le k\le m}\{t_k\}$, $\max_{1\le k\le m}\{p_{ik}\} < \max_{1\le k\le m}\{p_{jk}\}$, $r_i + \max_{1\le k\le m}\{p_{ik}\} > r_j$, $d_j > r_i + \max_{1\le k\le m}\{p_{ik}\} + \max_{1\le k\le m}\{p_{jk}\}$, and $d_l > r_j + \max_{1\le k\le m}\{p_{jk}\} + \max_{1\le k\le m}\{p_{ik}\}$, then σ dominates σ'.

Property 2-1 If $r_i \ge \max_{1\le k\le m}\{t_k\}$, $r_i + \max_{1\le k\le m}\{t_k\} < r_j$, $r_j + \max_{1\le k\le m}\{p_{jk}\} < d_j$, and $r_i + \max_{1\le k\le m}\{p_{ik}\} < d_i < r_j + \max_{1\le k\le m}\{p_{jk}\} + \max_{1\le k\le m}\{p_{ik}\}$, then σ dominates σ'.

Property 2-2 If $r_i \ge \max_{1\le k\le m}\{t_k\}$, $r_i + \max_{1\le k\le m}\{t_k\} < r_j$, $r_j + \max_{1\le k\le m}\{p_{jk}\} > d_j$, and $r_i + \max_{1\le k\le m}\{p_{ik}\} < d_i < r_j + \max_{1\le k\le m}\{p_{jk}\} + \max_{1\le k\le m}\{p_{ik}\}$, then σ dominates σ'.

Property 2-3 If $r_i \ge \max_{1\le k\le m}\{t_k\}$, $r_i + \max_{1\le k\le m}\{t_k\} < r_j$, $w_i > w_j$, and $r_j + \max_{1\le k\le m}\{p_{jk}\} < d_j < r_i + \max_{1\le k\le m}\{p_{ik}\} + \max_{1\le k\le m}\{p_{jk}\}$, then σ dominates σ'.

Property 2-4 If $r_i \ge \max_{1\le k\le m}\{t_k\}$, $r_i + \max_{1\le k\le m}\{t_k\} < r_j$, $r_j + \max_{1\le k\le m}\{p_{jk}\} > d_j$, and $r_i + \max_{1\le k\le m}\{p_{ik}\} < d_i$, then σ dominates σ'.

Property 2-5 If $r_i \ge \max_{1\le k\le m}\{t_k\}$, $r_i + \max_{1\le k\le m}\{t_k\} < r_j$, $d_j > r_i + \max_{1\le k\le m}\{p_{ik}\} + \max_{1\le k\le m}\{p_{jk}\}$, and $d_j > r_j + \max_{1\le k\le m}\{p_{jk}\} + \max_{1\le k\le m}\{p_{ik}\}$, then σ dominates σ'.

Property 3-1 If $\max\{t_k, r_j\} > \max\{t_k, r_i\}$ and $p_{ik} \le p_{jk}$ for $k = 1, 2, \ldots, m$, and $\max_{1\le k\le m}\{\max\{t_k, r_i\} + p_{ik} + p_{jk}\} < d_j$, and $\max_{1\le k\le m}\{\max\{t_k, r_j\} + p_{jk} + p_{ik}\} > d_i > \max_{1\le k\le m}\{\max\{t_k, r_i\} + p_{ik}\}$, then σ dominates σ'.

Property 3-2 If $\max\{t_k, r_j\} > \max\{t_k, r_i\}$ and $p_{ik} \le p_{jk}$ for $k = 1, 2, \ldots, m$, and $\max_{1\le k\le m}\{\max\{t_k, r_j\} + p_{jk}\} > d_j$, $\max_{1\le k\le m}\{\max\{t_k, r_i\} + p_{ik}\} < d_i$, and $\max_{1\le k\le m}\{\max\{t_k, r_j\} + p_{jk} + p_{ik}\} > d_i$, then σ dominates σ'.

Property 3-3 If $\max\{t_k, r_j\} > \max\{t_k, r_i\}$ and $p_{ik} \le p_{jk}$ for $k = 1, 2, \ldots, m$, $w_i > w_j$, and $\max_{1\le k\le m}\{\max\{t_k, r_i\} + p_{ik} + p_{jk}\} > d_j > \max_{1\le k\le m}\{\max\{t_k, r_j\} + p_{jk}\}$, then σ dominates σ'.

Property 3-4 If $\max\{t_k, r_j\} > \max\{t_k, r_i\}$ and $p_{ik} \le p_{jk}$ for $k = 1, 2, \ldots, m$, $\max_{1\le k\le m}\{\max\{t_k, r_j\} + p_{jk}\} > d_j$, and $\max_{1\le k\le m}\{\max\{t_k, r_i\} + p_{ik}\} < d_i$, then σ dominates σ'.

Property 3-5 If $\max\{t_k, r_j\} > \max\{t_k, r_i\}$ and $p_{ik} \le p_{jk}$ for $k = 1, 2, \ldots, m$, $\max_{1\le k\le m}\{\max\{t_k, r_i\} + p_{ik} + p_{jk}\} < d_j$, and $\max_{1\le k\le m}\{\max\{t_k, r_j\} + p_{jk} + p_{ik}\} < d_i$, then σ dominates σ'.

Next, we present one more property to justify a feasible sequence of the unscheduled orders. Assume that (π, π^c) is a schedule of orders in which π is a partial determined schedule with q orders and π^c is another undetermined part with $(n–q)$ orders. Furthermore, let π^{edd} be the sequence according to the earliest due date first rule (EDD) of undetermined $(n–q)$ order and t_k denote the finished time of the last order on M_k in π for $1 \le k \le m$.

