sustainability

MDPI

Article

Balancing Disassembly Line in Product Recovery to Promote the Coordinated Development of Economy and Environment

Jia Liu [1,2,*] and Shuwei Wang [3,4]

1 School of Management and Economics, University of Electronic Science and Technology of China, Chengdu 611731, China
2 School of Management and Economics, Tianfu College of Southwestern University of Finance and Economics, Mianyang 621000, China
3 School of Economics and Management, Southwest Jiaotong University, Chengdu 610031, China; wangshuwei@my.swjtu.edu.cn
4 School of Information and Technology, Tianfu College of Southwestern University of Finance and Economics, Mianyang 621000, China
* Correspondence: liujia@tfswufe.edu.cn; Tel.: +86-816-6354-888

Academic Editor: Marc A. Rosen
Received: 21 December 2016; Accepted: 16 February 2017; Published: 21 February 2017

Abstract: For environmentally conscious and sustainable manufacturing, many more manufacturers are acting to recycle and remanufacture their post-consumed products. The most critical process of remanufacturing is disassembly, since it allows for the selective extraction of the valuable components and materials from returned products to reduce the waste disposal volume. It is, therefore, important to design and balance the disassembly line to work efficiently due to its vital role in effective resource usage and environmental protection. Considering the disassembly precedence relationships and sequence-dependent parts removal time increments, this paper presents an improved discrete artificial bee colony algorithm (DABC) for solving the sequence-dependent disassembly line balancing problem (SDDLBP). The performance of the proposed algorithm was tested against nine other approaches. Computational results evidently indicate the superior efficiency of the proposed algorithm for addressing the environmental and economic concerns while optimizing the multi-objective SDDLBP.

Keywords: product recovery; environmental sustainable development; artificial bee colony; sequence-dependent disassembly line balancing

1. Introduction

Shorter product life cycles and ever-increasing consumption result in a growing amount of waste production, resource reduction, and even environmental deterioration. Enhanced public environmental awareness and stricter government regulations, coupled with extended manufacturer responsibility, make manufacturers responsible for their post-consumed products [1,2]. In addition, customer demand and production cost reduction are also the main driving forces for recycling and reusing discarded products. Product recovery seeks to retrieve valuable materials and parts through recycling, refurbishing, or remanufacturing from the used products. It can decrease the amount of residues sent to landfills and diminish the quantity of toxic substances released into the environment [3]. Some enterprises of the electronics industry, such as Dell, HP, and Xerox, are engaging the cost effectiveness in the practice of product recovery, because it has the potential to increase and bring environmental benefits [4]. See [5,6] for more information on environmentally-conscious manufacturing and product recovery.

The first crucial step of product recovery is disassembly which involves the separation of the desired parts/components and hazardous materials from the returned products [5]. After disassembly, the valuable parts/components can be reused in new products. The hazardous materials will be disposed of in an appropriate manner, while the waste is sent to landfills. Therefore, disassembly is an environmentally friendly and profitable activity. For improving disassembly efficiency and promoting disassembly industrialization, a disassembly line is the most suitable way to carry out disassembly operations [7]. Designing and balancing the disassembly line has gained increasing attention from researchers due to its vital role in environmentally-conscious manufacturing.

The disassembly line balancing problem (DLBP) requires the assignment of disassembly tasks to an ordered sequence of workstations with disassembly precedence relations, while optimizing the effectiveness of some measures [8]. The DLBP was firstly introduced by Gungor and Gupta [3,9], and then described as a multi-objective combinatorial problem [7]. To deal with DLBP, mathematical programming techniques have been formulated [10–12], but they were inadequate to solve practical large-sized instances. The DLBP has been proven to be NP-complete [13]. Therefore, heuristic and meta-heuristic approaches are more suitable for solving DLBP. The standard ant colony optimization (ACO) and genetic algorithms (GA) were applied for attaining an (near) optimal solution for DLBP [14,15]. Later, researchers improved the heuristic and meta-heuristic algorithms according to the character of DLBP. Considering repair algorithms and a diversification strategy as fitness evaluation approaches, Aydemir-Karadag and Turkbey used an effective GA to deal with multi-objective stochastic DLBP with station paralleling [16]. Ding et al. proposed a multi-objective ant colony optimization (MOACO) algorithm for solving DLBP. Niche technology and Pareto dominance concepts were used to obtain the Pareto optimal solutions [17]. Prakash et al. presented a constraint-based simulated annealing (CBSA) algorithm, which used the constraint-based genetic operators integrated with the SA approach to determine the ordering and disassembly schedule [18]. Considering the uncertainty character of disassembly systems, Kalayci et al. presented a fuzzy DLBP with fuzzy task processing times. A hybrid discrete artificial bee colony algorithm (HDABC) was proposed to solve this problem [19]. Tuncel et al. addressed the DLBP using a Monte Carlo-based reinforcement learning technique. This approach can operate DLBP in deterministic and stochastic environments [20]. Avikal et al. proposed a heuristic using a Kano model, fuzzy analytic hierarchy process (Fuzzy-AHP), and modified technique for order performance by similarity to ideal solution (M-TOPSIS) method-based technique for solving DLBP under a fuzzy environment [21].

However, it should be noted that all of the abovementioned literatures do not consider the interaction between disassembly tasks. Actually, whenever precedence-free tasks interact, their task times may be influenced based on their performed order [22]. It was named as a sequence dependent problem and firstly introduced by Scholl et al. in the assembly line balancing problems (ALBP) [23]. Several meta-heuristic approaches are applied to solve the sequence-dependent assembly line balancing problems (SDALBP) [24–26]. Similar to the SDALBP, Kalayci and Gupta firstly introduced the sequence-dependent disassembly line balancing problem (SDDLBP) and utilized ant colony optimization (ACO) to address this NP-complete problem [27]. The extension of the basic DLBP by considering sequence-dependent task times has great relevance in real-world disassembly line settings. Then a series of meta-heuristic methods were proposed to solve SDDLBP, such as artificial bee colony [22], simulated annealing [28], particle swarm optimization [29], tabu search [30], variable neighborhood search [31], and hybrid genetic algorithms [32]. In this paper, we deal with the SDDLBP and propose an improved discrete artificial bee colony algorithm (DABC) to solve this problem. We also compare the performance of the DABC algorithm with other meta-heuristic approaches to demonstrate the superiority of the proposed method.

