Int J Adv Manuf Technol (2015) 77:2105–2115
DOI 10.1007/s00170-014-6614-9

ORIGINAL ARTICLE

An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date

Erdal Caniyilmaz · Betül Benli · Mehmet S. Ilkay

Received: 4 February 2014 / Accepted: 19 November 2014 / Published online: 30 November 2014

Abstract The problem dealt with in this study has job sequence-dependent setup times under suitable machine constraints and due date constraints. The performance criterion for this problem is to minimize the sum of makespan and total tardiness. In the literature, an application of ABC algorithm for the problems which includes all the properties that we have dealt has not been discussed. Because there is no appropriate test data, a real-life data was collected from a factory. In this study, a new approach has been proposed for the solution with meta-heuristics of unrelated parallel machine scheduling problems which is a combinatorial problem. This new neighborhood approach provides different machine assignments for every candidate job sequences. This approach is used by integrating into ABC and GA. To evaluate the performances of the algorithms, the real-life problem was solved by using ABC and GA algorithms under similar conditions. It was found that all jobs can be completed in two shifts without the need for a third shift. Computational results show that ABC algorithm has better performance than GA.

Keywords Scheduling · Artificial bee colony algorithm · Combinatorial artificial bee colony · Unrelated parallel machine scheduling · Meta-heuristics

E. Caniyilmaz · B. Benli
Industrial Engineering Department, Engineering Faculty, Erciyes University, 38039 Kayseri, Turkey

M. S. Ilkay (✉)
Department of Business, Faculty of Economics and Administrative Sciences, Erciyes University, 38039 Kayseri, Turkey
e-mail: ilkay@erciyes.edu.tr

1 Introduction

Today, scheduling problems are one of the most important problems that businesses have to resolve. Scheduling means to determine which jobs can be processed by which machines in what order within a certain period of time for purposes set, such as ensuring that products are delivered to customers when promised, more efficient use of production resources, and minimization of the total completion time, in a manufacturing environment [1].

In job scheduling, the best result is investigated according to a predetermined optimization criterion. The most studied optimization criterion is makespan, which means minimization of the maximum completion time of a schedule. Other criteria commonly used as a performance measure are total tardiness, flow time, and minimizing the number of jobs delayed. At the transitions between the successive jobs to be performed on the machines, setup time is needed for reasons such as process variation and change of raw materials used. In good scheduling, similar jobs fulfilled successively will allow reduction of the setup time incurred. In addition, jobs to be completed without delay for a promised due date and delivery to customers are important for businesses in terms of prestige and customer satisfaction.

Solving methods commonly used in scheduling problems are mathematical programming, the branch-and-bound algorithm, dynamic programming, heuristics, and meta-heuristics. Among the most frequently used meta-heuristics are tabu search, genetic algorithms, simulated annealing, and ant colony and particle swarm algorithms [2]. In recent years, the foraging behavior, learning, and knowledge sharing capabilities of honeybees have become an important research area in the literature in the field of swarm intelligence [3]. In solving scheduling problems, algorithms that are created by modeling the behavior of honeybees have been used widely.

There are different unrelated parallel machine scheduling problems with different characteristics in the literature. The problem dealt with in this study has job sequence-dependent setup times under suitable machine constraints and due date constraints. The minimization of the sum of maximum completion time and tardiness was studied by using real-life production data obtained from a business. An application of ABC algorithm for the problem which includes these properties is not studied in the current literature. A neighborhood approach is integrated to ABC and GA to generate neighborhood solutions to solve this problem.

In the second part of the study, a detailed review of the literature is given on unrelated parallel machine scheduling. In the third part, the basic artificial bee colony algorithm is introduced. In the fourth part, the proposed neighborhood approach for a combinatorial ABC algorithm is introduced in detail. In the fifth part, a real-life implementation is shown. In addition, problem description and assumptions, data preparation, and implementation details of the ABC algorithm proposed are given in this part. In the sixth section, the experimental results obtained in the study are presented and interpreted. In the last part, an overall evaluation of the study is conducted.

2 Literature review

Studies in the literature on the problem of unrelated parallel machine scheduling can be classified as those dealing with exact methods and approximate methods according to solution methods. Exact methods guarantee the optimal solution for small-sized problems. However, it has been shown that these methods are not effective in cases that have large data size and constraints; they cannot even find a feasible solution due to increased complexity and computation time [4].

The branch-and-bound algorithm reaches a solution by eliminating several candidate solutions according to certain restrictive conditions [5]. Shim [6] showed that the subproblems generated by the proposed effective branching approach are smaller than those by other previous methods. It has been shown that there is no need to check dominance rules in unpromising nodes. In this way, the required time to find optimal solutions can be reduced. Gharehgozli et al. [7] proposed a fuzzy mixed-integer goal programming model for an unrelated parallel machine problem with sequence-dependent setup times and release dates to minimize the total weighted flow time and total weighted tardiness simultaneously. They considered the problem under the assumption of fuzzy processing times. The performance of the method is evaluated by producing small-sized random samples. The optimal solution can be obtained in a reasonable time for up to seven jobs and three machines by using Lingo software.