Property 4 If $\max_{1\le k\le m}\left\{\max\{t_k, r_j\} + p_{jk}\right\}_{j\in\pi^c} > \max_{j\in\pi^c}\{d_j\}$ holds, then (π, π^{edd}) dominates (π, π^c).

In the following, a lower bound is also proposed for the B&B method. Consider π as a partial schedule in which the sequence of the first q orders is scheduled, and π^{us} as the unscheduled orders with $(n–q)= n_1$ orders. Assume that $\widehat{p}_{(q+1)}, \widehat{p}_{(q+2)}, \ldots, \widehat{p}_{(q+n_1)}$ denote the increasing sequence of $\sum_{k=1}^{m} p_{q+1\,k}, \sum_{k=1}^{m} p_{q+2\,k}, \ldots, \sum_{k=1}^{m} p_{q+n_1\,k}$. Then, the lower bound (LB) can be easily computed and given by

$$LB = \sum_{i=1}^{q} w_i U_i + w_{\min} \sum_{i=1}^{n_1} I_{\{C_{[q]}+\tilde{p}_{(q+i)}>d_{\max}\}},$$

where $I_{\{t>a\}} = 1$, $I_{\{t\le a\}} = 0$, $d_{\max} = \max_{j\in\pi^{US}}\{d_j\}$, and $w_{\min} = \min_{j\in\pi^{US}}\{w_j\}$.

3.2 Artificial bee colony algorithm

The artificial bee colony (ABC) algorithm has been widely applied to solve a lot of combinatorial problems (Abba Ari et al. (2016), Basturk and Karaboga (2006), Kang et al. (2015, 2016a, b), Tereshko (2000), Tereshko and Lee (2002), Tereshko and Loengarov (2005), Kang and Li (2016), Teodorovic (2003), Lucic and Teodorovic (2002), Teodorovic and Dell'orco (2005), Benatchba et al. (2005), Wedde et al. (2004), Yang (2005a, b), Karaboga (2005), Aydoğdu et al. (2016), Singh and Mann (2017), Xin et al. (2017), and Xue et al. (2018)). Inspired by these observations, we use intelligent behaviors of honeybee swarms to solve problem under study.

Following the design of Karaboga (2005), the three elements of the ABC cover employed bees, onlookers, and scouts. In ABC, two ways, including the multi-point insert method by Chaurasia et al. (2016), Singh (2009), Sundar and Singh (2012), and three-point exchange method by Liang et al. (2002) and Chaurasia et al. (2016), are commonly adopted to generate a new neighborhood solution. The main advantages of two methods are to avoid trapping in a local searching and to explore the search space. The pseudo-code of ABC and the neighborhood local search method are provided in the following.

The main codes of the ABC are given below.

Input: N_b, N_{on}, *ITRN*, *limit*, *p_sout*.
Initialization: generate N_b employed bees and N_{on} onlooker bees randomly
  Decode each employed bee by the ranking method
  Compute the food source of each employed bee
  Determine the acceptance probability p_bee_i of each employed bee i
  Retain the best food source among m employed bees
Do while {j<*ITRN*}
  Do while {i< N_b}
    Improve each employed bee i by a local search
    Retain the best food source among N_b employed bees
    Choose a food source for each onlooker bee k according the probability p_bee_i by roulette wheel selection and generate a candidate solution according the following formula
    $v_{ij} = em\_bee_{ij} + w \times (em\_bee_{ij} - onlooker\_bee_{kj})$, where ($k \neq$ i)
    w=(rnunf()-0.5)*2.0
    Update employed bee i by v_{ij}
    Decode this employed bee by the ranking method
    Compute its food source and determine if its food source is better than the best one, then update the best one by the current food source.
    Do l=1, *limit*
      Employed bee i improved by local search
      Retain the local best food source and the improved employed bee i
    End do
    Determine if its food source is better than the best one, and then update the best one by the current food source.
    Generate a random r with 0 < r <1
    if (r < *p_scout*) then
      regenerate employed bee i
      decode employed bee i
      Compute its food source and determine if its food source is better than the best one, then update the best one by the current food source.
    endif
  End while
End while

The details of the neighborhood local search method are given below.

Input a solution π_i (or input an employed bee) and a acceptance probability p
Generate a solution π_k randomly
Decode π_i and π_k
Determine if π_k is the same as π_i, and then regenerate another π_k until a new one occurs.
Generate a random number r with 0 < r <1
If $r < p$, then
  Perform multi-point insert scheme to generate π_w from π_i and π_k
Else
  Perform three point exchange scheme on π_i to generate π_w
Endif
Output π_w

Let $\text{em\_bee}_i = (x_1, x_2, \ldots, x_n)$ denote an employed bee with $x_i \in R$, $i = 1, 2, \ldots, n$. Consider an employed bee as a schedule of the n orders in this study. During the execution of ABC, a random number encoding method is used to define a set of continuous real numbers to represent the codes of the orders. For example, given $\text{em\_bee}_\text{i} = (0.83, 0.72, 0.34, 0.23, 0.51)$ as an employed bee, it is decoded as a sequence $\sigma = (5, 4, 2, 1, 3)$ by the ranking method. To show the impact of parameter settings on the ABC algorithm, four types of ABC algorithms were constructed, each of which was assigned a different parameter setting. Specifically, the parameter settings of the four ABC algorithms are given in Table 1. In addition, we also compare four proposed hybrid ABCs with

Table 1 Parameter settings of the four types of ABC algorithms

Algorithm	Parameter settings			
	N_b	N_{on}	ITRN	limit
ABC1	20	20	30	5
ABC2	10	10	30	5
ABC3	20	20	30	10
ABC4	10	10	30	10

their corresponding four basic ABC algorithms where the number of *limit* is set to 0. They are referred as B-ABC1, B-ABC2, B-ABC3, and B-ABC4. Due to the setting of Table 1, B-ABC1 and B-ABC3, and B-ABC2 and B-ABC4 are with the same number of employed bees and the same number of onlookers, but they adopt different levels of limit.