The rest of the paper is organized as follows: Problem definition and formulation are given in Section 2. Section 3 introduces the basic ABC briefly. Section 4 presents the improved DABC algorithm. A comparative computational study is given in Section 5, and a conclusion is provided in Section 6.

2. Problem Definition and Formulation

In this study, the SDDLBP is concerned with the paced disassembly line for a single model of product that undergoes complete disassembly. The model assumptions of SDDLBP can be found in [27]. In SDDLBP, the tasks i and j with no precedence relationship become sequence dependent whenever they interact with each other during their task times. If we remove j before i, additional operating movements are required and/or the most efficient disassembly methods cannot be used for removing j due to the impact of i on j. Simply put, if a precedence-free sequence-dependent task j is arranged to be processed before task i is done, the total processing time of task j is the task time t_j and sequence-dependent time increment value sd_{ij} added together.

A disassembly example of an eight-part PC disassembly process is illustrated in Figure 1. The figure shows the disassemble tasks (nodes), task times (numbers in parentheses), precedence relationships (solid line arrows), sequence-dependent relationships (dashed line arrows), and sequence-dependent time increments (numbers with bottom line). For a feasible sequence {1, 2, 5, 3, 6, 8, 7, 4}, because task 2 is assigned to be removed before task 3, task 3 will prolong the total processing time of task 2 by $sd_{32} = 4$, i.e., the total removal time of task 2 increases from 10 to 14 considering sequence-dependent time increment sd_{32}. Similarly, task 5 is disassembled before task 6, so $sd_{65} = 3$ should be added to the removal time of task 5.

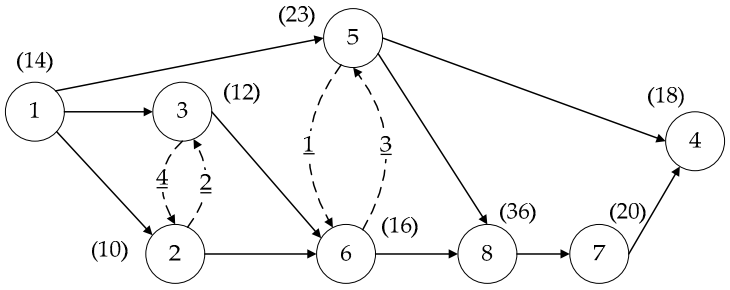

Figure 1. A simple example of an eight-part PC.

The parameters, decision variables, objective functions, and constraints of the proposed mixed integer non-linear programming model are defined as follows.

Parameters:

m Number of workstations; i.e., $k = 1, \ldots, m$
n Number of tasks, which is equal to the number of parts for removal; i.e., $i, j = 1, \ldots, n$
t_i Disassembly time of task i
sd_{ij} Sequence-dependent time increment influence of i on j
t_i' $t_i' = t_i + sd_{ij}$, total disassembly time of task i considering sequence-dependent relationships
P_l lth part in a disassembly sequence; e.g., for sequence {1, 5, 2, 3, 6, 8, 7, 4}, $P_2 = 5$
d_{Pl} Demand quantity for the lth part in a disassembly sequence
h_{Pl} Binary value; $h_{Pl} = 1$ if the lth part in a disassembly sequence is hazardous, else $h_{Pl} = 0$
ct Cycle time; maximum time available at each workstation
IP_i Set of parts for successors of task i

Decision variables:

$$z_k = \begin{cases} 1 \text{ if workstation } k \text{ is opened} \\ 0 \text{ otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1 \text{ if task } i \text{ is processed after task } j \\ 0 \text{ if task } i \text{ is processed before task } j \end{cases}$$

$$x_{jk} = \begin{cases} 1 \text{ if task } j \text{ is assigned to workstation } k \\ 0 \text{ otherwise} \end{cases}$$

Objective functions:

$$\min f_1 = \sum_{k=1}^{m} z_k \tag{1}$$

$$\min f_2 = \sum_{k=1}^{m} \left(ct - \sum_{j=1}^{n} t'_j \times x_{jk} \right)^2 \tag{2}$$

$$\min f_3 = \sum_{l=1}^{n} \left(l \times h_{P_l} \right) \tag{3}$$

$$\min f_4 = \sum_{l=1}^{n} \left(l \times d_{P_l} \right) \tag{4}$$

Constraints:

$$\sum_{k=1}^{m} x_{jk} = 1 \quad j = 1, \ldots, n \tag{5}$$

$$t'_j = t_j + \sum_{i=1}^{n} sd_{ij} \times y_{ij} \quad j = 1, \ldots, n \tag{6}$$

$$\sum_{j=1}^{n} t'_j \times x_{jk} \leq ct \times z_k \quad k = 1, \ldots, m \tag{7}$$

$$\left\lceil \frac{\sum_{i=1}^{n} t_i}{ct} \right\rceil \leq \sum_{k=1}^{m} z_k \leq n \tag{8}$$

$$x_{ik} \leq \sum_{k=1}^{m} x_{jk} \quad i = 1, \ldots, n, \forall j \in IP_i \tag{9}$$

In this paper, there are four objectives presented to address the environmental and economic concerns while optimizing the SDDLBP. Objective (1) minimizes the number of opened workstations in a given cycle time. This measure addresses investment, thus decreasing the devotion of human, equipment, and material resources which results in reducing the production costs and saving resources. Objective (2) distributes idle times across opened workstations evenly, which can also balance the workload among the workstations. This measure addresses efficiency, thus raising productivity and conserving energy, which lead to economic and environmental efficiency. Objective (3) processes hazardous parts/materials early. This measure addresses safety, thus recycling and reusing the uncontaminated subassemblies and handling hazardous materials, especially to protect the environment. Objective (4) removes high-demand parts as a priority. This measure addresses profit, thus obtaining demanded components for generating higher returns and reducing the quantity of residues for saving the environmentally-related costs. Economic and environmental concerns considered in SDDLBP are presented in Table 1.

Constraint (5) ensures that each task is assigned to only one workstation. Constraint (6) updates the removal time of disassembly tasks by considering sequence-dependent time increments. Constraint (7) forces the total operating time of each opened workstation to be less than or equal to the cycle time. Constraint (8) guarantees that the number of opened workstations cannot exceed the permitted number. Constraint (9) imposes that all the disassembly precedence relationships should be satisfied.

Table 1. Environmental and economic benefits considered in SDDLBP.

