

Journal of the Operational Research Society (2013) 64, 293–311 

www.palgrave-journals.com/jors/

# Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm

S-W Lin[1] and K-C Ying[2*]

[1]*Department of Information Management, Chang Gung University, Taoyuan, Taiwan, R.O.C; and*
[2]*Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei, Taiwan, R.O.C*

The order acceptance and scheduling (OAS) problem is an important topic for make-to-order production systems with limited production capacity and tight delivery requirements. This paper proposes a new algorithm based on Artificial Bee Colony (ABC) for solving the single machine OAS problem with release dates and sequence-dependent setup times. The performance of the proposed ABC-based algorithm was validated by a benchmark problem set of test instances with up to 100 orders. Experimental results showed that the proposed ABC-based algorithm outperformed three state-of-art metaheuristic-based algorithms from the literature. It is believed that this study successfully demonstrates a high-performance algorithm that can serve as a new benchmark approach for future research on the OAS problem addressed in this study.

*Journal of the Operational Research Society* (2013) **64,** 293–311. doi:10.1057/jors.2012.47;
published online 9 May 2012

**Keywords:** order acceptance and scheduling problem; single machine; artificial bee colony algorithm; sequence-dependent setup times

## 1. Introduction

Since the pioneering work of Guerrero and Kern (1988), the order acceptance and scheduling (OAS) problem has been recognized as an important topic and continuously attracted attention from researchers and practitioners. The OAS problem mainly focuses on two joint decisions: one is to decide which orders should be accepted for processing, and the other is to decide the processing sequence of the accepted orders (Slotnick, 2011). Therefore, a trade-off between the revenues and associated costs brought in by the accepting orders should be considered for making decisions (Guerrero and Kern, 1988). In practice, this is necessary for many make-to-order production systems (eg the printing and laminating company), with limited production capacity and tight order delivery deadline requirements (Herbots *et al*, 2007; Oğuz *et al*, 2010).

Over the past two decades, a diversity of OAS problems of different problem characteristics and objectives has been studied. For detailed discussion and taxonomy of the application models and available algorithms for various OAS problems, the reader is referred to the excellent literature surveys of Keskinocak and Tayur (2004) and Slotnick (2011). In this study, we focus on the OAS problem in a single machine environment. A number of exact methods have been applied to this problem, including (but not limited to) integer programming (Stern and Avivi, 1990), mixed-integer programming (Charnsirisakskul *et al*, 2004; Charnsirisakskul *et al*, 2006), mixed-integer linear programming (Yang and Geunes, 2003; Oğuz *et al*, 2010; Nobibon and Leus, 2011), dynamic programming (Lewis and Slotnick, 2002; Engels *et al*, 2003; Gordon and Strusevich, 2009) and branch-and-bound (Slotnick and Morton, 2007; Nobibon and Leus, 2011) algorithms. While relatively easy to conceptualize, the typical single machine OAS problem under most problem characteristics and performance criteria is *NP*-hard (Ghosh, 1997; Slotnick and Morton, 2007; Oğuz *et al*, 2010). For a problem of such complexity, the computational requirements for obtaining optimal solutions by exact methods are severe even for very moderately sized problems. Consequently, in practice, decision makers often seek approximation algorithms that generate near-optimum solutions for problems with a large number of orders, as is usual in industry, in a reasonable computational time.

Currently available approximation algorithms for solving single machine OAS problems can be classified into two categories: constructive heuristics (CHs) and improvement heuristics (IHs). For CHs such as those

*Correspondence: K-C Ying, Department of Industrial Engineering and Management, National Taipei University of Technology, No. 1, Sec. 3, Chung-Hsiao E. Road, Taipei 10608, Taiwan, R.O.C.

proposed by Stern and Avivi (1990), Engels *et al* (2003), Yang and Geunes (2003), Lee and Sung (2008), Oğuz *et al* (2010) and Cesaret *et al* (2012), the procedures add orders one at a time and inspect the effect of each addition on the objective function value. Once an order is accepted and sequenced, it is fixed and cannot be reversed. A common feature of these CHs is the non-robustness of their solutions. Although these CHs can yield solutions rapidly, there exists no CH that outperforms all other CHs given a variety of performance criteria and manufacturing environments. Further, the solution quality of CHs may not as be good as expected, especially for large scale problems (Slotnick, 2011).

In contrast, IHs such as those proposed by Akkan (1997), Lewis and Slotnick (2002), Charnsirisakskul *et al* (2006), Slotnick and Morton (2007), Yang and Geunes (2007) and Cesaret *et al* (2012) start with an initial solution and then provide a scheme for iteratively improving it to obtain a near-optimal solution. Recently, the formidable computational requirements of single machine OAS problems have resulted in numerous attempts to develop efficient IHs, leading to interest in metaheuristic-based algorithms. Examples of metaheuristic-based algorithms for single machine OAS problems include: Simulated Annealing (SA; Oğuz *et al*, 2010; Cesaret *et al*, 2012), Genetic Algorithm (GA; Chen *et al*, 2008; Rom and Slotnick, 2009), Extremal Optimization (EO; Chen *et al*, 2008), Tabu search (TS; Cesaret *et al*, 2012) and hybrid metaheuristic-based algorithms (Chen *et al*, 2008). The relevant literature reveals that some of these metaheuristic-based algorithms may provide excellent results for single machine OAS problems and constitute the state-of-the-art in available approaches (Cesaret *et al*, 2012).

The Artificial Bee Colony (ABC) is a novel swarm-based metaheuristic introduced by Karaboga (2005) for optimizing numerical problems. ABC-based algorithms have yielded satisfactory results on the numerical test function optimization problem (Karaboga and Akay, 2009), the data mining problem (Karaboga and Ozturk, 2009, 2011), the structural inverse analysis problem (Kang *et al*, 2009), the leaf-constrained minimum spanning tree problem (Singh, 2009), the assignment problem (Özbakir *et al*, 2010), the multi-objective design optimization of composites problem (Omkar *et al*, 2011), the visual target recognition problem (Xu and Duan, 2010) and the unrelated parallel machine scheduling problem (Ying and Lin, 2012). However, the performance of ABC-based algorithms typically depends on an efficient neighbourhood search mechanism to find the best solution within the neighbourhood. Without such a mechanism, ABC-based algorithms may become localized in a small area of the solution space, eliminating the possibility of finding a global optimum. The iterated greedy (IG) heuristic is an efficient local search (LS) method that generates a sequence of solutions by iterating greedy CHs through two main phases (Ruiz and Stützle, 2007): destruction and construction. In considering a more efficient solution to the OAS problem, this study proposes a new ABC-based algorithm that combines the destruction and construction phases of the IG algorithm in order to generate a solution that maximizes the total net revenue (*TNR*) for a single machine OAS problem with release dates and sequence-dependent setup times. To the best of our knowledge, this study is the first to use ABC-based algorithms for this purpose.

The remainder of this paper is organized as follows. After this brief introduction, the problem is formulated in Section 2. In Section 3, the proposed ABC-based algorithm is described in detail. Using an existing benchmark problem set, the effectiveness and efficiency of the proposed ABC-based algorithm are evaluated and its performance is compared to state-of-the-art metaheuristic-based algorithms from the relevant literature in Section 4. Finally, in Section 5, this study concludes with recommendations for future studies.

## 2. Problem definition

The single machine OAS problem considered in this study can be defined formally as follows. There exists a collection (or stream) of $n$ incoming orders to be processed on a single machine, where limited production capacity and order delivery requirements necessitate selective acceptance of the orders. Each incoming order $i$ ($i=1,2,\ldots,n$) is identified with the following non-negative integer data: a processing time $p_i$; a release date $r_i$, after which the order becomes available for processing; a preferred due date $d_i$, after which a tardiness penalty is incurred; a strict deadline $\bar{d}_i$ ($\bar{d}_i > d_i$), after which the customer will not take the order and no revenue can be gained from the order; a maximum revenue $Q_i$, gained by the manufacturer if order $i$ is accepted and its tardiness is zero; a weight $w_i = Q_i/(\bar{d}_i - d_i)$, which denotes the penalty per unit-time delay beyond $d_i$ in the delivery to the customer; and a sequence-dependent setup time $s_{ji}(i \neq j)$ incurred when order $i$ is processed immediately after order $j$ in the processing sequence. Since tardiness penalties cause loss of revenue, OAS decisions must be taken jointly.

Given the above definitions, the objective is to find a processing sequence of all accepted orders on the single machine that maximizes the *TNR* (revenues of accepted orders minus total weighted tardiness penalties), which can be formulated as:

$$TNR = \max \sum_{i=1}^{n} x_i [Q_i - w_i T_i]$$

where $x_i$ is an indicator that equals to 1 if order $i$ is accepted, and 0 otherwise, and $T_i = \max\{0, C_i - d_i\}$ denotes the tardiness of order $i$, in which $C_i$ is the completion time

of order $i$. Moreover, the following assumptions are invoked for the problem considered in this study:

- The number of orders, their processing times, release dates, due dates, deadlines, maximum revenue and sequence-dependent setup times are known in advance.
- At any time, the machine can process at most one order and is persistently available to process all scheduled orders as required.
- The schedule is non-pre-emptive, meaning once an order starts to be processed on the machine, the process cannot be interrupted before its completion.
- There are no precedence constraints among all orders.

Notably, the single machine OAS problem with sequence-dependent setup times as described above is strongly NP-hard (Oğuz *et al*, 2010). If all orders must be accepted, the problem reduces to the single machine scheduling problem with sequence-dependent setup times. On the other hand, if no scheduling is required, the problem is analogous to the knapsack problem.

## 3. The proposed ABC-based algorithm

The ABC was inspired by the intelligent foraging behaviour of a honeybee swarm. The algorithm posits foraging artificial bees that are grouped into three clusters: employed bees, onlookers and scouts. A bee that is currently exploiting a food source is called an employed bee. A bee that is waiting in the hive to make a decision regarding a food source is called an onlooker. A bee that is performing a random search for a new food source is called a scout. An employed bee that abandons a food source becomes a scout. When employed bees and onlookers perform the exploitation process in the search space, the scouts control the exploration process. In the ABC, each solution to the problem of interest is treated as a food source and represented by an $n$-dimensional real-valued vector, and the fitness of the solution is represented by the amount of nectar in the associated food resource. The number of employed bees and onlookers is set equal to the number of food sources in the population. In other words, for each food source, only one bee is employed. Like other swarm intelligence-based approaches, the ABC is iterative. It begins with a population of randomly generated food sources (solutions). A typical iteration of ABC proceeds as follows: one employed bee is placed on each food source, the number of onlookers placed on each food source depends on the amount of nectar it has, and scouts are sent to the search area to discover new food sources. Finally, the best food source located up to the present iteration is recorded to memory. This process is repeated until the termination condition is satisfied.

The following subsection gives a detailed discussion of the proposed ABC-based algorithm for solving the single machine OAS problem addressed in this study. First, the solution representation (encoding scheme) that makes a solution recognizable to the proposed algorithm and the initial solution generation approach is presented. Then, the fitness value, the neighbourhood solution, the LS approach, the employed bee phase, onlooker phase and scout phase are explained. Finally, the parameters and the procedures of the proposed ABC-based algorithm are described.

### *3.1. Representation of solution and initial solutions*

Unlike Cesaret *et al* (2012), this study represents a solution $\pi$ by a string of numbers $(\pi_1, \pi_2, \ldots, \pi_n)$ consisting of a permutation list of the preference sequence of the $n$ orders, which will be accepted for processing if its deadline constraints are not violated. For example, a solution 2-1-4 can be decoded from the permutation list of the preference sequence, $(2, 1, 5, 3, 4)$, if orders 5 and 3 are rejected due to a violation of their deadline constraints. This new representation of a solution is easier than that of Cesaret *et al* (2012) for generating neighbourhood solutions.

It should be noted that, when checking the deadline constraints, the release time of orders should be considered. That is, the current order must be waiting if its release time is later than the completion time of the previous order. It is clear that this solution representation always corresponds to a feasible solution that does not violate the deadline constraints of orders. Further, the solutions in the initial population are obtained from preference sequences that are generated by randomly sequencing the incoming orders.