3.3 A branch-and-bound algorithm

The depth-first rule is utilized in the branching procedure. The advantage of this policy is that the computer only keeps no more than $(n-1)$ nodes for the lower bounds throughout the procedure. In the branch-and-bound method (B&B), the orders are first scheduled in a forward way and select a systematically search and branch down each tree. The steps are summarized in the following:

Step 1: Initialization	Perform eight ABCs and choose the best schedule as the current best solution
Step 2: Branching	Apply the depth-first rule
Step 3: Eliminating	Utilize all dominances to determine if the node is active or non-active. For the non-active nodes Compute their lower bound by the proposed lower-bound formulas Compute the weighted number of tardy orders for the full node
Step 4: Bounding	If the lower bound is larger than the initial solution, cut that node and all nodes beyond it in the branch, or if the weighted number of tardy orders for the completed node less than the initial solution, update it as a new current best solution
Step 5: Stopping rule	Repeat to do step 2, step 3, and step 4 until all nodes are visited

4 Computational experiment

In this section, we evaluate the performances of the eight ABCs and the B&B through some computational tests. All the algorithms were coded in a FORTRAN and run on a personal computer with an Intel(R) Core(TM)7 Quad CPU 2.66 GHz with 4 GB RAM. Following the design of Lee (2013) and Leung et al. (2006b, 2007b, 2008a, b), we generated the normal component processing times from Uniform(1, 100) for the order instances. In addition, we generated the order due dates from another Uniform($Pbar(1-\tau-R/2), P(1-\tau+R/2)$), where $Pbar = \sum_{k=1}^{m}\sum_{i=1}^{n} p_{ik}/m$, and τ and R describe the tardiness factor and the range of the due dates, respectively. Six cases of τ and R were examined at $(\tau, R) = (0.5, 0.75)$, $(0.5, 0.5)$, $(0.5, 0.25)$ $(0.25, 0.25)$, $(0.25, 0.5)$, and $(0.25, 0.75)$. Following the design of Reeves (1995), we generated the release times from Uniform(0, $100n\lambda$) where λ and n denote the control variable and order size. The values of λ were set at 0.2, 0.5, 0.8. In this paper, we design two parts of the computational experiment comprising the small-size order and the big-size order. (Note that a job is equivalent to an order.)

Table 2 Performances of B&B as the value of each parameter changes

n	m	Node	CPU time	FS
14	2	1,893,119	38.03	1794
	3	1,601,208	38.68	1796
	4	1,356,169	37.34	1798
16	2	8,703,212	181.81	1595
	3	7,914,658	195.51	1596
	4	9,159,211	203.48	1611
λ				
14	0.2	4,305,450	102.38	1788
	0.5	522,206	11.13	1800
	0.8	228,389	0.51	1800
16	0.2	18,887,477	493.56	1284
	0.5	5,989,023	141.60	1720
	0.8	441,603	10.08	1798
τ				
14	0.25	2,464,975	57.25	2688
	0.50	768,689	18.77	2700
16	0.25	8,095,795	197.89	2279
	0.50	9,103,761	232.27	2523
R				
14	0.25	1,287,846	30.01	1798
	0.50	1,216,725	29.08	1797
	0.75	2,345,925	54.95	1793
16	0.25	3,302,633	84.74	1637
	0.50	8,021,892	206.12	1679
	0.75	14,790,753	354.39	1486

4.1 Results of small number of orders

For the small-size orders, we set the number of order sizes at $n = 14$, 16, and the number of three machine sizes at $m = 2$,

Table 3 Performances of AEP of ABCs as the value of each parameter changes

n	m	ABC1	ABC2	ABC3	ABC4	B-ABC1	B-ABC2	B-ABC3	B-ABC4
14	2	0.59	0.74	0.54	0.66	1.36	1.74	1.36	1.74
	3	0.47	0.57	0.42	0.51	1.20	1.34	1.20	1.34
	4	0.54	0.62	0.47	0.57	1.19	1.37	1.19	1.37
16	2	0.88	1.06	0.85	0.96	1.72	2.01	1.72	2.01
	3	0.94	1.19	0.95	1.07	1.77	2.06	1.77	2.06
	4	0.82	1.00	0.76	0.81	1.48	1.62	1.48	1.62
λ									
14	0.2	1.29	1.53	1.15	1.38	2.89	3.46	2.89	3.46
	0.5	0.25	0.31	0.23	0.29	0.63	0.73	0.63	0.73
	0.8	0.07	0.09	0.05	0.08	0.23	0.27	0.23	0.27
16	0.2	2.09	2.57	2.04	2.22	3.78	4.32	3.78	4.32
	0.5	0.45	0.55	0.44	0.50	0.92	1.06	0.92	1.06
	0.8	0.10	0.13	0.08	0.11	0.27	0.31	0.27	0.31
τ									
14	0.25	0.88	1.05	0.78	0.95	2.04	2.43	2.04	2.43
	0.50	0.20	0.24	0.17	0.21	0.46	0.54	0.46	0.54
	0.25	1.44	1.80	1.41	1.55	2.74	3.14	2.74	3.14
16	0.50	0.32	0.37	0.29	0.34	0.58	0.65	0.58	0.65
R									
14	0.25	0.35	0.44	0.30	0.39	0.88	1.02	0.88	1.02
	0.50	0.49	0.61	0.43	0.53	1.20	1.37	1.20	1.37
	0.75	0.77	0.89	0.70	0.82	1.67	2.06	1.67	2.06
16	0.25	0.46	0.56	0.41	0.49	0.98	1.11	0.98	1.11
	0.50	0.87	1.01	0.80	0.93	1.64	1.87	1.64	1.87
	0.75	1.31	1.68	1.35	1.42	2.36	2.71	2.36	2.71

3, 4. We examined one hundred problem instances for each category. Therefore, there were a total of 10,800 instances tested in this experiment. The B&B algorithm would stop and turn to run next instance once node number was over 10^8.