Objectives	Environmental Benefits	Economic Benefits
Objective 1: Minimize the number of opened workstations	Saves resources	Reduces the disassembly cost
Objective 2: Distribute idle times across opened workstations evenly	Conserves energy	Raises productivity
Objective 3: Process hazardous parts/materials early	Reduces pollution	Protects useful subassemblies
Objective 4: Remove high-demand subassemblies as a priority	Minimizes the space required for landfills	Maximizes profit

3. The Basic ABC Algorithm

The artificial bee colony (ABC) algorithm is a new member of swarm intelligence, which is based on the cooperative foraging behavior of a swarm of bees [33]. Due to the advantage of employing fewer control parameters, systematically incorporating exploration and exploitation mechanisms, the ABC algorithm is comparable, and sometimes superior, to other population-based algorithms. Therefore, it is suitable for solving large-scale complicated optimization problems [34].

In the ABC algorithm, each food source corresponds to a feasible solution and the nectar amount of a food source represents the quality (fitness) of the solution. The control parameters used in ABC are the number of food sources which is equal to the amount of employed bees or onlookers (*SN*), the number of trials after which a food source is to be abandoned (*uplimit*), and the termination iteration number (*maxIter*). There are three kinds of artificial bees cooperating to search for the optimal food source, named as the employed bees, onlookers, and scouts. Employed bees are responsible for exploiting food sources. Meanwhile, they gather and provide the food source information for the onlookers. Then the onlookers tend to choose better food sources to further exploit. If a food source is exhausted, i.e., the quality of a food source cannot be improved after the *uplimit* number of trials, the food source is assumed to be abandoned. The employed bee corresponding to that food source turns into a scout and explores food sources randomly. Whenever a new food source is found, the scout becomes an employed bee again [35].

Similar to other swarm intelligence-based approaches, ABC is an iterative algorithm. It starts with a group of initial food sources (i.e., solutions), then the following phases are performed by each type of bee, and are repeated until the termination iteration number is met [36]. The main steps of the ABC algorithm are given below.

(1) Initialization population phase

An initial population consists of *SN* food sources, and each of them is represented by an *n*-dimensional real-valued vector. Let $x_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ represents the *i*th food source, which can be produced by:

$$x_{ij} = LB_j + rand(0,1) \times \left(UB_j - LB_j\right) \; j = 1,2,\ldots,n \quad \text{and} \quad i = 1,2,\ldots,SN \tag{10}$$

where *rand* (0, 1) is a random number between [0, 1]; LB_j and UB_j are the lower and upper bounds of the dimension *j*, respectively.

(2) Employed bee phase

Each employed bee is responsible for finding a new solution x_i' in the neighborhood of its present position x_i using the following equation:

$$x'_{ij} = x_{ij} + \left(x_{ij} - x_{kj}\right) \times rand(-1,1) \; j = 1,2,\ldots,n \quad \text{and} \quad k = 1,2,\ldots,SN \tag{11}$$

where k and j are arbitrarily chosen indices. *rand* $(-1, 1)$ is a uniformly distributed real number between $[-1, 1]$. If the fitness value of x_i' is better than that of x_i, x_i' will replace x_i as a new food source, otherwise x_i is retained.

(3) Onlooker bee phase

An onlooker bee evaluates the nectar information shared by the employed bees and selects a food source x_i depending on its probability value P_i computed as follows:

$$P_i = \frac{f_i}{\sum_{i=1}^{SN} f_i} \tag{12}$$

where f_i is the objective value of x_i. The probability of the ith food source to be chosen increases with the increases of f_i-value. Once the food source x_i is selected, the onlooker bee will conduct a local search on x_i using the same method as the employed bees.

(4) Scout bee phase

In this phase, a food source x_i is assumed to be abandoned if it cannot be further improved through a fixed number of iterations. The corresponding employed bee then becomes a scout, and generates a new food source randomly by using Equation (10).

4. The Improved DABC Algorithm for SDDLBP

In order to raise search capacity and convergent speed, an improved DABC on the basis of ABC is proposed for solving the SDDLBP. In this algorithm, the permutation-based representation is used for presenting the solution. Random and heuristic initializations are blended to generate the initial solutions with diversity and high quality. Employed/onlooker bees apply a variable neighborhood descent (VND) strategy to locally search for enhancing the exploitation capability. With the purpose of breaking away from the local optimum and approaching to the global optimum rapidly, scout bees generate food sources by utilizing one point left/right operator to the best food source in the population. The following sub-sections details the sub-steps of the improved DABC.

4.1. Solution Representation

The SDDLBP is a discrete optimization problem. In the improved DABC algorithm, permutation-based representation is used. Each element of a solution string is an integer, which represents a task assigned to an opened workstation. As shown in Figure 1, there are eight tasks to be distributed to workstations, so the length of the solution string is 8. The sequence {1, 2, 5, 3, 6, 8, 7, 4} indicates that the tasks are assigned sequentially to workstations.

4.2. Population Initialization

Some heuristics are often used to create initial feasible solutions for the SDDLBP, such as longest processing time, maximum total number of successor tasks, maximum average ranked positional weight, and priority of positional constraints, and the details can be obtained in [37]. In the proposed algorithm, a mixed method of longest processing time and random selection are built to guarantee an initial population with high quality and diversity.

The generation procedure of a feasible solution is started by opening the first workstation, and tasks are assigned to this workstation successively until no more tasks can be assigned. Then, a new workstation is opened. In each iteration, the task from the assignable task set assigned to the current workstation is chosen according to a certain strategy. The procedure ends when all tasks are assigned [17]. The procedure is given in Figure 2.

Some remarks are given as follows:

(1) A task can be added to the available task set (C^*) if and only if it has not been assigned to a workstation and all of its predecessors have been assigned.

(2) A task can be added to the assignable task set (A^*) if and only if it belongs to the set of C^* and its total processing time considering sequence-dependent time increment is less than, or equal to, the idle time of the current workstation.

