Evolutionary Intelligence (2020) 13:415–425
https://doi.org/10.1007/s12065-019-00305-7

RESEARCH PAPER

Comparisons of metaheuristic algorithms for unrelated parallel machine weighted earliness/tardiness scheduling problems

Oğuzhan Ahmet Arık[1]

Received: 14 March 2019 / Revised: 3 July 2019 / Accepted: 10 October 2019 / Published online: 17 October 2019

Abstract
This paper investigates unrelated parallel machine scheduling problems where the objectives are to minimize total weighted sum of earliness/tardiness costs. Three different metaheuristic algorithms are compared with others to determine what kind (swarm intelligence based, evolutionary or single solution) of metaheuristics is effective to solve these problems. In this study, artificial bee colony (ABC), genetic algorithm and simulated annealing algorithm are chosen as swarm intelligence based algorithm, evolutionary algorithm and single solution algorithm. All proposed algorithms are created without modification in order to determine effectiveness of these metaheuristics. Experimental results show that ABC outperforms its opponents in view of solution quality as swarm intelligence based metaheuristic algorithm.

Keywords Artificial bee colony · Genetic algorithm · Simulated annealing · Earliness/tardiness · Parallel machine · Scheduling

1 Introduction

Weighted earliness/tardiness scheduling problems with common due date are significant for just-in-time (JIT) philosophy. Each company that accepts JIT philosophy wants to complete jobs on their due dates. This is significant for profitability of the company because each early or tardy job has a cost. If a job is completed before its due date, then this job is an early job and there are some costs for the company. For instance, the company has to stock this job until its due date, then the company will pay for stocking cost and/or insurance cost. If a job is completed after its due date then this job is a tardy job and there are some costs because of tardiness. For instance, the company will have to delay the shipment or the company will have to lose some of its earnings because of the contract signed with the costumer. Common due dates are used in order to determine a shipment date for the jobs that will be transshipped simultaneously in a common transportation vehicle (cargo, air-cargo, truck etc.). Most of costumers of the company are from different countries and orders of a costumer must be completed until shipment date of transportation vehicle (cargo, air-cargo, truck etc.). If some of orders in a shipment are completed before shipment date, the company has to stock these ones until the shipment date and there will be a holding, stocking or assuring costs for these stocks. On the contrary, if some of orders in a shipment completed after the shipment date, this situation leads a penalty cost such as loss of reputation, reshipment of tardy jobs and perhaps loss of costumer. Due dates or common due date can be determined by internal or external costumers. The internal costumer may want to have all jobs on a certain date, for instance the packaging department may want to package all jobs in a specific order on a their predetermined date. Furthermore, an external costumer may want to have all jobs simultaneously. In that sense, the external costumer sends its transportation vehicle to the company to pick up all of their order. Therefore the optimal schedule is important for decreasing total earliness/tardiness cost. Although finding the optimal schedule is critical for earliness/tardiness problems in JIT company, the problem is in the class of NP hard.

The common due date consideration in parallel machine problems has been investigated by researchers since 1990. While some of researchers have considered the common due date as a predetermined value, others have considered it as a decision variable. Adamopoulos and Pappis [1] considered common due date as a decision variable in parallel machine

✉ Oğuzhan Ahmet Arık
oaarik@nny.edu.tr

[1] Industrial Engineering Department, Nuh Naci Yazgan University, 38170 Kayseri, Turkey

scheduling problems and they proposed a heuristic procedure that is to minimize total cost of weighted common due date, earliness and tardiness cost. Their proposed heuristic's complexity is $O(n^3)$ so their proposed heuristic is a polynomial time heuristic for a NP-Hard problem. Even if the complexity of their proposed heuristic is polynomial, they only used small test instances to check the performance of the proposed algorithm. Bank and Werner [2] proposed some heuristic methods for parallel machine scheduling problem with pre-determined common due date and release dates where the objective is to minimize total cost of earliness and tardiness. Bank and Werner [2] presented two different type algorithms. The first type of algorithms was constructive algorithms and the other was iterative algorithms. They stated that the performance of constructive algorithms depends on some parameters settings and the performance of iterative algorithms depends on shifting jobs from one machine to another. They compared their proposed algorithms with some metaheuristics and they concluded that none of the tested variants is clearly superior to the remaining ones in general. Yang et al. [3] introduced an evolutionary strategy method for parallel machine earliness/tardiness scheduling problem with common due date. They stated that computational results of different scale problems show that the method can effectively solve the parallel machine scheduling problem with relatively large scale. Xiao and Li [4] considered the problem of assigning a common due date to a set of jobs and scheduling the jobs on a set of parallel machines so that the weighted sum of the due date, total earliness, and total tardiness is minimized. They developed a heuristic is to solve the problem. Mönch and Unbehaun [5] proposed three decomposition heuristics for the parallel machine batch scheduling with pre-determined common due date where the objective is to minimize total earliness/tardiness cost. Their first heuristic is to apply and exact algorithm for non-batching problem and then it uses some job sequencing rules and dynamic programming in order to form batches for the early and tardy job sets and sequence them optimally. Their second algorithm uses genetic algorithm in order to assign jobs to each machine and then, it uses again sequencing rules and dynamic programming techniques to the early and tardy jobs sets on each single machine in order to form batches. Their third algorithm firstly separated equally jobs as early and tardy and then uses again genetic algorithm and applies again dynamic programming and sequencing rules. They reported their computational experiments and stated that the property-based heuristic beats the genetic algorithm-based heuristics with respect to solution quality in case of a large number of jobs. Toksarı and Güner [6–9] investigated parallel machine scheduling problem under effects of learning and deterioration with common due date where the objective is to minimize total earliness/tardiness cost. Srirangacharyulu and Srinivasan [10] considered the problem of scheduling n jobs on two identical parallel machines in order to minimize the mean squared deviation of job completion times about a given common due date. They proposed a heuristic to provide quick solutions for problems of larger size. Drobouchevitch and Sidney [11] considered a problem of scheduling n identical nonpreemptive jobs with a common due date on m uniform parallel machines where the objective is to determine an optimal value of the due date and an optimal allocation of jobs onto machines so as to minimize a total cost of earliness, tardiness and due date values. They proposed a two-phase algorithm to solve the problem. Kim et al. [12] presented a mathematical programming model and two types of heuristics for parallel machine earliness/tardiness scheduling problem where the common due date is a decision variable and the objective is to minimize the sum of due-date assignment, earliness and tardiness penalties. Their two types of heuristics were: (a) a fast two-stage heuristic with obtaining an initial solution and improvement; and (b) two meta-heuristics, tabu search and simulated annealing, with new neighbourhood generation methods. They stated that each of the heuristic types outperforms the existing one. Beyranvand et al. [13] considered the quadratic programming formulation of the unrelated parallel machine scheduling problem with restrictive common due date. Awasthi et al. [14] investigated single and parallel machine earliness/tardiness scheduling problems with common due date. They presented an exact polynomial algorithms for optimizing a given job sequence for single and parallel machines with the run-time complexities of O($n\ log\ n$) and O($mn2log\ n$) respectively. Lin et al. [15] investigated the problem of minimizing total weighted earliness and tardiness penalties on identical parallel machines against a restrictive common due date. They proposed a fast ruin-and-recreate (FR&R) algorithm is proposed to obtain high-quality solutions to this complex problem. They stated that computational results provide evidence of the efficiency of FR&R, which consistently outperform existing algorithms when applied to benchmark instances.