### *3.2. Fitness value*

The quality of a solution is determined by the value of its fitness function. The fitness function of a solution is defined as $\text{fit}(\pi) = TNR(\pi)$, where $TNR(\pi)$ represents the $TNR$ of solution $\pi$. This equation reveals that a higher total revenue corresponds to a higher fitness value.

### *3.3. Neighbourhood solutions*

In this study, the employed bees generate food sources (solutions) in the neighbourhood close to their current positions. This process is structured by a permutation-based representation. On the basis of the permutation-based neighbourhood structure, the destruction and construction phases of the IG algorithm (Ruiz and Stützle, 2007) are applied to generate neighbourhood solutions of the current solution just as Ying and Lin (2012) implemented. The procedure for generating a neighbourhood solution is defined as $IG(\pi, \alpha)$ herein, where $\pi$ is the current solution and $\alpha$ denotes the number of removed

orders in the destruction phase. The two steps of $IG(\pi, \alpha)$, that is, the destruction and construction phases, are described briefly below.

*Step* 1 (*Destruction phase*): Randomly sample $\alpha$ orders from $\pi$ without replacement. Delete them from $\pi$ and add them to $\pi_R$ in the order in which they are chosen, where $\pi_R$ is the sequence of the $\alpha$ removed orders.

*Step* 2 (*Construction phase*): Sequentially reinsert the orders of $\pi_R$ into $\pi_D$ until a complete neighbourhood solution is obtained, where $\pi_D$ is the partial sequence of $\pi$ after removing the $\alpha$ sampled orders. During the construction procedure, the best (partial) solution, $\pi_{new}$, obtained by inserting the orders of $\pi_R$ into all possible positions of the current subsequence, is recorded.

To clearly demonstrate the destruction and construction phases, we give an example as follows: Suppose there are 15 orders to be scheduled and the current solution is shown in Figure 1. (Detailed data of this example can be found in Dataslack_15orders_Tao3R9_9.txt, for the benchmark problem set generated by Cesaret *et al* (2012).) In the destruction phase, orders 5, 9 and 8 are randomly sampled. Delete them from the current solution and add them to $\pi_R$ in the sequence in which they were chosen; we then obtain $\pi_D = (7, 13, 14, 4, 15, 10, 3, 1, 2, 11, 6, 12)$ and $\pi_R = (5, 9, 8)$. Subsequently, in the construction phase, the orders 5, 9 and 8 are sequentially reinserted into all possible positions of the current subsequence, and the best (partial) solution is recorded. Finally, a complete neighbourhood solution $\pi_{new} = (7, 13, 14, 5, 4, 15, 8, 10, 3, 1, 2, 11, 6, 9, 12)$ is obtained.

### 3.4. LS approach

In this study, an LS approach outlined in Figure 2 was applied to improve $\pi_{new}$. As shown in Figure 2, the LS approach improves $\pi_{new}$ by iteratively exchanging orders in the solution with orders positioned after them.

### 3.5. Employed bee phase

If the solution ($\pi^i$) corresponding to an employed bee $x_i$ ($\forall i$) equals $\pi_{best}$, a new solution $\pi^i_{new}$ for this employed bee is generated from the neighbourhood solutions described in Section 3.3. Otherwise, a new solution $\pi^i_{new}$ for this employed bee is generated by the following steps of order crossover (Davis, 1985; Gen and Cheng, 1996):

*Step* 1: Select a substring from $\pi_{best}$ at random.

*Step* 2: Produce a proto-child by copying the substring into its corresponding positions.

*Step* 3: Delete the orders that are already in the substring from the second parent. The resulting sequence of orders contains the orders that the proto-child needs.

*Step* 4: Place the orders into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

If $\pi^k_{new}$ is better than the best solution ($\pi^*$) found so far by the proposed ABC-based algorithm, or the relative difference of fitness function values between $\pi^k_{new}$ and $\pi^*$, that is $[fit(\pi^*) - fit(\pi^k_{new})]/fit(\pi^*)$, is smaller than a predetermined value (*threshold*), the LS approach is applied to improve $\pi^k_{new}$.

A new neighbourhood solution $\pi^k_{new}$ is always accepted if it is better than $\pi^k$. Therefore, once $\pi^k_{new}$ is obtained, it is compared to $\pi^k$. If $fit(\pi^k_{new}) \geq fit(\pi^k)$, then $\pi^k$ is replaced by $\pi^k_{new}$ and $\pi^k_{new}$ becomes a new member of the population; otherwise $\pi^k$ is retained.

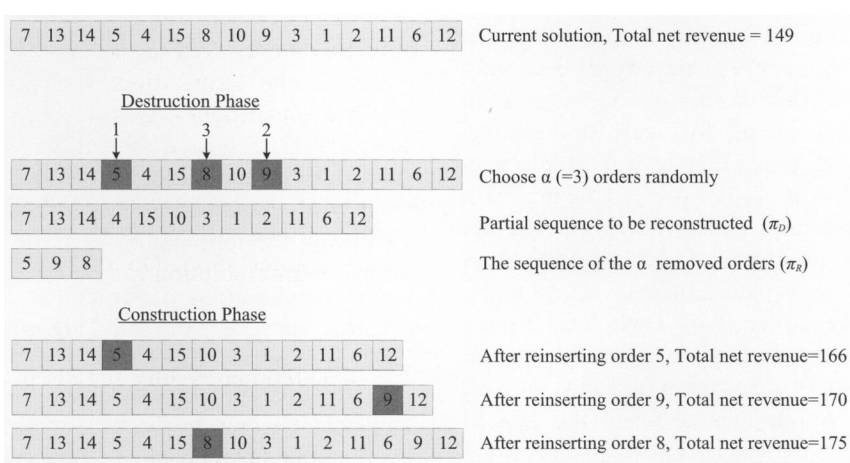


**Figure 1** An example of the destruction and construction phases for generating a neighbourhood solution.

```
Procedure Local_Search (π_new)
Begin
  for (i = 1; i ≤ N − 1; i++)
    {
      for (j = i + 1; j ≤ N; j++)
      {
        π_temp = solution obtained by exchange the i^th and j^th orders of π_new;
        If (TNR(π_temp) is better than TNR(π_new))
        {
          π_new = π_temp;
        }
      }
    }
  return π_new;
End
```

**Figure 2** Local search procedure.

### 3.6. Onlooker phase

An onlooker evaluates the nectar information taken from all the employed bees and selects a solution (food source) $\pi^k$ with a probability $p_k$, given by $p_k = f_k / \sum_{k=1}^{SN} f_k$, where $f_k$ is the amount of nectar (fitness value) of $\pi^k$, and $SN$ is the number of food sources. Clearly, a higher $f_k$ corresponds to a higher probability that $\pi^k$ is selected. Once a solution $\pi^k$ is selected by an onlooker, a new solution ($\pi^k_{new}$) is generated using the neighbouring procedure described in Section 3.3. The LS approach as applied in the employed bee phase may then be implemented on $\pi^k_{new}$. If $\pi^k_{new}$ obtained from the LS procedure has at least as much nectar as $\pi^k$, then $\pi^k_{new}$ will replace $\pi^k$ and become a new member of the population.

### 3.7. Scout phase

If $\pi^k$ cannot be further improved in a predetermined number of trials, *limit*, then it is abandoned, and the corresponding employed bee becomes a scout. The scout generates a new food source randomly.

### 3.8. Parameters

The proposed ABC-based algorithm starts with six parameters: *SN*, *threshold*, *limit*, $\alpha$, $G_{\max}$ and $G_{non-improving}$. *SN* represents the number of food sources, which equals the number of the employed bees and onlookers. That is, the colony size is $2 \times SN$. *limit* is the number of trials without improvement after which a food source is abandoned. $\alpha$ denotes the number of removed orders in the destruction phase of IG ($\pi, \alpha$), and *threshold* is used to determine whether the LS approach is applied after the neighbourhood solution has been generated. $G_{\max}$ is the maximum number of iterations and the $G_{non-improving}$ is the allowable number of generations for which the best obtained objective function value is not improved by the proposed ABC-based algorithm.

### 3.9. Procedures of the proposed ABC-based algorithm

Figure 3 presents the pseudocode of the proposed ABC-based algorithm. In the first step, the initial population $\text{Pop} = (\pi^1, \pi^2, \ldots, \pi^{SN})$ is generated randomly. Subsequently, the fitness value is calculated for each solution in the initial population. In the following main loop, each iteration involves procedures in the following sequence. First, the employed bees are placed on their food sources. Second, the onlookers are placed on the food sources in a manner that depends on the amounts of nectar of each source. Third, the scouts are sent to the search area to discover new food sources. Fourth, the best food source found so far is saved to memory ($\pi^*$). At the end of each iteration, the termination condition is checked. If the number of iterations $G$ exceeds $G_{\max}$ or $\pi^*$ is not improved upon in $G_{non-improving}$ successive generations, then the procedure is terminated; otherwise, a new iteration will begin and the iteration number $G$ is increased by one. Following the termination of the procedures, the (near-) global optimal solution can be derived from $\pi^*$.

## 4. Computational results and discussion

Recently, Cesaret *et al* (2012) presented three algorithms for the same OAS problem addressed in this study, that is, *m-ATCS*, *ISFAN* and *TS*, where *m-ATCS* is a modified

```
ABC (SN, threshold, limit, α, G_max and G_non-improving)
{
  Generate initial population Pop = (π^1, π^2, ..., π^SN) randomly and improve the initial food sources
    (solutions) by the local search approach;
  G=0; G_non=0;
  Calculate the fitness of each food source fit(π^k), k ∈ Pop;
  Do ( G ≤ G_max and G_non ≤ G_non-improving )
  {
    G=G+1; G_non= G_non +1;
    //Place the employed bees on their food sources;
    For k = 1, 2, ..., SN
    {
      Obtain new solution π^i_new by order crossover or IG( π^k , α );
      If [fit(π*) − fit(π^k_new)]/ fit(π*) < threshold or fit(π^k_new) ≥ fit(π*) , then perform LS( π^k_new );
      If fit(π^k_new) ≥ fit(π^k) , then replace π^k by π^k_new ;
    }
    //Place the onlooker bees on the food sources depending on their nectar amounts
    For k = 1, 2, ..., SN
    {
      Selects a food source π^k depending on its probability value p_k calculated by
        p_k = f_k / Σ_{k=1}^{SN} f_k ;
      Obtain new solution π^k_new by IG( π^k , α );
      If [fit(π*) − fit(π^k_new)]/ fit(π*) < threshold or fit(π^k_new) ≥ fit(π*) , then perform LS( π^k_new );
      If fit(π^k_new) > fit(π^k) , then replace π^k by π^k_new ;
    }
    // Send the scout to the search area for discovering a new food source;
    For k = 1, 2, ..., SN
    {
      If (food source π^k is not changed in successive limit times), then regenerate π^k randomly;
    }
    // Memorize the best food source found so far
    For k = 1, 2, ..., SN
    {
      If fit(π^k_new) ≥ fit(π*)
      {
        π* = π^k ;
        G_non = 0;
      }
    }
  }
  Output the best food source found so far ( π* );
}
```

**Figure 3** The pseudocode of the proposed ABC approach.

apparent tardiness cost rule-based approach, ISFAN is an SA-based algorithm, and *TS* is a TS-based algorithm. The analytical results of Cesaret *et al* (2012) show that *TS* outperforms *m-ATCS* and *ISFAN*, and is the state-of-art algorithm for this problem. Cesaret *et al* (2012) also presented computational results found by a mixed integer linear programming formulation (MILP) given in (Oğuz *et al*, 2010). They reported the best feasible solution found by the CPLEX solver within the 3600 second time limit as the *MILP* solution.

To compare the performance of the proposed ABC-based algorithm with these leading algorithms on the same basis, the proposed ABC-based algorithm was coded in *C* computer language and run on a PC with an Intel Pentium IV (3.0 GHz) CPU and 4 GB of RAM. The test problems and the computational results of the proposed ABC-based algorithm against these best-so-far algorithms are discussed in the following subsections.