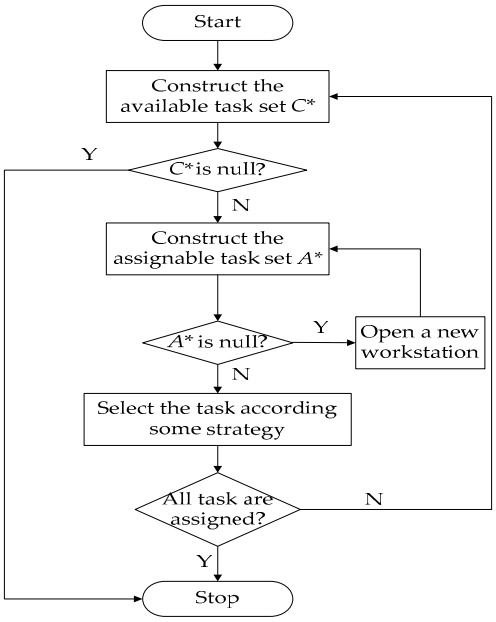

Figure 2. The generation procedure of the feasible solution.

4.3. The Improved Employed Bee Phase

In this study, a new exploitation mechanism based on the VND for employed bees is developed. As to the permutation-based neighborhood structure, some neighborhood structures, such as insert operator, swap operator, one point left/right operator, and 2-opt operator are commonly used to generate neighboring solutions [31].

In order to enhance the quality of new food sources, a series of experiments has been conducted to find the best neighborhood structures. Compared with other neighborhood structures, the insert operator can obtain much better results for the SDDLBP. Thus, in the proposed algorithm, the set of multi-insert operator (N_1) (Figure 3a), insert operator (N_2) (Figure 3b), and local-insert operator (N_3) (Figure 3c) are combined as neighborhoods (N_k). Three operators are orderly utilized to generate neighboring solutions, and then the current food source will be replaced by the better one between the incumbent and the new one. The flowchart of the proposed VND is given in Figure 4.

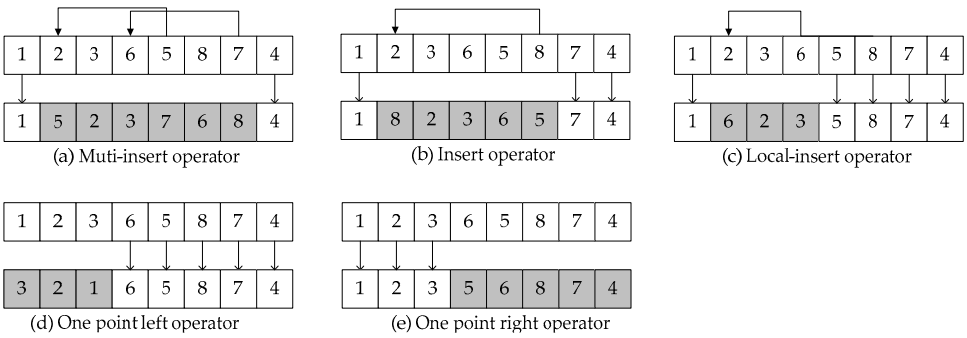

Figure 3. Neighborhood structures. (**a**) Multi-insert operator; (**b**) Insert operator; (**c**) Local-insert operator; (**d**) One point left operator; (**e**) One point right operator.

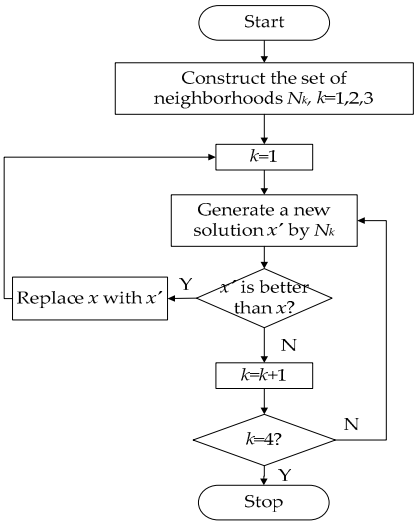

Figure 4. The flowchart of the proposed VND.

4.4. The Improved Onlooker Bee Phase

In the basic ABC, the onlooker bee evaluates and selects a food source based on the probability proportional to the nectar amount of that food source. It is time consuming to calculate the fitness of each food source. In addition, the SDDLBP is a multi-objective optimization problem. No matter which objective function is selected as the fitness evaluation function, there is no significant difference in fitness value among solutions during the late stage of iteration. In this study, we proposed the tournament selection with the size of 4 to replace the fitness proportional selection due to the simplicity and time-saving of the tournament operator [38]. In a four-way tournament, four food sources are randomly chosen from the population, and then the best one is selected. Once an onlooker bee selects a food source, it will produce a modification using the same exploitation mechanism as the employed bee in Section 4.3.

4.5. The Improved Scout Phase

In the basic ABC algorithm, the solution will be discarded if it is has not been improved for consecutive *uplimit* iterations. Then the employed bee becomes a scout and explores new food sources randomly in the predefined search scope. Actually, the employed bee exploits the food source with no more improvement in the early iterations, it does not necessarily mean the solution is trapped in a local optimum. If the solution which can be further improvement is discarded, it will cost much more time to search for the global optimum. Thus, in this study, scout bees do not explore new food sources until half of the number of iterations have finished.

The best food source found so far in the population often contains richer nectar than others during the iterative optimization process, so the search space around it may be the most promising region [36]. Therefore, it is more effective for a scout to produce a high-quality food source based on the best solution in the population. The new solution is constructed by randomly performing one point left/right operator to the best solution, which keeps a partial sequence from the best solution and also reconstructs a new subsequence. Examples of the two neighborhood structures are displayed in Figure 3d,e.

4.6. The Pseudo-Code of the Improved DABC Algorithm

In the improved DABC algorithm, the initial food sources are produced by random and heuristic approaches. Then employed/onlooker bees are hired for exploiting better food sources locally by the VND strategy. Finally, the scouts explore new food sources based on the best food source in the population. The pseudo-code of the improved DABC algorithm is described in Figure 5.

```
Begin
    Parameter initialization
    //Generate the initial population
    for(i = 1 to SN)
        Construct one food source x_i
    end for
    //Evaluate the population and remember the best solution x_best
    for(i = 1 to SN)
        Evaluate the fitness function f(x_i)
        if f(x_i) > f(x_best) then x_best = x_i
    end for
    while(k < maxIter)   //Stopping criterion is not satisfied
        //Employed bees phase
        for(i = 1 to SN)
            Produce new solution x_i' using VND
            Evaluate the fitness function f(x_i')
            if f(x_i') > f(x_i) then x_i = x_i'; limit = 0 else limit = limit + 1
        end for
        //Onlooker bees phase
        for(i = 1 to SN)
             Randomly pick up four food sources from the population;
             Choose the best food source x_i with tournament selection;
             Apply VND to x_i generate a new food source x_i' for onlookers;
             Evaluate the fitness function f(x_i')
            if f(x_i')> f(x_i) then x_i = x_i'; limit = 0 else limit = limit + 1
        end for
        //Scout bees phase
        if(k > maxIter/2)     // Scout bees do not explore new food sources until maxIter/2
            for(i = 1 to SN)
                if(limit >= uplimit)   // The solution should be discarded if it is not improved during uplimit
                     Generate a new food x_i' source with one point left/right operator to x_best
                     Evaluate the fitness function f(x_i')
                       if f(x_i') > f(x_i) then x_i = x_i'; limit = 0
                  end if
            end for
        end if
        //Update the best solution found so far
        for(i = 1 to SN)
           if f(x_i) > f(x_best) then x_best = x_i
        end for
    end while
End
```

Figure 5. The pseudo-code of the improved DABC algorithm.