Although earliness/tardiness parallel machine scheduling problem is a well-known and classical problem in the literature, the problem is in NP hard class. For combinatorial optimization problem such as scheduling problems in NP hard class, metaheuristics promise to find near optimal solution for these problems within a reasonable time. Metaheuristic algorithms are classified into swam intelligence based algorithms, evolutionary algorithms and single solution algorithms. All of these algorithms are adaptable for different types of combinatorial problems (see [16–19]). In fact, a hybrid method including any two different algorithms can be used in order to solve and to find near-optimal or optimal solutions for these problems. For unrelated parallel machine weighted earliness/tardiness scheduling problem, in order to find what kind of algorithms are more efficient than others,

this study investigates the performance of ABC, GA and SA algorithms that are well-known examples of swam intelligence based algorithms, evolutionary algorithms and single solution algorithms. The difference of the paper from the literature for the problem and its variants is to make a comparison among different types of well-known metaheuristic algorithms and to suggest one of alternatives for the readers.

2 Mathematical model for unrelated parallel machine weighted earliness/tardiness scheduling problems

In this section, a position-dependent mixed integer programming model for unrelated parallel machine weighted earliness/tardiness scheduling problems with common due-date is introduced. If a job is completed after its due date then that job is a tardy job and tardiness of that job can be calculated as follows:

$$T_i = \max(C_i - D, 0) \quad \forall i \tag{1}$$

where T_i is tardiness of the ith job, C_i is completion time of the ith job and D is the common due-date for all jobs. If a job is completed before its due date then that job is an early job and earliness of that job can be calculated as follows:

$$E_i = \max(D - C_i, 0) \quad \forall i \tag{2}$$

where E_i is tardiness of the ith job. In a JIT company, jobs are wanted to be completed on its due-dates. Earliness and tardiness are not desired because each early/tardy job has some costs such as insurance, holding, stocking, additional cargo, etc., for the company. The company may define costs of earliness and tardiness amounts of jobs and then the total cost of earliness and tardiness can be calculated as follows:

$$f = \alpha \sum_{i=1}^{n} E_i + \beta \sum_{i=1}^{n} T_i \tag{3}$$

where α is the cost for one time-unit earliness and β is the cost for one time-unit tardiness. A position-dependent mixed integer linear programming model for unrelated parallel machine weighted earliness/tardiness scheduling problems with common due-date is as follows:

Indexes

i: index for jobs,
j: index for machines,
r: Index for position numbers on machines,

Parameters

P_{ij}: processing time of job i for machine j,
D: common due date for all jobs,
α: weight for early jobs,
β: weight for tardy jobs,
n: number of jobs
m: number of machines

Decision variables

$P_{[r],j}$: processing time of the job at position r of machine j,
$C_{[r],j}$: completion time of the job at position r of machine j,
C_i: completion time of job i,
$X_{i,r,j}$: if job i is assigned at position r of machine j, then it is equal to 1, else 0
E_i: earliness time of job i
T_i: tardiness time of job i

Objective function

$$\min z: \ \alpha \sum_{i=1}^{n} E_i + \beta \sum_{i=1}^{n} T_i \tag{4}$$

Constraints

$$P_{[r],j} = \sum_{i}^{n} X_{i,r,j} P_{ij} \quad \forall r, j \tag{5}$$

$$C_{[r],j} = P_{[r],j} + C_{[r-1],j} \quad \forall r, j \tag{6}$$

$$C_{[0],j} = 0 \quad \forall j \tag{7}$$

$$C_i = \sum_{r}^{n} \sum_{j}^{m} X_{i,r,j} C_{[r],j} \quad \forall i \tag{8}$$

$$E_i \geq D - C_i \quad \forall i \tag{9}$$

$$T_i \geq C_i - D \quad \forall i \tag{10}$$

$$\sum_{i}^{n} x_{i,r,j} \leq 1 \quad \forall r, j \tag{11}$$

$$\sum_{r}^{n} \sum_{j}^{m} x_{i,r,j} = 1 \quad \forall i \tag{12}$$

$$x_{i,r,j} + \sum_{I=1}^{n} x_{I,r+1,j} \leq 2 \quad (I \neq i), \ (r = 1, 2, \ldots, n-1) \ \forall i, j \tag{13}$$

$$x_{i,r+1,j} \leq x_{I,r,j} \quad (I \neq i), \ (r = 1, 2, \ldots, n-1) \ \forall i, j \tag{14}$$