### *4.1. Test problems*

To evaluate the performance of the proposed ABC-based algorithm, computational simulations and comparisons

were executed based on 1500 instances from the benchmark problem set of Cesaret *et al* (2012). The same benchmark instances were also used by the *MILP*, *m-ATCS*, *ISFAN* and *TS*. The instances of this benchmark problem set consist of the following three factors: number of orders ($n$), due date range ($R$) and tardiness factor ($\tau$). The number of orders was set to 10, 15, 20, 25, 50 and 100; both the due date range and tardiness factor were set to 0.1, 0.3, 0.5, 0.7 and 0.9. For each combination of the three factors, 10 instances were randomly generated. Therefore, the test bed was comprised of $6 \times 5 \times 5 \times 10 = 1500$ benchmark instances.

These instances were randomly generated as follows (Cesaret *et al*, 2012): for each order $i$ ($i = 1, 2, \ldots, n$), a processing time $p_i$, a maximum revenue $Q_i$, and a release date $r_i$ were generated from the uniform distribution [0, 20], [0, 20] and $[0, \tau P_T]$, respectively, where $P_T$ is the total processing time of all orders. The sequence-dependent setup times $s_{ji}$ ($j = 1, 2, \ldots, n, j \neq i$) of each order $i$ ($i = 1, 2, \ldots, n$) were generated from the uniform distribution [1, 10]. The preferred due date $d_i$ of each order $i$ ($i = 1, 2, \ldots, n$) was generated as $d_i = r_i + \max_{j=0,1,\ldots,n} s_{ji} + \max\{slack, p_i\}$, where *slack*was drawn from a discrete uniform distribution in the interval $[P_T(1-\tau-R/2), P_T(1-\tau+R/2)]$. The strict deadline $\bar{d}_i$ of each order $i$ ($i = 1, 2, \ldots, n$) was generated from the formula $\bar{d}_i = d_i + Rp_i$. All of the above generated data are integers. The files for these test instances are available via the website (http://home.ku.edu.tr/~coguz/Research/Dataset_OAS.zip).

### 4.2. Results and discussion

Parameter selection can affect the quality of results. In this study, the optimized parameter values were chosen taking into consideration both solution quality and computational efficiency. To determine the appropriate values of experimental parameters, an extensive computational testing was performed. In the preliminary optimization stage, the following combinations of the parameter values were tested on 20 randomly selected instances: $SN = 4, 6, 8, 10, 12$; $threshold = 0.005, 0.01, 0.015, 0.02$; $\alpha = 3, 4, 5, 6$; $limit = 100, 200$; $G_{\max} = 600, 900, 1200$; $G_{non-improving} = 100, 200, 300$. Finally, the following parameter values seem to offer the best performance within a reasonable computational time: $SN = 6$, $threshold = 0.01$, $\alpha = 3$, $limit = 100$, $G_{\max} = 900$, $G_{non-improving} = 200$. Therefore, these parameter values were adopted for all subsequent experiments in this study.

To validate and verify the performance of the proposed ABC-based algorithm, the *TNR* obtained for each of the 1500 test instances was recorded and compared with the solution obtained by the *MILP*, *m-ATCS*, *ISFAN* and *TS*, respectively, using the percentage deviation of the *TNR* from an upper bound (UB) (% Deviation from *UB*) as follows.

$$\%\ Deviation\ from\ UB = \frac{(UB - TNR^h)}{UB} \times 100\%$$

where $TNR^h$ denotes the *TNR* obtained by algorithm $h$, and *UB* is the corresponding *UB* proposed by Cesaret *et al* (2012).

In this section, the computational results of the maximum (*Max.*), average (*Ave.*) and minimum (*Min.*) *% Deviation from UB* of each subset obtained by the *MILP*, *m-ATCS*, *ISFAN*, *TS* and *ABC* were summarized and discussed. The average run time (CPU time in seconds) and the number of optimal solutions out of 10 instances of each subset were also reported. Because the *m-ATCS* takes less than a second, it was not shown in the tables. Since Cesaret *et al* (2012) did not point out how many times the *MILP*, *m-ATCS*, *ISFAN* and *TS* were executed for each instance, in this study each instance was solved by the proposed approach once to ensue that the comparisons were fairly made. The computational results of the *MILP*, *m-ATCS*, *ISFAN* and *TS* cited from the literature (Cesaret *et al*, 2012) were run on a workstation with a 3.0 GHz Intel Xeon processor and 4 GB of RAM.

Due to the computational difficulty of the problem, *MILP* can be solved to optimality only for $n = 10$ and some cases of $n = 15$, 20 and 25. For problem set $n = 10$, the solution obtained by the proposed ABC-based algorithm is almost the same as the optimal solution. As shown in Table 1, the proposed ABC-based algorithm obtained 247 optimal solutions out of 250 test instances, while the *MILP*, *m-ATCS*, *ISFAN* and *TS* obtained 249, 19, 93 and 188 optimal solutions, respectively. Even though the proposed ABC-based algorithm obtained three near-optimal solutions for problem set $n = 10$, the maximal *% Deviation from UB* of these solutions is only 6%. Thus, their TNR is uniformly very close to optimal values. The computational time required by the proposed ABC-based algorithm is less than 0.03 s for all instances of $n = 10$, while MILP requires about 227 s on the average. Since MILP is deterministic and the proposed ABC-based algorithm is non-deterministic, there is a trade-off between solution quality and computing time. When the order number to be scheduled is less than 10, the practitioner can use MILP to get an optimal solution. When the order number is bigger than 10, the trade-off clearly favours the proposed ABC-based algorithm, which can reach a near-optimal solution in a short time.

Table 2 shows that the ABC-based algorithm is competitive with *MILP* and *TS* algorithm for the problem set $n = 15$ by finding 101 optimal solutions out of 250 test instances with 3% *Deviation from UB* on the average. The *TS* algorithm and *MILP* found the optimal solutions in 91 and 102 out of 250 test instances, and give 3% and 6% *Deviation from UB* on the average. When $\tau$ equals 0.1, 0.3 and 0.5, the *MILP* found the optimal solutions in only 3

**Table 1** Performance comparisons for problem set $n = 10$

| $\tau$ | $R$ | *Deviation from UB* | | | | | | | | | | | | | | | *Number of optimal solutions* | | | | | *Average run times (s)* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *MILP* | | | *m-ATCS* | | | *ISFAN* | | | *TS* | | | *ABC* | | | *MILP* | *m-ATCS* | *ISFAN* | *TS* | *ABC* | *MILP* | *ISFAN* | *TS* | *ABC* |
| | | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | | | | | | | | | |
| 0.1 | 0.1 | 0 | 0 | 0 | 25 | 9 | 0 | 10 | 3 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 10 | 1 | 6 | 6 | 10 | 1123.90 | 22.93 | 0.00 | 0.02 |
| | 0.3 | 0 | 0 | 0 | 19 | 9 | 2 | 9 | 3 | 0 | 3 | 1 | 0 | 2 | 0 | 0 | 10 | 0 | 3 | 5 | 9 | 1119.90 | 21.32 | 0.00 | 0.02 |
| | 0.5 | 6 | 1 | 0 | 27 | 9 | 0 | 8 | 3 | 0 | 8 | 1 | 0 | 6 | 1 | 0 | 9 | 1 | 4 | 8 | 9 | 1037.20 | 16.35 | 0.00 | 0.02 |
| | 0.7 | 0 | 0 | 0 | 23 | 10 | 0 | 10 | 3 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 6 | 8 | 10 | 435.00 | 12.69 | 0.01 | 0.03 |
| | 0.9 | 0 | 0 | 0 | 21 | 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 9 | 7 | 10 | 289.63 | 10.63 | 0.00 | 0.02 |
| 0.3 | 0.1 | 0 | 0 | 0 | 19 | 10 | 0 | 12 | 4 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 2 | 9 | 10 | 466.20 | 28.90 | 0.00 | 0.02 |
| | 0.3 | 0 | 0 | 0 | 33 | 14 | 0 | 10 | 2 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 10 | 1 | 6 | 6 | 10 | 372.40 | 28.12 | 0.00 | 0.02 |
| | 0.5 | 0 | 0 | 0 | 40 | 16 | 0 | 19 | 7 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 10 | 1 | 2 | 6 | 9 | 257.30 | 24.24 | 0.01 | 0.03 |
| | 0.7 | 0 | 0 | 0 | 36 | 12 | 1 | 16 | 6 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 10 | 0 | 1 | 6 | 10 | 310.30 | 19.71 | 0.00 | 0.02 |
| | 0.9 | 0 | 0 | 0 | 42 | 21 | 5 | 29 | 6 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 5 | 4 | 10 | 118.20 | 16.84 | 0.00 | 0.03 |
| 0.5 | 0.1 | 0 | 0 | 0 | 24 | 14 | 4 | 20 | 9 | 0 | 6 | 1 | 0 | 0 | 0 | 0 | 10 | 0 | 3 | 9 | 10 | 22.10 | 33.33 | 0.00 | 0.03 |
| | 0.3 | 0 | 0 | 0 | 34 | 14 | 0 | 20 | 5 | 0 | 5 | 1 | 0 | 0 | 0 | 0 | 10 | 0 | 4 | 8 | 10 | 52.60 | 32.48 | 0.00 | 0.02 |
| | 0.5 | 0 | 0 | 0 | 23 | 15 | 3 | 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 4 | 9 | 10 | 13.90 | 30.99 | 0.00 | 0.02 |
| | 0.7 | 0 | 0 | 0 | 40 | 16 | 1 | 32 | 6 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 4 | 4 | 10 | 24.30 | 27.27 | 0.00 | 0.03 |
| | 0.9 | 0 | 0 | 0 | 35 | 15 | 1 | 28 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 3 | 7 | 10 | 26.70 | 23.85 | 0.00 | 0.03 |
| 0.7 | 0.1 | 0 | 0 | 0 | 34 | 14 | 0 | 23 | 6 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 4 | 9 | 10 | 1.30 | 42.10 | 0.00 | 0.03 |
| | 0.3 | 0 | 0 | 0 | 27 | 14 | 0 | 23 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 2 | 5 | 10 | 10 | 1.50 | 42.67 | 0.00 | 0.02 |
| | 0.5 | 0 | 0 | 0 | 49 | 24 | 4 | 12 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 3 | 9 | 10 | 1.60 | 37.37 | 0.00 | 0.03 |
| | 0.7 | 1 | 0 | 0 | 26 | 13 | 0 | 19 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 1 | 8 | 10 | 1.70 | 38.71 | 0.00 | 0.03 |
| | 0.9 | 0 | 0 | 0 | 36 | 20 | 9 | 19 | 6 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 1 | 6 | 10 | 1.90 | 28.31 | 0.00 | 0.03 |
| 0.9 | 0.1 | 0 | 0 | 0 | 17 | 8 | 0 | 11 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 3 | 7 | 10 | 10 | 1.00 | 59.88 | 0.00 | 0.03 |
| | 0.3 | 0 | 0 | 0 | 29 | 11 | 0 | 18 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 2 | 2 | 9 | 10 | 1.00 | 54.97 | 0.00 | 0.03 |
| | 0.5 | 0 | 0 | 0 | 22 | 9 | 0 | 27 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 3 | 5 | 8 | 10 | 1.00 | 46.37 | 0.00 | 0.03 |
| | 0.7 | 0 | 0 | 0 | 34 | 15 | 6 | 39 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 1 | 9 | 10 | 1.00 | 45.53 | 0.00 | 0.00 |
| | 0.9 | 0 | 0 | 0 | 40 | 19 | 2 | 27 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 2 | 8 | 10 | 1.00 | 43.45 | 0.00 | 0.03 |
| Total average | | 0 | 0 | 0 | 30 | 14 | 2 | 18 | 5 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 9.96 | 0.76 | 3.72 | 7.52 | 9.88 | 227.31 | 31.56 | 0.00 | 0.02 |
| Total | | | | | | | | | | | | | | | | | 249 | 19 | 93 | 188 | 247 | | | | |