5. Computational Results and Comparisons

In this section, the performance of the improved DABC algorithm is tested by P10 and P25 benchmark instances. P10 abbreviation represents a 10-part product disassembly and P25 represents a 25-part cellular telephone disassembly instance. The knowledge database and precedence relationships of these instances can be obtained in [27]. All of the algorithms are coded in Microsoft Visual C++ 6.0 and tested on a personal computer with an Intel(R) Core I3 1.80 GHZ, 3 GB RAM. The results of the improved DABC are compared with other meta-heuristic approaches for the SDDLBP reported in the literature. Each instance is carried out for 30 replications in order to calculate the average and standard deviation as statistics for the solution quality.

5.1. Calibrating Uplimit

As pointed out in Section 3, the ABC algorithm has three control parameters: *SN*, *uplimit* and the termination iteration number *maxIter*. Karaboga and Basturk found that the colony size of 50 can provide an acceptable convergence speed for a search [39]. Hence, in the proposed algorithm, the number of food sources (*SN*) is set to 25 and, additionally, it is equal to the number of employed or onlooker bees. In this section, we calibrate the improved DABC algorithm for solving the SDDLBP by taking into account the parameter *uplimit*. As mentioned earlier, if the improved solutions x' cannot be found around the food source x within consecutive *uplimit* iterations, this food source x will be abandoned to prevent premature convergence to local optima. Thus, the performance of the DABC algorithm on solving optimization problems is quite sensitive to the setting of *uplimit* [40].

It is important to choose an appropriate value for *uplimit* spent on exploiting the local area of a food source. Fewer iterations may lead to the fact that the local area of a food source cannot be fully exploited. It is possible to abandon a better solution which can be further improved. As a result, it reduces the convergence speed. On the contrary, if more iterations are performed on the local area of a food source, the employed bees will continue to exploit the food source which has been exhausted. It results in a lower efficiency of search.

For the above reason, we tested the impact of the parameter *uplimit* on the performance of the improved DABC algorithm for solving the SDDLBP. Figure 6 shows different values of *uplimit* with respect to the average computational time of finding an optimal solution. As can be seen, when the *uplimit* is set to 7, the proposed algorithm performs the best for both of the two instances.

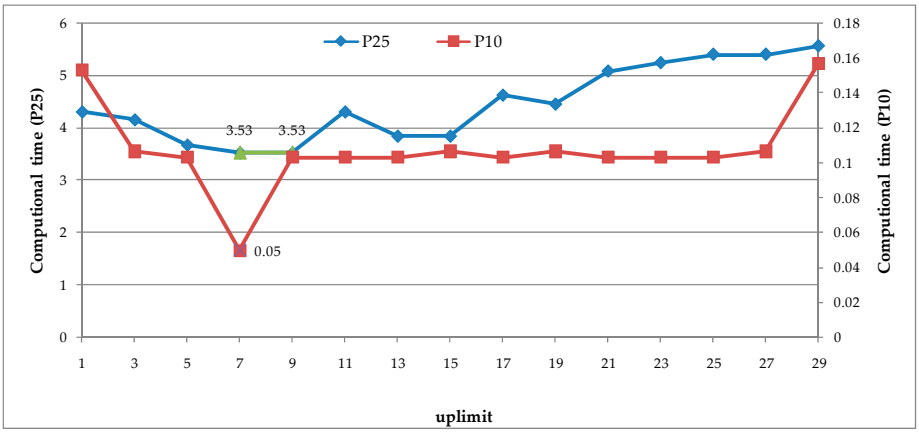

Figure 6. Effect of the parameter *uplimit*.

5.2. Comparison with the Basic ABC Algorithm

To make a fair comparison, the parameters of the improved DABC and the basic ABC algorithms were set the same. Figures 7 and 8 depict the converging processes of the two algorithms for P10 and P25, respectively. It is quite clear that the improved DABC algorithm converges much faster than the

basic ABC algorithm at an earlier stage and soon tends to stability. The improved DABC algorithm finds the optimal solution in $0.05t$ time for P10 and $3.75t$ time for P25, while the basic ABC cannot find the optimal solution in these times. It shows the improved DABC algorithm have a very good convergence due to the effective balance between exploration and exploitation strategies. We can draw a conclusion that the improved DABC algorithm yields solutions of higher quality against the basic ABC at the same computational time. The improved DABC algorithm is superior to the basic ABC for solving the SDDLBP.

