A Novel Improved Discrete ABC Algorithm for Manpower Scheduling Problem in Remanufacturing

Debabrota Basu[1], Shantanab Debchoudhury[1],
Kai-Zhou Gao[2], and Ponnuthurai Nagaratnam Suganthan[2]

[1] Dept. of Electronics & Telecommunication Engineering,
Jadavpur University, Kolkata, India
[2] School of EEE, Nanyang Technological University, Singapore, 639798
basudebabrota29@gmail.com

Abstract. Remanufacturing technique is a widely used approach in modern industries. But the very first step of this technique is disassembling. This disassembling operation requires an efficient employee pool and their allocation to several steps of disassembling. In this paper, we have proposed a improved ABC algorithm that can be used to solve the manpower scheduling problem for the disassembling operation in remanufacturing industry. We test this algorithm on several instances along with some existing state-of-art algorithms. The results prove the efficiency of this algorithm to solve manpower scheduling problem in remanufacturing.

Keywords: Remanufacturing, Man-power scheduling, Artificial Bee Colony algorithm, ID-ABC.

1 Introduction

Remanufacturing is a process of product recovery used in industries [1]. In this process, used products or modules are disassembled to its basic components among which the faulty components are cleaned, repaired and replaced with new ones. After this process of disassembling and repairing these new sets of basic components are reassembled to produce the main product or module which will be in sound working condition.

In this point of time when humanity is facing the problem of global warming and the lack of renewable energy and resources remanufacturing is an important process. Remanufacturing can help us to use the resources again and again without any degradation of product quality and performance. Thus, remanufacturing is considered as the *ultimate form of recycling* and now-a-days it is an interesting topic for researchers [2]. Even remanufacturing is practically used in several countries across the globe for remanufacturing of several products like aerospace, air-conditioning units, bakery equipments, computer and telecommunication equipment, defence equipments, robots, vending machines, motor vehicles and many more. This environmental edge of remanufacturing process [3] has inspired us to deal with this problem.

B.K. Panigrahi et al. (Eds.): SEMCCO 2013, Part I, LNCS 8297, pp. 738–749, 2013.

But efficient disassembling of these products or modules require an efficient employee pool who will have a knowledge of these equipments and the knowledge of steps through which the product or the module can be disassembled to its basic components from which it can be reassembled to build up the actual product which can work as efficient as a new one. Here comes the problem of scheduling man force in disassembling process of remanufacturing industry. Because after one product or its part is divided into sub-components then disassembling of those sub-components become independent of each other. Besides this, every employee cannot do every operation of the disassembling. Rather there exists a certain set of employees with certain skill sets for each of the operations involved in disassembling. Thus, the parallelism and presence of constraints make this problem a challenging combinatorial optimization problem which can hardly be solved by hands for large or real time cases.

There are several heuristics and meta-heuristics which have already been developed to produce efficient solutions for several combinatorial problems like job shop scheduling problems, man power scheduling problem[4]-[6] etc. Shifting Bottleneck Procedure [7] has proved to be the most successful heuristic for scheduling problems. But the shortcoming of these algorithms is they are too much problem specific. But due to robustness and exceptional performance of several bio-inspired algorithms, now-a-days new meta-heuristic approaches are invented and used for solving these scheduling problems. Goldberg [8] first used Genetic Algorithm to solve the scheduling problems. Ho *et al* [9] used modified Tabu Search technique to solve man power scheduling problem for airline catering. But there is hardly any significant work in solving the manpower scheduling for disassembling industry where the problem follows a parallel tree pattern rather than two ended graphical pattern followed by other scheduling problems.

Here, we have proposed a novel meta-heuristic approach, named as Improved Discrete ABC (ID-ABC) based on ABC algorithm to solve this problem. Artificial Bee Colony (ABC) algorithm [10] is an efficient global optimization tool which was first developed by Karaboga *et al.* Though it was first proposed as a continuous optimizer, later researchers have developed several modifications of it and even developed discrete versions of it to solve several combinatorial optimization problems like job shop scheduling problem [11]-[12], travelling salesman problem [13] etc.

Rest of the article is presented as follows: in the next section, we have described the mathematical formulation of the man power scheduling problem for disassembling process using integer programming. After that in section 3, the basics of ABC algorithms is described in brief and in section 4, the improved discrete ABC algorithm (ID-ABC) is proposed with its pseudo code. In the very next section, this algorithm is tested on several problem instances and the result is compared with several state-of-art algorithms, which proves the efficiency of this algorithm to solve the scheduling problem. In section 6 we have stated the conclusion of our paper where we have explored the efficiency of our algorithm and the possible future works that can be done using this model and algorithm.

