A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion

Kai Zhou Gao a,*, Ponnuthurai Nagaratnam Suganthan a, Tay Jin Chua b, Chin Soon Chong b, Tian Xiang Cai b, Qan Ke Pan c

Abstract

This study addresses the scheduling problem in remanufacturing engineering. The purpose of this paper is to model effectively to solve remanufacturing scheduling problem. The problem is modeled as flexible job-shop scheduling problem (FJSP) and is divided into two stages: scheduling and re-scheduling when new job arrives. The uncertainty in timing of returns in remanufacturing is modeled as new job inserting constraint in FJSP. A two-stage artificial bee colony (TABC) algorithm is proposed for scheduling and re-scheduling with new job(s) inserting. The objective is to minimize makespan (maximum completion time). A new rule is proposed to initialize bee colony population. An ensemble local search is proposed to improve algorithm performance. Three re-scheduling strategies are proposed and compared. Extensive computational experiments are carried out using fifteen well-known benchmark instances with eight instances from remanufacturing. For scheduling performance, TABC is compared to five existing hybrid algorithms. For re-scheduling performance, TABC is compared to six simple heuristics and proposed hybrid heuristics. The results and comparisons show that TABC is effective in both scheduling stage and rescheduling stage.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Many researchers studied flexible job shop scheduling problem (FJSP) in the past decade. FJSP is a generation of classic job shop scheduling problem (JSP). In FJSP, an operation can be processed on more than one candidate machines. For solving FJSP problem, two sub-problems have to be considered, machine assignment and operation sequencing. Machine assignment is to assign a processing machine for each operation while operation sequencing is to schedule all operations on machines to obtain feasible and quality solution. Therefore, FJSP is more complicated than JSP problem and is classified as NP-hard problem (Bruker & Schlie, 1990).

The first study to address FJSP was by Bruker and Schlie who proposed a polynomial algorithm for two jobs FJSP. In recent years, many heuristics and meta-heuristics have been employed for FJSP problem, for example tabu search (TS) (Brandimarte, 1993), genetic algorithm (GA) (Gao, Sun, & Gen, 2008), particle swarm optimization (PSO) (Zhang, Shao, Li, & Gao, 2009), simulated annealing (SA) (Li, Pan, & Gao, 2011; Li, Pan, Suganthan, & Chua, 2011), ant colony optimization (ACO) (Xing, Chen, Wang, Zhao, & Xiong, 2010), parallel variable neighborhood search (PVNS) (Yazdani, Amiri, & Zandieh, 2010) and hybrid algorithms based on different heuristics and meta-heuristics.

Among different approaches, artificial bee colony (ABC) is a widely employed swarm intelligence algorithm for solving FJSP problems. ABC algorithm is a relatively new population-based meta-heuristic approach that is based on the collective behavior of self-organized systems. It was first proposed to solve the multi-variable and multi-modal continuous functions (Karaboga, 2005). Many comparative studies have shown that the performance of the ABC algorithm is competitive to other population-based algorithms with the advantage of employing fewer control parameters in the continuous space (Karaboga, 2009; Karaboga & Akay, 2009; Karaboga & Basturk, 2007, 2008). The ABC algorithm has received intensive interest from researchers in scheduling fields.

Li (Li et al., 2011; Li, Pan, Suganthan, et al., 2011) proposed a Pareto-based ABC algorithm for multi-objective FJSP problem. In
ABC algorithm was also employed for other shop scheduling problems. Wang (Wang, Zhou, Xu, & Liu, 2012; Wang, Zhou, Xu, & Wang, 2012) proposed an effective ABC algorithm for FJSP to minimize makespan. The novelty of the algorithm is employ a combination of several heuristics rules for initializing food sources. Crossover and mutation operators were designed to generate new neighboring food sources for the employed bees. A critical path based local search was proposed to improve the intensification capability for the onlooker bees. Wang (Wang et al., 2012; Wang, Zhou, Xu, & Wang, 2012) also proposed an enhanced Pareto-based artificial bee colony algorithm for multi-objective FJSP with makespan, the total workload of machines and the workload of the critical machine. In addition, a reconstruction and select strategy is employed to determine the survival of the bees and Taguchi method based parameter setting is investigated in design of experiment. For FJSP with fuzzy processing time, Wang (Wang, Wang, Xu, & Liu, 2013; Wang, Yin, & Qin, 2013; Wang, Zhou, Xu, & Liu, 2013) proposed a hybrid ABC algorithm with variable neighborhood search (VNS).

ABC algorithm was also employed for other shop scheduling problems. Lei (Lei & Guo, 2013) proposed a modified ABC for JSP with lot streaming. An effective two-phase decoding procedure is applied in which a schedule is first built and then transportation tasks are dispatched. Swap and insertion were used in the employed bee phase and the onlooker bee phase respectively to produce new solutions. Zhang (Zhang, Song, & Wu, 2013) proposed a hybrid ABC for JSP with total weighted tardiness objective.