$$P_{[r],j}, C_{[r],j} \geq 0 \quad \forall r, j \tag{15}$$

$$C_i, E_i, T_i \geq 0 \ \forall i \tag{16}$$

$$X_{i,r,j} \in \{0, 1\} \tag{17}$$

The objective function (4) is to minimize the sum of total weighted earliness/tardiness costs for all jobs. Constraint (5) shows the calculation for actual processing times of jobs assigned at position r of machine j. Constraint (6) ensures that completion time of the job at position r of machine j is equal to sum the of the actual time of same job and previous job's completion time in same machine. Constraint (7) shows that each machine is ready to process jobs at the beginning. Constraint (8) is to determine completion time of job i by transforming machine and position indexes. Constraint (9) assures that earliness of a job must be equal to time interval between common due date and early job's completion time. Constraint (10) assures that tardiness of a job must be equal to time interval between common due date and tardy job's completion time. Constraint (11) ensures that a job can be assigned at a position of a machine at most once. Constraint (12) shows that each job must be assigned on a position of a machine. Constraint (13) assures that each job must come after another job and no position number is idle between two jobs. Constraint (14) shows that if a job assigned on previous position, then the current position may be used for another job or not. For Constraints (13–14), I index is an alias of i index and it is also index for jobs. Constraints (15–17) show that $X_{i,r,j}$ are binary decision variables and other decision variables are non-negative.

3 Solution approaches

In this section, three different metaheuristics are introduced for unrelated parallel machine weighted earliness/tardiness scheduling problems with common due-date. These are artificial bee colony (ABC), genetic algorithm (GA) and simulated annealing (SA) algorithm. These metaheuristics are well known samples of swarm intelligence based algorithms, evolutionary algorithms and single solution algorithms. The solution (encoding) representation of each proposed algorithm is directly encoding by using numbers of positions and machines i.e., if there are 4 jobs and 3 machines then there are 12 genes in a chromosome (solution). Figure 1 illustrates an example for encoding in a solution for this study.

The encoding strategy illustrated in Fig. 1 is common for all proposed metaheuristics to make a fair comparison.

3.1 Artificial bee colony algorithm

The artificial bee colony (ABC) algorithm is a swarm based meta-heuristic algorithm that was introduced by Karaboga [20] for optimizing numerical problems. It was inspired by

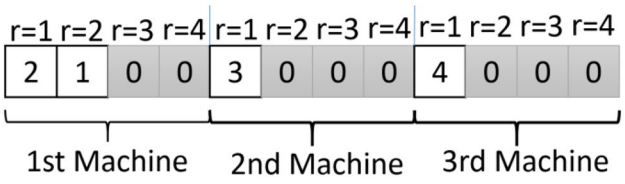

Fig. 1 Encoding strategy for a solution for all proposed metaheuristics in this study

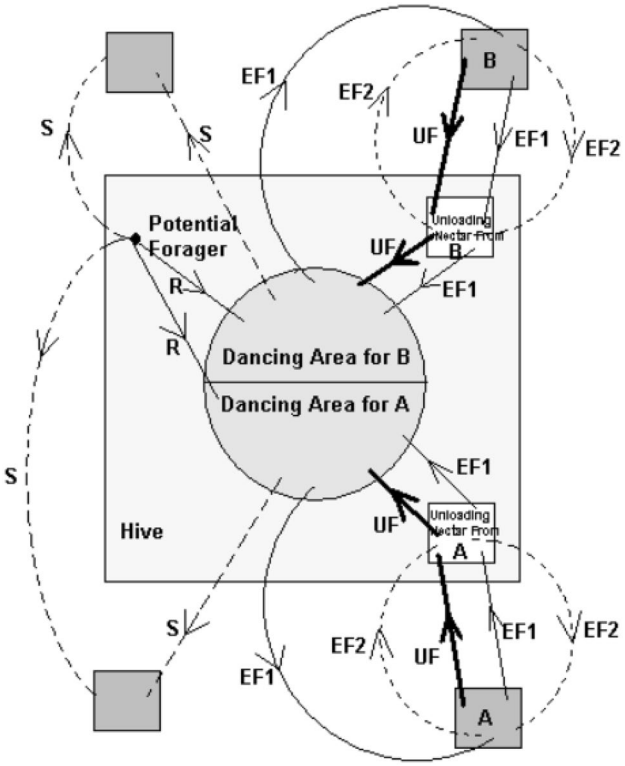

Fig. 2 The behaviour of honey bee foraging for nectar [20]

the intelligent foraging behavior of honey bees as seen in Fig. 2. As a well-known optimization approach, ABC algorithm provides robust and near-optimal solutions. In basic structure of ABC, there is a population including individuals called food positions and the individuals are evaluated with artificial bees in order to discover the best food resource or the area of good food resources. Artificial bees are classified in three groups as employed bees, onlooker bees and scout bees. As an assumption in ABC, the number of food resources is equal to the number of employed bees. In initialization phase, scout bees fly randomly to food resources and discover these resources' nectar amounts. Then, employed bees exploit the food resources by using the knowledge gained from scout bees. Then feedbacks of employed bees are used for selection of effective food resources and onlooker bees fly to neighbors of effective food resources. If an employed bee whose food

source has been exhausted, then that bee becomes a scout bee. This process goes on until a stopping condition has been occurred. The general schema of ABC algorithm is given in Fig. 3. ABC algorithm is a swarm based algorithm that combines benefits of local and global searches and it also has a feedback structure that disables to investigate bad solutions that have been previously investigated.

Karaboga proposed ABC algorithm for continuous optimization problems where the objective is to maximize or minimize a $f(\overrightarrow{x_m})$ function and each member x_{mi} of $\overrightarrow{x_m}$ vector is bounded with a lower bound l_i and an upper bound u_i. In this study, solutions in proposed ABC are represented as in Fig. 1. Let us assume $\vec{\beta}$ is vector for genes in a solution and there are nm (n is number of jobs and m is number of machines) genes $\vec{\beta} = (\beta_1, \beta_2, \beta_3, \ldots, \beta_{nm})$. There is a $f\left(\vec{\beta}\right)$ function as follows:

$$f\left(\vec{\beta}\right) = \alpha \sum_{i=1}^{n} E_i\left(\vec{\beta}\right) + \beta \sum_{i=1}^{n} T_i\left(\vec{\beta}\right) \tag{18}$$

where $E_i\left(\vec{\beta}\right)$ is the earliness of ith job for $\vec{\beta}$ solution and $T_i\left(\vec{\beta}\right)$ is the tardiness of ith job for $\vec{\beta}$ solution. $E_i\left(\vec{\beta}\right)$ and $T_i\left(\vec{\beta}\right)$ are calculated by using Eqs. (1–2) and completion times $C_i\left(\vec{\beta}\right)$ of jobs in $\vec{\beta}$ solution.