2 Mathematical Modeling of Scheduling Problem

Let us consider the following tree structure for the arrangement of operations in the disassembling process where a single product or module has to be divided in n basic components through a set of operations $\mathbb{O}$. Here, we may consider each of the operations as the nodes of the tree where the main product will be the root of the tree and the final components will be the leaves. Now, there is a set of m employees $\mathbb{E}$ working on this process where every node i has its own set of eligible employees E_i such that, $E_i \subseteq \mathrm{E}$ and $\bigcup_{i \in \mathbf{O}} E_i = \mathrm{E}$. The time required for the j^{th} employee to perform the i^{th} operation is given by t_{ij}, where $j \in E_i$. To clarify the problem statement let us consider the following example: there are 4 employees and a product has to be disassembled into 4 basic components, where the disassembling process will follow the tree as shown in fig 1. Where 5 leaves represent 5 basic components that has to be derived from the original one and the root node or operation 0 represents the basic module.

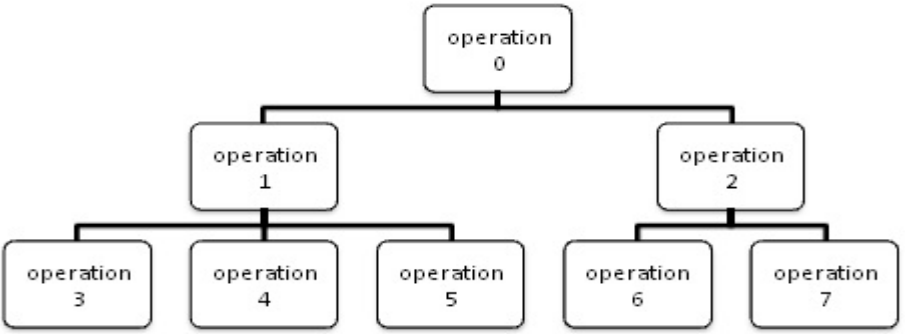

Fig. 1. The disassembly operation tree for 5components, 4employee problem

Now, the tree shown above contains 7 nodes representing 7 disassembled modules where each of them have their own eligible employee list and processing time list for each of them. As shown in table 1 below:

Table 1. List of eligible employees and corresponding processing times for each operation

Operations	*Employee(Processing_time)*		
1	4(10)	3(15)	
2	1(5)	3(10)	
3	1(6)	2(9)	4(8)
4	3(7)	4(10)	
5	2(5)	3(6)	
6	2(4)	4(7)	
7	1(11)	3(8)	4(10)

Before presenting the problem statement let us state the basic problem statements and assumptions of this problem:

1. Each employee is ready to perform the operation at the time t=0 and there is no time they need to take between the two successive operations.
2. Two operations belonging to two different subtrees are independent of each other but the operations belonging to the same subtree are not independent because it cannot be fulfilled until its parent unit (node) is disassembled.
3. No interruption of an operation is allowed-each must be scheduled into a single contiguous time interval.
4. The processing times, the employee list and the operation division tree are known previously, as shown in fig. 1. and table 1.
5. There is a certain non-zero waiting time for the employees working on the nodes after the level 1 of the tree. We can represent the waiting time for j^{th} employee to perform the i^{th} operation is given by w_{ij}, where $j \in E_i$ and

$$w_{ij} = \max\left\{ \sum_{i\in\mathbb{C}} p_{ij}, \sum_{i\in\mathbb{Q}} p_{ij} \right\} \quad where, \quad \mathbb{C} = set\ of\ all\ nodes\ from\ i^{th}\ operation\ to\ the\ root\ node \tag{1}$$

$$\mathbb{Q} = set\ of\ all\ nodes\ from\ where\ j^{th}\ employee\ is\ assigned\ before\ i^{th}\ operation$$

Now, if $\mathbb{S}$ is a possible and valid permutation of the operations and corresponding employee list, then the makespan of the disassembling process can be described as:

$$C(i) = p_{ij,} \qquad \text{if the operation } i \text{ is in the level 1} \tag{2}$$

$$C(i) = p_{ij} + w_{ij}, \qquad \text{if the operation } i \text{ is not in the level 1} \tag{3}$$

$$C(\mathbf{S}) = \max\{C(i)\}, \quad \text{if the operation } i \text{ is a leaf i.e., basic component} \tag{4}$$

The objective of our problem is to find such a permutation of operations and employees $\mathbb{S}$, such that the makespan $C(\mathbf{S})$ will be minimum.

