Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 3407646, 14 pages
https://doi.org/10.1155/2018/3407646

Research Article

Optimum Assembly Sequence Planning System Using Discrete Artificial Bee Colony Algorithm

Özkan Özmen,[1] **Turgay Batbat**,[2] **Tolgan Özen**,[3] **Cem Sinanoğlu**,[1] **and Ayşegül Güven**[2]

[1]*Department of Industrial Design Engineering, Erciyes University, Kayseri 38030, Turkey*
[2]*Department of Biomedical Engineering, Erciyes University, Kayseri 38030, Turkey*
[3]*Software's Functional Management Branch Office, Turkish Air Force, Ankara 06580, Turkey*

Correspondence should be addressed to Turgay Batbat; turgaybatbat@erciyes.edu.tr

Received 20 November 2017; Revised 13 February 2018; Accepted 6 March 2018; Published 12 April 2018

Academic Editor: Abhishek K. Gupta

Assembly refers both to the process of combining parts to create a structure and to the product resulting therefrom. The complexity of this process increases with the number of pieces in the assembly. This paper presents the assembly planning system design (APSD) program, a computer program developed based on a matrix-based approach and the discrete artificial bee colony (DABC) algorithm, which determines the optimum assembly sequence among numerous feasible assembly sequences (FAS). Specifically, the assembly sequences of three-dimensional (3D) parts prepared in the computer-aided design (CAD) software AutoCAD are first coded using the matrix-based methodology and the resulting FAS are assessed and the optimum assembly sequence is selected according to the assembly time optimisation criterion using DABC. The results of comparison of the performance of the proposed method with other methods proposed in the literature verify its superiority in finding the sequence with the lowest overall time. Further, examination of the results of application of APSD to assemblies consisting of parts in different numbers and shapes shows that it can select the optimum sequence from among hundreds of FAS.

1. Introduction

With increasing international competition, efficient methods for manufacturing and producing goods at low cost are essential. This is particularly important in the manufacturing sector, where products are designed, assembled, and simulated with the aid of computers before they are produced. Assembly, the process of combining parts to create a structure or the result of this operation, is also the name given to the product. Assembly sequence planning (ASP) is a very complicated procedure which becomes more difficult as the number of parts increases, as this results in more complexities. These complexities in assembly processes necessitate that assembly sequences be well planned and thoroughly scrutinized. Hence, research is actively being conducted in this field with the objective of developing an ideal method for ASP [1–3].

In this paper, assembly planning system design (APSD), a computer program developed based on a previously proposed matrix-based approach and the discrete artificial bee colony (DABC) algorithm, is proposed for automatically generating assembly sequences and determining the optimum sequence without user intervention. The program runs on the computer-aided design (CAD) software AutoCAD using the visual basics for applications (VBA) module and a CAD database. APSD can also work with data from other 3D CAD software programs by first converting it from the 3D CAD data of those software into dwg or ACIS data format. With the capability of counting, identifying, and labelling the components of a 3D model, APSD can automatically determine the relations between parts in terms of contact and translational functions. In the software, these relations are represented in matrix form and used to evaluate assembly sequences for automatic sequence generation and determination of whether connected parts can form a feasible assembly. Then, according to the assembly time criterion, an optimum assembly sequence is selected from the feasible assembly

sequences (FAS). DABC algorithm is used for optimisation and compared with the literature to evaluate the performance.

2. Literature Review

Initially, products sequencing planning in assembly lines was performed manually. However, with the development of related software and solid modelling programs, the first solid models of assembly products were developed in CAD programs. Subsequently, by means of various theorems, their assembly sequencing was determined. Finally, the optimum ordering was investigated.

The idea of combining assembly modelling with the 3D model of a product using a feature-based method was first introduced by Eng et al. [4]. Their primary approach involved using disassembly to determine the assembly. More specifically, they used the degree of freedom between two features of connected parts to distinguish kinematic conditions, boundary box control to determine collisions, and user interaction to set the restrictions on precedence relations. In another feature-based study, Zha and Du [5] used STEP standards to show modelling information and to manage and convert that information into assembly data. They proposed an assembly editor that could differentiate all feature connections and generate assembly sequences based on feature identification techniques. However, in their system, information on the location and liaison relations between parts must be detected for the automatic generation of the assembly sequences. This was for the purpose of moving parts along the axes of the Cartesian coordinate system in a CAD environment in order to diagnose the intersections between the parts and generate the assembly sequences. This idea was also utilised by Gottipolu and Ghosh [6] to detect the relational information between parts based on a geometric model of the assembly. In their approach, for every couple, the contact relations and disassembly orientations between parts along the coordinate axes are described in terms of two functions: contact and translational. The feasibility of the assemblies are then examined under two compulsory conditions with the implementation of logical operations: connectivity and precedence constraints. Then, after assessing all the numerous feasible sequences, the optimum sequences are shown as a table of assembly states and assembly tasks in a hierarchical fashion, starting from individual parts in the unassembled state to the completed assembly. This assembly sequence table is also supplied with revised features to use strategic constraints under several quantitative and qualitative criteria for analysis of suitable sequences and to determine the final sequence [7].