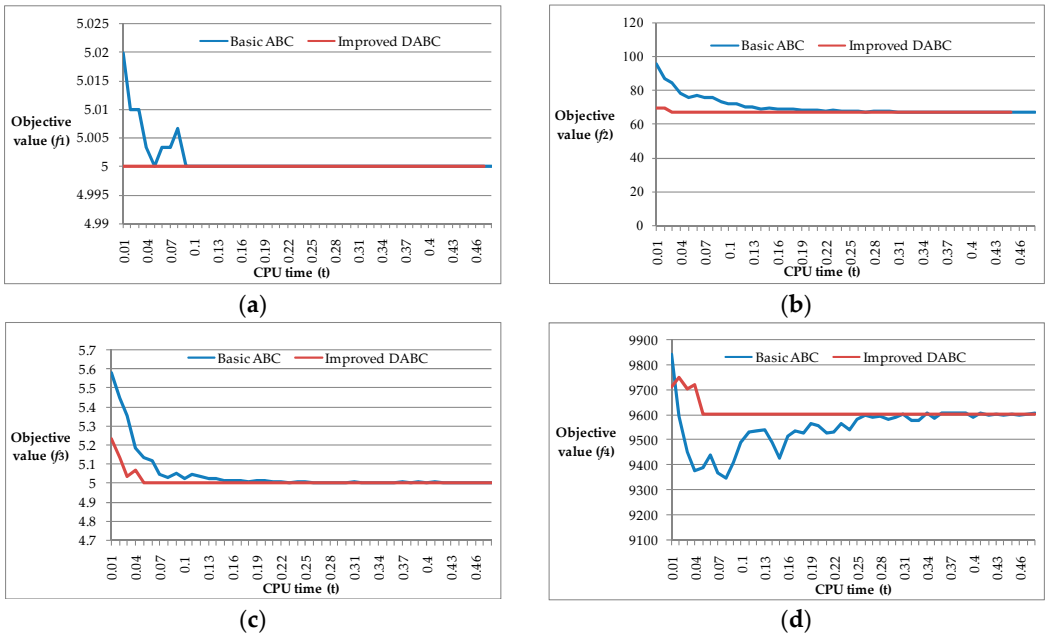

Figure 7. The convergence curves for P10. (**a**) Converging process of f_1; (**b**) Converging process of f_2; (**c**) Converging process of f_3; (**d**) Converging process of f_4.

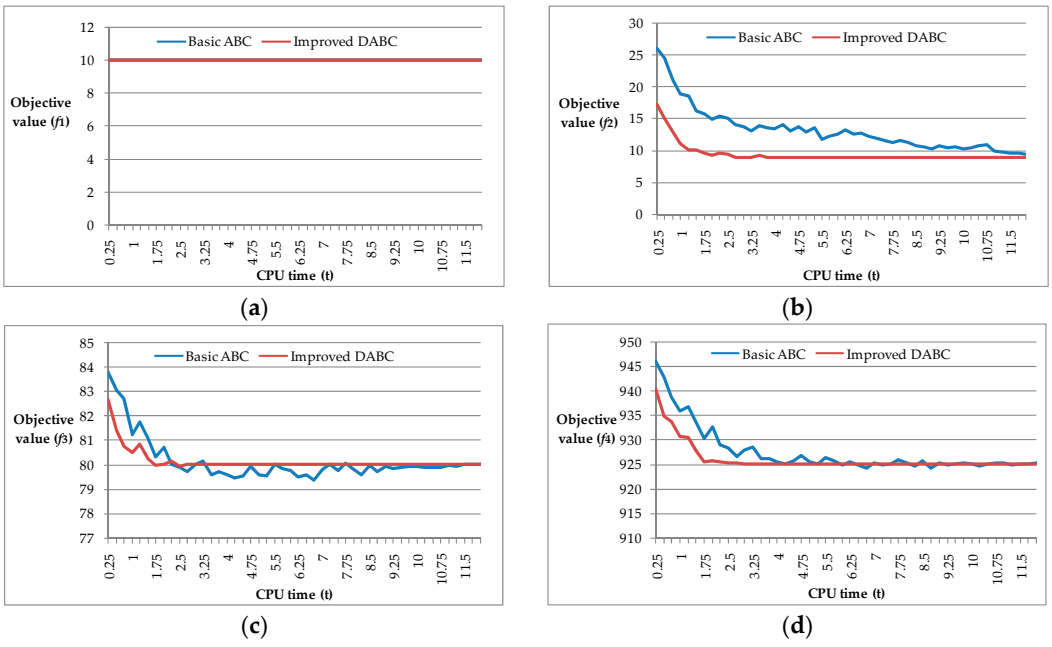

Figure 8. The convergence curves for P25. (**a**) Converging process of f_1; (**b**) Converging process of f_2; (**c**) Converging process of f_3; (**d**) Converging process of f_4.

5.3. Comparison with Existing Meta-Heuristic Algorithms Reported in the Literature

There are nine existing meta-heuristics for solving the SDDLBP in the published literature, including ant colony optimization (ACO), artificial bee colony (ABC), particle swarm optimization (PSO), genetic algorithm (GA), river formation dynamics (RFD), tabu search (TS), simulated annealing (SA), variable neighborhood search (VNS), and hybrid genetic algorithm (VNSGA) .

Since the 10-part product instance is a relatively small example, the exhaustive search method can find optimal solution in 215t time on average [27]. As can be seen in Table 2, it is easier for each method to get optimality in less than 6.0t time. However, the improved DABC is the fastest one to reach optimal solutions in just 0.05t time, on average, showing the best performance. Table 3 shows the comparison of the objective function values between an optimal solution sequence and a non-optimized sequence. For the optimal sequence DS_1 {6, 1, 5, 10, 7, 4, 8, 9, 2, 3}, the objective function values are found to be: $f_1 = 5, f_2 = 67, f_3 = 5, f_4 = 9{,}605$, and the objective function values of the ordinary feasible sequence DS_2 {5, 10, 9, 1, 6, 4, 7, 8, 3, 2} are found to be: $f_1 = 6, f_2 = 602, f_3 = 7, f_4 = 11{,}895$. The comparison results confirm that the optimal sequence DS_1 is effective and outperforms the ordinary feasible sequence DS_2 on all measures of performance. It is meaningful and valuable to design and balance the disassembly line to increase the disassembly efficiency and realizing the coordinated development of economy and environment.

Table 2. Average (AVG) and standard deviation (SD) for objectives for P10.

Method	f_1		f_2		f_3		f_4		t	
	AVG	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**
ABC	5	0	67	0	5	0	9605	0	2.4	1.73
ACO	5	0	67	0	5	0	9605	0	5.36	5.61
GA	5	0	67	0	5	0	9605	0	5.3	4.62
PSO	5	0	67	0	5	0	9605	0	0.81	0.16
RFD	5	0	67	0	5	0	9605	0	3.62	3.45
SA	5	0	67	0	5	0	9605	0	0.59	0.71
TS	5	0	67	0	5	0	9605	0	0.87	1.05
VNS	5	0	67	0	5	0	9605	0	0.83	-
VNSGA	5	0	67	0	5	0	9605	0	0.65	0.35
Our method	5	0	67	0	5	0	9605	0	0.05	0.03

Table 3. Comparison of the objective function values of an optimal sequence and a feasible sequence for P10.