Approximate methods have been developed that find solutions close to the optimal solution in a reasonable time to solve problems for which it is not possible to find the optimal solution by using exact methods in an effective way. Approximate methods can be divided into three groups: heuristics, approximation algorithms, and meta-heuristics. Heuristic algorithms are designed according to the characteristics of the problem dealt with and/or a specific objective function. Local optimum solutions are then found [8]. Approximation algorithms are theoretically valuable but are not used much in practice. Inability to find a solution close enough to the optimal solution, complexity, and difficulty in implementation are the reasons for this. Therefore, lots of heuristics have been developed and used in practice [9].

Kuei and Wei [10] considered a scheduling problem with release dates to minimize makespan, total weighted completion time, and total weighted tardiness individually. They developed several mixed-integer programming models to find the optimal solutions for small-sized problems and proposed several dispatching rules to find solutions close to the optimum quickly for large-sized problems. The results show that the proposed dispatching rules outperform other previous dispatching rules. Low et al. [11] considered a two-stage hybrid flow shop scheduling problem that has alternative parallel machines working alongside at each stage. It is assumed that the operation can be partially substituted by other machines and the machines are named according to their technical functions, so they can perform jobs belonging to a certain type. Researchers combined the modified Johnson's rule and the First-Fit rule to minimize makespan. Peyro and Ruiz [12] considered a problem in which it is desirable to use only a subset of the available parallel machines. They proposed a three-phased solution procedure to minimize makespan. A procedure was developed to rank machines for the first phase and make a selection of the machines for the second phase. In the third phase, the reduced model, which is obtained by using solutions that are produced in the first and second phases, is solved with CPLEX. They also proposed, instead of CPLEX, a fast local search algorithm and an iterated greedy search method for the third phase.

Approximation algorithms can be considered as a type of heuristic with performance guarantee. A ρ-approximation algorithm is a polynomial-time algorithm that finds a solution within ρ times the optimal value. ρ refers to the performance guarantee of the algorithm. A polynomial-time approximation scheme (PTAS) is a type of polynomial-time approximation algorithm with performance guarantee $1+\varepsilon$ for all $\varepsilon>0$ [13] [14]. Muratore et al. [15] considered a scheduling problem in which jobs can be performed on a certain subset of the machines, where the machine sets are nested. They developed a polynomial-time approximation scheme to minimize makespan. The results show that it guarantees a performance with $(\alpha+9)/\alpha\leq 1+\varepsilon$, where α is a constant positive integer.

Svensson [16] proposed a polynomial-time algorithm for the restricted assignment problem to minimize makespan. The algorithm finds solutions that approximates within a factor $33=17+\varepsilon\approx1,9412$ for $\varepsilon>0$.

A meta-heuristic algorithm is a general-purpose iterative search process which can also guide subordinate heuristics [17]. The ability to avoid local best solutions is an important feature of meta-heuristics. Moghaddam et al. [18] considered a problem in which some precedence relations exist between jobs and sequence-dependent setup times. The jobs have non-common due dates and release times. They proposed a genetic algorithm to solve the bi-objective scheduling problem, where the objectives are the number of tardy jobs and the total completion time of the jobs. The results show that the proposed algorithm is effective for small- and large-sized problems. Chyu and Cheng [19] proposed a hybrid pareto converging genetic algorithm for a scheduling problem with job sequence- and machine-dependent setup times to minimize total weighted flow time and total weighted tardiness simultaneously. The proposed algorithm has better results compared to the multi-objective simulated annealing algorithm. Logendran et al. [20] addressed a problem in which the dynamic release of jobs and dynamic availability of machines are assumed to reflect real-life problems better. They developed six different search algorithms based on tabu search for a problem with job sequence-dependent setup times to minimize the weighted tardiness of jobs. Bozorgirad and Logendran [4] considered a sequence-dependent group scheduling problem which needs setup time between the groups. The job release times and the machine availability times are considered to be dynamic to comply with real problems. They developed meta-heuristic algorithms based on tabu search to optimize both the total weighted completion times and total weighted tardiness of jobs simultaneously. These two criteria are given a weight that reflects the importance of the producer and customers. The proposed algorithm produces solutions close to the optimum found by CPLEX in a short time. Chen and Chen [21] considered a problem with sequence-dependent setup times. They developed a solution procedure which integrates the virtues of the variable neighborhood descent approach and tabu search to minimize the weighted number of tardy jobs. Kim et al. [22] proposed a simulated annealing algorithm for a problem with job sequence-dependent setup time to minimize total tardiness. Chen [23] addressed a problem with sequence-dependent and machine-dependent setup times and proposed a simulated annealing approach to minimize total tardiness. Computational results show that the proposed heuristic obtains better results than a random descent heuristic by improving the initial solutions effectively. Wang et al. [24] considered a scheduling problem with splitting jobs. They developed a hybrid differential evolution approach which includes a new crossover method and a new mutation method for global search according to the job splitting constraint and a local search method for better performance. Lin et al. [25] developed an ant colony algorithm that integrates new methods of producing the initial solution, local search, and selection of machine to minimize total weighted tardiness. The proposed procedure improved the performance of the algorithm.