To evaluate the performance of the B&B, we recorded the average and maximum number of nodes and the average and maximum execution times (in seconds). To evaluate the performance of eight versions of ABCs, we recorded the average error percentage (AEP) and maximum error percentage. The AEP is defined as

$$\text{AEP} = \left[(H_i - O^*)/O^*\right] \times 100\%,$$

where H_i is the objective obtained by using eight ABCs and O^* is the objective obtained using the B&B.

The performance of the B&B is given in Table 2, where the last column FS is the number of instances the B&B succeeded in achieving the optimal solution. The performances of the eight versions of ABCs are given in Table 3.

As given in Table 2, the number of machines had little impact on the performance of the B&B. Regarding the impact of parameter λ on the performance of the B&B, it appears

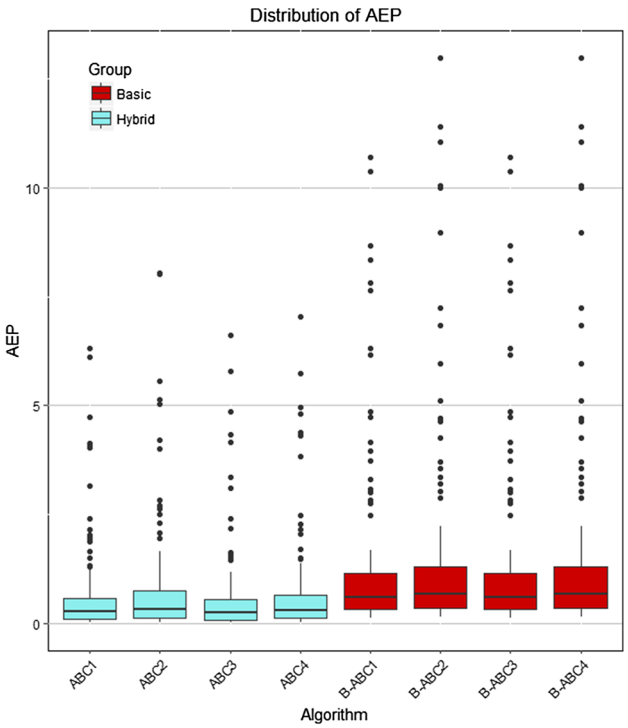

Fig. 1 Box plot of AEP for eight algorithms at $n = 14,16$

Table 4 ANOVA table for the small number of jobs

Source of variation	*df*	Sum of squares	Mean square	*F* value	$Pr > F$
Model	15	1856.5261	123.7684	68.2100	<0.0001
Algorithm	7	154.8160	22.1166	12.1900	<0.0001
Size	1	33.8278	33.8278	18.6400	<0.0001
Machine	2	5.4796	2.7398	1.5100	0.2215
λ	2	1046.4183	523.2092	288.3600	<0.0001
τ	1	485.6488	485.6488	267.6600	<0.0001
R	2	130.3356	65.1678	35.9200	<0.0001
Error	848	1538.6547	1.8145		
Corrected total	863	3395.1807			

Table 5 Results of Fisher's LSD for the four ABCs and four B-ABCs at $n = 14,16$

Pairwise comparison Between algorithms	Pairwise mean difference $\lvert\overline{\text{AEP}_i} - \overline{\text{AEP}_j}\rvert$	LSD ($\alpha = 0.05$) = 0.3598 Difference > LSD?
ABC1 versus ABC2	|0.7084–0.8642|	No
ABC1 versus ABC3	|0.7084–0.6647|	No
ABC1 versus ABC4	|0.7084–0.7623|	No
ABC1 versus B-ABC1	|0.7084–1.4539|	Yes
ABC1 versus B-ABC2	|0.7084–1.6919|	Yes
ABC1 versus B-ABC3	|0.7084–1.4539|	Yes
ABC1 versus B-ABC4	|0.7084–1.6919|	Yes
ABC2 versus ABC3	|0.8642–0.6647|	No
ABC2 versus ABC4	|0.8642–0.7623|	No
ABC2 versus B-ABC1	|0.8642–1.4539|	Yes
ABC2 versus B-ABC2	|0.8642–1.6919|	Yes
ABC2 versus B-ABC3	|0.8642–1.4539|	Yes
ABC2 versus B-ABC4	|0.8642–1.6919|	Yes
ABC3 versus ABC4	|0.6647–0.7623|	No
ABC3 versus B-ABC1	|0.6647–1.4539|	Yes
ABC3 versus B-ABC2	|0.6647–1.6919|	Yes
ABC3 versus B-ABC3	|0.6647–1.4539|	Yes
ABC3 versus B-ABC4	|0.6647–1.6919|	Yes
ABC4 versus B-ABC1	|0.7623–1.4539|	Yes
ABC4 versus B-ABC2	|0.7623–1.6919|	Yes
ABC4 versus B-ABC3	|0.7623–1.4539|	Yes
ABC4 versus B-ABC4	|0.7623–1.6919|	Yes
B-ABC1 versus B-ABC2	|1.4539–1.6919|	No
B-ABC1 versus B-ABC3	|1.4539–1.4539|	No
B-ABC1 versus B-ABC4	|1.4539–1.6919|	No
B-ABC2 versus B-ABC3	|1.6919–1.4539|	No
B-ABC2 versus B-ABC4	|1.6919–1.6919|	No
B-ABC3 versus B-ABC4	|1.4539–1.6919|	No

in Table 2 that the AEPs of CPU time and nodes tend to decrease for both $n = 14$ and $n = 16$ as the value of parameter λ increases, when the other parameters are fixed. For parameter τ, Table 2 shows that the AEPs of nodes and CPU time decrease for $n = 14$, while they increase for $n = 16$. There is no apparent pattern of impact of parameters τ and R on the performance of the B&B.