If we set number of employed bees (number of food researches or individual in the population) in proposed ABC as SN (solution number) then there are SN different $\vec{\beta}$ vectors such as $\vec{\beta}_l$ where $l = 1, 2, \ldots, SN$. So the purpose of the forecasting is to find best forecasted values close to the observed values then there is a best $\vec{\beta}^*$ vector that have optimum or near optimum as follows:

$$f\left(\vec{\beta}^*\right) = \min\left(f\left(\vec{\beta}_1\right), f\left(\vec{\beta}_2\right), \ldots, f\left(\vec{\beta}_{SN}\right)\right) \tag{19}$$

In initialization phase of ABC, all the vectors $\vec{\beta}_l$ ($l = 1, 2, \ldots, SN$) are initialized by scout bees. Since each food source $\vec{\beta}_l$, is a set of genes in solution representation in Fig. 1, each $\vec{\beta}_l$ vector holds nm variables, (β_{lt}, $t = 1, 2, 3, \ldots, nm$), which are to be optimized so as to find minimum $f\left(\vec{\beta}^*\right)$. For initialization of all $\vec{\beta}_l$ ($l = 1, 2, 3, \ldots SN$), a procedure illustrated in Fig. 4 is used.

The procedure illustrated in Fig. 4 is also used in proposed GA in this study. Employed bees search for a new set of genes or a new solution ($\vec{v}_l$) that have more suitable to be the best candidate. The searching procedure for a neighboring solution by using existing $\vec{\beta}_l$ is simply swapping two random genes' positions in $\vec{\beta}_l$ for proposed ABC algorithm. After searching procedure taking place, a repair operation

Fig. 3 Artificial bee colony algorithm for optimization problems [20]

Send the scouts onto the initial food sources
REPEAT
Send the employed bees onto the food sources and determine their nectar amounts
Calculate the probability value of the sources with which they are preferred by the onlooker bees
Send the onlooker bees onto the food sources and determine their nectar amounts
Stop the exploitation process of the sources exhausted by the bees
Send the scouts into the search area for discovering new food sources, randomly
Memorize the best food source found so far
UNTIL (requirements are met)

Begin, Define *n* (number of job), Define *m* (number of machines), Declare *K = n*m* (number of genes in a chromosome), Define *P* (population size), Declare *POP(P,K)* (solution population), For *p=1* to *P* For *k=1* to *K* If *k ≤ n* then *POP(p,k) = k* Else *POP(p,k) = 0* End If Next *k,* Next *p,*	For *p=1* to *P* For *k=1* to *K* *X1 = Random integer [1,K],* *X2 = Random integer [1,K],* *Y1 = POP(p,X1),* *Y2 = POP(p,X2),* *POP(p,X1) = Y2,* *POP(p,X2) = Y1,* Next *k,* Next *p,*	For *p=1* to *P* For *k=1* to *K* If (*k mod n*) = 0 then *Nk = k,* For *o = 1* to *n* For *j= Nk – (n-1)* to *Nk-1* If *POP(p,j) = 0* and *POP(p,j+1)>0* then *POP(p,j) = POP(p,j+1)* *POP(p,j+1) = 0* End if Next *j,* Next *o,* End If Next *k,* Next *p,* **Stop.**
*** Randomly assign all jobs on the first machine.**	***Randomly assign jobs from the first machine to other machines.**	*** Regulate the schedule and fix infeasible solutions in the population.**

Fig. 4 Pseudo code for generating initial population

regulates the solution as in the proposed GA in this study, if necessary.

Employed bees find a new $\vec{v}_l$ and they evaluate its profitability (fitness). After producing a new set of genes $\vec{v}_l$, its fitness is calculated and a greedy selection is applied between $\vec{v}_l$ and $\vec{\beta}_l$. The fitness value of $\vec{\beta}_l$ is illustrated with $fitness_l\left(\vec{\beta}_l\right)$ notation and it is calculated as follows:

$$fitness_l\left(\vec{\beta}_l\right) = 1/\left(1 + f\left(\vec{\beta}_l\right)\right) \tag{20}$$

In ABC algorithm, employed bees give information about their own sets of genes to onlooker bees. Then, onlooker bees start selecting probabilistically their sets by using the feedback from employed bees. This selection phase is done with a fitness based selecting technique and the probability p_l of the set $\vec{\beta}_l$ can be determined as follows:

$$p_l = \frac{fitness_l\left(\vec{\beta}_l\right)}{\sum_{l=1}^{SN} fitness_l\left(\vec{\beta}_l\right)} \forall l \tag{21}$$

After a set $\vec{\beta}_l$ for an onlooker bee is probabilistically chosen, a neighbourhood set $\vec{v}_l$ is determined by swapping operation, and its fitness value is computed by using Eq. (20). As in the employed bees phase, a greedy selection is applied between $\vec{v}_l$ and $\vec{\beta}_l$. Thus, the number of onlooker bees recruiting better solution spaces is increased. In this phase, to disable to inefficient set, a counter $failure_l\left(\vec{\beta}_l\right)$ for each $\vec{\beta}_l$ takes places. If $fitness_l\left(\vec{\beta}_l\right)$ is better than $fitness_l(\vec{v}_l)$, then $failure_l\left(\vec{\beta}_l\right)$ increases one. If $fitness_l\left(\vec{\beta}_l\right)$ is not better than $fitness_l(\vec{v}_l)$, then $failure_l\left(\vec{\beta}_l\right)$ is set as zero. If $failure_l\left(\vec{\beta}_l\right)$ reaches a pre-determined *limit*, then the employee bee dealing with $\vec{\beta}_l$ becomes a scout bee that abandons $\vec{\beta}_l$ and finds a random set of genes. The number of scout bees is generally limited for preventing ABC becoming a random search algorithm. The algorithm runs until a pre-determined stopping condition occurrence.