The ABC algorithm is a swarm-intelligence-based algorithm that was inspired by the behavior of foraging bees. Rodriguez et al. [26] proposed an ABC algorithm to minimize total weighted completion times. Jobs have to be processed on a set of unrelated parallel machines. A local search method to enhance the exploitation capability of the basic ABC algorithm and a neighborhood operator based on the iterated greedy procedure were combined. They compared ABC to different approaches. Computational results showed that this algorithm represents a competitive alternative to the existing methods. Ying and Lin [27] discussed a problem with sequence- and machine-dependent setup times under due date constraints to minimize total tardiness. The results showed that the proposed ABC algorithm outperforms existing algorithms for most benchmark problems. Lin and Ying [28] presented a hybrid artificial bee colony algorithm for the problem with machine-dependent and job sequence-dependent setup times to minimize makespan. Computational results indicate that the proposed algorithm outperforms existing algorithms.

There are different neighborhood approaches in the literature to solve this type of problems using meta-heuristics. Rodriguez et al. [26] proposed a neighborhood operator based on the destructive-constructive procedure used in iterated greedy algorithms. Lin and Ying [28] also presented a destructive-constructive procedure to generate neighborhood solutions. This method is based on setting partial sequence of solutions by removing and reinserting until a complete neighborhood solution is reconstructed. Li et al. [32] proposed a discrete ABC algorithm for flexible job shop scheduling problem. They described two types of neighborhood structures in the discrete ABC algorithm. Each solution contains two vectors; the routing vector and the scheduling vector. In the first one, neighboring solutions are generated from the routing vector by considering some workload rules. In the second one, neighboring solutions are generated from scheduling vector by applying some traditional swap, insert, and reverse moves. Chen and Chen [21] proposed a hybrid meta-heuristic which integrates the principles of the variable neighborhood descent approach and tabu search. Four reduced-size neighborhood structures using swap and insert methods are proposed in the meta-heuristic. Kim et al. [22] generated neighbor solutions considering the lots rather than just items. The proposed simulated annealing approach uses six job or item rearranging techniques: lot interchange, lot insert, lot merge, lot split, item interchange, and item insert.

Studies in the literature mentioned above are summarized in Table 1.

3 The basic ABC algorithm

In nature, there are many swarms which interact with each other with the purposes of finding food, finding a good mate, providing a defense against attackers, and so on. The ABC algorithm is a swarm-intelligence-based algorithm that models the intelligent foraging behavior of real honey bee colonies [29]. The foraging behavior of bee colonies has three basic components: employed bees, unemployed bees, and food sources. The quality of a food source is determined according to the closeness to the hive, richness of food sources, and similar characteristics. Employed bees are bees which gather information by hovering around specific food sources and sharing this information with the unemployed bees in the hive. Onlooker and scout bees are called unemployed bees. Onlooker bees investigate solutions to the neighboring food sources according to the information brought by employed bees. Scout bees search randomly for new food sources in the area. Employed bees take on the role of regional research, while scout bees take on the role of global research.

The ABC algorithm is a quite simple, flexible, and robust algorithm. It has three control parameters including colony size, limit or abandonment criteria of the food source, and maximum cycle number as a stopping criterion. The ABC algorithm has fewer control parameters than other algorithms. The values of the control parameters play an important role in the performance of the algorithms. The ABC carries out global exploration and exploitation in a balanced way by using greedy strategies and neighbor search mechanisms [30]. The basic ABC algorithm's detailed pseudo-code is given in Table 2.

The calculations of the equations referred to in Table 2 are as follows:

$$x_{ij} = LB_j + \left(UB_j - LB_j\right) * r(0,1)\, i = 1,\ldots KS\, j = 1,\ldots N \tag{1}$$

With x_{ij} equality, food sources are generated randomly between the values of the parameters' lower and upper limits. r is a randomly generated number according to a uniform distribution in the range of [0–1].

$$y_{ij} = x_{ij} + \left(x_{ij} - x_{kj}\right)^{*} r(-1,1) \tag{2}$$

By Eq. (2), candidate solutions are created. j is an integer randomly generated to express one of the parameters of the food sources. k is an integer randomly generated and refers to one of the food sources. r is a randomly generated number according to a uniform distribution in the range of [−1,1]. Solution x_k is a solution randomly selected in the neighborhood of the solution x_i. If the difference between x_{ij} and x_{kj} decreases, the amount of change in x_{ij} parameters will also be reduced. Thus, the local optimal solution closer to the reduction of the amount of change is provided adaptively.

$$g_i = \left\{ \begin{array}{c} \dfrac{1}{1+f_i}, f_i \geq 0 \\ 1 + abs(f_i), f_i < 0 \end{array} \right\} \tag{3}$$

By Eq. (3), the fitness value of food sources is calculated. Herein, f_i is the value of the objective function of the corresponding food source.