FJSP has strong industry background, such as semiconductor manufacturing process, automobile assembly process and mechanical manufacturing systems etc. al. For actual industry related scheduling, many constraints or uncertain conditions have to be considered when solving FJSP. Mousakhan (2013) considered sequence-dependent setup time in FJSP with total tardiness. A mathematical model was developed to formulate FJSP with sequence-dependent setup time and an iteration based meta-heuristic was proposed for solving the same problem. Wang Ling and Wang Shengyao (Wang, Wang, et al., 2013; Wang, Yin, et al., 2013; Wang, Zhou, et al., 2013) studied FJSP with fuzzy processing time using ABC and estimation of distribution algorithm (EDA). The influence of parameter setting was considered in both ABC and EDA. The novelty of the algorithm is that a left-shift scheme was employed for improving the scheduling solution in decoding stage. In addition, crossover based exploitation and variable neighborhood search (VNS) were employed for improving the performance of ABC. Xiong (Xiong, Xing, & Chen, 2013) researched into robust scheduling multi-objective FJSP with random machine breakdowns. Two surrogate measures for robustness were developed. One was for machine breakdown and another was for the location of float times and machine breakdown at the same time. Al-Hinai (Al-Hinai & Elmekawy, 2011) researched robust and stable scheduling for FJSP with random machine breakdowns using a two-stage hybrid genetic algorithm. The first stage considered general FJSP while the second stage was for machine breakdown in the decoding space. Calleja (Calleja & Pastor, 2014) and Wang (Wang, Wang, et al., 2013; Wang, Yin, et al., 2013; Wang, Zhou, et al., 2013) also considered and studied FJSP with constraints, for example, transfer batches and machine disruption. Li (2014) researched on the FJSP with maintenance activities using ABC algorithm. A self-adaptive strategy was proposed to generate new neighboring solutions and tabu search based local search was employed to improve performance. For multi-objective function, the algorithm was compared to ten existing algorithms to verify algorithm’s effectiveness.

In this study, we consider the FJSP problem in remanufacturing environment. Remanufacturing is the process of disassembly and recovery at the module level and, eventually, at the component level (Lund, 1984). It requires the repair or replacement of worn out or obsolete components and modules. Parts subject to degradation affecting the performance or the expected life of the whole are replaced. Remanufacturing is a form of a product recovery process that differs from other recovery processes in its completeness: a remanufactured machine should match the same customer expectations as new machines (Krupp, 1992). Guide (2000) considered the production planning when inputs have different and uncertain quality levels and discussed different decision variables in remanufacturing engineering. Junior (2012) reviewed the literatures on production planning and control in remanufacturing. Seventy-six papers were examined and classified. However, there are few literatures on reprocessing scheduling in remanufacturing. Uncertainty in timing of returns is one of seven major complicating characteristics in remanufacturing (Ferguson, 2009; Krupp, 1993). Li Yongjian (Li, Chen, and Cai, 2006) proposed a dynamic programming approach to derive the optimal solution in the case with large returned products and different arriving time. Li Jianzhi (Li, González, and Zhi, 2009) proposed a simulation optimization model with a prioritized stochastic batch arrival mechanism to plan and control the remanufacturing process. The uncertainty in timing and quantity are factors cannot be controlled by remanufacturers. It means that new returned product(s) or job(s) may need to be inserted into the ongoing existing scheduling solution. It is therefore important to handle this uncertainty in remanufacturing scheduling. As many discrete manufacturing systems, remanufacturing processes can be modeled as FJSP problem.

Building on the successful application of ABC for solving FJSP, a two-stage artificial bee colony (TABC) algorithm is proposed for FJSP with new job inserting. The first stage is for the general FJSP scheduling problem while the second stage is for rescheduling after new job inserting. To improve the algorithm performance, we add crossover operator and critical path based local search method.

The motivation to design two-stage ABC algorithm is scheduling and rescheduling FJSP with new job inserting. The first stage is to schedule the existing job at start stage. After that, the scheduling solution will be executed in shop floor. The second stage is to reschedule for new job inserting. After new job inserting, the second stage will be activated to reschedule the new inserted job(s) and the existing jobs’ operations that are not yet started at the inserting time. The second stage will be executed repeatedly for each new job inserting. In the first stage, all machines and jobs have the same start time. In the second stage, one machine is available for rescheduling after completing the current operation if there is an operation on this machine at new job inserting time. The jobs and machines may have different start time for rescheduling. The objective considered in this paper is to minimize the maximum complete time (makespan).

The remainder of this paper is organized as follows. Section 2 describes the FJSP with new job inserting. In Section 3, the ABC algorithm is introduced. The two-stage ABC algorithm is proposed in Section 4. Experiment design, comparison and discussion are in Section 5. We conclude this paper with future work in Section 6.

2. Description of FJSP with new job inserting

In FJSP, each job consists of a sequence of operations. An operation can be executed by one machine out of a set of candidate machines. Each operation of a job must be processed only on one machine at a time, while each machine can process only one operation at a time. The following notations and assumptions are used for the formulation of FJSP.
(1) Let \( J = \{ j_i \} \), \( 1 \leq i \leq n \), indexed \( i \) be a set of \( n \) jobs to be scheduled. \( q_i \) denotes total number of operations of job \( j_i \).
(2) Let \( M = \{ M_k \} \), \( 1 \leq k \leq m \), indexed \( k \) be a set of \( m \) machines.
(3) Each job \( j_i \) consists of a predetermined sequence of operations. Let \( O_{i,b} \) be operation \( h \) of \( j_i \).
(4) Each operation \( O_{i,b} \) can be processed without interruption on one of a set of candidate machines \( M(O_{i,b}) \). Let \( P_{i,b,h} \) be the processing time of \( O_{i,b} \) on machine \( M_k \).
(5) Decision variables \( x_{i,b,h} = \begin{cases} 1, & \text{if machine } k \text{ is selected for the operation } O_{i,b} \; \text{otherwise} \end{cases} 
\) where \( c_i \) denotes the completion time of the operation \( O_{i,b} \); \( c_i \) denotes the completion time of the job \( j_i \).

The re-scheduling problem becomes to FJSP with different scheduling schemes. Hence, both machine start time and job start time must be considered for re-scheduling in this strategy. The re-scheduling problem becomes FJSP with different machine starting times.