**Table 2** Performance comparisons for problem set $n = 15$

| $\tau$ | $R$ | *Deviation from UB* | | | | | | | | | | | | | | | *Number of optimal solutions* | | | | | *Average run times (s)* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *MILP* | | | *m-ATCS* | | | *ISFAN* | | | *TS* | | | *ABC* | | | *MILP* | *m-ATCS* | *ISFAN* | *TS* | *ABC* | *MILP* | *ISFAN* | *TS* | *ABC* |
| | | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | | | | | | | | | |
| 0.1 | 0.1 | 13 | 8 | 1 | 17 | 10 | 4 | 15 | 8 | 1 | 4 | 3 | 1 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 37.67 | 0.04 | 0.04 |
| | 0.3 | 17 | 7 | 1 | 23 | 16 | 8 | 12 | 6 | 2 | 8 | 3 | 0 | 7 | 3 | 1 | 0 | 0 | 0 | 1 | 0 | 3600.00 | 31.26 | 0.01 | 0.03 |
| | 0.5 | 10 | 6 | 1 | 18 | 11 | 4 | 8 | 4 | 2 | 4 | 2 | 0 | 4 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 3600.00 | 25.47 | 0.01 | 0.04 |
| | 0.7 | 21 | 7 | 1 | 29 | 11 | 3 | 9 | 4 | 0 | 4 | 2 | 0 | 4 | 1 | 0 | 0 | 0 | 1 | 1 | 5 | 3600.00 | 23.02 | 0.01 | 0.03 |
| | 0.9 | 15 | 6 | 0 | 28 | 10 | 0 | 11 | 5 | 0 | 6 | 1 | 0 | 6 | 1 | 0 | 1 | 1 | 3 | 4 | 5 | 3241.70 | 19.62 | 0.01 | 0.04 |
| 0.3 | 0.1 | 13 | 10 | 4 | 24 | 15 | 7 | 8 | 6 | 3 | 8 | 4 | 3 | 6 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 44.64 | 0.01 | 0.04 |
| | 0.3 | 22 | 11 | 4 | 41 | 19 | 6 | 19 | 9 | 4 | 11 | 5 | 2 | 11 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 41.25 | 0.01 | 0.04 |
| | 0.5 | 22 | 11 | 4 | 29 | 19 | 9 | 12 | 7 | 5 | 8 | 5 | 1 | 8 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 34.50 | 0.01 | 0.03 |
| | 0.7 | 20 | 11 | 3 | 36 | 20 | 6 | 34 | 11 | 3 | 8 | 4 | 2 | 8 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 32.61 | 0.01 | 0.04 |
| | 0.9 | 16 | 9 | 0 | 31 | 18 | 3 | 13 | 6 | 0 | 7 | 4 | 0 | 7 | 3 | 0 | 1 | 0 | 1 | 2 | 2 | 3444.50 | 30.32 | 0.01 | 0.04 |
| 0.5 | 0.1 | 18 | 12 | 2 | 27 | 20 | 12 | 19 | 11 | 2 | 13 | 7 | 2 | 13 | 7 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 55.18 | 0.01 | 0.04 |
| | 0.3 | 15 | 11 | 7 | 33 | 22 | 8 | 18 | 10 | 4 | 14 | 8 | 4 | 14 | 8 | 4 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 51.61 | 0.01 | 0.05 |
| | 0.5 | 26 | 15 | 7 | 35 | 26 | 10 | 20 | 12 | 4 | 15 | 9 | 6 | 15 | 9 | 4 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 45.57 | 0.01 | 0.04 |
| | 0.7 | 12 | 7 | 0 | 44 | 26 | 11 | 25 | 9 | 4 | 11 | 6 | 1 | 11 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 39.85 | 0.01 | 0.05 |
| | 0.9 | 18 | 7 | 0 | 40 | 21 | 7 | 21 | 8 | 0 | 18 | 6 | 0 | 18 | 6 | 0 | 1 | 0 | 1 | 1 | 1 | 3369.80 | 41.74 | 0.01 | 0.05 |
| 0.7 | 0.1 | 0 | 0 | 0 | 26 | 15 | 0 | 7 | 3 | 0 | 4 | 1 | 0 | 2 | 0 | 0 | 10 | 1 | 1 | 7 | 9 | 83.90 | 55.74 | 0.01 | 0.04 |
| | 0.3 | 0 | 0 | 0 | 30 | 14 | 0 | 5 | 3 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 10 | 1 | 1 | 7 | 10 | 88.60 | 58.06 | 0.01 | 0.05 |
| | 0.5 | 0 | 0 | 0 | 44 | 20 | 12 | 12 | 4 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 1 | 7 | 10 | 101.30 | 59.85 | 0.01 | 0.05 |
| | 0.7 | 8 | 1 | 0 | 33 | 19 | 8 | 20 | 7 | 0 | 8 | 2 | 0 | 8 | 1 | 0 | 9 | 0 | 2 | 3 | 6 | 883.90 | 64.45 | 0.01 | 0.05 |
| | 0.9 | 0 | 0 | 0 | 29 | 17 | 6 | 16 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 1 | 9 | 8 | 288.50 | 53.13 | 0.01 | 0.05 |
| 0.9 | 0.1 | 0 | 0 | 0 | 27 | 10 | 0 | 14 | 4 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 5 | 9 | 10 | 1.00 | 116.49 | 0.01 | 0.05 |
| | 0.3 | 0 | 0 | 0 | 35 | 17 | 8 | 10 | 4 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 3 | 9 | 9 | 1.00 | 87.96 | 0.01 | 0.06 |
| | 0.5 | 0 | 0 | 0 | 24 | 17 | 2 | 12 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 4 | 10 | 8 | 6.10 | 74.97 | 0.01 | 0.05 |
| | 0.7 | 0 | 0 | 0 | 25 | 16 | 3 | 33 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 2 | 10 | 7 | 4.40 | 69.81 | 0.01 | 0.05 |
| | 0.9 | 0 | 0 | 0 | 39 | 21 | 3 | 28 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 2 | 10 | 8 | 12.40 | 69.23 | 0.01 | 0.05 |
| Total average | | 11 | 6 | 1 | 31 | 17 | 6 | 16 | 7 | 1 | 7 | 3 | 1 | 6 | 3 | 1 | 4.08 | 0.16 | 1.12 | 3.64 | 4.04 | 2189.08 | 50.56 | 0.01 | 0.04 |
| Total | | | | | | | | | | | | | | | | | 102 | 4 | 28 | 91 | 101 | | | | |

out of 150 test instances for the problem set $n = 15$ within the 1-h limit. The average run time of the *MILP* increased to 2189 s, while both the *TS* and the proposed ABC-based algorithms run in less than 0.05 s on average.

Table 3 shows the improvement rate of the proposed ABC-based algorithm to TS with respect to *%Deviation from UB*, that is

$$\frac{PDUB_{TS} - PDUB_{ABC}}{\text{Max}(PDUB_{TS}, PDUB_{ABC})} \times 100\%$$

where $PDUB_{TS}$ and $PDUB_{ABC}$ denote *%Deviation from UB* obtained by the ABC-based algorithm and *TS* for each problem set, respectively. Notably, if $PDUB_{TS} = PDUB_{ABC}$, the improvement rate is set as zero. To give an idea about the robustness of the proposed ABC-based algorithm, we re-solved the proposed algorithm five times for each instance and recorded the average improvement rate (%) and the best improvement rate (%) in Table 3. As shown in Table 3, the average improvement rates of ABC are 14.65%, 13.42%, 18.06%, 21.73%, 26.30% and 20.70% for $n = 10$, 15, 20, 25, 50 and 100 problem sets, respectively. In addition, the best improvement rates of ABC are 15.77%, 18.77%, 25.20%, 31.38%, 34.71% and 39.88% for $n = 10$, 15, 20, 25, 50 and 100 problem sets, respectively. From these data, it is clear that the proposed ABC-based algorithm not only outperforms the state-of-art *TS* algorithm but also shows itself to be a robust approach for the addressed problem.

If all orders are accepted, and no order is tardy, this solution can be deemed as the optimal solution. As shown in Tables 4–7, when $n = 20$, 25, 50 and 100, the proposed ABC-based algorithm can obtain a greater number of optimal solutions than *MILP* and *TS*. All the *Max.*, *Avg.* and *Min. % Deviation from UB* of the proposed ABC-based algorithm are less than or equal to the best values obtained by all of the compared approaches. That is, the proposed ABC-based algorithm can obtain more near-optimal solutions, especially when the number of orders increases.

Although the proposed ABC-based algorithm was run on an analogous computer (ie, one with a 3.0 GHz CPU and 4 GB of RAM) to the computer used by Cesaret *et al* (2012) for the algorithms *MILP*, *m-ATCS*, *ISFAN* and *TS*, the computation time may vary depending on software, coding and computational budgets. Therefore, in this study, the computation times are not reported for comparison purposes. However, as shown in Tables 1, 2 and 4–7, the average run time required by the proposed ABC-based algorithm is very short. For small-sized test instances ($n = 10$, 15, 20 and 25), the proposed ABC-based algorithm requires a slightly longer computational time than *TS*. Nevertheless, it is still acceptable because the computational time required for the ABC-based algorithm is less than 1 s. As for large-sized problems ($n = 50$ and 100), the computational time required for the ABC-based