An onlooker bee chooses a food source depending on the probability of appropriating the nectar amount the food source has. In the ABC algorithm, the probabilistic selection process is made using fitness values corresponding to the amount of nectar.

$$p_i = \frac{g_i}{\sum_{j=1}^{KB} g_j} \tag{4}$$

The ratio of the fitness value of a source to the total fitness values of all the sources shows the probability of a source being selected relative to the other sources as shown in Eq. (4).

4 The proposed neighborhood approach for a combinatorial ABC algorithm

The ABC originally was developed for continuous optimization problems. It can be also used for combinatorial optimization problems by some modifications. There are different neighborhood approaches in the literature to solve these types of problems using meta-heuristics. In this study, an alternative neighborhood approach is developed and integrated into ABC and GA to generate neighborhood solutions. It is based on providing random machine assignments for each candidate job sequence.

The steps of the proposed neighborhood approach with ABC algorithm are explained below. First, the control parameters of the algorithm are determined. The dimension of the problem is the number of jobs that will be scheduled. c is the parameter which shows the number of machine assignments for each candidate job sequence.

Because scheduling is a ranking problem, the random permutation function that creates job sequence randomly

Table 1 Studies in the literature related to unrelated parallel machines

Article	Objective function	Job sequence-dependent setup time	Machine sequence-dependent setup time	Due date	Release time	Weight	Precedence	Job splitting	Processing set restrictions
[6]	$\sum wjFj$			✓		✓			
[7]	$\sum wjFj + \sum wjTj$	✓		✓	✓	✓			
[10]	$C_{\max}$, $\sum wjTj$, $\sum w_jC_j$			✓	✓	✓			
[11]	$C_{\max}$								✓
[12]	$C_{\max}$								✓
[15]	$C_{\max}$								✓
[16]	$C_{\max}$								✓
[18]	$\sum C_j + \sum U_j$	✓	✓	✓	✓		✓		
[19]	$\sum w_jC_j + \sum w_jT_j$	✓	✓	✓					
[20]	$\sum w_jT_j$	✓	✓	✓	✓	✓			✓
[4]	$\sum w_jC_j + \sum w_jT_j$	✓	✓	✓	✓	✓			✓
[21]	$\sum w_jU_j$	✓		✓		✓	✓		
[22]	$\sum T_j$	✓	✓	✓					
[23]	$\sum T_j$	✓	✓	✓					
[24]	$C_{\max}$							✓	✓
[25]	$\sum w_jT_j$			✓		✓			
[26]	$\sum w_jC_j$			✓		✓			
[27]	$\sum T_j$	✓	✓	✓					
[28]	$C_{\max}$	✓	✓						

was used instead of Eq. (1) which is used to generate random food sources in a continuous space.

Figure 1 represents an example of generating a new job sequence. It is used instead of Eq. (2) in the original ABC algorithm. Assume that we have an $\overrightarrow{x_i}$ candidate job sequence for a scheduling problem consisting of seven jobs as follows. To produce a new job sequence from this candidate job sequence, the *j*th parameter of $\overrightarrow{x_i}$ job sequence determined randomly is replaced by the *j*th parameter of $\overrightarrow{x_k}$ candidate job sequence which again is randomly determined.

For the calculation of the value of objective function, a method was proposed for assigning machines to each job sequence. In this context, a c parameter was defined and random machine assignments are performed c times for each candidate job sequence generated. The job sequence and machine assignment providing the best objective function value is kept in memory. Fitness value is calculated according to the objective function value.

In Fig. 2, a small problem is presented to illustrate the application of this method. Assume that we have seven jobs and three machines. Suitable machine list for jobs has been given in Table 3.

Random machine assignments are performed c times for the job sequence given in Fig. 2. This operation is repeated for each candidate job sequence. The job sequence and machine assignment providing the best objective function value is kept in memory.

The objective function's aim is to minimize the sum of makespan and total tardiness. In this study, a large coefficient value is given for the total tardiness to ensure that the total tardiness is zero.

The objective function values of the candidate job sequences are kept in a vector to compare with the objection function values of the neighbor solutions that will be produced next.