228 Strategy II is to re-schedule both new job(s) and the existing jobs’ operations that are not yet scheduled. The revised schedule for existing jobs’ operations may be changed. Consider a scenario that some machines may be processing some operations of existing jobs at the new job inserting time. The machines are available when the processing operations are completed. The existing jobs are also available for re-scheduling when the corresponding operations are completed. It means that jobs on machines may have different start times after re-scheduling.

There are three different re-scheduling strategies proposed.

248 (1) Strategy I is to reschedule the new job(s). Existing scheduling scheme is remained. In this re-scheduling strategy, the machines are available when all the assigned operations are completed. It means that machines may have different start time when re-scheduling is implemented. Hence, machine start time must be considered for re-scheduling in this strategy. The re-scheduling problem becomes FJSP with different machine starting times.

268 (2) Strategy II is to re-schedule both new job(s) and the existing jobs’ operations that are not yet scheduled. The revised schedule for existing jobs’ operations may be changed. Consider a scenario that some machines may be processing some operations of existing jobs at the new job inserting time. The machines are available when the processing operations are completed. The existing jobs are also available for re-scheduling when the corresponding operations are completed. It means that maybe jobs have different start time and machines have different available time after re-scheduling. Hence, both machine start time and job start time must be considered for re-scheduling in this strategy. The re-scheduling problem becomes FJSP with different machine starting times and different job starting times.

288 (3) Strategy III is available-time-block re-scheduling based on existing scheduling scheme. Similar to Strategy I, the existing scheduling scheme is also remained. The new job(s) are scheduled on all gaps of all machines after inserting time. It means that one machine is available if it has free-time-gap on the existing scheduling schema. In this condition, this machine can be employed for scheduling new job(s). After considered all available-time-block on each machine, the same scheduling strategy in Strategy I is employed to schedule remaining operations. Compared to Strategy I, Strategy III has higher time complexity.

To explain FJSP problem and the three different re-scheduling strategies more clearly, two examples are shown in following content. Fig. 1(a) shows a Gantt chart for 3-jobs and 3-machines FJSP problem. The numbers of operations in three jobs are Job 1, 3, Job 2, 2 and Job 3, 2, respectively. The makespan value in this scheduling solution is 10. The completion time of three machines is M1, 8, M2, 7 and M3, 10. Fig. 1(b) shows the new job, Job 4, arrives at time 3 and will be inserted into existing schedule at or after time 3. The Job 4 has three operations. Fig. 1(c) and (d) show re-scheduling scheme using strategy I and II. In Fig. 1(c), existing scheduling scheme is remained and the Job 4 are scheduled when the assigned operation on all machines are completed. The start times of three machines for re-scheduling are M1, 8, M2, 7 and M3, 10. In Fig. 1(d), both new Job 4 and all no start operations of existing three jobs are rescheduled. The available time of M1 and M3 is 3 while the available time of Job 2, Job 3 and Job 4 is also 3. M2 is processing the first operation of Job 1 when the Job 4 inserts at Time 3. M2 and Job 1 are available when the first operation of Job 1 is completed. Therefore, the available time of M2 and Job 1 is Time 4. In Fig. 1(d), the start time of Operation 1.2 and Operation 1.3 are delayed than that in Fig. 1(b). Fig. 1(e) and (f) show the Strategy III for another example. New job 5 comes and inserts into existing scheduling schema at Time 4. In Fig. 1(f), the operation 5.1 and 5.2 are assigned at free gap of machine M2 and M3.

3. Introduction to ABC algorithm

ABC algorithm is a population-based meta-heuristic proposed by Karaboga (Karaboga, 2005, 2009; Karaboga and Basturk, 2008). ABC is inspired from the foraging behavior of bee colony. There are three kinds of bees, namely, employed bees, onlooker bees and scout bees in ABC algorithm. A bee that is currently exploiting a food source is called an employed bee. A bee waiting in the hive for making decision to choose a food source is named as an onlooker. A bee carrying out a random search for a new food source is called a scout. Each solution to the problem under consideration is called a food source, whereas the fitness of the solution is corresponded to the nectar amount of the associated food resource.

The main steps of the basic ABC algorithm are as follows.

(1) Initialization of the parameters and population phase

The parameter of ABC are the number of food sources \( SN \), the number of trials after which a food source is to be abandoned (limit) and the termination criterion. The number of food sources is equal to the number of employed bees or onlooker bees. The initialization of population is to fill the population with \( SN \) number of randomly generated food sources, \( n \)-dimensional real-valued vectors.

Let \( X_i = \{ x_{i1}, x_{i2}, \ldots, x_{in} \} \) represent the \( i \)-th food source in the population. The food sources are generated as follows:

\[
x_{i1} = LB_1 + (UB_1 - LB_1) \times r_j \quad 1 \leq j \leq n \quad \text{and} \quad i = 1, 2, \ldots, SN
\]

where \( r \) is a uniform random number in the range [0, 1]; \( LB \) and \( UB \) are the lower and upper bounds for the dimension \( j \) respectively. The food sources are randomly assigned to employed bees and the corresponding finesses are evaluated.

(2) Employed bee phase

In this phase, each employed bee \( X_i \) generates a new food source \( X_{new} \) in the neighborhood of its present position as follows:

\[
x_{new,j} = x_{i,j} + (x_{i,j} - x_{best,j}) \times r
\]

where \( k \in \{ 1, 2, \ldots, SN \} \) and \( k \neq i \) and \( j \in \{ 1, 2, \ldots, n \} \) are randomly chosen indexes. \( r \) is a uniformly distributed real number in [-1, 1]. \( X_{new} \) will be compared to \( X_i \). If the fitness of \( X_{new} \) is equal
or better than that of $X_i$, $X_{\text{new}}$ will replace $X_i$ as a new food source; otherwise $X_i$ is retained.