3 Artificial Bee Colony Algorithm (ABC)

Artificial bee colony algorithm (ABC) is one of the most popular and efficient swarm algorithm developed in recent times [10]. In this algorithm three bee phases are repeated iteratively until the termination condition is reached. The employed bees are responsible for exploiting the food sources, and they share food sources information with onlooker bees which are waiting in the hive to make the decision to choose food sources. This phase acts as a positive feedback mechanism; whereas scout bees carry out a random search near the hive for new food sources, which activates the negative

feedback mechanism. Each food sources are represented by n-dimensional real-valued vectors which denote solutions to the problem under consideration, whereas the nectar amount of the food resource is evaluated by the fitness value.

Now, movement of onlookers is controlled by probability of selecting the source, which is given by the following equation,

$$p_k = \frac{f_k}{\sum_{k=1}^{SN} f_k} \tag{5}$$

where, p_k : probability of selecting the k[th] employed bee, SN: no. of employed bees, and, f_k : the fitness value of the position where k[th] employed bee is placed.

Again, depending on the nectar densities as shown above the new positions are calculated as follows,

$$x_{ij}(t+1) = x_{ij}(t) + \phi_{ij} * (x_{ij}(t) - x_{kj}(t)) \tag{6}$$

x_i : Position of the onlooker bee, t: iteration number, x_k :randomly chosen employed bee, j : dimension of the solution, ϕ_{ij} : series of random variables in the range [-1,1].

This step is known as greedy selection strategy.

But, to activate the negative feedback mechanism and to keep the exploration property of ABC intact movement of scout bees is conducted according to,

$$x_{ij} = \min_j + \text{rand}(0,1) * (\max_j - \min_j) \tag{7}$$

4 Improved Discrete ABC (ID-ABC)

ABC algorithm is actually not formulated for discrete domains. It works in the continuous domain with its unique positive and negative feedback mechanism which makes it a strong optimization tool. But here our problem is mainly a combinatorial optimization problem. To adapt ABC algorithm in this discrete domain and to work effectively on this problem we have used some novel techniques with the state of art ABC algorithm which evolves this Improved Discrete ABC (IDABC) algorithm. The key processes of IDABC are discretization of ABC algorithm for the problem described in section (2), destruction and construction based iterative greedy (IG) heuristic [14] and the RIS local search algorithm with priority rules, which are discussed elaborately in the next subsections.

4.1 Intitialization

In this algorithm, we have considered each food source $\mathbf{S_k}$ as a job vector $\vec{\mathbf{J}}_k$ consisting of a sequence of jobs which has correspondence with another employee vector

$\overrightarrow{\mathbf{W}_k}$, where corresponding employees are selected for the jobs are sequenced. We can express it mathematically as follows:

$$\mathbf{S_k} = (\vec{\mathbf{J}}_k, \overrightarrow{\mathbf{W}_k}) \qquad where \;\; k = 1, 2, ...NP \tag{7}$$

where, $\vec{J}_k = (j_1, j_2, ..., j_n) \quad where, \;\; j_i \in \mathbf{O}$

and, $\overrightarrow{W_k} = (e_1, e_2, ..., e_n) \quad where, \;\; e_i \in E_{j_i}$

The solution space which includes ***NP*** such food sources is generated randomly. Here, each of the food sources is also randomly assigned with a destruction size d_k. This destructor size is allocated on the food sources as they will generate their neighbors depending on the destruction and construction process of iteratively greedy (IG) heuristic which is discussed in the section (4.5).

4.2 Employeed Bee Phase

The employed bees generate food sources in the neighbourhood of their current positions. It is basically the search phase of ABC algorithm where the swarm explores the search space. For neighbour generation, we have taken three different methods: insert, swap and construction destruction with size d_k. In case of every employed bee of the population any one of the above perturbation methods are adopted in a random manner. It enriches the neighbourhood structure and diversifies the population. The swap operator randomly selects two positions on the corresponding food sources and interchanges their corresponding positions. The insert operator takes any random position of the food source and inserts it somewhere else in the food source randomly. The destruction and construction method is as shown below,

Pseudocode 1. **Destruction and Construction Process**

```
1  //Destruction Process//
   for iter = 1to d_k
      S_Dk = remove one job and the corresponding employee
           from the sequence S_k and insert it into the
           sequence S_Rk
   end
2  //Construction Process//
   for iter = 1to d_k
       S = best permutation obtained by inserting i^th job
           and the corresponding employee pair from the
           sequence S_Rk and insert it into the sequence S_D
   end
```

After generating a pool of such food sources, RIS local search along with the priority rules of scheduling is applied to further improve the solution quality.