The next phases of the algorithm are repeated until the stopping criterion:

(a) The fitness values of new solutions are calculated according to their objective function value. A greedy selection is applied between the current and new solution and the best one is picked.
(b) The probability values p_i are calculated based on fitness values. A new solution is generated from the candidate solution which is chosen according to its probability value, and the greedy selection process is applied between new and chosen.
(c) The counters are checked and the generation of new solutions is ended for those that reach the limit value. These solutions are replaced with a candidate solution that is produced randomly.

The detailed pseudo-code of the proposed algorithm is given in Table 4.

Table 2 The basic ABC algorithm's detailed pseudo-code [29]

```
Step 1: Generate initial values for the food sources $x_{ij}$ randomly.
        ($i=1, \ldots, KS$   $j=1, \ldots, N$)                         (1)
Step 2: Reset counters of the unimproved solution and cycle number
        ($L_i = 0$)           ($C=0$)
Step 3: Calculate fitness values for each food source $f(x_i)$
Step 4: REPEAT
    a) For each food source $x_i$:
        i.   Generate a new food source, $y_i$, for the employed bee of the solution,
             $x_i \rightarrow y_i$                                      (2)
        ii.  Compute fitness value for $f(y_i)$.                       (3)
        iii. Apply the greedy selection process between $y_i$ and $x_i$ and choose the better one. If $x_i$ solution
             is unimproved, increase the value of the unimproved counter by one,
             $L_i = L_i + 1$,
             If solution is improved, reset the counter, $L_i = 0$.
    b) Calculate probability values $p_i$, based on fitness values used by onlooker bees when they choose
       food sources.                                                   (4)
    c) For each onlooker bee:
        i.   Choose a food source according to $p_i$ value and produce a new food source for onlooker bees.
             (5)
        ii.  Apply the greedy selection process between $y_i$ and $x_i$ and select the better one. If $x_i$ solution is
             unimproved, increase the value of the unimproved counter by one,
             $L_i = L_i + 1$,
             If solution is improved, reset the counter, $L_i = 0$.
    d) For each food source $x_i$
       If ($L_i$ >limit) THEN
       Replace $x_i$ with a new solution randomly generated for scout bee
    e) Memorize the best solution so far
    f) $C=C+1$
Step 5: UNTIL {$C$=maximum cycle number}
```

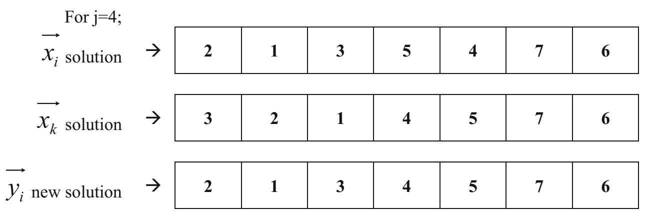

Fig. 1 New solution generation

5 Real-life implementation

The study was conducted using real-life data obtained from a time study in the quilting work center in a large factory which produces sofas, armchairs, and bed bases in Kayseri, Turkey. Quilting machines are machines which texture pattern on

		Solution representation						
1	Job sequence	2	1	3	4	5	7	6
	Machines assigned	2	1	2	3	1	1	2

		Solution representation						
2	Job sequence	2	1	3	4	5	7	6
	Machines assigned	3	1	2	1	2	1	2

. . .
. . .
. . .

		Solution representation						
c	Job sequence	2	1	3	4	5	7	6
	Machines assigned	2	1	2	1	1	1	2

Fig. 2 An example of solution representation

combined fabric, foam, and fiber, each of which has different characteristics. There are seven quilting machines which run in parallel at the factory. These machines are considered as unrelated parallel machines. The semi-finished goods which are completed in this work center are sent to the next stage in the tailor work center.

The factory operates three shifts a day and each shift lasts 7.5 h. However, in the third shift, just three machines continue to work until the completion of production. A weekly production schedule of the factory is considered for this study. To assess the performance of the proposed method, the results obtained is compared with the current schedule results and GA results. When weekly jobs are scheduled according to the sequence specified by the business, all jobs are completed in 5380 min, that is, makespan is 5380 min.

5.1 Problem description and assumptions

The problem dealt with in this study has job sequence-dependent setup times under suitable machine constraints and due date constraints. The performance criterion for this problem is to minimize the sum of makespan and total tardiness. This problem can be expressed as $R|S_{ij}, M_j|C_{\max}+\sum T_j$. An application of ABC algorithm for this problem has not been studied in the current literature.

It is assumed that the machines are continuously available and the jobs do not have release times. The operation is not interrupted until completion of the job. There are no weight and precedence relations among jobs.

In this study, 131 jobs, which must be to be completed in 1 week according to the factory's weekly plan, are discussed. It is not possible to assign every job to every machine due to features such as the width, length, thickness, and pattern of fabric. In this regard, the 131 jobs dealt with were divided into 31 groups according to the use of similar raw materials, raw material size, and pattern of fabric. Therefore, the machine sets for each job belonging to a group are determined. The job groups and machine sets for each job, and the processing times on machines are given in the following link (http://iibf.erciyes.edu.tr/ilkay/data.pdf). Setup time is required for jobs belonging to different groups when they are performed on the same machine consecutively. The setup times for each transition among groups are calculated. If two jobs which belong to the same group are processed on the same machine consecutively, setup time is not required. The setup times among groups are given also in the link above.