(3) Onlooker bee phase

An onlooker bee evaluates all the employed bees and selects a food source $X_i$ depending on its probability value $p_i$ calculated by the following expression:

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$  

where $f_i$ is the nectar amount or the fitness value of the $i$th food source $X_i$. The higher the $f_i$ is, the more ability that the $i$th food source is selected.

Once the food source $X_i$ is selected, the onlooker bee will execute the update $X_i$ using Eq. (3). If the new food source has equal or better fitness value than $X_i$, the new food source will replace $X_i$ as a new member in the population.

(4) Scout bee phase

If a food source $X_i$ cannot be improved through a predetermined number of trials limit, the food source is to be abandoned and the corresponding employed bee becomes a scout. The scout produces a new food source randomly as follows:

$$x_{ij} = LB_j + (UB_j - LB_j) \times r$$  

where $r$ is a uniform random number in the range $[0, 1]$. If $x_{ij}$ is no better than $X_i$, then put it to a new hive; otherwise it becomes a new food source.

(5) Repeat (2)-(4) until the termination is satisfied.

4. Two-stage TABC for DFJSP

4.1. Encoding and decoding

A solution in this paper consists of two vectors corresponding to the machine assignment and operation scheduling sub-problems of the FJSP with new job inserting. Using ABC algorithm, a bee is therefore composed of two parts, Machine Assignment vector (hereafter called MA) and Operation Sequencing vector (hereafter called OS). MA is a sequence for assigning machine for each operation while OS is a sequence for ordering the operations on selected machines. Fig. 2(a) illustrates a machine assignment vector (MA) while Fig. 2(b) shows the corresponding operation sequence (OS). In Fig. 2(a), the first line is the operation of all jobs while the second line is the MA used in the proposed algorithm. In MA, each element represents the machine selected for the corresponding operation. For example, the first value “3” means machine 3 is selected for processing operation $O_{11}$. In OS, the same elements represent the different operations of the same job. For example, the first value “2” is the first operation of job 2. The second value “1” is the first operation of job 1. The third value “3” is the first operation of job 3. The fourth value “2” represents the second operation of job 2 because it is the second time value “2” appearing in OS. In the same way, the fifth value “4” is the first operation of job 4 while the sixth value “4” represents the second operation of job 4. The seventh value “1” is the second operation of job 1 while the eighth value “3” represents the second operation of job 3. The last four values “2, 4, 3, 1” represent the third operations of corresponding jobs.

Pinedo (2002) divided schedules into three classes: non-delay schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule. It has been verified in the above literature that active schedule contains the optimal schedule, active schedule and semi-schedule.
For MA initialization, random rule and global minimum processing time rule are employed to select one processing machine for each operation. In addition, a minimum completion time rule is also employed for MA initialization. In this rule, two machines will be considered for each operation that has more than two selectable machines. The operation has minimum processing time on machine M1 while the second selected machine M2 has earliest feasible time. The completion times of this operation on above two machines are compared. The machine with the smaller completion time will ultimately be selected for processing the operation. This rule considers the completion time of each operation to makespan optimization. For OS initialization, random rule, most work remaining rule and most number of operations remaining rule are employed. All the above initialization methods do machine assignment first, and then perform the operation sequencing. One initializing rule with reverse process, reverse heuristic (RH), is proposed. In this heuristic, the operation sequence is decided randomly first, then the machine assignment is decided based on the operation sequence. The steps of this heuristic are as follows:

Step 1: All operations of all jobs are shuffled randomly to obtain operation sequence.

Step 2: Calculate the completion time of each operation $O_{ij}$ on all selectable machines. The machine with the minimum completion time is selected for processing operation $O_{ij}$.

### 4.3. Exploitation search

According to the characteristic of FJSP with new job inserting and the encoding strategies, we use crossover operators for the OS part of a new solution. There are several crossover operations proposed during the past decades, such as partial-mapped crossover, order crossover, cycle crossover and so on. In this study, we employed a new crossover operator based on order crossover. We obtain two new solutions from the current two solutions. The proposed crossover operator works for the OS part as follows:

**Step 1:** Generate a random number $R$ from 1 to number of jobs;

**Step 2:** Copy the values from the OS part of S1 to the corresponding positions in New S1 where the values are less than or equal to $R$.

**Step 3:** Copy the values from the OS part of S2 to the corresponding positions in New S2 where the values are larger than $R$.

**Step 4:** From the OS part of S2, copy the values which do not appear in New S1 to the vacant positions in New S1 from left to right according to the order of the sequence in S2.

**Step 5:** From the OS part of S1, copy the values which do not appear in New S2 to the vacant positions in New S2 from left to right according to the order of the sequence in Harmony 1.

The procedure is illustrated in Fig. 5.