An Optimal Sequence (DS_1)						**A Feasible Sequence (DS_2)**							
Task	**Workstations**					**Task**	**Workstations**						
	1	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**	**6**	
6	14 (+ 2 + 1)					5	23 (+ 4 + 4)						
1	14 (+ 4)					10		10					
5		23 (+ 4)				9		14 (+ 3)					
10		10				1			14 (+ 4)				
7			19			6			14				Time to remove parts
4			17			4				17			
8				36		7				19			
9					14	8					36		
2					10 (+ 3)	3					12 (+ 2)		
3					12	2						10	
Total time 35	37	36	36	39		Total time 31	27	32	36	36	24		
Idle time 5	3	4	4	1		Idle time 9	13	8	4	4	16		
$f_1 = 5; f_2 = 5^2 + 3^2 + 4^2 + 4^2 + 1^2 = 67; f_3 = 5 \times 1 = 5;$ $f_4 = 1 \times 750 + 5 \times 295 + 8 \times 360 + 9 \times 500 = 9605$						$f_1 = 6; f_2 = 9^2 + 13^2 + 8^2 + 4^2 + 4^2 + 16^2 = 602; f_3 = 7 \times 1 = 7;$ $f_4 = 3 \times 360 + 5 \times 750 + 7 \times 295 + 10 \times 500 = 11{,}895$							

For the 25-part SDDLBP instance, it is impractical to use the exhaustive search to obtain the optimal solution in the vast search space (25!). The near-optimal solutions obtained by the meta-heuristic algorithms should be accepted. Table 4 shows the solutions found within a reasonable time of 350t time using improved DABC and other meta-heuristic algorithms. It is obvious that VNS, VNSGA, and improved DABC produce better results compared with the other seven approaches, and robustly reach the best-so-far solution in all of the experiments. Further, considering the time performance, the improved DABC finds an optimal solution in 3.75t time, making it superior to the other algorithms. The efficiency and robustness of the improved DABC have been further confirmed on the P25 instance.

Table 4. Average (AVG) and standard deviation (SD) for objectives for P25.

Method	f_1		f_2		f_3		f_4		t	
	AVG	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**	**AVG**	**SD**
ABC	10	0	10.07	2.16	80	1.14	925.57	5.07	124.6	146.79
ACO	10	0	17.77	1.41	82.8	1.32	949.37	6.31	244.67	161.49
GA	10	0	12.13	2.56	79.77	0.73	924.9	2.4	156.37	138.27
PSO	10	0	13.97	1.96	80.63	2.46	932.5	11.53	40.74	24.71
RFD	10	0	16	0	80.6	0.62	939.83	2.29	222.25	119.24
SA	10	0	11.7	1.82	83.43	3.22	940.93	12.4	297.91	141.19
TS	10	0	13.3	1.7	83.1	2.87	941.3	12.51	273.02	142.14
VNS	10	0	9	0	80	0	925	0	40.86	-
VNSGA	10	0	9	0	80	0	925	0	35.55	20.02
Our method	10	0	9	0	80	0	925	0	3.75	0.53

In short, the comparative results show the highly effective performance of the improved DABC algorithm for solving the multi-objective SDDLBP and addressing the environmental and economic concerns. It has higher search efficiency and stability against the mentioned several meta-heuristic algorithms. The superiority of the improved DABC algorithm should be attributed to the combination of local search and global search with an appropriate balance between exploitation and exploration.

6. Conclusions

Since disassembly contributes to sustainable production and environmental protection, it is vital to design and balance the disassembly line to work efficiently. In this paper, four objectives are considered to address the environmental and economic concerns while solving the multi-objective SDDLBP, and an effective DABC algorithm is presented to search for (near) optimal disassembly sequences. In the proposed algorithm, randomness and heuristics are blended to generate the initial population with high quality and diversity. The employed/onlooker bees exploit new food sources using a VND strategy based on three neighborhood operators to expand the search space and speed up the convergence. To accelerate the speed of escaping from local optima and find high-quality solutions, the scout bees explore new food sources by performing a one point left/right operator to the best food source of the current iteration.

The performance of the improved DABC algorithm for SDDLBP was compared with nine approaches presented in the literature, including GA, ACO, RFD, PSO, TS, SA, ABC, VNS, and VNSGA. All of the algorithms were tested on two scenarios, including a small-sized and a large-sized disassembly instance. Computational results evidently indicate the higher efficiency of the proposed approach with regard to fitness measures and time complexity. The improved DABC algorithm is simple and flexible. It is also superior and robust for solving the SDDLBP. The optimal disassembly sequences obtained by the DABC algorithm can achieve the objectives of getting higher profits and better protecting the environment. Therefore, balancing the disassembly line will make a great contribution to realize the coordinated development of the economy and environment.

For further research, the uncertainty of disassembly task times should be taken into account because of the task-time variability pertaining to human factors and product condition. It would

also be worthwhile studying mixed-model sequence-dependent disassembly lines in light of different structures of returned products. In addition, another research direction may consider a more applied problem which allows for partial or incomplete disassembly due to economic and environmental factors.

Acknowledgments: The authors thank the editors and anonymous referees who commented on this manuscript.