**Table 3** The improvement rate of the proposed ABC-based algorithm to TS with respect to *% Deviation from UB*

| $n$ | $\tau$ | $R$ | *Average improvement rate %* | *Best improvement rate %* | $n$ | $\tau$ | $R$ | *Average improvement rate %* | *Best improvement rate %* | $n$ | $\tau$ | $R$ | *Average improvement rate %* | *Best improvement rate %* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.1 | 0.1 | 37.75 | 37.75 | 20 | 0.1 | 0.1 | 30.92 | 46.43 | 50 | 0.1 | 0.1 | 10.08 | 25.41 |
| | | 0.3 | 40.11 | 47.81 | | | 0.3 | 31.25 | 41.05 | | | 0.3 | 21.32 | 33.46 |
| | | 0.5 | 6.20 | 9.14 | | | 0.5 | 17.92 | 23.09 | | | 0.5 | 45.74 | 60.67 |
| | | 0.7 | 16.36 | 16.36 | | | 0.7 | 16.11 | 16.11 | | | 0.7 | 42.29 | 54.42 |
| | | 0.9 | 28.10 | 28.10 | | | 0.9 | 45.78 | 63.33 | | | 0.9 | 80.00 | 80.00 |
| | 0.3 | 0.1 | 9.43 | 9.43 | | 0.3 | 0.1 | 11.51 | 24.78 | | 0.3 | 0.1 | −2.18 | 12.69 |
| | | 0.3 | 24.68 | 37.69 | | | 0.3 | 19.83 | 29.69 | | | 0.3 | 16.46 | 26.38 |
| | | 0.5 | 35.14 | 35.63 | | | 0.5 | 29.37 | 35.47 | | | 0.5 | 26.92 | 32.85 |
| | | 0.7 | 34.43 | 38.16 | | | 0.7 | 23.93 | 24.69 | | | 0.7 | 41.99 | 54.47 |
| | | 0.9 | 47.10 | 47.10 | | | 0.9 | 23.84 | 30.19 | | | 0.9 | 66.85 | 76.71 |
| | 0.5 | 0.1 | 9.86 | 9.86 | | 0.5 | 0.1 | 17.07 | 24.28 | | 0.5 | 0.1 | 25.54 | 34.63 |
| | | 0.3 | 19.66 | 9.66 | | | 0.3 | 14.65 | 21.51 | | | 0.3 | 24.83 | 35.36 |
| | | 0.5 | 8.92 | 8.92 | | | 0.5 | 17.83 | 21.06 | | | 0.5 | 18.11 | 26.63 |
| | | 0.7 | 9.64 | 9.64 | | | 0.7 | 9.18 | 21.07 | | | 0.7 | 24.71 | 37.60 |
| | | 0.9 | 9.48 | 9.48 | | | 0.9 | 12.10 | 15.37 | | | 0.9 | 38.12 | 52.30 |
| | 0.7 | 0.1 | 10.00 | 10.00 | | 0.7 | 0.1 | 7.59 | 15.20 | | 0.7 | 0.1 | 27.17 | 36.94 |
| | | 0.3 | 0.00 | 0.00 | | | 0.3 | 6.15 | 8.76 | | | 0.3 | 20.37 | 26.14 |
| | | 0.5 | 9.64 | 9.64 | | | 0.5 | 0.10 | 7.40 | | | 0.5 | 20.95 | 26.78 |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0.7 | 0.00 | 0.00 | | | 0.7 | 1.65 | 3.12 | | | 0.7 | 17.64 | 24.62 |
| | | 0.9 | 9.85 | 9.85 | | | 0.9 | −0.91 | 10.60 | | | 0.9 | 21.54 | 26.03 |
| | 0.9 | 0.1 | 0.00 | 0.00 | | 0.9 | 0.1 | 8.00 | 10.00 | | 0.9 | 0.1 | 9.63 | 13.32 |
| | | 0.3 | 0.00 | 0.00 | | | 0.3 | 13.85 | 15.83 | | | 0.3 | 15.67 | 17.38 |
| | | 0.5 | 0.00 | 0.00 | | | 0.5 | 21.11 | 27.59 | | | 0.5 | 19.20 | 21.99 |
| | | 0.7 | 0.00 | 0.00 | | | 0.7 | 27.64 | 35.25 | | | 0.7 | 14.83 | 17.98 |
| | | 0.9 | 0.00 | 0.00 | | | 0.9 | 45.00 | 58.10 | | | 0.9 | 9.77 | 12.93 |
| Ave. | | | 14.65 | 15.77 | | | | 18.06 | 25.20 | | | | 26.30 | 34.71 |
| 15 | 0.1 | 0.1 | 11.76 | 22.00 | 25 | 0.1 | 0.1 | 26.57 | 39.63 | 100 | 0.1 | 0.1 | 20.77 | 35.04 |
| | | 0.3 | −3.09 | 21.40 | | | 0.3 | 34.92 | 40.24 | | | 0.3 | 29.55 | 42.39 |
| | | 0.5 | −2.55 | 11.33 | | | 0.5 | 45.07 | 52.91 | | | 0.5 | 35.55 | 52.05 |
| | | 0.7 | 40.13 | 51.32 | | | 0.7 | 25.20 | 28.75 | | | 0.7 | 70.43 | 85.00 |
| | | 0.9 | 12.19 | 12.19 | | | 0.9 | 37.24 | 46.67 | | | 0.9 | 56.00 | 60.00 |
| | 0.3 | 0.1 | 24.75 | 34.44 | | 0.3 | 0.1 | 7.45 | 18.93 | | 0.3 | 0.1 | 17.89 | 33.10 |
| | | 0.3 | 27.02 | 30.05 | | | 0.3 | 28.68 | 39.92 | | | 0.3 | 26.85 | 35.16 |
| | | 0.5 | 15.51 | 17.78 | | | 0.5 | 23.37 | 29.92 | | | 0.5 | 19.82 | 29.34 |
| | | 0.7 | 28.05 | 31.04 | | | 0.7 | 17.11 | 30.98 | | | 0.7 | 58.15 | 70.73 |
| | | 0.9 | 22.43 | 22.54 | | | 0.9 | 23.29 | 32.64 | | | 0.9 | 58.90 | 81.87 |
| | 0.5 | 0.1 | 6.83 | 6.83 | | 0.5 | 0.1 | 20.62 | 28.92 | | 0.5 | 0.1 | 14.64 | 23.91 |
| | | 0.3 | 3.49 | 9.40 | | | 0.3 | 17.46 | 21.92 | | | 0.3 | 18.36 | 28.15 |
| | | 0.5 | 3.39 | 4.00 | | | 0.5 | 12.05 | 22.53 | | | 0.5 | 10.47 | 21.01 |
| | | 0.7 | 13.25 | 14.95 | | | 0.7 | 38.13 | 48.49 | | | 0.7 | 30.26 | 38.45 |
| | | 0.9 | 3.49 | 3.56 | | | 0.9 | 23.27 | 30.89 | | | 0.9 | 32.90 | 49.81 |
| | 0.7 | 0.1 | 17.99 | 27.99 | | 0.7 | 0.1 | 21.05 | 24.62 | | 0.7 | 0.1 | 19.35 | 27.01 |
| | | 0.3 | 25.99 | 28.29 | | | 0.3 | 11.44 | 14.36 | | | 0.3 | 28.81 | 35.68 |
| | | 0.5 | 17.95 | 28.19 | | | 0.5 | 10.97 | 13.98 | | | 0.5 | 20.01 | 26.16 |
| | | 0.7 | 49.90 | 55.55 | | | 0.7 | 13.02 | 15.07 | | | 0.7 | 24.04 | 30.29 |
| | | 0.9 | 1.40 | 10.00 | | | 0.9 | 29.59 | 31.32 | | | 0.9 | 30.87 | 38.34 |
| | 0.9 | 0.1 | 10.00 | 10.00 | | 0.9 | 0.1 | 34.31 | 42.31 | | 0.9 | 0.1 | 26.72 | 32.95 |
| | | 0.3 | 10.00 | 10.00 | | | 0.3 | 1.61 | 24.80 | | | 0.3 | 28.97 | 32.85 |
| | | 0.5 | −5.45 | 1.00 | | | 0.5 | 38.13 | 48.49 | | | 0.5 | 25.55 | 28.35 |
| | | 0.7 | −8.00 | 0.83 | | | 0.7 | 4.17 | 29.68 | | | 0.7 | 24.91 | 28.18 |
| | | 0.9 | 4.59 | 4.59 | | | 0.9 | −1.37 | 26.62 | | | 0.9 | 27.38 | 31.08 |
| Ave. | | | 13.42 | 18.77 | | | | 21.73 | 31.38 | | | | 20.70 | 39.88 |

**Table 4** Performance comparisons for problem set $n = 20$

| $\tau$ | $R$ | *Deviation from UB* | | | | | | | | | | | | | | | *Number of optimal solutions* | | | | | *Average run times (s)* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *MILP* | | | *m-ATCS* | | | *ISFAN* | | | *TS* | | | *ABC* | | | *MILP* | *m-ATCS* | *ISFAN* | *TS* | *ABC* | *MILP* | *ISFAN* | *TS* | *ABC* |
| | | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | | | | | | | | | |
| 0.1 | 0.1 | 19 | 11 | 6 | 17 | 11 | 6 | 10 | 7 | 2 | 5 | 3 | 1 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 3600.00 | 56.87 | 0.10 | 0.06 |
| | 0.3 | 19 | 12 | 2 | 30 | 17 | 5 | 13 | 9 | 3 | 5 | 3 | 1 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 54.07 | 0.04 | 0.06 |
| | 0.5 | 22 | 10 | 3 | 23 | 17 | 4 | 8 | 5 | 1 | 3 | 2 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 40.73 | 0.03 | 0.05 |
| | 0.7 | 23 | 11 | 0 | 23 | 12 | 6 | 10 | 5 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5 | 6 | 3415.50 | 35.16 | 0.04 | 0.05 |
| | 0.9 | 15 | 7 | 1 | 24 | 10 | 0 | 17 | 6 | 0 | 4 | 1 | 0 | 3 | 1 | 0 | 0 | 2 | 1 | 3 | 7 | 3600.00 | 32.22 | 0.03 | 0.07 |
| 0.3 | 0.1 | 29 | 18 | 7 | 28 | 20 | 9 | 18 | 11 | 7 | 8 | 5 | 2 | 7 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 69.44 | 0.03 | 0.07 |
| | 0.3 | 21 | 14 | 4 | 27 | 20 | 12 | 15 | 9 | 5 | 6 | 5 | 3 | 6 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 63.65 | 0.04 | 0.07 |
| | 0.5 | 25 | 16 | 7 | 26 | 17 | 9 | 12 | 8 | 5 | 8 | 5 | 2 | 6 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 59.10 | 0.03 | 0.07 |
| | 0.7 | 25 | 12 | 5 | 31 | 17 | 5 | 12 | 7 | 2 | 7 | 3 | 1 | 7 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 55.62 | 0.04 | 0.07 |
| | 0.9 | 20 | 10 | 2 | 28 | 18 | 5 | 13 | 7 | 1 | 5 | 2 | 0 | 5 | 2 | 0 | 0 | 0 | 0 | 2 | 2 | 3444.50 | 53.07 | 0.04 | 0.08 |
| 0.5 | 0.1 | 30 | 13 | 6 | 26 | 20 | 12 | 13 | 10 | 7 | 8 | 6 | 4 | 6 | 5 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 79.69 | 0.03 | 0.08 |
| | 0.3 | 26 | 17 | 7 | 63 | 26 | 11 | 18 | 12 | 5 | 9 | 6 | 3 | 9 | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 77.91 | 0.07 | 0.07 |
| | 0.5 | 23 | 16 | 9 | 37 | 27 | 19 | 17 | 13 | 9 | 10 | 7 | 3 | 9 | 5 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 75.72 | 0.04 | 0.09 |
| | 0.7 | 27 | 14 | 8 | 35 | 23 | 13 | 17 | 12 | 7 | 11 | 5 | 1 | 11 | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 70.42 | 0.03 | 0.09 |
| | 0.9 | 21 | 12 | 5 | 38 | 26 | 10 | 23 | 14 | 5 | 11 | 6 | 0 | 11 | 6 | 0 | 0 | 0 | 0 | 1 | 1 | 3600.00 | 73.10 | 0.05 | 0.07 |
| 0.7 | 0.1 | 26 | 12 | 6 | 30 | 22 | 17 | 20 | 14 | 8 | 15 | 10 | 5 | 14 | 9 | 5 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 92.53 | 0.05 | 0.09 |
| | 0.3 | 14 | 10 | 6 | 33 | 24 | 10 | 18 | 13 | 10 | 14 | 9 | 6 | 13 | 8 | 6 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 97.47 | 0.04 | 0.10 |
| | 0.5 | 27 | 10 | 0 | 47 | 35 | 19 | 24 | 14 | 5 | 20 | 9 | 0 | 20 | 9 | 0 | 2 | 0 | 0 | 2 | 1 | 3048.20 | 105.50 | 0.04 | 0.09 |
| | 0.7 | 20 | 8 | 0 | 40 | 25 | 9 | 24 | 13 | 0 | 12 | 7 | 0 | 12 | 6 | 0 | 3 | 0 | 1 | 3 | 2 | 2954.40 | 100.07 | 0.07 | 0.10 |
| | 0.9 | 17 | 10 | 0 | 44 | 29 | 7 | 34 | 17 | 8 | 13 | 8 | 0 | 12 | 7 | 0 | 1 | 0 | 0 | 1 | 0 | 3242.80 | 104.19 | 0.04 | 0.08 |
| 0.9 | 0.1 | 0 | 0 | 0 | 18 | 10 | 1 | 8 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 3 | 0 | 10 | 1.70 | 138.66 | 0.06 | 0.11 |
| | 0.3 | 0 | 0 | 0 | 26 | 15 | 7 | 24 | 7 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 10 | 0 | 1 | 8 | 8 | 8.70 | 144.22 | 0.03 | 0.12 |
| | 0.5 | 0 | 0 | 0 | 43 | 20 | 8 | 24 | 8 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 8 | 7 | 80.10 | 127.94 | 0.03 | 0.11 |
| | 0.7 | 0 | 0 | 0 | 46 | 21 | 11 | 13 | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 3 | 8 | 6 | 114.00 | 119.66 | 0.03 | 0.11 |
| | 0.9 | 13 | 1 | 0 | 39 | 27 | 12 | 18 | 12 | 0 | 13 | 2 | 0 | 13 | 1 | 0 | 9 | 0 | 1 | 6 | 4 | 467.00 | 118.01 | 0.03 | 0.10 |
| Total average | | 18 | 10 | 3 | 33 | 20 | 9 | 17 | 10 | 4 | 8 | 4 | 1 | 7 | 4 | 1 | 2.24 | 0.08 | 0.4 | 1.88 | 2.32 | 2831.07 | 81.80 | 0.04 | 0.07 |
| Total | | | | | | | | | | | | | | | | | 56 | 2 | 10 | 47 | 58 | | | | |