For the impact of the above parameters on the performance of the eight types of ABCs, Table 3 shows no apparent impact of the parameter m. The average AEP first decreases but then

Table 6 Performances of RPD of ABCs as the value of each parameter changes

n	m	ABC1	ABC2	ABC3	ABC4	B-ABC1	B-ABC2	B-ABC3	B-ABC4
40	2	0.06	0.09	0.04	0.07	0.22	0.26	0.22	0.26
	5	0.05	0.08	0.04	0.07	0.20	0.23	0.20	0.23
	10	0.05	0.08	0.03	0.07	0.19	0.22	0.19	0.22
80	2	0.03	0.05	0.02	0.04	0.12	0.14	0.12	0.14
	5	0.03	0.05	0.02	0.04	0.11	0.13	0.11	0.13
	10	0.03	0.05	0.02	0.04	0.11	0.13	0.11	0.13
λ									
40	0.2	0.09	0.14	0.06	0.11	0.32	0.38	0.32	0.38
	0.5	0.05	0.07	0.03	0.06	0.17	0.20	0.17	0.20
	0.8	0.03	0.05	0.02	0.04	0.11	0.13	0.11	0.13
80	0.2	0.05	0.08	0.04	0.07	0.19	0.22	0.19	0.22
	0.5	0.03	0.04	0.02	0.04	0.09	0.11	0.09	0.11
	0.8	0.02	0.03	0.01	0.02	0.06	0.07	0.06	0.07
τ									
40	0.25	0.07	0.11	0.05	0.09	0.26	0.30	0.26	0.30
	0.50	0.04	0.06	0.03	0.05	0.15	0.17	0.15	0.17
	0.25	0.04	0.06	0.03	0.05	0.14	0.17	0.14	0.17
80	0.50	0.03	0.04	0.02	0.03	0.08	0.10	0.08	0.10
R									
40	0.25	0.05	0.08	0.03	0.07	0.20	0.23	0.20	0.23
	0.50	0.05	0.08	0.04	0.07	0.20	0.24	0.20	0.24
	0.75	0.05	0.09	0.04	0.07	0.21	0.24	0.21	0.24
80	0.25	0.03	0.05	0.02	0.04	0.11	0.13	0.11	0.13
	0.50	0.03	0.05	0.02	0.04	0.11	0.13	0.11	0.13
	0.75	0.03	0.05	0.02	0.04	0.11	0.13	0.11	0.13

increases as m increases for instances of $n=14$; however, an opposite trajectory is observed for instances of $n=16$. As clearly given in Table 3, the average AEP of the four types of ABC algorithms tends to decrease as the value of λ or τ increases. On the contrary, the average AEP of the eight types of ABCs increases as the value of parameter R increases. Table 3 also confirms that the ABC3, on average, slightly outperformed the other algorithms.

As for the performance of eight ABCs, the box plots in Fig. 1 show the AEP distributions of all the proposed methods. To compare the solution quality among the four basic ABCs (B-ABC1, B-ABC2, B-ABC3, and B-ABC4) and the four hybrid ABCs (ABC1, ABC2, ABC3, and ABC4), analysis of variance (ANOVA) was conducted. The results are given in Table 4. With a p value of less than 0.0001, follow-up Fisher's least significant difference tests were subsequently conducted to find out the differences of solution quality among the eight algorithms. The reports are presented in Table 5. At the 0.05 level of significance, the AEPs were divided into two groups: the AEPs of ABC1, ABC2, ABC3, and ABC4 were in one group; the AEPs of the B-ABC1, B-ABC2, B-ABC3, and B-ABC4 were in the other group. The ABC3 algorithm with the smallest value of AEP is recommended for small number of orders.

4.2 Results of big number of orders

For the big number of jobs, we set the number of order sizes at $n=40$ and 80, and number of three machine sizes at $m=2$, 5, 10. One hundred instance problems were randomly examined for each case. Consequently, there were also a total of 10,800 problem instances examined in this experiment. To evaluate the performance of the eight versions of ABCs, we reported the mean and maximum RPD. The RPD is defined as

$$\text{RPD} = \left[(H_i - H^*)/H^*\right] \times 100\%,$$

where H_i is the objective obtained by using four ABCs and $H^*=\min\{\text{eight versions of ABC algorithms}\}$. The computational results are summarized in Table 4.

As given in Table 6, the RPDs of the four types of ABC algorithms strictly decrease as the values of parameters λ and τ increase for both $n=40$ and $n=80$. It appears in Table 6 that for both instances of $n=40$ and $n=80$, there are few changes