4.3 Onlooker Bee Phase

In this phase, the greedy selection process is applied on the food sources to select out the best set of food sources. Generally, systems like roulette wheel are used here to select the best set of food sources. The onlooker bees utilize the same method as used by the employed bee to produce a new neighbouring solution S_{new} in the neighbour of the selected food sources. Then among the final pool of food sources the selection is done on the basis of tournament selection. Here, 2 of the food sources will be chosen randomly and the source with minimum makespan will be retained for the next step. Again, RIS search with priority is used on them to give legal and good quality solutions.

4.4 Scout Bee Phase

In the basic ABC algorithm, a scout bee produces a food source randomly in the predefined search space. This will decrease the search efficacy, since the best food source in the population often carries better information than others during the search process, and the search space around it could be the most promising region. Therefore, in the DABC algorithm, a tournament selection with the size of two is again used to discard a solution in such a way that two random food sources are picked up from the population and the worst one is selected. Then the scout generates a food source by performing a DC with the best destruction size d_{best} to the best solution in the population and replaced with the food source determined by tournament selection.

4.5 IG Algorithm

In the IG algorithm used here, the destruction phase is concerned with removing randomly d number of jobs and corresponding employee components from a previously constructed solution S to generate two partial solutions S_R and S_D, whereas construction phase is related to the reconstruction of a complete solution S_{new}, by using a greedy constructive heuristic. Here, we have employed NEH heuristic as described in [15]. An acceptance criterion is then used to decide whether or not the reconstructed solution will replace the incumbent solution.

Pseudocode 2. **Iterative Greedy algorithm**

```
1 S0 = initially generated solution
2 S = Local Search (S0)
3 While (termination condition is not met)
            SD = Destruction (S)
            Snew = Construction (SD, SR)
            Snew* = Local Search (Snew)
            S = Aceeptance_criterion (S, Snew)
  end while
```

4.6 Reference Insertion Search With Priority Rules for Constraint Handling

The main purpose of the local search is to generate a better solution or food source from the neighbourhood of a given solution or food source. In this paper we have adopted a reference insert neighbourhood based local search, which has been regarded as superior to the swap or exchange neighbourhood [16].

Pseudocode 3. **RIS Local search algorithm with priority rule**

1 Set i =1

2 Set $\vec{J} = (j_1, j_2, ..., j_n)$, $\overrightarrow{W} = (e_1, e_2, ..., e_n)$ and $\mathbf{S} = (\vec{\mathrm{J}}, \overrightarrow{\mathrm{W}})$

3 Extract a certain job j_i randomly without repetition from $\vec{J}$ and remove it from permutation. In a similar manner the corresponding employee will be extracted from the employee vector $\overrightarrow{W}$.

4 For k = 1 to n

a Re-insert j_i in another different position of the permutation and adjust permutation accordingly by not changing the relative positions of the other jobs to get $\mathbf{S}^*$

b If $\mathbf{S}^*$ is better than $\mathbf{S}$, then let $S = S^*$

end for

5 $i = i+1$.

6 If i ≤ n, then go to step 2 else stop.

Here, we have to decide whether $\mathbf{S}^*$ is better than $\mathbf{S}$ or not. For that we have used the superiority of feasible (SF) approach [17] for constrained optimization, where the priority of a certain solution, here permutation or food source, is decided based on lexicographic ordering and constraint violation and objective function value are distinguished. The aim of this approach is to optimize constraint optimization problems by a lexicographic order where constraint violation gets higher priority over objective function value.

According to superiority of feasible (SF) approach the thumb-rules to decide the superiority of a certain solution $\mathbf{S}$ over $\mathbf{S}^*$ can be given as follows:

1) $\mathbf{S}^*$ is feasible and $\mathbf{S}$ is not.
2) $\mathbf{S}^*$ and $\mathbf{S}$ are both feasible and $\mathbf{S}^*$ has a smaller objective function value than $\mathbf{S}$.