**Table 5** Performance comparisons for problem set $n=25$

| $\tau$ | $R$ | *Deviation from UB* | | | | | | | | | | | | | | | *Number of optimal solutions* | | | | | *Average run times (s)* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *MILP* | | | *m-ATCS* | | | *ISFAN* | | | *TS* | | | *ABC* | | | *MILP* | *m-ATCS* | *ISFAN* | *TS* | *ABC* | *MILP* | *ISFAN* | *TS* | *ABC* |
| | | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max (%)* | *Ave. (%)* | *Min (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | | | | | | | | | |
| 0.1 | 0.1 | 19 | 12 | 8 | 20 | 14 | 9 | 12 | 8 | 4 | 6 | 4 | 1 | 4 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 83.62 | 0.08 | 0.09 |
| | 0.3 | 23 | 13 | 6 | 32 | 16 | 6 | 11 | 7 | 3 | 6 | 3 | 2 | 6 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 65.37 | 0.06 | 0.09 |
| | 0.5 | 9 | 7 | 5 | 21 | 12 | 3 | 11 | 7 | 3 | 4 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 61.45 | 0.06 | 0.08 |
| | 0.7 | 11 | 5 | 1 | 23 | 13 | 5 | 9 | 5 | 2 | 4 | 1 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 4 | 6 | 3600.00 | 53.97 | 0.06 | 0.10 |
| | 0.9 | 10 | 4 | 1 | 24 | 8 | 2 | 10 | 6 | 1 | 2 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 4 | 7 | 3600.00 | 56.69 | 0.05 | 0.10 |
| 0.3 | 0.1 | 24 | 15 | 9 | 29 | 19 | 13 | 12 | 9 | 5 | 5 | 4 | 2 | 5 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 93.23 | 0.10 | 0.11 |
| | 0.3 | 20 | 11 | 7 | 24 | 18 | 7 | 12 | 10 | 7 | 7 | 5 | 3 | 6 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 89.71 | 0.09 | 0.10 |
| | 0.5 | 8 | 5 | 2 | 26 | 18 | 10 | 10 | 7 | 5 | 6 | 3 | 2 | 5 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 80.76 | 0.08 | 0.12 |
| | 0.7 | 9 | 5 | 1 | 32 | 17 | 8 | 24 | 11 | 6 | 6 | 2 | 1 | 5 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 78.54 | 0.08 | 0.12 |
| | 0.9 | 27 | 18 | 11 | 27 | 16 | 6 | 18 | 9 | 5 | 4 | 2 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 2 | 2 | 3600.00 | 79.35 | 0.07 | 0.12 |
| 0.5 | 0.1 | 17 | 10 | 6 | 29 | 19 | 12 | 14 | 11 | 7 | 7 | 6 | 3 | 7 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 110.84 | 0.07 | 0.12 |
| | 0.3 | 13 | 9 | 4 | 29 | 21 | 11 | 17 | 11 | 6 | 9 | 5 | 3 | 7 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 109.09 | 0.08 | 0.12 |
| | 0.5 | 17 | 10 | 3 | 30 | 20 | 8 | 20 | 13 | 8 | 8 | 5 | 2 | 6 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 110.30 | 0.09 | 0.16 |
| | 0.7 | 13 | 8 | 2 | 38 | 26 | 17 | 17 | 13 | 7 | 11 | 6 | 2 | 12 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3600.00 | 121.02 | 0.08 | 0.19 |
| | 0.9 | 42 | 20 | 6 | 49 | 28 | 16 | 25 | 15 | 6 | 7 | 4 | 1 | 7 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 109.74 | 0.08 | 0.11 |
| 0.7 | 0.1 | 19 | 10 | 2 | 26 | 20 | 12 | 27 | 18 | 10 | 18 | 9 | 3 | 16 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 142.09 | 0.06 | 0.14 |
| | 0.3 | 17 | 12 | 8 | 33 | 26 | 18 | 23 | 17 | 9 | 14 | 10 | 7 | 12 | 9 | 5 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 141.36 | 0.08 | 0.11 |
| | 0.5 | 21 | 13 | 7 | 43 | 33 | 26 | 26 | 19 | 13 | 15 | 12 | 7 | 14 | 10 | 5 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 143.57 | 0.08 | 0.18 |
| | 0.7 | 15 | 9 | 2 | 40 | 29 | 15 | 21 | 15 | 9 | 14 | 8 | 2 | 12 | 7 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 140.45 | 0.07 | 0.15 |
| | 0.9 | 17 | 10 | 0 | 53 | 29 | 8 | 24 | 17 | 10 | 15 | 10 | 3 | 14 | 8 | 0 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 152.11 | 0.08 | 0.15 |
| 0.9 | 0.1 | 0 | 0 | 0 | 20 | 12 | 4 | 25 | 8 | 0 | 6 | 1 | 0 | 5 | 1 | 0 | 10 | 0 | 1 | 5 | 8 | 5.41 | 188.99 | 0.06 | 0.18 |
| | 0.3 | 0 | 0 | 0 | 22 | 17 | 10 | 8 | 5 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 10 | 0 | 0 | 8 | 6 | 8.30 | 187.45 | 0.06 | 0.17 |
| | 0.5 | 12 | 3 | 0 | 45 | 27 | 14 | 44 | 12 | 1 | 12 | 4 | 0 | 12 | 3 | 0 | 6 | 0 | 0 | 3 | 3 | 1452.61 | 176.50 | 0.07 | 0.19 |
| | 0.7 | 22 | 7 | 0 | 45 | 29 | 17 | 31 | 13 | 0 | 25 | 8 | 0 | 21 | 7 | 0 | 6 | 0 | 1 | 4 | 4 | 1446.35 | 177.66 | 0.08 | 0.16 |
| | 0.9 | 21 | 6 | 0 | 58 | 30 | 20 | 31 | 13 | 2 | 22 | 7 | 0 | 19 | 6 | 0 | 4 | 0 | 0 | 4 | 2 | 2176.18 | 169.41 | 0.09 | 0.17 |
| Total average | | 16 | 9 | 4 | 33 | 21 | 11 | 19 | 11 | 5 | 9 | 5 | 2 | 8 | 4 | 1 | 1.44 | 0 | 0.08 | 1.36 | 1.72 | 3083.55 | 116.93 | 0.07 | 0.13 |
| Total | | | | | | | | | | | | | | | | | 36 | 0 | 2 | 34 | 43 | | | | |

**Table 6** Performance comparisons for problem set $n = 50$

| $\tau$ | $R$ | *Deviation from UB* | | | | | | | | | | | | | | | *Number of optimal solutions* | | | | | *Average run times (s)* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *MILP* | | | *m-ATCS* | | | *ISFAN* | | | *TS* | | | *ABC* | | | *MILP* | *m-ATCS* | *ISFAN* | *TS* | *ABC* | *MILP* | *ISFAN* | *TS* | *ABC* |
| | | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | | | | | | | | | |
| 0.1 | 0.1 | 81 | 24 | 9 | 20 | 15 | 9 | 12 | 8 | 6 | 3 | 2 | 1 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 264.38 | 1.19 | 0.68 |
| | 0.3 | 22 | 16 | 7 | 18 | 15 | 9 | 12 | 9 | 7 | 4 | 2 | 1 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 250.16 | 0.98 | 0.60 |
| | 0.5 | 25 | 17 | 11 | 19 | 13 | 6 | 10 | 8 | 5 | 2 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 238.33 | 0.89 | 0.80 |
| | 0.7 | 18 | 13 | 9 | 24 | 14 | 7 | 22 | 9 | 4 | 16 | 3 | 0 | 16 | 2 | 0 | 0 | 0 | 0 | 1 | 4 | 3600.00 | 220.68 | 0.58 | 0.73 |
| | 0.9 | 27 | 10 | 2 | 10 | 4 | 1 | 11 | 7 | 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 10 | 3600.00 | 288.67 | 0.46 | 1.14 |
| 0.3 | 0.1 | 33 | 24 | 13 | 24 | 16 | 11 | 13 | 11 | 8 | 3 | 3 | 2 | 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 338.06 | 1.61 | 0.69 |
| | 0.3 | 48 | 28 | 18 | 30 | 20 | 11 | 15 | 12 | 9 | 5 | 4 | 3 | 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 343.90 | 1.49 | 0.44 |
| | 0.5 | 33 | 24 | 18 | 33 | 18 | 9 | 15 | 11 | 8 | 5 | 3 | 1 | 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 328.47 | 1.16 | 0.80 |
| | 0.7 | 26 | 20 | 13 | 24 | 14 | 5 | 18 | 11 | 7 | 3 | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 3600.00 | 346.78 | 0.85 | 0.72 |
| | 0.9 | 34 | 21 | 13 | 26 | 17 | 3 | 11 | 9 | 8 | 3 | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 3600.00 | 345.54 | 0.68 | 1.01 |
| 0.5 | 0.1 | 31 | 28 | 19 | 27 | 20 | 13 | 19 | 14 | 10 | 5 | 4 | 3 | 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 430.80 | 1.36 | 0.64 |
| | 0.3 | 86 | 42 | 25 | 30 | 25 | 18 | 20 | 16 | 11 | 8 | 6 | 3 | 6 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 448.20 | 1.18 | 0.45 |
| | 0.5 | 44 | 31 | 17 | 34 | 25 | 15 | 21 | 16 | 7 | 8 | 4 | 2 | 7 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 438.71 | 1.35 | 0.77 |
| | 0.7 | 38 | 21 | 10 | 31 | 24 | 12 | 18 | 15 | 9 | 5 | 3 | 2 | 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 471.10 | 1.22 | 1.02 |
| | 0.9 | 22 | 18 | 8 | 35 | 27 | 17 | 19 | 15 | 6 | 6 | 4 | 2 | 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 495.41 | 1.16 | 0.72 |
| 0.7 | 0.1 | 25 | 21 | 13 | 30 | 22 | 17 | 21 | 16 | 12 | 9 | 7 | 4 | 6 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 561.75 | 1.44 | 0.97 |
| | 0.3 | 23 | 18 | 11 | 33 | 24 | 16 | 20 | 16 | 11 | 11 | 7 | 4 | 9 | 6 | 4 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 565.54 | 1.52 | 0.95 |
| | 0.5 | 52 | 29 | 13 | 46 | 33 | 23 | 22 | 19 | 14 | 13 | 9 | 7 | 11 | 7 | 5 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 582.52 | 1.59 | 0.94 |
| | 0.7 | 28 | 21 | 9 | 35 | 31 | 24 | 29 | 19 | 12 | 18 | 9 | 2 | 15 | 8 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 609.67 | 1.27 | 0.95 |
| | 0.9 | 26 | 19 | 14 | 41 | 29 | 19 | 28 | 21 | 17 | 18 | 11 | 6 | 14 | 8 | 4 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 617.63 | 1.15 | 1.09 |
| 0.9 | 0.1 | 18 | 14 | 8 | 30 | 24 | 17 | 22 | 17 | 12 | 18 | 13 | 8 | 17 | 11 | 7 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 684.75 | 1.25 | 1.48 |
| | 0.3 | 22 | 16 | 11 | 42 | 32 | 21 | 28 | 23 | 18 | 23 | 17 | 13 | 18 | 14 | 9 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 686.89 | 1.26 | 1.32 |
| | 0.5 | 21 | 15 | 5 | 45 | 39 | 33 | 26 | 21 | 18 | 23 | 17 | 10 | 17 | 13 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 672.19 | 1.42 | 1.19 |
| | 0.7 | 25 | 17 | 9 | 46 | 37 | 26 | 72 | 27 | 17 | 21 | 16 | 11 | 18 | 12 | 6 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 780.25 | 1.43 | 1.24 |
| | 0.9 | 30 | 21 | 14 | 44 | 36 | 24 | 29 | 22 | 16 | 19 | 16 | 11 | 16 | 12 | 10 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 673.72 | 1.37 | 1.03 |
| Total average | | 34 | 21 | 12 | 31 | 23 | 15 | 21 | 15 | 10 | 10 | 7 | 5 | 8 | 5 | 3 | 0.00 | 0.00 | 0.00 | 0.16 | 0.72 | 3600.00 | 467.36 | 1.19 | 0.89 |
| Total | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 4 | 18 | | | | |