Author Contributions: Jia Liu conceived the research idea and co-wrote the paper. Shuwei Wang co-wrote and revised the paper. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xie, M.; Wang, J.; Chen, K. Coordinated development analysis of the "resources-environment-ecology-economy-society" complex system in china. *Sustainability* **2016**, *8*, 582. [CrossRef]
2. Barba-Sánchez, V.; Atienza-Sahuquillo, C. Environmental proactivity and environmental and economic performance: Evidence from the winery sector. *Sustainability* **2016**, *8*, 1014. [CrossRef]
3. Gungor, A.; Gupta, S.M. Issues in environmentally conscious manufacturing and product recovery: Survey. *Comput. Ind. Eng.* **1999**, *36*, 811–853. [CrossRef]
4. Akçalı, E.; Çetinkaya, S. Quantitative models for inventory and production planning in closed-loop supply chains. *Int. J. Prod. Res.* **2011**, *49*, 2373–2407. [CrossRef]
5. Ilgin, M.A.; Gupta, S.M. Environmentally conscious manufacturing and product recovery (ecmpro): A review of the state of the art. *J. Environ. Manag.* **2010**, *91*, 563–591. [CrossRef] [PubMed]
6. Jasiulewicz-Kaczmarek, M.; Saniuk, A. Human factor in sustainable manufacturing. In *Universal Access in Human-Computer Interaction. Access to the Human Environment and Culture*; Springer International Publishing: Basel, Switzerland, 2015; Volume 9178, pp. 444–455.
7. Güngör, A.; Gupta, S.M. Disassembly line in product recovery. *Int. J. Prod. Res.* **2002**, *40*, 2569–2589. [CrossRef]
8. Özceylan, E.; Paksoy, T. Reverse supply chain optimisation with disassembly line balancing. *Int. J. Prod. Res.* **2013**, *51*, 5985–6001. [CrossRef]
9. Gungor, A.; Gupta, S.M. A solution approach to the disassembly line balancing problem in the presence of task failures. *Int. J. Prod. Res.* **2001**, *39*, 1427–1467. [CrossRef]
10. Altekin, F.T.; Kandiller, L.; Ozdemirel, N.E. Profit-oriented disassembly-line balancing. *Int. J. Prod. Res.* **2008**, *46*, 2675–2693. [CrossRef]
11. Koc, A.; Sabuncuoglu, I.; Erel, E. Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an and/or graph. *IIE Trans.* **2009**, *41*, 866–881. [CrossRef]
12. Altekin, F.T.; Akkan, C. Task-failure-driven rebalancing of disassembly lines. *Int. J. Prod. Res.* **2012**, *50*, 4955–4976. [CrossRef]
13. Mcgovern, S.M.; Gupta, S.M. Combinatorial optimization analysis of the unary np-complete disassembly line balancing problem. *Int. J. Prod. Res.* **2007**, *45*, 4485–4511. [CrossRef]
14. Lu, C.; Huang, H.Z.; Fuh, J.Y.H.; Wong, Y.S. A multi-objective disassembly planning approach with ant colony optimization algorithm. *Proc. IME B J. Eng. Manuf.* **2008**, *222*, 1465–1474. [CrossRef]
15. Mcgovern, S.M.; Gupta, S.M. A balancing method and genetic algorithm for disassembly line balancing. *Eur. J. Oper. Res.* **2007**, *179*, 692–708. [CrossRef]
16. Aydemir-Karadag, A.; Turkbey, O. Multi-objective optimization of stochastic disassembly line balancing with station paralleling. *Comput. Ind. Eng.* **2013**, *65*, 413–425. [CrossRef]
17. Ding, L.P.; Feng, Y.X.; Tan, J.R.; Gao, Y.C. A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. *Int. J. Adv. Manuf. Technol.* **2010**, *48*, 761–771. [CrossRef]
18. Prakash, P.; Ceglarek, D.; Tiwari, M.K. Constraint-based simulated annealing (cbsa) approach to solve the disassembly scheduling problem. *Int. J. Adv. Manuf. Technol.* **2012**, *60*, 1125–1137. [CrossRef]
19. Kalayci, C.B.; Hancilar, A.; Gungor, A.; Gupta, S.M. Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm. *J. Manuf. Syst.* **2014**, *37*, 672–682. [CrossRef]
20. Tuncel, E.; Zeid, A.; Kamarthi, S. Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *J. Intell. Manuf.* **2014**, *25*, 647–659. [CrossRef]

21. Avikal, S.; Jain, R.; Mishra, P.K. A kano model, ahp and m-topsis method-based technique for disassembly line balancing under fuzzy environment. *Appl. Soft Comput.* **2014**, *25*, 519–529. [CrossRef]
22. Kalayci, C.B.; Gupta, S.M. Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Expert Syst. Appl.* **2013**, *40*, 7231–7241. [CrossRef]
23. Scholl, A.; Boysen, N.; Fliedner, M. The sequence-dependent assembly line balancing problem. *OR Spectr.* **2008**, *30*, 579–609. [CrossRef]
24. Andrés, C.; Miralles, C.; Pastor, R. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *Eur. J. Oper. Res.* **2008**, *187*, 1212–1223. [CrossRef]
25. Yolmeh, A.; Kianfar, F. An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Comput. Ind. Eng.* **2012**, *62*, 936–945. [CrossRef]
26. Hamta, N.; Ghomi, S.M.T.F.; Jolai, F.; Shirazi, M.A. A hybrid pso algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *Int. J. Prod. Econ.* **2013**, *141*, 99–111. [CrossRef]
27. Kalayci, C.B.; Gupta, S.M. Ant colony optimization for sequence-dependent disassembly line balancing problem. *J. Manuf. Technol. Manag.* **2013**, *24*, 413–427. [CrossRef]
28. Kalayci, C.B.; Gupta, S.M. Simulated annealing algorithm for solving sequence-dependent disassembly line balancing problem. *IFAC Proc. Vol.* **2013**, *46*, 93–98. [CrossRef]
29. Kalayci, C.B.; Gupta, S.M. A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. *Int. J. Adv. Manuf. Technol.* **2013**, *69*, 197–209. [CrossRef]
30. Kalayci, C.B.; Gupta, S.M. A tabu search algorithm for balancing a sequence-dependent disassembly line. *Prod. Plan. Control* **2014**, *25*, 149–160. [CrossRef]
31. Kalayci, C.B.; Polat, O.; Gupta, S.M. A variable neighbourhood search algorithm for disassembly lines. *J. Manuf. Technol. Manag.* **2015**, *26*, 182–194. [CrossRef]
32. Kalayci, C.B.; Polat, O.; Gupta, S.M. A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Ann. Oper. Res.* **2016**, *242*, 321–354. [CrossRef]
33. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-TR06; Erciyes University: Kayseri, Turkey, 2005.
34. Karaboga, D.; Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
35. Szeto, W.Y.; Wu, Y.; Ho, S.C. An artificial bee colony algorithm for the capacitated vehicle routing problem. *Eur. J. Oper. Res.* **2011**, *215*, 126–135. [CrossRef]
36. Pan, Q.K.; Tasgetiren, M.F.; Suganthan, P.N.; Chua, T.J. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Inf. Sci. Int. J.* **2011**, *181*, 2455–2468. [CrossRef]
37. Tapkan, P.; Ozbakir, L.; Baykasoglu, A. Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. *Appl. Soft Comput.* **2012**, *12*, 3343–3355. [CrossRef]
38. Pan, Q.-K. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. *Eur. J. Oper. Res.* **2016**, *250*, 702–714. [CrossRef]
39. Karaboga, D.; Basturk, B. On the performance of artificial bee colony (abc) algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. [CrossRef]
40. Karaboga, D.; Ozturk, C. Neural networks training by artificial bee colony algorithm on pattern classification. *Neural Netw. World* **2009**, *19*, 279–292.