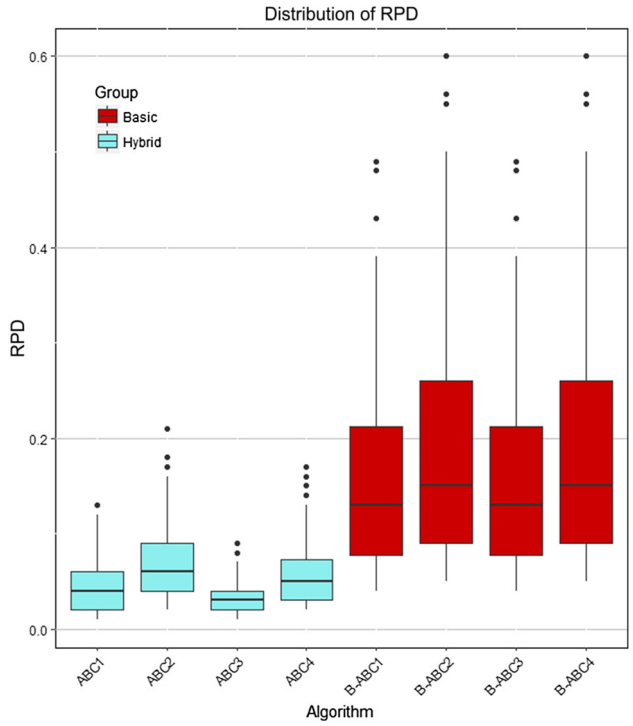

Fig. 2 Box plot of RPD for eight algorithms at $n=40, 80$

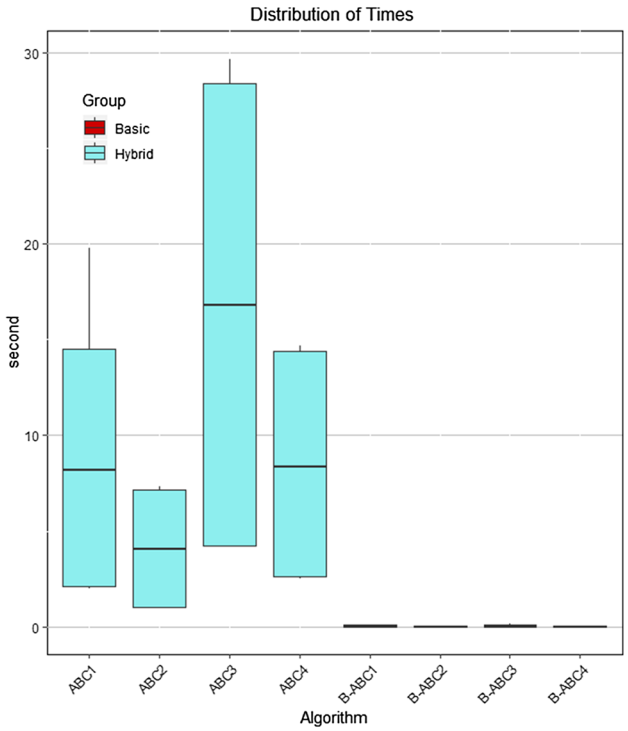

Fig. 3 Box plot of CPU time for eight algorithms at $n=40$ and 80

for the RPDs of the four types of ABC algorithms as the values of parameters R and m increase.

As demonstrated in Fig. 2, the box plots of RPDs show that the four hybrid ABCs (ABC1, ABC2, ABC3, and ABC4) outperformed the four basic ABCs (B-ABC1, B-ABC2, B-ABC3, and B-ABC4) for large numbers of orders. Meanwhile, as shown in Fig. 3, four hybrid ABCs did consume much CPU time than the four basic ABCs for large numbers of orders. An ANOVA was conducted to compare the solution quality among the four basic ABCs and the four hybrid ABCs. Table 7 shows the ANOVA test results. With a p value of less than 0.0001, follow-up Fisher's least significant difference tests were subsequently conducted to find out the differences of solution quality among the eight algorithms for large number of orders. The results are listed in Table 8. At the 0.05 significance level, the RPDs were divided into several groups: ABC3 in one group, ABC1 in the second group, ABC2 and ABC4 in the third group, B-ABC2 and B-ABC4 in the fourth group. Moreover, the box plots in Fig. 2 also show that the RPDs of the four ABC3 have less dispersion; thus, the ABC3 with both accuracy and stability is recommended for big number of orders.

Based on the above test results, we conclude that the ABC3, on average, outperformed the other seven ABCs.

5 Conclusions and suggestions

Recently, the OS models have become important and challenging topics in a lot of practical services and manufacturing environments. Different from previous research assumption that the orders are available at time zero, this paper investigates the condition that orders come in different release times. In this study, we made several main contributions. First, a lower bound and several dominance properties were derived to be used in a B&B for the small-size orders. Second, eight versions of ABCs were also presented to obtain approximate solutions for the big number of orders. Computational experiments were conducted to evaluate the performances of the B&B and eight versions of ABCs, as well as the impacts of parameters on their performances. The experimental results show that, incorporating with the proposed lower bound and dominance properties, the B&B can solve a problem instance up to $n=16$ within a reasonable CPU time. In addition, the experimental tests further illustrate that the eight types of ABC algorithms performed satisfactorily in terms of efficacy and efficiency for problem instances of both small and big number of orders. Based on the results of the four versions of ABC algorithms, ABC-3 is recommended for the problem under study because of its superior performance and its quickness in achieving good-quality solutions in term of the AEPs and RPDs. For some interesting issues, one may extend our model to the bi-criterion case or in different machine environments for the future study.

Table 7 ANOVA table for the big number of jobs

Source of variation	*df*	Sum of squares	Mean square	*F* value	$Pr > F$
Model	15	7.0539	0.4703	191.2000	<0.0001
Algorithm	7	3.3018	0.4717	191.7800	<0.0001
Size	1	0.7807	0.7807	317.4200	<0.0001
Machine	2	0.0208	0.0104	4.2300	0.0149
λ	2	2.2138	1.1069	450.0500	<0.0001
τ	1	0.7353	0.7353	298.9700	<0.0001
R	2	0.0015	0.0007	0.3000	0.7433
Error	848	2.0857	0.0025		
Corrected total	863	9.1396			

Table 8 Results of Fisher's LSD for the four ABCs and four B-ABCs at $n = 40, 80$