#### 4.4. Ensemble local search

Critical path theory is employed for solving the FJSP problem by many researchers (Gao et al., 2008; Zhang, Li, Guan, & Rao, 2007; Li et al., 2011; Li, Pan, Suganthan, et al., 2011). In one solution, there may exist one or more critical paths. Each operation on one critical path is called a critical operation. This paper proposed an ensemble local search on machine. The machines are sorted based on the number of critical operations processed on them. The first ensemble local search is executed on the machine with the maximum critical operations. There are five local search operators, one Insert, one Swap, two Inserts, two Swaps, one Insert and one Swap. These five operators are stored in an operator pool. At the beginning, each operator has the same ratio to be selected for generating a neighbor solution. The operator improving the solution will be remained in operator pool while the operator not improving the solution will be deleted from the operator pool. The ensemble local search will be terminated when all the operators have been deleted from operator pool. The next machine will be executed ensemble local search until the maximum local search

---

**Procedure:** Find the earliest idle time interval for operation $O_{ij}$

```
Int f=0;
For i=1 to MSche-1 / MSche is the number of operations the machine has processed
  If the interval between operation i and i+1 equal or larger than the processing time of $O_{ij}$
    Insert $O_{ij}$ between operation i and i+1;
    The start time of $O_{ij}$ equal the end time of operation i;
    f=f+1;
    Break;
  Endif
Endfor
If f=0
  Insert $O_{ij}$ after the final operation which has been scheduled;
Endif
```

Fig. 3. The process of finding the earliest idle time.
476 iterations is met. The detail steps of ensemble local search can be described as follows:

1. Set \( l = 1 \).
2. Sort all machines based on the number of critical operations.
3. Set \( m = 1 \).
4. Ensemble local search
   4.1. Select a local search operator from the operator pool.
   4.2. Generate neighbor solution.
   4.3. If the neighbor solution is better than the current one, remain the operator; otherwise, delete the operator from the operator pool.
5. Set \( l = l + 1 \), if \( l \leq L \) (\( L \) is the maximum iteration), repeat (4) until the operator pool is empty; otherwise, stop and return the solution.
6. If the solution is improved, update the solution and go to (2).
7. Set \( m = m + 1 \), if \( m \leq M \) (\( M \) is machine number), go to (4); otherwise, go to (3).

4.5. Framework of two-stage TABC

The TABC algorithm employed multiple initialization strategies, novel machine assignment and operation sequencing operator for generating new solutions, ensemble local search method on machine. The exploitation and exploration are balanced and stressed in this algorithm. The proposed TABC algorithm includes two stages: initiating scheduling stage and re-scheduling stage. At the initiating stage, the start time of all jobs and machines is at time 0. After the initiating stage, the solution with the best objective value will be output, and all operations will be processed on corresponding machines based on the best solution. On new job(s) coming and inserting into the shop floor, the re-scheduling stage will be activated and reschedule the new job(s) and the operations that have not started processing. All not started operations and new job(s) will proceed based on the re-scheduling best solution. The computational procedure of TABC algorithm for scheduling and re-scheduling can be described as follows:

4.5.1. Initializing scheduling stage

1. Set parameters, including the number of employed bees, the number of onlooker bees and the number of scout bees. In TABC algorithm, the number of employed bees equal to the number of food sources.
2. Initialize population with multiple strategies shown in Section 4.2, evaluate each solution and determine the best food source.
3. Employed bee phase. For all populations, repeat the following sub-steps:
   3.1. For each pair of current solutions, generate two new solutions by using the strategy presented in Section 4.3.
   3.2. Improve the two new solutions based on the ensemble local search method shown in Section 4.4.
   3.3. If the new solutions are better than or equal to the current solutions, update the population.
4. Onlooker bee phase. For all populations, repeat the following sub-steps:
   4.1. Select two food sources for two onlooker bees by using tournament selection.
   4.2. Generate two new solutions for the two onlookers by using the strategy presented in Section 4.3.
   4.3. Improve the two new solutions based on the ensemble local search method shown in Section 4.4.
   4.4. If the new solutions are better than or equal to the current solutions, update the population.
5. Scout bee phase. If a solution is not been improved during the last limit number of trials, abandon it, generate a new solution randomly and execute the ensemble local search on it.
6. Update the best solution. If the termination criterion is reached, return the best solution; otherwise, go to step (3).

4.5.2. Re-scheduling stage

1. The new job(s) inserting in the scheduled and executing sequence.

Please cite this article in press as: Gao, K. Z., et al. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. Expert Systems with Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.06.004
(8) Calculate and record the start time of each job and each machine based on the inserting time.
(9) Re-calculate the operation number of existing jobs and add the new job(s) to the re-scheduling job set.
(10) Execute re-scheduling.
(11) Link up the re-scheduling results to executing scheduling results based on the inserting time.

5. Experiment evaluation and comparisons

5.1. Experimental setup

To test the performance of the proposed two-stage TABC algorithm, extensive experimental evaluation and comparisons with existing algorithms are provided using well-known FJSP benchmark sets. Two sets of instances are considered in this paper: (1) the first data set are five Kacem instances (Kacem, Hammadi, and Borne, 2002a, 2002b), (2) the second data set, called BRdata, is a set of 10 problems by Brandimare. The Kacem benchmark set is composed of 5 instances with the size ranging from 4 jobs, 5 machines and 11 operations to 15 jobs, 10 machines and 60 operations. The Brandimare benchmark set includes 10 problems with the size ranging from 10 jobs, 6 machines and 55 operations to 20 jobs, 16 machines and 240 operations.

To test the performance of TABC for FJSP with new job inserting and the three re-scheduling strategies provided in Section 2, an instance set from remanufacturing enterprise is considered. This instance set includes eight instances with the size ranging from 5 jobs, 4 machines and 23 operations to 20 jobs, 15 machines and 355 operations. There are 35 new jobs inserted to existing scheduling sequence of eight instances. Each inserting has different inserting time, job number and operation number. The detail information is shown in Table 1. The first two columns are instance number and new job inserting order. The third column is the inserting time of corresponding inserting order. The last three columns are the corresponding job number, machine number and operation number. The first row of each instance shows the job number, machine number and operation number at initializing scheduling (see Tables 2 and 3).