Thus, finally we can summarize the total MoDABC algorithm as follows:

Pseudocode 4. **IDABC algorithm**

1 Set parameters
2 //Initialization//
  Establish initial populations randomly with NP food sources where each food source S contains two n dimensional vectors, **J** presenting a sequence of jobs and **E** a sequence of employee for the corresponding jobs.
3 Assign a destruction size d_i to each food source in the population which is assigned randomly in the range [1, n-1].
4 Evaluate population and find S_{best} and d_{best}.
5 //Employed Bee Phase//
  Repeat the following for each employed bee S_i
  a Generate a new food source by $S_{new} = DC_i(S_i, d_i)$
  b Apply local search algorithm to S_{new}
  c if $f(S_{new}) < f(S_i), S_i = S_{new}$
    else $d_i = rand() \% (n-1)$
  d if $f(S_{new}) < f(S_{best}), S_{best} = S_{new}$ and $d_{best} = d_i$
6 //Onlooker Bee Phase//
  Repeat the following for each onlooker bee S_k

  a Select a food source S_k = Tournament Selection($S_k \in \mathbb{S}$)
  b Generate a new food source by $S_{new} = DC_k(S_k, d_k)$
  c Apply RIS local search to S_{new}
  d if $f(S_{new}) < f(S_k)$, $S_k = S_{new}$
    else $d_k = rand()\%(n-1)$
  e if $f(S_{new}) < f(S_{best}), S_{best} = S_{new}$ and $d_{best} = d_i$
7 //Scout Bee Phase//
  Repeat the following steps for each scout bee π_k

  a Select a food source S_k = Tournament Selection($S_k \in \mathbb{S}$)
  b Generate a new food source by $S_{new} = DC_k(S_k, d_{best})$
  c Apply VNS local search to S_{new}
  d if $f(S_{new}) < f(S_k)$, $S_k = S_{new}$
    else $d_k = rand()\%(n-1)$
  e if $f(S_{new}) < f(S_{best}), S_{best} = S_{new}$
  f if the stopping criterion is not met, got to Step 5,
    else stop and return S_{best}.

5 Experimental Results and Discussions

We have tested our proposed algorithm on 10 different problem instances generated in accordance with real time circumstances each with different employee numbers and different final components. Four are relatively easy scheduling problems where number of operators is limited to a maximum of 6 and depth of disassembling is at most 4. Four cases require assignment of medium level of difficulty where depth of tree is at most 7 and number of operators involved is at most 10. The remaining two cases pose a difficult assignment challenge with involvement of 15 operators and a depth level of 10.

Table 2. Comparison of results of MoDABC and other algorithms for several instances

Instance No.	Makespan obtained and algorithmic run time					
	GA		ABC		ID-ABC	
	Makespan	RunTime(sec)	Makespan	RunTime(sec)	Makespan	RunTime (sec)
1	20	0.59634	20	0.54972	20	0.50070
2	45	0.63181	35	0.52531	35	0.50090
3	45	0.75019	36	0.50924	30	0.50281
4	41	0.80111	41	0.73677	41	0.68710
5	135	1.33270	101	1.17296	100	0.99364
6	110	1.23306	83	1.23467	83	1.06776
7	116	1.10739	85	0.97186	85	0.82784
8	114	1.11338	83	0.96689	83	0.92827
9	198	1.76803	128	1.57978	125	1.66963
10	189	1.89628	157	1.87293	157	1.74956

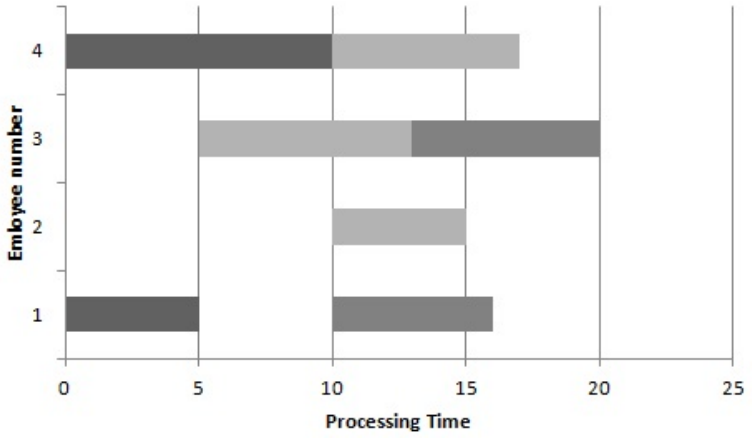

Fig. 1. Gantt chart obtained using ID-ABC for a 5components, 4 employee problem

The efficiency of ID-ABC is shown in table 2 by comparing its results with state-of-art algorithms like ABC and GA. Besides this, the gnat-chart [18] for instance number 1 with 5 employee and 4 components generated by ID-ABC is shown in the fig 2.