3.2 Genetic algorithm

Genetic algorithm (GA) is a metaheuristic search method for optimization problems and it is well-known example of evolutionary algorithms. GA uses stochastic random search methods to generate new alternative solutions from initially generated solution population. Each solution in the initial population is evaluated by comparing with others in order to determine how much proper (fitness value) this candidate to be the best solution. Then, these candidates are transmitted into matching pool for crossover process in order to randomly generate new solutions from good solutions in the current population. After some population generations, crossover process drives solutions in the population to be similar to each other. Thus, a mutation process randomly takes place to protect the diversity of the population. After these processes, solutions in the new population are evaluated again. If the stopping criterion occurs, then GA stops and displays the best solution. Otherwise, GA continues to search for better solutions until stopping criteria occurs.

The general schema of GA in this study is illustrated with a pseudo code in Fig. 5. Encoding strategy of GA in this study illustrates the indexes of jobs assigned on any position number of machines. This strategy makes computationally easy to execute GA operations. This encoding strategy is common for all methods investigated in this paper and can be seen in Fig. 1. Evaluation and selection steps in this GA do not let the worse solution in the current population to transmit to matching pool. After crossover and mutation steps, if the feasibility of solutions is going wrong, then a repair operation takes place to fix infeasible solutions. Thus, GA only searches for the fitness of objective function instead of the fitness of constraints of the model. These are the advantages of proposed GA.

Generating initial population In order to generate an initial solution for the problem, the pseudocode in Fig. 4 is proposed in this study. After generating randomly an initial population, feasibilities of candidate solutions in the population must be controlled. The pseudo code in Fig. 4 checks the feasibility and fixes solutions in the population if it is required.

Evaluating The evaluation step in GA depends on fitness values of solutions in the population. Let $k\,(k = 1, \ldots, P)$ is the index for population and P is the size of population. The fitness value f_k of solution k in the current population is calculated as follows:

$$f_k = \max\left\{\left(\alpha\sum_{i=1}^{n} E_i^1 + \beta\sum_{i=1}^{n} T_i^1\right), \ldots, \left(\alpha\sum_{i=1}^{n} E_i^P + \beta\sum_{i=1}^{n} T_i^P\right)\right\} - \left(\alpha\sum_{i=1}^{n} E_i^k + \beta\sum_{i=1}^{n} T_i^k\right) \tag{22}$$

where E_i^k is earliness of job i and T_i^k is tardiness of job i in solution k in the current population.

Selection The selection step in this GA is roulette wheel selection. Selection probability values of solutions are used to determine which solutions will be transmitted into the matching pool. The selection probability values are calculated as follows:

$$P_k = \frac{f_k}{\sum f_k}, \tag{23}$$

where P_k is the selection probability for solution k by comparing its fitness value with sum of all fitness values.

Crossover Crossover step in GA takes place after the selection step. This step is required to generate new alternative solutions by using a selected solution pair from the matching pool. Crossover is a random step with a probability p_c. In this study, two points crossover method is preferred. Once crossover step occurs for a selected pair of solutions, a repair operation may be required in order to fix infeasible solutions. Figure 6 illustrates an example for crossover and repair steps.

Mutation Mutation is another random step with a probability p_m in GA in order to protect solution diversity in the population because crossover step in GA makes solutions in the population get similar to each other after some generations. In this study, randomly selected two genes are replaced with each other. Then, if it is required, a repair step may take place to fix infeasibility. Figure 7 illustrates an example for mutation and repair steps.

Begin,
Define K (number of solutions in a population),
Define T (number of generation),
Declare $S_{best}=\{\emptyset\}$ (Best Solution),
Declare $Z_{best} = 0$ (Best objective func. value),
Generate initial population $P_0=\{S_1 \ldots S_K\}$,
Evaluate solutions $S_k \; \forall \; k \in K$ and calculate fitness values f_k of solutions in P_0,
Find the best solution S^* and its objective func. value Z^* of the current population,
Set $S_{best}= S^*$ and $Z_{best}= Z^*$,
For $t=1$ to T
    Define number of solution Q_k that will be transmitted into new population,
    Select solution k and transmit it Q_k times into new population,
    Randomly run **Crossover** operation for selected pair of solution in new population,
    Randomly run **Mutation** operation for selected single solution in new population,
    Evaluate solutions $S_k \; \forall \; k \in K$ and calculate fitness values f_k of solutions in P_0,
    Find best solution S^* and its objective func. value Z^* of new population,
    If $Z_{best} \geq Z^*$ then
        Set $S_{best}= S^*$ and $Z_{best}= Z^*$,
    End if
Next t,
Display Z_{best},
Display S_{best},
Stop.