The TABC algorithm is coded in C++ and implemented on an Intel ® Core™2 Duo CPU P8600 @ 2.40 GHz PC with 4 GB RAM. Based on our previous research (Li et al., 2011; Taşgetiren, Pan, Suganthan, and Oner, 2013), the population size is fixed at 50. The sizes of employed bees, onlooker bees and scout bees are 50, 100 and 10, respectively. The predetermined number of trials is fixed at 50. The probability of crossover operation is 0.4, and the maximum iteration of ensemble local search is set to 5 times the number of available machines. The maximum generation is 3000. Each instance is carried out 30 replications.

5.2. Discussion and comparison of three re-scheduling strategies

For the eight instances from remanufacturing enterprise, the re-scheduling is based on the initiating scheduling or last time re-scheduling for new job(s) inserting. In this section, we just focus on the minimal makespan over 30 runs. To compare the three re-scheduling strategies, we also calculated the relative percentage increase (RPI) as follows:

$$RPI(C_M) = \frac{C_M - C^*_M}{C^*_M} \times 100$$

where \(C_M\) is the makespan value obtained in the ith replication, \(C^*_M\) is the best makespan value found by three re-scheduling strategies. Obviously, the smaller the RPI value, the better result the re-scheduling strategy produces. To show the algorithm performance, we also calculated the average relative percentage increase (ARPI) of each re-scheduling strategy. For each instance, the average run time over 30 replications also recorded for the performance of TABC algorithm and three re-scheduling strategies.

To illustrate the three re-scheduling strategies more clearly, the first time new job inserting re-scheduling of instance 2 is used as an example. Fig. 6 shows the initiating scheduling Gantt chart. The makespan value is 36. It means that all existing jobs will be completed and all machines are available at time “36” if there is no new job(s) inserting. The new job, Job 9, inserts at time “12”. At this time, the machine release time, job release time and corresponding operation number are shown in Table 4. For example, the fourth operation of job 1, O1.4, is processing on machine M3 on time “12”. This operation will be completed on time “13” on machine M3. Hence, the release time of job 1 is on time “13” and the start operation is O1.5. At the same time, release time of machine M3 is also on time “13”. In the same way, all job release time and machine can be obtained. For the new job J9, the release time is on time “12” and the first start operation is operation O9.1. The three re-scheduling strategies’ Gantt charts are shown in Figs. 7–9. Fig.7 is the Gantt chart of re-scheduling strategy I. The new inserting job, Job 9 is re-scheduling from time “36” and the final complete time is 63. Fig.8 shows the Gantt chart of re-scheduling strategy II. The re-scheduled operations include Job 9’s operations and the existing jobs’ operations that have no start time new job inserting. Fig.9 shows the Gantt chart of re-scheduling strategy III. The new inserting jobs’ operations have different and new operation sequence. The Job 9’s operations are also included in the new operation sequences. The final complete time is 44 by re-scheduling strategy II. The Gantt chart by re-scheduling strategy III is shown in Fig.9. It can be seen that the existing jobs’ operations are the same with initiating scheduling, shown in Fig.6. The first three operations of Job 9 are inserted into the free-time-gap of existing operation sequence. The fourth operation’s start time is time “35” which is also earlier...
It is clear that the re-scheduling strategy II is more appropriate than re-scheduling strategy III if the new Job 9 has the same priority with the existing jobs. The re-scheduling strategy III should be selected if the existing jobs are priority completed. In practice

<table>
<thead>
<tr>
<th>Instance</th>
<th>Inserting order</th>
<th>Re-sche I</th>
<th>Re-sche II</th>
<th>Re-sche III</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cm</td>
<td>Time (s)</td>
<td>ARPI (%)</td>
<td>Cm</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>25</td>
<td>1.032</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>36</td>
<td>0.156</td>
<td>33.33</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>36</td>
<td>0.219</td>
<td>12.50</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>36</td>
<td>4.203</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>63</td>
<td>0.235</td>
<td>38.64</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>77</td>
<td>0.312</td>
<td>57.14</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>55</td>
<td>5.641</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>77</td>
<td>0.281</td>
<td>32.76</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>111</td>
<td>0.261</td>
<td>68.18</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>44</td>
<td>7.938</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69</td>
<td>1.020</td>
<td>35.29</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>103</td>
<td>1.111</td>
<td>71.67</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>3.842</td>
<td>31.39</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Inserting order</th>
<th>Re-sche I</th>
<th>Re-sche II</th>
<th>Re-sche III</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cm</td>
<td>Time (s)</td>
<td>ARPI (%)</td>
<td>Cm</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>79</td>
<td>19.813</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>106</td>
<td>0.904</td>
<td>29.27</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>132</td>
<td>0.817</td>
<td>48.31</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>73</td>
<td>25.157</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>97</td>
<td>1.008</td>
<td>32.88</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>115</td>
<td>0.939</td>
<td>49.35</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>96</td>
<td>53.594</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>140</td>
<td>2.423</td>
<td>26.13</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>168</td>
<td>1.020</td>
<td>35.29</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>91</td>
<td>112.078</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>116</td>
<td>2.865</td>
<td>24.73</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>147</td>
<td>2.800</td>
<td>37.70</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>16.731</td>
<td>30.14</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. The Gantt chart of initiating scheduling.

Table 4
Machine release time, job release time and corresponding operations.

<table>
<thead>
<tr>
<th>Order</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job release time</td>
<td>13</td>
<td>12</td>
<td>17</td>
<td>13</td>
<td>19</td>
<td>18</td>
<td>13</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>Job start operation</td>
<td>1.5</td>
<td>2.4</td>
<td>3.3</td>
<td>4.4</td>
<td>5.5</td>
<td>6.6</td>
<td>7.6</td>
<td>8.4</td>
<td>9.1</td>
</tr>
<tr>
<td>Machine release time</td>
<td>16</td>
<td>17</td>
<td>13</td>
<td>18</td>
<td>19</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

Please cite this article in press as: Gao, K. Z., et al. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. Expert Systems with Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.06.004
remanufacturing environment, re-scheduling strategy III may also be selected if the operations of existing jobs cannot be moved from assigned machine. Although three re-scheduling obtain different performance, each of them may be selected in practice shop floor.