**Table 7** Performance comparisons for problem set $n = 100$

| $\tau$ | $R$ | *Deviation from UB* | | | | | | | | | | | | | | | *Number of optimal solutions* | | | | | *Average run times (s)* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *MILP* | | | *m-ATCS* | | | *ISFAN* | | | *TS* | | | *ABC* | | | *MILP* | *m-ATCS* | *ISFAN* | *TS* | *ABC* | *MILP* | *ISFAN* | *TS* | *ABC* |
| | | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | *Max. (%)* | *Ave. (%)* | *Min. (%)* | | | | | | | | | |
| 0.1 | 0.1 | 56 | 44 | 37 | 19 | 15 | 9 | 13 | 9 | 8 | 3 | 2 | 1 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 682.82 | 16.76 | 3.98 |
| | 0.3 | 88 | 51 | 42 | 20 | 17 | 11 | 10 | 9 | 7 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 666.81 | 17.26 | 8.22 |
| | 0.5 | 60 | 51 | 43 | 19 | 14 | 11 | 11 | 9 | 6 | 3 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 690.50 | 10.87 | 7.61 |
| | 0.7 | 60 | 54 | 46 | 16 | 11 | 5 | 12 | 9 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 6 | 3600.00 | 701.30 | 6.21 | 6.25 |
| | 0.9 | 70 | 60 | 38 | 11 | 6 | 0 | 16 | 12 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 10 | 3600.00 | 744.60 | 3.53 | 9.08 |
| 0.3 | 0.1 | 68 | 62 | 52 | 24 | 19 | 15 | 13 | 12 | 10 | 4 | 3 | 1 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 941.03 | 22.37 | 4.89 |
| | 0.3 | 75 | 64 | 52 | 23 | 18 | 15 | 17 | 13 | 11 | 5 | 3 | 2 | 4 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 964.58 | 20.10 | 5.29 |
| | 0.5 | 76 | 66 | 56 | 21 | 18 | 14 | 17 | 14 | 10 | 4 | 2 | 1 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 988.50 | 16.77 | 4.88 |
| | 0.7 | 84 | 73 | 63 | 32 | 20 | 15 | 15 | 14 | 12 | 3 | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3600.00 | 1028.52 | 9.48 | 6.16 |
| | 0.9 | 82 | 66 | 47 | 18 | 15 | 10 | 17 | 13 | 9 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3600.00 | 1020.23 | 7.58 | 8.59 |
| 0.5 | 0.1 | 87 | 77 | 71 | 24 | 22 | 17 | 18 | 16 | 12 | 5 | 4 | 2 | 4 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1243.25 | 25.96 | 5.27 |
| | 0.3 | 86 | 61 | 47 | 28 | 23 | 17 | 18 | 15 | 12 | 6 | 4 | 3 | 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1288.58 | 28.87 | 6.94 |
| | 0.5 | 88 | 70 | 56 | 31 | 24 | 20 | 19 | 17 | 14 | 5 | 4 | 3 | 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1305.33 | 20.56 | 5.86 |
| | 0.7 | 92 | 71 | 55 | 34 | 23 | 10 | 21 | 17 | 13 | 4 | 3 | 2 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1385.12 | 15.57 | 6.57 |
| | 0.9 | 100 | 66 | 48 | 36 | 24 | 15 | 24 | 18 | 12 | 5 | 2 | 1 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1419.77 | 12.15 | 6.76 |
| 0.7 | 0.1 | 86 | 67 | 49 | 28 | 22 | 17 | 19 | 17 | 13 | 6 | 5 | 3 | 5 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1646.30 | 33.60 | 6.04 |
| | 0.3 | 66 | 57 | 45 | 34 | 24 | 18 | 21 | 17 | 13 | 11 | 7 | 4 | 7 | 5 | 2 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1659.33 | 26.62 | 6.25 |
| | 0.5 | 65 | 55 | 46 | 37 | 26 | 20 | 24 | 18 | 14 | 13 | 6 | 4 | 12 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1660.40 | 22.30 | 6.77 |
| | 0.7 | 76 | 60 | 42 | 38 | 31 | 22 | 23 | 19 | 16 | 13 | 7 | 3 | 7 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1690.41 | 26.36 | 6.78 |
| | 0.9 | 64 | 53 | 39 | 37 | 31 | 23 | 24 | 18 | 15 | 13 | 8 | 5 | 8 | 6 | 4 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1657.41 | 17.84 | 7.42 |
| 0.9 | 0.1 | 48 | 37 | 31 | 25 | 22 | 18 | 20 | 17 | 14 | 12 | 9 | 7 | 9 | 7 | 5 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1288.09 | 19.13 | 13.63 |
| | 0.3 | 50 | 40 | 28 | 36 | 31 | 25 | 26 | 20 | 16 | 23 | 15 | 9 | 12 | 9 | 5 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1807.60 | 26.32 | 9.82 |
| | 0.5 | 54 | 38 | 23 | 39 | 33 | 25 | 25 | 21 | 19 | 18 | 16 | 13 | 17 | 12 | 9 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1802.09 | 21.51 | 6.46 |
| | 0.7 | 48 | 38 | 27 | 40 | 36 | 26 | 24 | 21 | 15 | 20 | 16 | 11 | 13 | 11 | 8 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1798.96 | 22.70 | 7.52 |
| | 0.9 | 39 | 35 | 27 | 40 | 35 | 28 | 28 | 21 | 13 | 22 | 16 | 12 | 17 | 11 | 4 | 0 | 0 | 0 | 0 | 0 | 3600.00 | 1765.10 | 17.48 | 8.61 |
| Total average | 71 | 57 | 44 | | 28 | 22 | 16 | 19 | 15 | 12 | 8 | 6 | 4 | 6 | 4 | 2 | 0.00 | 0.00 | 0.00 | 0.20 | 0.84 | 3600.00 | 1288.09 | 19.13 | 7.03 |
| Total | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 5 | 21 | | | | |

**Table 8** Paired *t*-tests on % *Deviation from UB*

| *ABC v.s.* | *MILP* | *m-ATCS* | *ISFAN* | *TS* |
|---|---|---|---|---|
| *Max. % Deviation from UB* | | | | |
| Difference | −19.04% | −25.06% | −12.49% | −1.40% |
| Degree of freedom | 149 | 149 | 149 | 149 |
| *t*-value | 9.192 | 38.841 | 20.100 | 11.490 |
| One-tail significance | 1.57E-16 | 3.83E-80 | 1.44E-44 | 1.40E-22 |
| *Ave. % Deviation from UB* | | | | |
| Difference | −13.83% | −16.35% | −7.37% | −0.86% |
| Degree of freedom | 149 | 149 | 149 | 149 |
| *t*-value | 8.619 | 41.081 | 26.839 | 13.137 |
| One-tail significance | 4.55E-15 | 1.84E-83 | 3.07E-59 | 5.68E-27 |
| *Min. % Deviation from UB* | | | | |
| Difference | −9.48% | −8.38% | −4.08% | −0.49% |
| Degree of freedom | 149 | 149 | 149 | 149 |
| *t*-value | 7.228 | 16.645 | 12.381 | 6.246 |
| One-tail significance | 1.18E-11 | 4.10E-36 | 5.88E-25 | 2.09E-09 |

algorithm is less than that of the *TS* algorithm, while the solution quality of the ABC-based algorithm is better than that of the *TS* algorithm. Even for problem sizes involving up to 100 orders, the proposed ABC-based algorithm obtained very good solutions for a reasonable computational expense. As such, there is no doubt that the proposed ABC-based algorithm is an efficient algorithm that can be applied to real-world problems.

To further verify the effectiveness, one-tailed paired *t*-tests on the *Max.*, *Avg.* and *Min. % Deviation from UB* were performed to compare the proposed ABC-based algorithm with the *MILP*, *m-ATCS*, *ISFAN* and *TS*. At confidence level $\alpha = 0.05$, the results in Table 8 show that the proposed ABC-based algorithm significantly outperforms the *MILP*, *m-ATCS*, *ISFAN* and *TS* algorithms with respect to *Max.*, *Avg.* and *Min. % Deviation from UB.*

The reason that the proposed ABC-based algorithm outperforms the existing algorithms is probably because it combines the advantages of both the IG and ABC algorithms. To confirm whether this combination can improve the performance or not, a comparison is made of the ABC algorithm with the destruction and construction phases of the IG heuristic, and with the typical swap and insertion operators for generating neighbourhood solutions. The analytical results are listed in Table 9. As shown in Table 9, unlike the combination with the swap and insertion operators, the ABC algorithm that combines with the destruction and construction phases of the IG heuristic can achieve a lower average percentage deviation from UB values. It is clear that the simple but effective perturbation mechanism of the IG heuristic—the destruction and construction procedures—can significantly reduce the search effort and improve the solution quality.

Judging from the above experimental results, it is clear that the proposed ABC-based algorithm is not only effective, but also efficient when compared to the best known algorithms on the OAS problem addressed in this study. This study appears to have made a step towards reducing the gap between theoretical progress and industrial practice for the OAS problem on a single machine with release dates and sequence-dependent setup times.

## 5. Conclusions and recommendations for future studies

This paper proposes an ABC-based algorithm for solving the OAS problem on a single machine with release dates and sequence-dependent setup times. To empirically evaluate the applicability of the proposed ABC-based algorithm, its performance was compared with that of the best-so-far algorithms on the same benchmark problem set. The computational and analytical results prove that the performance of the proposed ABC-based algorithm is superior to that of the state-of-the-art metaheuristic-based algorithms. We believe that this study successfully developed and demonstrated a high-performance algorithm that can serve as a new benchmark approach for future research on this problem.

The following recommendations are made for future research on OAS problems. First, additional efficient and effective metaheuristic-based algorithms for this problem, including hybrid metaheuristic-based algorithms, deserve future research. Second, it would be interesting to develop additional exact methods and dispatching rules for this problem. Third, an extension of the single machine OAS

**Table 9** Performance comparison of the ABC algorithm combined with the destruction and construction (D&C) phases of IG, and with the swap and insertion operators