Table 3 Jobs and suitable machines

Jobs	Suitable machines
1	1, 2
2	1, 2, 3
3	2
4	2, 3
5	1, 2
6	1, 2, 3
7	1

6 Experimental results

To obtain good solutions for the scheduling problem, the proposed neighborhood approach is integrated into ABC and GA. Deciding on the parameter values of the algorithm for this problem emerges as a distinct problem. In this study, to evaluate the results of different parameter alternatives, the value of a parameter was changed and the others kept constant. The proximity of the results shows that the algorithm is robust.

The meta-heuristics were run with the different parameter values given in Tables 5 and 6.

In Table 5, the first of the three values given for the colony size represents the initial value of colony size; the second is the amount of increase in the colony size, and the third is the end value. A value ($CS*D$) obtained by multiplying colony size (CS) and dimension of the problem (D) is proposed for the limit value [31]. The last column in Table 5 shows the objective function values obtained by running the proposed ABC algorithm with the different parameter values. As a result, the best value of the objective function value was obtained as 5025 which means that all jobs can be completed in two shifts without the need for a third shift. Not having a large value for the objective function indicates that

Table 4 Detailed pseudo-code of the proposed algorithm

```
Step 1: Generate random permutations of candidate job sequence that represent food sources
Step 2: Reset counters of the unimproved solution and cycle number (Li = 0)    (C=0)
Step 3: Calculate fitness values for each food  source  $f(x_i)$  (*)
        (*) function_1(all candidate job sequences)
                function _1.1 (one of candidate job sequences)
                        Perform random machine assignment for this candidate job sequence by the number of
                        given values of parameter c, and calculate objective function value for this candidate
                        job sequence
                end_1.1;
                        return_1.1    the best objective function value among all machine assignments and the
                        best machine assignment;
            end_1;
                return_1 the objective function values of all candidate job sequences
Step 4: REPEAT
                a)  For each food source  $x_i$:
                    i.    Generate a new food source, $y_i$ , for the employed bee of the solution,
                          $x_i \rightarrow y_i$                                   (2)
                    ii.   Compute fitness value for $f(y_i)$.            (3)
                    iii.  Apply the greedy selection process between $y_i$ and $x_i$ and choose the better one. If $x_i$
                          solution is unimproved, increase the value of unimproved counter by one,
                                $L_i = L_i + 1$,
                          If solution is improved reset the counter, $L_i = 0$.
                b)  Calculate probability values $p_i$, based on fitness values used by onlooker bees' when they
                    choose food sources.                                  (4)
                c)  For each onlooker bee:
                    i.    Choose a food source according to $p_i$ value; produce a new food source for onlooker
                          bees.                                           (5)
                    ii.   Apply the greedy selection process between $y_i$ and $x_i$ and select the better ones. If $x_i$
                          solution is unimproved, increase the value of unimproved counter by one,
                                $L_i = L_i + 1$,
                          If solution is improved reset the counter, $L_i = 0$.
                d)  For each food source $x_i$
                    If ($L_i$ >limit) THEN
                          Replace $x_i$  with a new solution randomly generated for scout bee
                e)  Memorize the best solution so far
                f)  C=C+1
Step 5: UNTIL {C=maximum cycle number}
```

Table 5 Parameter experiments and results of ABC Algorithm

Number	c	Cycle time	Colony size (CS)	Limit	Objective function value (min)
	For 3 shifts				
1	50	350	50:50:350	0.5*CS*D	5059
2	50	500	50:50:350	0.5*CS*D	5075
3	50	750	50:50:350	0.5*CS*D	5063
4	50	1000	50:50:350	0.5*CS*D	5051
5	30	500	50:50:350	0.5*CS*D	5030
6	50	500	50:50:350	0.5*CS*D	5075
7	100	500	50:50:350	0.5*CS*D	5043
8	150	500	50:50:350	0.5*CS*D	5025
	For 2 shifts				
9	150	500	100:100:400	0.5*CS*D	5100
10	200	500	100:100:400	0.5*CS*D	5051
11	250	500	100:100:400	0.5*CS*D	5049
12	500	500	100:100:400	0.5*CS*D	5037
13	250	750	300:100:600	0.5*CS*D	5063
14	250	1000	300:100:600	0.5*CS*D	5071

there is no delay. Therefore, 5025 represents only the value of makespan.