Fig. 5 Pseudo code for genetic algorithm

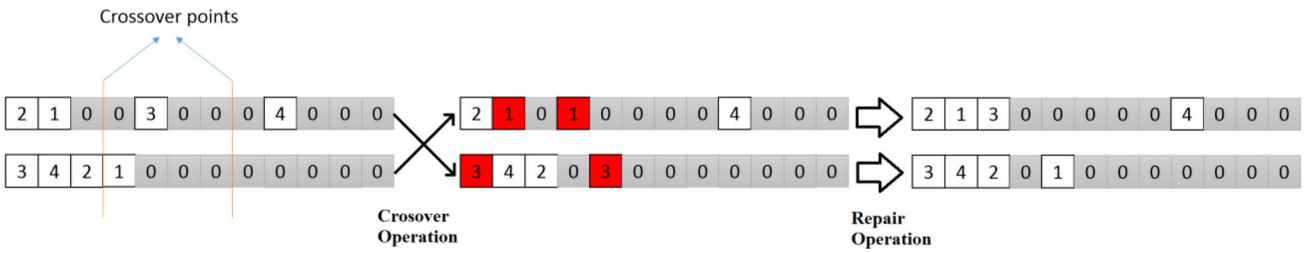

Fig. 6 Crossover and repair steps

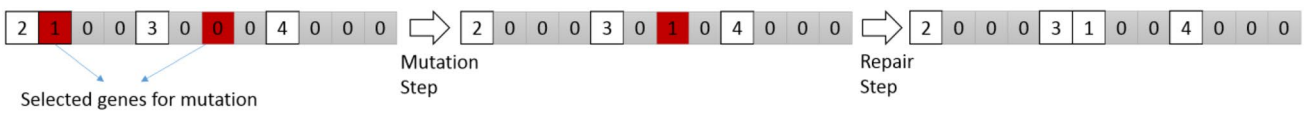

Fig. 7 Mutation and repair steps

Repair The chromosome in Figs. 6 and 7 represents a solution for 4 jobs and 3 machines. There are 12 (4*3) genes in each chromosome. The first four-gene in the chromosome belongs to the first machine's jobs and the second four-gene belongs to the second machine's jobs. When crossover or mutation operation takes place as seen in Fig. 6, a repair operation regulates sequence of genes to disable idle genes get in between two genes having jobs. Furthermore, if there is an idle gene before a gene having job, then idle gene and that gene are replaced as illustrated in Figs. 6 and 7.

3.3 Simulated annealing algorithm

Simulated annealing (SA) is another metaheuristic method in order to deal with optimization problems. SA is inspired from the annealing process in metal work. SA's best advantage by comparing it with other search methods is avoiding from a local optimum by using stochastic search. SA starts with an initial system temperature and this temperature is decreased at each iteration. At each system temperature, SA looks for neighboring solutions in order to improve its current best solution. While temperature is decreasing at each iteration, number of searching for neighboring solutions at each iteration is being increased by SA. Furthermore, if SA sticks around a local optimum, then a stochastic movement for new search point takes place to release SA from the local optimum. This process ends when the system temperature reaches the stopping temperature. Figure 8 shows the pseudo code for SA in this study. In this proposed SA, searching procedure for neighboring solutions is simply swapping two random genes' positions as in ABC algorithm. Encoding strategy of SA in this study illustrates the indexes of jobs assigned on any position number of machines. This strategy makes computationally easy to execute SA operations. This

Begin,
Set initial temperature T_{init},
Set stopping temperature T_{stop},
Set current temperature $T = T_{init}$,
Set temperature decrease rate a,
Generate randomly initial solution S_{init},
Calculate the objective function value of initial solution f_{init},
Set best solution $S_{best} = S_{init}$,
Set best objective function value $f_{best} = f_{init}$,
Set $x = 0$,
Do until $T \leq T_{stop}$

Set nm number of movements to be sought in each temperature, $nm = \left\lceil n^2 * \sqrt{\frac{T}{x}} \right\rceil + 1$

For $i = 1$ to nm
Generate a neighbor solution S_n for current best solution,
Calculate the obj. function value f_n for S_n,
Set $\Delta f = f_n - f_{best}$,
If $\Delta f \leq 0$ then
$f_{best} = f_n$
$S_{best} = S_n$
Else
Generate a random number $r \in [0,1]$
If $r \leq e^{-(\Delta f/T)}$ then
$f_{best} = f_n$
$S_{best} = S_n$
End if
End if
Next ***i***
Set $x = x + 1$,
Decrease temperature $T = a * T$,
Loop
Display best schedule S_{best},
Display best obj. function value f_{best},
Stop.

Fig. 8 Pseudo code for SA

encoding strategy is common for all methods investigated in this paper and can be seen in Fig. 1.

4 Experimental results

In this section, some test instances are generated to determine which proposed metaheuristic is better than others. 16 test instances group and 10 instances in each group are generated. Numbers of jobs in instances are 100, 200, 300 and 400. Numbers of machines in instances are 5, 10, 15 and 20. The processing times are uniformly generated $P_{ij} = U[1, 20]$ for 160 instances. As earliness and tardiness weights (α, β), two sets are used {(2, 3); (3, 2)} for all test instances. The common due date D for each instances are determined with the way proposed by Bank and Werner [2] as follows:

$$D = \left\lfloor \frac{1}{3}\frac{n}{m}\bar{P} \right\rfloor \tag{24}$$

where

$$\bar{P} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij}}{nm} \tag{25}$$

Each proposed metaheuristic was coded with C# programming language and MS Access database by using a standard desktop having i5 CPU and 8 GB RAM. The parameter settings of metaheuristic are determined by considering general parameter settings in the literature. For proposed GA, crossover and mutation probabilities are set as 0.85 and 0.15. The initial temperature and temperature decrease rate of proposed SA are set as 10^6 and 0.8. The numbers of employee bees, onlooker bees and scout bees are set as 30, 30 and 1 for proposed ABC. The limit value for failures of proposed ABC is equal to 15 times of the product of number of machines and number of jobs for each instance. Each proposed metaheuristic is executed 10 times until predetermined solution time limits for each test instances. Average solution values of each test instance group for earliness and tardiness weights $(\alpha = 2, \beta = 3)$ are given in Table 1. Average solution values of each test instance group for earliness and tardiness weights $(\alpha = 3, \beta = 2)$ are given in Table 2.

As seen from Table 1, ABC algorithm gives better results than its opponents in average. All results obtained from all proposed metaheuristics for $\alpha = 2$ and $\beta = 3$ are analyzed with ANOVA for objective function values. The ANOVA results for total weighted earliness/tardiness values obtaining by different methods give us a p value as zero. Therefore, we can say that methods are significantly different considering solution quality. Furthermore, the interval plot in Fig. 9 show us ABC gives better solution in a more narrow interval.