| *N* | $\tau$ | *R* | *Swap and Insertion (%)* | *D&C phases (%)* | *N* | $\tau$ | *R* | *Swap and Insertion (%)* | *D&C phases (%)* | *N* | $\tau$ | *R* | *Swap and Insertion (%)* | *D&C phases (%)* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.1 | 0.1 | 0.2[a] | 0.1 | 20 | 0.1 | 0.1 | 2.9 | 1.9 | 50 | 0.1 | 0.1 | 2.2 | 1.9 |
| | | 0.3 | 0.6 | 0.3 | | | 0.3 | 2.3 | 1.7 | | | 0.3 | 1.9 | 1.8 |
| | | 0.5 | 1.0 | 0.7 | | | 0.5 | 1.8 | 1.4 | | | 0.5 | 1.0 | 1.0 |
| | | 0.7 | 0.1 | 0.1 | | | 0.7 | 0.5 | 0.5 | | | 0.7 | 2.2 | 1.9 |
| | | 0.9 | 0.1 | 0.0 | | | 0.9 | 0.4 | 0.5 | | | 0.9 | 0.0 | 0.0 |
| | 0.3 | 0.1 | 0.3 | 0.1 | | 0.3 | 0.1 | 4.8 | 3.8 | | 0.3 | 0.1 | 3.2 | 2.6 |
| | | 0.3 | 0.6 | 0.1 | | | 0.3 | 4.1 | 3.7 | | | 0.3 | 3.0 | 3.0 |
| | | 0.5 | 0.2 | 0.2 | | | 0.5 | 4.5 | 3.7 | | | 0.5 | 2.1 | 1.9 |
| | | 0.7 | 0.4 | 0.1 | | | 0.7 | 2.7 | 2.5 | | | 0.7 | 0.8 | 0.7 |
| | | 0.9 | 0.5 | 0.1 | | | 0.9 | 1.9 | 1.7 | | | 0.9 | 0.6 | 0.5 |
| | 0.5 | 0.1 | 0.4 | 0.1 | | 0.5 | 0.1 | 4.8 | 4.6 | | 0.5 | 0.1 | 3.6 | 3.1 |
| | | 0.3 | 0.1 | 0.1 | | | 0.3 | 6.2 | 5.4 | | | 0.3 | 4.9 | 4.5 |
| | | 0.5 | 0.1 | 0.1 | | | 0.5 | 6.6 | 5.5 | | | 0.5 | 4.5 | 3.8 |
| | | 0.7 | 1.0 | 0.1 | | | 0.7 | 5.9 | 4.7 | | | 0.7 | 2.6 | 2.3 |
| | | 0.9 | 0.4 | 0.1 | | | 0.9 | 6.4 | 5.5 | | | 0.9 | 2.5 | 2.2 |
| | 0.7 | 0.1 | 0.4 | 0.0 | | 0.7 | 0.1 | 9.7 | 8.8 | | 0.7 | 0.1 | 6.0 | 4.6 |
| | | 0.3 | 0.5 | 0.0 | | | 0.3 | 9.1 | 8.5 | | | 0.3 | 6.7 | 5.7 |
| | | 0.5 | 0.0 | 0.0 | | | 0.5 | 9.9 | 8.7 | | | 0.5 | 7.9 | 7.1 |
| | | 0.7 | 1.3 | 0.0 | | | 0.7 | 7.3 | 6.5 | | | 0.7 | 8.3 | 7.6 |
| | | 0.9 | 0.0 | 0.0 | | | 0.9 | 8.1 | 7.1 | | | 0.9 | 8.1 | 8.0 |
| | 0.9 | 0.1 | 0.0 | 0.0 | | 0.9 | 0.1 | 0.5 | 0.0 | | 0.9 | 0.1 | 12.9 | 11.3 |
| | | 0.3 | 0.0 | 0.0 | | | 0.3 | 0.6 | 0.1 | | | 0.3 | 15.2 | 13.8 |
| | | 0.5 | 1.3 | 0.0 | | | 0.5 | 0.8 | 0.1 | | | 0.5 | 13.5 | 12.8 |
| | | 0.7 | 0.0 | 0.0 | | | 0.7 | 0.3 | 0.1 | | | 0.7 | 12.8 | 11.5 |
| | | 0.9 | 0.1 | 0.0 | | | 0.9 | 1.6 | 1.4 | | | 0.9 | 13.1 | 12.4 |
| Ave. | | | 0.4 | 0.1 | | | | 4.2 | 3.5 | | | | 5.6 | 5.0 |
| 15 | 0.1 | 0.1 | 2.6 | 2.3 | 25 | 0.1 | 0.1 | 3.3 | 2.7 | 100 | 0.1 | 0.1 | 1.8 | 1.8 |
| | | 0.3 | 3.5 | 3.0 | | | 0.3 | 2.4 | 2.2 | | | 0.3 | 1.5 | 1.3 |
| | | 0.5 | 1.8 | 1.6 | | | 0.5 | 1.7 | 1.4 | | | 0.5 | 0.7 | 0.7 |
| | | 0.7 | 1.4 | 1.0 | | | 0.7 | 0.7 | 0.6 | | | 0.7 | 0.2 | 0.1 |
| | | 0.9 | 1.2 | 1.2 | | | 0.9 | 0.3 | 0.3 | | | 0.9 | 0.0 | 0.0 |
| | 0.3 | 0.1 | 3.9 | 3.2 | | 0.3 | 0.1 | 3.7 | 3.1 | | 0.3 | 0.1 | 2.3 | 2.1 |
| | | 0.3 | 4.1 | 3.9 | | | 0.3 | 4.2 | 3.4 | | | 0.3 | 2.3 | 2.1 |
| | | 0.5 | 4.5 | 4.0 | | | 0.5 | 2.5 | 2.4 | | | 0.5 | 2.0 | 1.7 |
| | | 0.7 | 3.4 | 3.0 | | | 0.7 | 2.2 | 2.0 | | | 0.7 | 0.8 | 0.6 |
| | | 0.9 | 3.5 | 2.9 | | | 0.9 | 1.6 | 1.3 | | | 0.9 | 0.3 | 0.3 |
| | 0.5 | 0.1 | 7.5 | 7.0 | | 0.5 | 0.1 | 5.5 | 4.9 | | 0.5 | 0.1 | 3.9 | 3.4 |
| | | 0.3 | 7.7 | 7.5 | | | 0.3 | 5.0 | 4.5 | | | 0.3 | 3.4 | 3.0 |
| | | 0.5 | 9.8 | 9.2 | | | 0.5 | 5.3 | 4.5 | | | 0.5 | 3.6 | 3.1 |
| | | 0.7 | 6.4 | 6.1 | | | 0.7 | 3.1 | 2.6 | | | 0.7 | 2.0 | 1.8 |
| | | 0.9 | 6.1 | 6.1 | | | 0.9 | 3.8 | 3.4 | | | 0.9 | 1.4 | 1.4 |

**Table 9** *Continued*

| $N$ | $\tau$ | $R$ | *Swap and Insertion (%)* | *D&C phases (%)* | $N$ | $\tau$ | $R$ | *Swap and Insertion (%)* | *D&C phases (%)* | $N$ | $\tau$ | $R$ | *Swap and Insertion (%)* | *D&C phases (%)* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.7 | 0.1 | 0.3 | 0.3 | | 0.7 | 0.1 | 8.6 | 7.7 | | 0.7 | 0.1 | 5.0 | 4.0 |
| | | 0.3 | 0.6 | 0.1 | | | 0.3 | 9.5 | 8.6 | | | 0.3 | 5.1 | 4.7 |
| | | 0.5 | 0.7 | 0.1 | | | 0.5 | 11.6 | 10.5 | | | 0.5 | 5.8 | 5.3 |
| | | 0.7 | 1.7 | 1.0 | | | 0.7 | 7.4 | 6.9 | | | 0.7 | 6.7 | 5.4 |
| | | 0.9 | 0.1 | 0.1 | | | 0.9 | 8.6 | 8.1 | | | 0.9 | 6.8 | 5.9 |
| | 0.9 | 0.1 | 0.5 | 0.0 | | 0.9 | 0.1 | 1.3 | 0.6 | | 0.9 | 0.1 | 8.4 | 6.6 |
| | | 0.3 | 0.3 | 0.0 | | | 0.3 | 1.9 | 0.2 | | | 0.3 | 11.1 | 9.4 |
| | | 0.5 | 0.7 | 0.0 | | | 0.5 | 3.1 | 2.6 | | | 0.5 | 12.3 | 11.9 |
| | | 0.7 | 0.2 | 0.0 | | | 0.7 | 7.5 | 6.8 | | | 0.7 | 12.6 | 11.2 |
| | | 0.9 | 0.0 | 0.0 | | | 0.9 | 6.1 | 6.1 | | | 0.9 | 12.2 | 11.3 |
| Ave. | | | 2.9 | 2.6 | | | | 4.4 | 3.9 | | | | 4.5 | 4.0 |

[a]Average % Deviation from UB.

problem involving different performance criteria, such as those including finished goods inventory costs, may prove both interesting and useful. Fourth, revising the proposed ABC-based algorithm to solve multi-criteria single machine OAS problems is a possible direction for future research. Sixth, another open issue is to analyse whether the ABC-based algorithm offers any advantage over other meta-heuristic-based algorithms for the OAS problem in terms of time complexity or average case complexity. Finally, the stochastic single machine OAS problem with job release date constraints, while practical, is a more complex problem worthy of further studies.

*Acknowledgements*—The authors thank Professor Ceyda Oğuz for providing their benchmark problem set and solutions to us. We are also grateful to the anonymous referees for their useful and constructive comments and suggestions. This research was partially supported by the National Science Council of the Republic of China (Taiwan) under Contract nos. NSC 99-2410-H-182-023-MY2 and NSC 99-2628-E-027-003.

## References

Akkan C (1997). Finite-capacity scheduling-based planning for revenue-based capacity management. *European Journal of Operational Research* **100**(1): 170–179.

Cesaret B, Oğuz C and Sibel SF (2012). A tabu search algorithm for order acceptance and scheduling. *Computers & Operations Research* **39**(6): 1197–1205.

Charnsirisakskul K, Griffin PM and Keskinocak P (2004). Order selection and scheduling with leadtime flexibility. *IIE Transactions* **36**(7): 697–707.

Charnsirisakskul K, Griffin PM and Keskinocak P (2006). Pricing and scheduling decisions with leadtime flexibility. *European Journal of Operational Research* **171**(1): 153–169.

Chen YW, Lu YZ and Yang GK (2008). Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling. *International Journal of Advanced Manufacturing Technology* **36**(9): 959–968.

Davis L (1985). Applying adaptive algorithms to epistatic domains. In: Joshi A (ed). *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI'85)*. Vol. 1, San Francisco, pp 162–164.

Engels DW, Karger DR, Kolliopoulos SG, Sengupta S, Uma RN and Wein J (2003). Techniques for scheduling with rejection. *Journal of Algorithms* **49**(1): 175–191.

Gen M and Cheng R (1996). *Genetic Algorithms and Engineering Design*. John Wiley & Sons, Inc.: USA.

Ghosh JB (1997). Job selection in a heavily loaded shop. *Computers & Operations Research* **24**(2): 141–145.

Gordon VS and Strusevich VA (2009). Single machine scheduling and due date assignment with positionally dependent processing times. *European Journal of Operational Research* **198**(1): 57–62.

Guerrero HH and Kern GM (1988). How to more effectively accept and refuse orders. *Production and Inventory Management* **29**(4): 59–62.

Herbots J, Herroelen W and Leus R (2007). Dynamic order acceptance and capacity planning on a single bottleneck resource. *Naval Research Logistics* **54**(8): 874–889.

Kang F, Li J and Xu Q (2009). Structural inverse analysis by hybrid simplex artificial bee colony algorithms. *Computers & Structures* **87**(13): 861–870.

Karaboga D (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization*. Technical Report, Engineering Faculty, Computer Engineering Department, Erciyes University.

Karaboga D and Akay B (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation* **214**(1): 108–132.

Karaboga D and Ozturk C (2009). Neural networks training by artificial bee colony algorithm on pattern classification. *Neural Network World* **19**(10): 279–292.

Karaboga D and Ozturk C (2011). A novel clustering approach: Artificial bee colony (ABC) algorithm. *Applied Soft Computing* **11**(1): 652–657.

Keskinocak P and Tayur S (2004). Due date management policies. In: Simchi-Levi D, Wu SD and Shen ZJ (eds). *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*. Kluwer: Boston, pp 485–556.

Lee IS and Sung CS (2008). Single machine scheduling with outsourcing allowed. *International Journal of Production Economics* **111**(2): 623–634.

Lewis HF and Slotnick SA (2002). Multi-period job selection: Planning work loads to maximize profit. *Computers & Operations Research* **29**(8): 1081–1098.

Nobibon FT and Leus R (2011). Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment. *Computers & Operations Research* **38**(1): 367–378.

Oğuz C, Salman FS and Yalçin ZB (2010). Order acceptance and scheduling decisions in make-to-order systems. *International Journal of Production Economics* **125**(1): 200–211.

Omkar SN, Senthilnath J, Rahul K, Narayana Naik G and Gopalakrishnan S (2011). Artificial bee colony (ABC) for multi-objective design optimization of composite structures. *Applied Soft Computing* **11**(1): 489–499.

Özbakir L, Baykasoglu A and Tapkan P (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation* **215**(11): 3782–3795.

Rom WO and Slotnick SA (2009). Order acceptance using genetic algorithms. *Computers & Operations Research* **36**(6): 1758–1767.

Ruiz R and Stützle T (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal Operational Research* **177**(3): 2033–2049.

Singh A (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing* **9**(2): 631–652.

Slotnick SA (2011). Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research* **212**(1): 1–11.

Slotnick SA and Morton TE (2007). Order acceptance with weighted tardiness. *Computers & Operations Research* **34**(10): 3029–3042.

Stern HI and Avivi Z (1990). The selection and scheduling of textile orders with due dates. *European Journal of Operational Research* **44**(1): 11–16.

Xu C and Duan H (2010). Artificial bee colony (ABC) optimized edge potential function (EPF) approach to target recognition for low-altitude aircraft. *Pattern Recognition Letters* **31**(13): 1759–1772.

Yang B and Geunes J (2003). *Heuristic approaches for solving single resource scheduling problems with job-selection flexibility*. Technical Report, Department of Industrial and Systems Engineering, University of Florida.

Yang B and Geunes J (2007). A single resource scheduling problem with job-selection flexibility, tardiness costs and controllable processing times. *Computers & Industrial Engineering* **53**(3): 420–432.

Ying KC and Lin SW (2012). Unrelated parallel machine scheduling with sequence- and machine-dependent setup times and due date constraints. *International Journal of Innovative Computing, Information and Control* **8**(5): 3279–3297.

*Received April 2011;*
*accepted January 2012 after one revision*