The parameters used in the MoDABC algorithm vary over a wide range depending on the problem dimensions. But we have generally taken the dimension of each food source as the number of operations in the problem, total number of food sources ***NP*** as the product of total number of components and the number of employee. But there is no restriction like that but it gives kind of thumb rule for the population size. Again, the number of onlooker bee is twice of ***NP*** and the size of the scout population is 0.25****NP***. The other two algorithms are also run with the same population sizes for each problem.

6 Conclusions and Future Possibilities

In this paper, we have proposed the manpower scheduling for disassembling process which is one of the foremost and basic step of remanufacturing industry. We have also presented a tree type structure and a mathematical formulation of makespan for this problem. This formulation opens a scope of application of several algorithms in this problem and using this model we can also develop many efficient heuristics and meta-heuristics in future which can solve the problem efficiently. Here, we have proposed a meta-heuristic ID-ABC which proves itself very efficient to solve this scheduling problem when compared with other state-of-art algorithms. But still there is scope of improvements in this algorithm which will make it more robust and applicable for even other discrete problems too. We can mold this algorithm with some efficient local search heuristic to make it more efficient for other combinatorial optimization problems. So, both the problem and the algorithm can be considered as future topics for research.

References

1. Remanufacturing, link: http://en.wikipedia.org/wiki/Remanufacturing
2. The Remanufacturing institute, link: http://reman.org/AboutReman_main.htm
3. http://www.remanufacturing.org.uk/
4. Lau, H.C.: On the complexity of manpower scheduling. Computers Ops. Res. 23(1), 93–102 (1996)
5. Shahrezaei, P.S., Moghaddam, R.T., Kazemipoor, H.: Solving a multi-objective multi-skilled manpower scheduling model by a fuzzy goal programming approach. Applied Mathematical Modelling 37, 5424–5443 (2013)
6. Pan, Q.K., Suganthan, P.N., Chua, T.J., Cai, T.X.: Solving manpower scheduling problem in manufacturing using mixed-integer programming with a two-stage heuristic algorithm. Int. J. Adv. Manuf. Technol. 46, 1229–1237 (2010)
7. Joseph, A., Egon, B., Daniel, Z.: The Shifting Bottleneck Procedure for Job-shop Scheduling. Management Science 34(3) (March 1988) (printed in USA)

8. Goldberg, D.E.: Genetic Algorithms in search. Addison-Wesley (1994)
9. Ho, S.C., Leung, J.M.Y.: Solving a manpower scheduling problem for airline catering using metaheuristics. European Journal of Operational Research 202, 903–921 (2010)
10. Karaboga, D., Basturk, B.: A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. Journal of Global Optimization 39(3), 459–471 (2007)
11. Li, J., Pan, Q., Xie, S., Wang, S.: A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. Int. J. of Computers, Communications & Control VI(2), 286–296 (2011)
12. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review, 1–37 (2012)
13. Li, L., Cheng, Y., Tan, L., Niu, B.: A Discrete Artificial Bee Colony Algorithm for TSP Problem. In: Huang, D.-S., Gan, Y., Premaratne, P., Han, K. (eds.) ICIC 2011. LNCS, vol. 6840, pp. 566–573. Springer, Heidelberg (2012)
14. Jacobs, L.W., Brusco, M.J.: A local search heuristic for large set-covering problems. Naval Research Logistics Quarterly 42(7), 1129–1140 (1995)
15. Nawaz, M., Enscore Jr., E.E., Ham, I.A.: Heuristic algorithm for the m-machine, n-job flow shop sequencing problem. OMEGA 11(1), 91–95 (1983)
16. Ruben, R., Stutzle, T.: An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. Eur. J. Oper. Res. 187, 1143–1159 (2008)
17. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186, 311–338 (2000)
18. Geraldi, J., Lechter, T.: Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today. International Journal of Managing Projects in Business 5(4), 578–594 (2012)