5.3. Comparison with existing algorithms

In initializing scheduling phase, the job start time and machine release time are the same, zero. Hence, the problem is a general FJSP problem. In re-scheduling phase, the job start time may be different and the machine release time may be different. The problem becomes FJSP problem with different job start time and different machine release time. In addition, the job number is also different in different re-scheduling operators. In this section, we compare the TABC algorithm to several existing algorithms for FJSP in initializing phase. For the re-scheduling phase, we compare TABC algorithm with several simple and combination heuristics. The results and discussions are shown in Sections 5.3.1 and 5.3.2.

5.3.1. Initializing scheduling

There are many existing meta-heuristics for solving flexible job shop scheduling problem with makespan criterion. To test the performance of proposed algorithm, we compare TABC algorithm to five competitive meta-heuristics, parallel variable neighborhood search (PVNS), knowledge-based ant colony optimization (KBACO), tabu search algorithm with efficient neighborhood structure (TSPCB) effective artificial bee colony algorithm (EABC) and a simple and effective evolutionary algorithm (SEA) (Chiang & Lin, 2013). The makespan values for all the algorithms are shown in Table 5.
For five Kacem instances, all compared algorithms can obtain the best known makespan as long as the instances were solved. The only exception is that PVNS did not reach the best known makespan for instance MK10.

5.3.2. Re-scheduling for new job inserting

Several simple heuristics are proposed for machine assignment and operation sequencing in FJSP problem. For example, machine with minimum workload heuristic (MSC) is for machine assignment while job with maximum remain work (MSB) and job with maximum remain operations (MSC1) are two heuristics for operation sequencing. In this section, we improve some simple heuristics and develop combined heuristics based on simple heuristics. For example, random job order in machine with minimum workload heuristic (RMSC), MSC + MSB heuristic, RMSC + MSB heuristic, MSC + MSC1 heuristic, RMSC + MSC1 heuristic etc. are combined. MSC1 heuristic is used to obtain the operation sequence in RH heuristic. Based on the initializing scheduling results of simple heuristics and the combination heuristics, we compare TABC algorithm to EABC, TSPCB and develop combined heuristics based on simple heuristics.