Pairwise comparison Between algorithms	Pairwise mean difference $\lvert\overline{\text{RPD}_i} - \overline{\text{RPD}_j}\rvert$	LSD ($\alpha = 0.05$) = 0.0013 Difference > LSD?
ABC1 versus ABC2	|0.0436–0.0687|	Yes
ABC1 versus ABC3	|0.0436–0.0030|	Yes
ABC1 versus ABC4	|0.0436–0.0558|	No
ABC1 versus B-ABC1	|0.0436–0.1559|	Yes
ABC1 versus B-ABC2	|0.0436–0.1839|	Yes
ABC1 versus B-ABC3	|0.0436–0.1559|	Yes
ABC1 versus B-ABC4	|0.0436–0.1839|	Yes
ABC2 versus ABC3	|0.0687–0.0030|	Yes
ABC2 versus ABC4	|0.0687–0.0558|	No
ABC2 versus B-ABC1	|0.0687–0.1559|	Yes
ABC2 versus B-ABC2	|0.0687–0.1839|	Yes
ABC2 versus B-ABC3	|0.0687–0.1559|	Yes
ABC2 versus B-ABC4	|0.0687–0.1839|	Yes
ABC3 versus ABC4	|0.0030–0.0558|	Yes
ABC3 versus B-ABC1	|0.0030–0.1559|	Yes
ABC3 versus B-ABC2	|0.0030–0.1839|	Yes
ABC3 versus B-ABC3	|0.0030–0.1559|	Yes
ABC3 versus B-ABC4	|0.0030–0.1839|	Yes
ABC4 versus B-ABC1	|0.0558–0.1559|	Yes
ABC4 versus B-ABC2	|0.0558–0.1839|	Yes
ABC4 versus B-ABC3	|0.0558–0.1559|	Yes
ABC4 versus B-ABC4	|0.0558–0.1839|	Yes
B-ABC1 versus B-ABC2	|0.1559–0.1839|	Yes
B-ABC1 versus B-ABC3	|0.1559–0.1559|	No
B-ABC1 versus B-ABC4	|0.1559–0.1839|	Yes
B-ABC2 versus B-ABC3	|0.1839–0.1559|	Yes
B-ABC2 versus B-ABC4	|0.1839–0.1839|	No
B-ABC3 versus B-ABC4	|0.1559–0.1839|	Yes

Acknowledgements Authors thank the Associate Editor and three anonymous referees for their helpful comments on the earlier versions of our paper. This paper was supported in part by the Ministry of Science Technology (MOST) of Taiwan under Grant Nos. MOST 103-2410-H-035-022-MY2 and MOST 105-2221-E-035-053-MY3. Authors thank Prof. Kunjung Lai of the Department of Statistics, Feng Chia University, for helping us to edit the English language.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

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

References

Abba Ari AA, Yenke BO, Labraoui N, Damakoa I, Gueroui A (2016) A power efficient cluster-based routing algorithm for wireless sensor networks: honeybees swarm intelligence based approach. J Netw Comput Appl 69:77–97

Ahmadi R, Bagchi U, Roemer TA (2005) Coordinated scheduling of customer orders for quick response. Naval Res Logist 52:493–512

Armentano VA, Ronconi DP (1999) Tabu search for total tardiness minimization in flowshop scheduling problems. Comput Oper Res 26:219–235

Aydoğdu I, Akın A, Saka MP (2016) Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution. Adv Eng Softw 92:1–14

Basturk B, Karaboga D (2006) An artificial bee colony (ABC) algorithm for numeric function optimization. In: IEEE swarm intelligence symposium 2006, May 12–14, Indianapolis, IN, USA

Benatchba K, Admane L, Koudıl M (2005) Using bees to solve data-mining problem expressed as a max-sat one, artificial intelligence and knowledge engineering applications: a bioinspired approach. In: First international work-conference on the interplay between natural and artificial computation, IWINAC 2005, Las Palmas, Canary Islands, Spain, June 15–18

Blocher JD, Chhajed D, Leung M (1998) Customer order scheduling in a general job shop environment. Decis Sci 29(4):951–981

Chaurasia SN, Sundar S, Singh A (2016) Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release date. Oper Res Int J 17(1):275–295

Cheng S-R, Yin Y, Wen C-H, Lin W-C, Wu C-C, Liu J (2017) A two-machine flowshop scheduling problem with precedence constraint on two jobs. Soft Comput 21(8):2091–2103

Erel E, Ghosh JB (2007) Customer order scheduling on a single machine with family setup times: complexity and algorithms. Appl Math Comput 185:11–18

Framinan JM, Perez-Gonzalez P (2017) New approximate algorithms for the customer order scheduling problem with total completion time objective. Comput Oper Res 78:181–192

French S (1982) Sequencing and scheduling: an introduction to the mathematics of the job shop. Ellis Horwood Limited, Chichester

Hsu SY, Liu CH (2009) Improving the delivery efficiency of the customer order scheduling problem in a job shop. Comput Ind Eng 57:856–866

Kang F, Li J (2016) Artificial bee colony algorithm optimized support vector regression for system reliability analysis of slopes. J Comput Civ Eng 30(3):04015040

Kang F, Salgado R, Li J (2015) System probabilistic stability analysis of soil slopes using Gaussian process regression with Latin hypercube sampling. Comput Geotech 63:13–25

Kang F, Xu Q, Li J (2016a) Slope reliability analysis using surrogate models via new support vector machines with swarm intelligence. Appl Math Model 40(11–12):6105–6120

Kang F, Li J-S, Li J (2016b) System reliability analysis of slopes using least squares support vector machines with particle swarm optimization. Neurocomputing 209:46–56

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department

Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW, Bohlinger JD (eds) Complexity of computer computations. Plenum, New York, pp 85–103

Lee I-S (2013) Minimizing total tardiness for the order scheduling problem. Int J Prod Econ 144:128–134

Leung JYT, Li H, Pinedo M (2005a) Order scheduling in an environment with dedicated resources in parallel. J Sched 8:355–386

Leung JYT, Li H, Pinedo M, Sriskandarajah C (2005b) Open shops with jobs overlap-revisited. Eur J Oper Res 163(2):569–571