The proposed neighborhood approach is integrated to GA and evaluated its performance with GA. As seen in Table 6, the same values were given for *c* and iteration value to compare performances. The maximum value of colony size value given in ABC was given for the population size values in GA. Mutation rate is 0.1 and crossover rate is 0.5 for each experiment. As seen in the last column in Table 6, the best value of the objective function was obtained as 5134.

The results show that ABC has better performance in terms of the objective function value. So, the firm will gain 355 min with the schedule obtained by the proposed ABC algorithm compared to the schedule that the factory suggests. The best job sequence is shown in Table 7, and which jobs assigned to which machines is shown in Table 8.

Table 6 Parameter experiments and results of GA

Number	*c*	Iteration	Population size	Mutation rate	Crossover rate	Objective function value (min)
	For 3 shifts					
1	50	350	350	0.1	0.5	5409
2	50	500	350	0.1	0.5	5134
3	50	750	350	0.1	0.5	5289
4	50	1000	350	0.1	0.5	5206
5	30	500	350	0.1	0.5	5300
6	50	500	350	0.1	0.5	5378
7	100	500	350	0.1	0.5	5421
8	150	500	350	0.1	0.5	5201
	For 2 shifts					
9	150	500	400	0.1	0.5	5303
10	200	500	400	0.1	0.5	5230
11	250	500	400	0.1	0.5	5294
12	500	500	400	0.1	0.5	5238
13	250	750	600	0.1	0.5	5273
14	250	1000	600	0.1	0.5	5276

Table 7 The best job sequence

Job sequence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Jobs	59	97	93	99	35	131	36	119	14	118	115	84	21	6	22	57	123	11	120	86	55	70	77	78	82
Job sequence	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Jobs	125	75	53	19	10	27	26	76	114	102	81	4	122	24	111	42	2	20	5	112	98	51	92	33	124
Job sequence	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
Jobs	46	90	116	1	106	30	28	130	44	89	9	18	94	129	85	34	47	72	121	108	79	60	43	39	67
Job sequence	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
Jobs	3	16	56	61	45	110	88	103	117	62	68	15	17	32	40	48	113	31	105	54	126	73	74	7	49
Job sequence	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
Jobs	66	29	87	13	128	58	37	38	96	8	52	69	65	12	83	100	101	109	127	64	25	91	41	50	104
Job sequence	126	127	128	129	130	131																			
Jobs	107	71	63	80	95	23																			

7 Conclusion

To create a schedule for a certain period of time that defines different purposes, like which jobs should be assigned to which machines and in which sequence, is very important for firms in terms of cost, efficiency, reliability, etc. This topic is a difficult task for decision-makers because of the size of data and different constraints in application. To solve such difficult problems, general-purpose, flexible, and simple algorithms inspired by nature have been developed in recent years.

Recently, the ABC algorithm, which was developed by modeling honey bees' intelligent behavior in foraging, is in great interest by researchers. The reason for this ABC algorithm is simpler, more flexible, and has less control parameters when compared to other meta-heuristics.

The problem dealt with in this study has job sequence-dependent setup times under suitable machine constraints and due date constraints. The performance criterion for this problem is to minimize the sum of makespan and total tardiness. In the literature, an application of ABC algorithm for the problems which includes all the properties that we have dealt has not been discussed. Because there is no appropriate test data, real-life data was collected from a factory.

In the study, a new approach has been proposed for the solution with meta-heuristics of unrelated parallel machine scheduling problems which is a combinatorial problem. This new neighborhood approach provides different machine assignments for every candidate job sequences. This approach is used by integrating into ABC and GA. To evaluate the performances of the algorithms, the real-life problem was solved by using ABC and GA algorithms under similar conditions. It was found that all jobs can be completed in two shifts without the need for a third shift. Computational results show that ABC algorithm has better performance than GA.

As a future study, this approach can be applied to other type of scheduling problems such as flow shop and job shop.

Table 8 Job-machine assignments

Machines	Jobs assigned to machines																									
1	59	36	125	111	112	1	72	16	32	113	7	87	13	107	80											
2	93	123	120	77	27	42	2	20	46	85	34	68	15	128	58	52	63	23								
3	99	119	57	55	78	75	53	24	98	51	90	30	56	62	40	73	37	65	91							
4	131	14	118	86	26	76	5	92	28	79	43	105	54	8	101	25										
5	115	6	22	81	122	130	9	94	129	108	39	67	110	103	96	100	41	50	95							
6	84	21	82	33	124	47	121	61	88	117	74	49	66	29	38	109	64									
7	97	35	11	70	19	10	114	102	4	116	106	44	89	18	60	3	45	17	48	31	126	69	12	83	127	104	71