Table 1 Average solutions of each test instance group for earliness and tardiness weights $(\alpha = 2, \beta = 3)$

# of jobs	# of machines	GA	SA	ABC
100	5	10,998.10	10,660.96	6757.48
	10	8599.18	4638.92	6633.18
	15	7168.72	2475.80	6529.02
	20	6325.08	2173.22	6055.82
200	5	45,853.78	45,995.52	27,474.70
	10	32,550.32	22,993.08	22,016.18
	15	26,869.38	13,431.34	21,744.40
	20	22,934.06	8548.06	20,618.86
300	5	105,891.64	97,018.80	61,661.68
	10	73,109.92	52,909.92	44,662.72
	15	59,288.08	33,783.36	44,330.24
	20	50,088.28	21,287.92	42,330.88
400	5	189,709.04	161,330.10	106,966.16
	10	131,987.66	87,281.46	73,296.06
	15	104,607.16	63,380.84	73,106.84
	20	90,388.30	40,784.38	69,535.60
	Average	60,398.04	41,793.36	39,607.49

Table 2 Average solutions of each test instance group for earliness and tardiness weights $(\alpha = 3, \beta = 2)$

# of jobs	# of machines	GA	SA	ABC
100	5	9644.08	15,929.20	8472.58
	10	6458.62	6946.88	5613.22
	15	5234.32	3507.08	4948.66
	20	4437.66	2366.06	4370.74
200	5	40,453.62	65,414.26	34,831.58
	10	25,754.36	33,941.94	20,518.40
	15	20,187.26	20,041.12	17,285.84
	20	16,807.00	11,877.70	15,506.52
300	5	92,042.34	135,129.96	79,287.30
	10	58,640.36	74,558.06	44,721.48
	15	45,449.22	50,519.82	36,905.56
	20	37,296.20	31,218.36	32,171.60
400	5	164,315.70	228,654.88	142,258.82
	10	104,044.30	119,000.02	77,466.20
	15	80,129.84	90,735.54	61,997.76
	20	67,351.34	60,275.66	55,012.42
	Average	48,640.39	59,382.28	40,085.54

As seen from Table 2, ABC algorithm gives better results than its opponents in average. All results obtained from all proposed metaheuristics for $\alpha = 3$ and $\beta = 2$ are analyzed with ANOVA for objective function values. The ANOVA results for total weighted earliness/tardiness values obtaining by different methods give us a p value as 0.001. Therefore, we can say that methods are significantly different

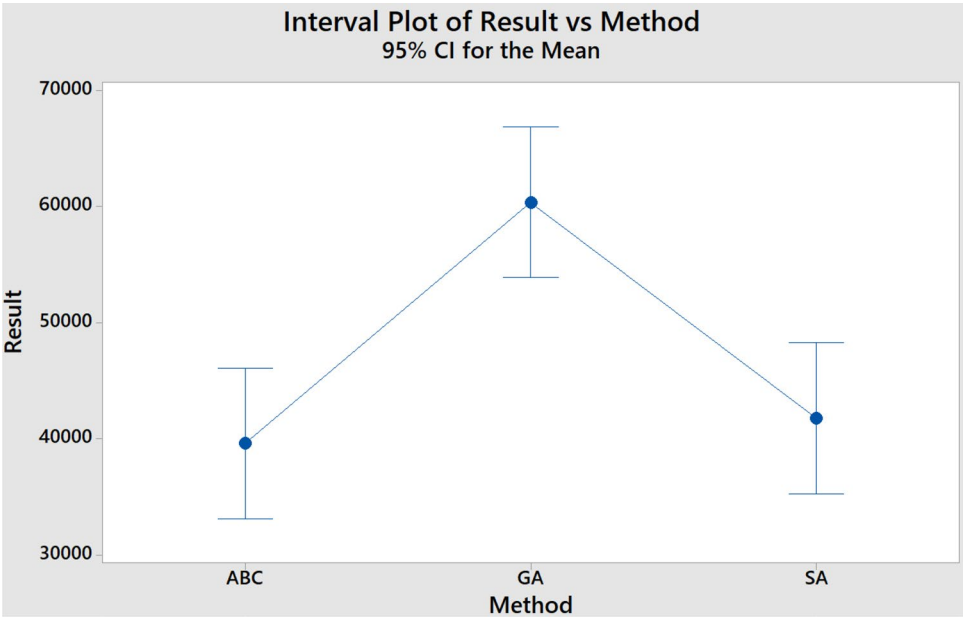

Fig. 9 Interval plots f results obtained proposed algorithms ($\alpha = 2$ and $\beta = 3$)

considering solution quality. Furthermore, the interval plot in Fig. 10 show us ABC gives better solution in a more narrow interval.

As a conclusion of above analyses for the problem, we can say that ABC algorithm as swarm intelligence based metaheuristic outperforms well-known samples of evolutionary and single solution metaheuristics. Each type of metaheuristic has own its advantages. For a fair comparison, the encoding strategies of all methods are the same and this strategy is illustrated in Fig. 1. For instance, evolutionary algorithms have a population based structure to help searching multiple solutions, a selection mechanism that helps to focus on better candidate solutions, and an escape mechanism from local optima. Single point or solution methods such as SA are relatively easy to code and to apply arbitrary problems and cost functions. Swarm intelligence based algorithms such as ABC imitate intelligent swarms' self organizing and information sharing models among individuals in the swarm so positive or negative feedbacks of individuals in the swarms help to improve the quality of the solution. Furthermore, each metaheuristic can be hybrid with other heuristics or metaheuristic considering advantages and disadvantages of alternative methods.

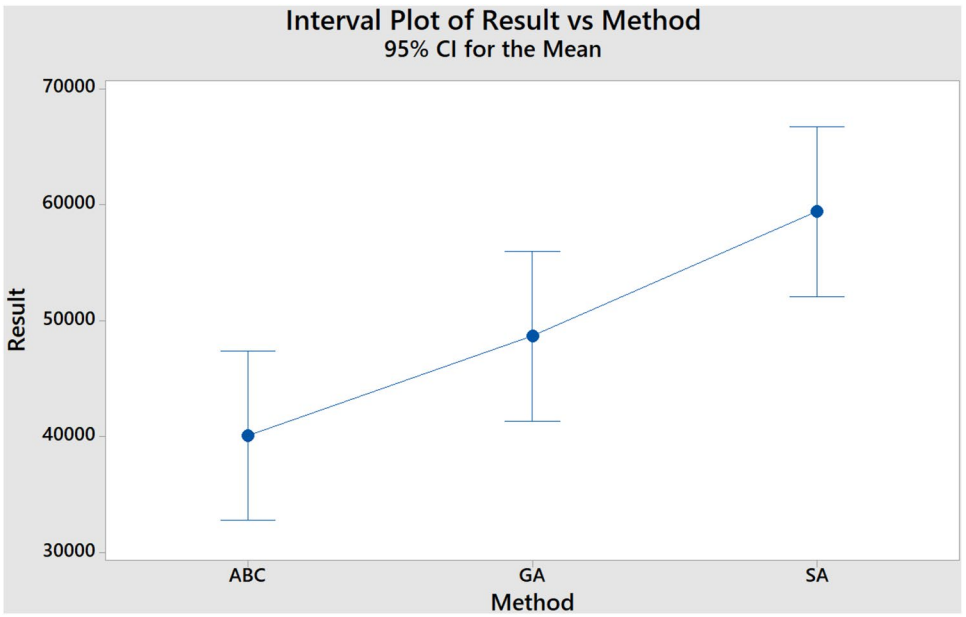

Fig. 10 Interval plots f results obtained proposed algorithms ($\alpha = 3$ and $\beta = 2$)