Table 5

<table>
<thead>
<tr>
<th>Instance</th>
<th>PVNS</th>
<th>KBACO</th>
<th>TSPCB</th>
<th>EABC</th>
<th>SEA</th>
<th>TABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kacem 1</td>
<td>N/A</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>0.00</td>
</tr>
<tr>
<td>Kacem 2</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>0.00</td>
</tr>
<tr>
<td>Kacem 3</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>2.97</td>
</tr>
<tr>
<td>Kacem 4</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>1.4</td>
</tr>
<tr>
<td>Kacem 5</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>2.97</td>
</tr>
<tr>
<td>MK01</td>
<td>40</td>
<td>39</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>0.53</td>
</tr>
<tr>
<td>MK02</td>
<td>26</td>
<td>29</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>0.81</td>
</tr>
<tr>
<td>MK03</td>
<td>204</td>
<td>204</td>
<td>204</td>
<td>204</td>
<td>204</td>
<td>0.00</td>
</tr>
<tr>
<td>MK04</td>
<td>60</td>
<td>65</td>
<td>62</td>
<td>60</td>
<td>61</td>
<td>0.83</td>
</tr>
<tr>
<td>MK05</td>
<td>173</td>
<td>173</td>
<td>172</td>
<td>173</td>
<td>173</td>
<td>0.00</td>
</tr>
<tr>
<td>MK06</td>
<td>60</td>
<td>67</td>
<td>65</td>
<td>60</td>
<td>65</td>
<td>2.02</td>
</tr>
<tr>
<td>MK07</td>
<td>141</td>
<td>144</td>
<td>140</td>
<td>139</td>
<td>139</td>
<td>0.91</td>
</tr>
<tr>
<td>MK08</td>
<td>523</td>
<td>523</td>
<td>523</td>
<td>523</td>
<td>523</td>
<td>0.00</td>
</tr>
<tr>
<td>MK09</td>
<td>307</td>
<td>311</td>
<td>310</td>
<td>307</td>
<td>307</td>
<td>1.81</td>
</tr>
<tr>
<td>MK10</td>
<td>208</td>
<td>229</td>
<td>214</td>
<td>208</td>
<td>225</td>
<td>4.98</td>
</tr>
<tr>
<td>No. CM</td>
<td>2+6</td>
<td>5+3</td>
<td>5+4</td>
<td>5+8</td>
<td>5+3</td>
<td>8+5</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>Instance</th>
<th>Inserting order</th>
<th>MSC</th>
<th>RMSC</th>
<th>RMSC + MSB</th>
<th>RMSC + MSC1</th>
<th>MSC1 + RH</th>
<th>RH</th>
<th>TABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>35</td>
<td>43.4</td>
<td>43.4</td>
<td>43.4</td>
<td>27.9</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>43</td>
<td>31</td>
<td>31.75</td>
<td>31.75</td>
<td>34.5</td>
<td>34</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>56</td>
<td>34</td>
<td>34.31</td>
<td>34.31</td>
<td>34.5</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>90</td>
<td>57.3</td>
<td>57.3</td>
<td>57.3</td>
<td>49.0</td>
<td>49</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>107</td>
<td>64.1</td>
<td>64.1</td>
<td>64.1</td>
<td>57.0</td>
<td>57</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>102</td>
<td>63.2</td>
<td>63.2</td>
<td>63.2</td>
<td>63.0</td>
<td>63</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>106</td>
<td>81.8</td>
<td>81.8</td>
<td>81.8</td>
<td>69.0</td>
<td>69</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>123</td>
<td>83</td>
<td>83.57</td>
<td>83.57</td>
<td>71.0</td>
<td>71</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>131</td>
<td>84.6</td>
<td>84.6</td>
<td>84.6</td>
<td>78.0</td>
<td>78</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>145</td>
<td>97.8</td>
<td>97.8</td>
<td>97.8</td>
<td>86.0</td>
<td>86</td>
<td>86</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>111</td>
<td>71.0</td>
<td>71.0</td>
<td>71.0</td>
<td>58.0</td>
<td>58</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>116</td>
<td>75.8</td>
<td>75.8</td>
<td>75.8</td>
<td>67.0</td>
<td>67</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>140</td>
<td>94.4</td>
<td>94.4</td>
<td>94.4</td>
<td>81.0</td>
<td>81</td>
<td>74</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>195</td>
<td>113.5</td>
<td>113.5</td>
<td>113.5</td>
<td>97.0</td>
<td>97</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>199</td>
<td>125.5</td>
<td>125.5</td>
<td>125.5</td>
<td>110.0</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>216</td>
<td>131.5</td>
<td>131.5</td>
<td>131.5</td>
<td>113.0</td>
<td>113</td>
<td>110</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>174</td>
<td>101.7</td>
<td>101.7</td>
<td>101.7</td>
<td>91.0</td>
<td>91</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>184</td>
<td>111.5</td>
<td>111.5</td>
<td>111.5</td>
<td>97.0</td>
<td>97</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>192</td>
<td>113.2</td>
<td>113.2</td>
<td>113.2</td>
<td>106.0</td>
<td>106</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>220</td>
<td>111.2</td>
<td>111.2</td>
<td>111.2</td>
<td>106.0</td>
<td>106</td>
<td>97</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>263</td>
<td>122</td>
<td>122.3</td>
<td>122.3</td>
<td>120.0</td>
<td>120</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>267</td>
<td>142</td>
<td>142.6</td>
<td>142.6</td>
<td>137.0</td>
<td>137</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>303</td>
<td>165.9</td>
<td>165.9</td>
<td>165.9</td>
<td>153.0</td>
<td>153</td>
<td>140</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>223</td>
<td>110</td>
<td>110.2</td>
<td>110.2</td>
<td>99.0</td>
<td>99</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>258</td>
<td>118</td>
<td>118.5</td>
<td>118.5</td>
<td>108.0</td>
<td>108</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>275</td>
<td>134</td>
<td>134.1</td>
<td>134.1</td>
<td>118.0</td>
<td>118</td>
<td>110</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>268</td>
<td>140</td>
<td>140.9</td>
<td>140.9</td>
<td>123.0</td>
<td>123</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>168.3 90.5 99.8 5.2 110.0 5.1 85.8 93.6 4.5 84.3 81.3 90.3 5.0 69.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
It can be seen from Table 6 that MSC has the worst average for all compared algorithms, there are several special results. For example, MSC1 + RH heuristic obtain smaller first re-scheduling minimum result (71) than the second re-scheduling result (77). These situations mean that, after inserting new job, re-scheduling operator may remain or even reduce makespan value in some special situation. To compare different heuristics and TABC algorithm clearly, Figs. 10 and 11 show the initializing scheduling results and all re-scheduling results for the sixth instance and eighth instance. It is clear that TABC has better results than other heuristics in initializing scheduling and re-scheduling operator phase.

In summary, TABC algorithm is effective and efficiency for solving FJSP problem. For scheduling and re-scheduling FJSP with new job inserting, combination heuristics and RH has better performance than simple heuristics. TABC algorithm obtains better results than compared simple and combination heuristics in both initializing scheduling and re-scheduling phase. For all eight remanufacturing instances, TABC algorithm can get the same result in all 30 repeats. It shows that the TABC algorithm has very good convergence and stability. Compared against six heuristics, the disadvantage of TABC algorithm is requiring more time to get the best solution. Compare to the actual job shop processing time of operations in remanufacturing, the computation time to obtain best solution can be negligible.

6. Conclusions and future work

This paper modeled the remanufacturing scheduling problem as two stage FJSP with new job inserting. This is the first work to model the uncertainty in timing of returns as the new job inserting constraint in FJSP. A two-stage artificial bee colony algorithm is proposed to solve scheduling and rescheduling with new job inserting. Three re-scheduling strategies are proposed for rescheduling and compared. The best one is used in the TABC algorithm. Except scheduling stage, the TABC algorithm realize dynamic and multiple rescheduling processes for remanufacturing. Ensemble local search is proposed to improve algorithm’s performance. The performance of TABC algorithm is verified by comparing against eleven heuristic and meta-heuristics algorithms. In benchmark FJSP instances and eight remanufacturing cases are solved by the TABC algorithm effectively. These are some directions for future works. Additional local search heuristics can be developed and experimented to see if they are useful for solving some instances. Further, the developed algorithm can be applied to practical remanufacturing scheduling problems in order to serve remanufacturing enterprise.

Uncited references


Acknowledgments

This research is partially supported by foundation of A*Star (Agency for Science, Technology and Research, Singapore): Remanufacturing Scheduling Systems under Grant 112 290 4021, the National Science Foundation of China under Grant 61104179.

References

Computing, 45(4), 369–375.