References

1. Pinedo M (2012) Scheduling: theory, algorithms, and systems. Springer Science Business Media, New York
2. Allahverdi A, Ng C, Cheng T, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187(3):985–1032
3. Akyol S, Alataş B (2012) Güncel Sürü Zekası Optamizasyon Algoritmaları. Nevşehir Universitesi Fen Bilimleri Enstitusu Dergisi 1(1):36–50
4. Bozorgirad MA, Logendran R (2012) Sequence-dependent group scheduling problem on unrelated-parallel machines. Expert Syst Appl 39(10):9021–9030
5. Eren T, Güner E (2002) Tek ve paralel makineli problemlerde çok olcutlu cizelgeleme problemleri için bir literatur taramasi. Gazi Universitesi Muhendislik Mimarlik Dergisi 17(4):37–69
6. Shim SO (2009) Generating subproblems in branch and bound algorithms for parallel machines scheduling problem. Comput Ind Eng 57(3):1150–1153. doi:10.1016/j.cie.2009.02.013
7. Gharehgozli A, Tavakkoli-Moghaddam R, Zaerpour N (2009) A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. Robot Cim-Int Manuf 25(4):853–859
8. Ponnambalam S, Aravindan P, Chandrasekaran S (2001) Constructive and improvement flow shop scheduling heuristics: an extensive evaluation. Prod Plan Control 12(4):335–344
9. Fan L, Zhang F, Gongming W, Zhiyong L (2009) An effective scheduling algorithm for linear makespan minimization on unrelated parallel machines. International Conference on High Performance Computing, Kochi,16-19 Dec 2009, pp 40–49
10. Lin YK, Lin CW (2013) Dispatching rules for unrelated parallel machine scheduling with release dates. Int J Adv Manuf Tech 67(1–4):269–279
11. Low CY, Hsu CJ, Su CT (2008) A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines. Comput Oper Res 35(3):845–853
12. Fanjul-Peyro L, Ruiz R (2012) Scheduling unrelated parallel machines with optional machines and jobs selection. Comput Oper Res 39(7):1745–1753
13. Li K, Yang S (2009) Non-identical parallel-machine scheduling research with minimizing total weighted completion times: models, relaxations and algorithms. Appl Math Model 33(4):2145–2158
14. Güler A (2008) Tamsayılı Programlama Problemleri İçin Garanti Değerli Algoritmalar, MSc Thesis, Turkey
15. Muratore G, Schwarz UM, Woeginger GJ (2010) Parallel machine scheduling with nested job assignment restrictions. Oper Res Lett 38(1):47–50
16. Svensson O (2012) Santa Claus schedules jobs on unrelated machines. Siam J Comput 41(5):1318–1341
17. Osman IH, Laporte G (1996) Metaheuristics: a bibliography. Ann Oper Res 63(5):511–623
18. Tavakkoli-Moghaddam R, Taheri F, Bazzazi M, Izadi M, Sassani F (2009) Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. Comput Oper Res 36(12):3224–3230
19. Chyu C-C, Chang W-S (2010) A Pareto evolutionary algorithm approach to bi-objective unrelated parallel machine scheduling problems. Int J Adv Manuf Technol 49(5–8):697–708
20. Logendran R, McDonell B, Smucker B (2007) Scheduling unrelated parallel machines with sequence-dependent setups. Comput Oper Res 34(11):3420–3438
21. Chen C-L, Chen C-L (2009) Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. Int J Adv Manuf Technol 43(1–2):161–169
22. Kim D-W, Kim K-H, Jang W, Frank Chen F (2002) Unrelated parallel machine scheduling with setup times using simulated annealing. Robot Cim-Int Manuf 18(3):223–231
23. Chen J-F (2009) Scheduling on unrelated parallel machines with sequence-and machine-dependent setup times and due-date constraints. Int J Adv Manuf Technol 44(11–12):1204–1212
24. Wang W-L, Wang H-Y, Zhao Y-W, Zhang L-P, Xu X-L (2012) Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm. Comput Oper Res 40(5):1196–1206
25. Lin CW, Lin YK, Hsieh HT (2013) Ant colony optimization for unrelated parallel machine scheduling. Int J Adv Manuf Tech 67(1–4):35–45
26. Rodriguez FJ, García-Martínez C, Blum C, Lozano M (2012) An artificial bee colony algorithm for the unrelated parallel machines scheduling problem. In: Parallel Problem Solving from Nature-PPSN XII. Springer, pp 143–152
27. Ying KC, Lin SW (2012) Unrelated parallel machine scheduling with sequence- and machine-dependent setup times and due date constraints. Int J Innov Comput I 8(5A):3279–3297
28. Lin SW, Ying KC (2014) ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. Comput Oper Res 51:172–181
29. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Techn Rep TR06, Erciyes Univ Press, Erciyes
30. Karaboğa D (2004) Yapay zeka optimizasyon algoritmaları. Atlas Yayin Dagitim, Istanbul, pp 75–112
31. Akay B (2009) Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi Algoritmasının Performans Analizi. PhD Thesis, Turkey
32. Li JQ, Pan QK, Tasgetiren MF (2014) A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. Appl Math Model 38:2145–2158