Leung JYT, Li H, Pinedo M (2006a) Approximation algorithms for minimizing total weighted completion time of orders on identical machines in parallel. Naval Res Logist 53:243–260

Leung JYT, Li H, Pinedo M (2006b) Scheduling orders for multiple product types with due date related objectives. Eur J Oper Res 168:370–389

Leung JYT, Li H, Pinedo M (2007a) Scheduling orders for multiple product types to minimize total weighted completion time. Discrete Appl Math 155:945–970

Leung JYT, Li H, Pinedo M, Zhang J (2007b) Minimizing total weighted completion time when scheduling orders in a flexible environment with uniform machines. Inf Process Lett 103:119–129

Leung JYT, Li H, Pinedo M (2008a) Scheduling orders on either dedicated or flexible machines in parallel to minimize total weighted completion time. Ann Oper Res 159:107–123

Leung JYT, Lee CY, Ng CW, Young GH (2008b) Preemptive multiprocessor order scheduling to minimize total weighted flowtime. Eur J Oper Res 190:40–51

Liang H, Yao X, Newton C, Hoffman D (2002) A new evolutionary approach to cutting stock problems with and without contiguity. Comput Oper Res 29:1641–1659

Lin BMT, Kononov AV (2007) Customer order scheduling to minimize the number of late jobs. Eur J Oper Res 183:944–948

Lin WC, Yin Y, Cheng SR, Cheng TCE, Wu CH, Wu C-C (2017) Particle swarm optimization and opposite-based particle swarm optimization for two-agent multi-facility customer order scheduling with ready times. Appl Soft Comput 52:877–884

Lucic P, Teodorovic D (2002) Transportation modeling: an artificial life approach. In: ICTAI, pp 216–223

Monch L, Balasubramanian H, Fowler JW, Pfund ME (2005) Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. Comput Oper Res 32:2731–2750

Reeves C (1995) Heuristics for scheduling a single machine subject to unequal job release times. Eur J Oper Res 80:397–403

Singh A (2009) An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl Soft Comput 9:625–631

Singh S, Mann P (2017) Energy efficient clustering protocol based on improved metaheuristic in wireless sensor networks. J Netw Comput Appl 83:40–52

Sundar S, Singh A (2012) A swarm intelligence approach to the early/tardy scheduling problem. Swarm Evol Comput 4:25–32

Sung CS, Yoon SH (1998) Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines. Int J Prod Econ 54:247–255

Teodorovic D (2003) Transport modeling by multi-agent systems: a swarm intelligence approach. Transp Plan Technol 26–4(August):289–312

Teodorovic D, Dell'orco M (2005) Bee colony optimisation—a cooperative learning approach to complex transportation problems. In: 10th EWGT Meeting, Poznan, 13–16

Tereshko V (2000) Reaction–diffusion model of a honeybee colony's foraging behaviour. In: Schoenauer M et al (eds) Parallel problem solving from nature VI, Lecture Notes in Computer Science, vol 1917. Springer, Berlin, pp 807–816

Tereshko V, Lee T (2002) How information mapping patterns determine foraging behaviour of a honey bee colony. Open Syst Inf Dyn 9:181–193

Tereshko V, Loengarov A (2005) Collective decision-making in honey bee foraging dynamics. In: Computing and information systems, pp 1352–94049

Wagneur E, Sriskandarajah C (1993) Open shops with jobs overlap. Eur J Oper Res 71:366–378

Wang G, Cheng TCE (2007) Customer order scheduling to minimize total weighted completion time. Omega 35:623–626

Wang D-J, Kang C-C, Shiau Y-R, Wu C-C, Hsu P-H (2017a) A two-agent single-machine scheduling problem with late work criteria. Soft Comput 21:2015–2033

Wang DJ, Yin Y, Wu W-H, Wu W-H, Wu C-C, Hsu PH (2017b) A two-agent single-machine scheduling problem to minimize the total cost with release dates. Soft Comput 21:806–816

Wedde HF, Farooq M, Zhang Y (2004) BeeHive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior, ant colony, optimization and swarm intelligence. In: 4th international workshop, ANTS 2004, Brussels, Belgium, September 5–8

Wu C-C, Liu DS, Lin TY, Yang TH, Chung IH, Lin WC (2018) Bicriterion total flowtime and maximum tardiness minimization for an order scheduling problem. Comput Ind Eng 117:156–213

Xin Y, Wang Y-D, Xie ZQ, Yang J (2017) A cooperative scheduling method based on the device load feedback for multiple tasks scheduling. J Netw Comput Appl 99:110–119

Xu J, Wu C-C, Yin Y, Zhao CL, Chiou Y-T, Lin WC (2016a) An order scheduling problem with position-based learning effect. Comput Oper Res 74:175–186

Xu X, Zhao Y, Wu M, Zhou Z, Ma Y (2016b) Stochastic customer order scheduling to minimize long-run expected order cycle time. Ann Oper Res. https://doi.org/10.1007/s10479-016-2254-9

Xue Y, Jiang J, Zhao B, Ma T (2018) A self-adaptive artificial bee colony algorithm based on global best for global optimization. Soft Comput 22(9):2935–2952

Yang XS (2005a) Engineering optimizations via nature-inspired virtual bee algorithms, Lecture Notes in Computer Science, 3562. Springer, Berlin

Yang J (2005b) The complexity of customer order scheduling problems on parallel machines. Comput Oper Res 32:1921–1939

Yang J, Posner ME (2005) Scheduling parallel machines for the customer order problem. J Sched 8:49–74

Yoon SH, Sung CS (2005) Fixed pre-assembly scheduling on multiple fabrication machines. Int J Prod Econ 96:109–118

Zhao Y, Xu X, Li H, Liu Y (2016) Prioritized customer order scheduling to maximize throughput. Eur J Oper Res 255(2):345–356