5 Conclusion

In this study, we investigated metaheuristic algorithm alternatives among well-known samples of swarm intelligence based, evolutionary and single solution algorithms for unrelated parallel machine earliness/tardiness scheduling problems with common due date. ABC, GA and SA were chosen for comparison in view of solution quality in a pre-determined time limitation. All proposed algorithms are created without modification in order to determine solution qualities of these metaheuristics. Furthermore, the encoding strategies of all methods are the same for a fair comparison. Experimental results show that ABC outperforms its opponents in view of solution quality as swarm intelligence based metaheuristic algorithm. Future researchers may use a hybrid algorithm having main structure and advantages of ABC for unrelated parallel machine earliness/tardiness scheduling problems with common due date. Furthermore, ABC algorithm can be used for the same problem with other constrains such as release dates, setup times and.

Compliance with ethical standards

Conflict of interest The author declared that there is no conflict of interest.

References

1. Adamopoulos GI, Pappis CP (1998) Scheduling under a common due-date on parallel unrelated machines. Eur J Oper Res 105:494–501. https://doi.org/10.1016/S0377-2217(97)00057-X
2. Bank J, Werner F (2001) Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. Math Comput Model 33:363–383. https://doi.org/10.1016/S0895-7177(00)00250-8
3. Yang Y-J, Liu M, Wu C (2001) Evolutionary strategy method for parallel machine earliness/tardiness scheduling problem with common due date. Tien Tzu Hsueh Pao/Acta Electron Sin 29:1478–1481
4. Xiao W-Q, Li C-L (2002) Approximation algorithms for common due date assignment and job scheduling on parallel machines. IIE Trans Inst Ind Eng 34:466–477. https://doi.org/10.1080/07408170208928883
5. Mönch L, Unbehaun R (2007) Decomposition heuristics for minimizing earliness–tardiness on parallel burn-in ovens with a common due date. Comput Oper Res 34:3380–3396. https://doi.org/10.1016/j.cor.2006.02.003
6. Toksarı MD, Güner E (2008) Minimizing the earliness/tardiness costs on parallel machine with learning effects and deteriorating jobs: a mixed nonlinear integer programming approach. Int J Adv Manuf Technol 38:801–808. https://doi.org/10.1007/s00170-007-1128-3
7. Toksarı MD, Güner E (2009) Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration. Comput Oper Res 36:2394–2417. https://doi.org/10.1016/j.cor.2008.09.012
8. Toksarı MD, Güner E (2010) The common due-date early/tardy scheduling problem on a parallel machine under the effects of time-dependent learning and linear and nonlinear deterioration. Expert Syst Appl 37:92–112. https://doi.org/10.1016/j.eswa.2009.05.014
9. Toksarı MD, Güner E (2010) Parallel machine scheduling problem to minimize the earliness/tardiness costs with learning effect and deteriorating jobs. J Intell Manuf 21:843–851. https://doi.org/10.1007/s10845-009-0260-3
10. Srirangacharyulu B, Srinivasan G (2011) Minimising mean squared deviation of job completion times about a common due date in multimachine systems. Eur J Ind Eng 5:424–447. https://doi.org/10.1504/EJIE.2011.042740
11. Drobouchevitch IG, Sidney JB (2012) Minimization of earliness, tardiness and due date penalties on uniform parallel machines with identical jobs. Comput Oper Res 39:1919–1926. https://doi.org/10.1016/j.cor.2011.05.012
12. Kim J-G, Kim J-S, Lee D-H (2012) Fast and meta-heuristics for common due-date assignment and scheduling on parallel machines. Int J Prod Res 50:6040–6057. https://doi.org/10.1080/00207543.2011.644591
13. Beyranvand MS, Peyghami MR, Ghatee M (2012) On the quadratic model for unrelated parallel machine scheduling problem with restrictive common due date. Optim Lett 6:1897–1911. https://doi.org/10.1007/s11590-011-0385-0
14. Awasthi A, Lässig J, Kramer O (2014) Common due-date problem: exact polynomial algorithms for a given job sequence. In: Proceedings - 15th international symposium on symbolic and numeric algorithms for scientific computing, SYNASC 2013, 2014, pp 258–264
15. Lin S-W, Ying K-C, Chiang Y-I, Wu W-J (2016) Minimising total weighted earliness and tardiness penalties on identical parallel machines using a fast ruin-and-recreate algorithm. Int J Prod Res 54:6879–6890. https://doi.org/10.1080/00207543.2016.1190041
16. Guha R, Ghosh M, Kapri S et al (2019) Deluge based genetic algorithm for feature selection. Evol Intell. Special Issue:1–11. https://doi.org/10.1007/s12065-019-00218-5
17. Sharma S, Kumar S, Sharma K (2019) Improved Gbest artificial bee colony algorithm for the constraints optimization problems. Evol Intell. Special Issue:1–7. https://doi.org/10.1007/s12065-019-00231-8
18. Yang Z, Wang L, Cai Y, Kimie K (2018) A polychromatic sets theory based algorithm for the input/output scheduling problem in AS/RSs. Evol Intell 12:333–340
19. Islam MR, Saifullah CMK, Mahmud MR (2019) Chemical reaction optimization: survey on variants. Evol Intell. 12:395–420. https://doi.org/10.1007/s12065-019-00246-1
20. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Engineering Faculty, Computer Engineering Department, Erciyes University