Lee and Kumara [8] proposed an approach for analysis and efficient production of the assembly sequencing of a complex assembly design before its schematic design stage. In their approach, every part is disassembled and recorded in a sweeping table. Then, assembly and disassembly sequences are produced using matrices and sweeping tables. Dini and Santochi [9] proposed a method based on a mathematical model of the product obtained by determination of three matrices (interaction, contact, and connection). For each subassembly and product, all the possible assembly sequences are produced and the matrix numbers reduced depending on specified optimisation criteria. Zhang et al. [10] developed a procedure for automatic production of all appropriate assembly sequences for the assembly of the body of a car. Their developed procedure is based on the mathematical model of the car body obtained through its connecting and matching matrices, which represent the precedence constraints between the components and the subassembly. Possible subassemblies are then automatically determined on the basis of whether they satisfy specified mathematical conditions. Ciszak [11] proposed a new concept based on graph theory and heuristic multipurpose optimisation that utilises two kinds of matrices (collision and assembly) to generate assembly sequences. Wu et al. [12] proposed an approach that employs assembly knowledge to ASP problems and presented an appropriate method for articulating geometric information and nongeometric knowledge. In their proposed approach, an assembly connection graph is built based on knowledge in the engineering, design, and manufacturing areas. Complicated and low-performance calculations in the assembly design process are bypassed in their approach, which they demonstrated via an assembly planning example.

Depending on the geometry and number of parts in an assembly, hundreds or even thousands of FAS can exist. Determining the optimum assembly sequence from among these FAS is a major issue. Various researchers are currently actively searching for solutions to this problem using artificial intelligence algorithms. For example, Guo et al. [13] proposed an approach based on the shuffled frog leaping algorithm to optimise the ASP of maintenance activities in radioactive environments. In their proposed approach, the geometrical feasibility of the assembly sequences is tested with the help of interference matrices along six axes. Further, to evaluate the fitness function, assembly directional and gripper changes are made. Experimental results indicated that their proposed method yielded better results than approaches based on classical genetic algorithm and particle swarm optimisation. Motavalli and Islam [14] proposed a multicriteria algorithm to find the best assembly sequence among all FAS based on simulated annealing (SA). Their proposed multicriteria algorithm considers both the assembly time and reorientation to find the optimum sequence. Subsequently, Choi et al. [15] applied genetic algorithm (GA) to the problem and obtained better results than using SA. Further, Karthik and Deb [16] compared GA and an hybrid cuckoo-search genetic algorithm (CS-GA) and found that they resulted in the same assembly time. In addition, Mukred et al. [17] applied the particle swarm algorithm (PSA) and the binary particle swarm algorithm (BPSA) to the same problem and found that PSA produced the optimised solution with the shorter assembly time.

3. Mathematical Model

In this section, our proposed mathematical model is explained from three aspects: (1) minimisation of assembly time, (2) reorientation, and (3) matrix-based assembly

sequencing [15, 18]. Minimum assembly time and cost are achieved by considering factors that play a major role, such as the setup time, transfer time, number of tool changes, and proper fixture selection. Therefore, to evaluate the performance of DABC, the following two criteria are discussed: (1) minimisation of assembly time, including setup and actual assembly; (2) minimisation of the number of reorientations to satisfy geometrical constraints, as the parts or subassemblies to be assembled could differ in terms of geometry. When the DABC algorithm is considered in APSD, only assembly time is considered.

3.1. Minimisation of Assembly Time. Assembly time consists of setup time and actual assembly time, which is assumed to be always constant regardless of the sequence. Every component of an assembly demands proper setup. The setup time depends on the geometry of the component itself and the components assembled previously. The following function can be used to predict the setup time for a component:

$$T_{\text{Setup}}(i) = p_{i0} + \sum_{j=1}^{n} p_{ij} x_{ij}, \tag{1}$$

where i is component to be assembled, p_{i0} is setup time for product i being the first component in the assembly, p_{ij} is contribution to the setup time due to the presence of part j when entering part i, x_{ij} = {1, if component j has already assembled, for $i = 1, \ldots, n$; 0, otherwise, for $i = 1, \ldots, n$}

The total assembly time is the summation of the setup time and actual assembly time:

$$\min T_{\text{Assembly}} = \sum_{i=1}^{n} \left(T_{\text{Setup}}(i) + A_i \right), \tag{2}$$

where A_i is the assembly time for component i. The objective function for minimising the assembly time is as follows:

$$Z_1 = \min T_{\text{Assembly}}. \tag{3}$$

3.2. Reorientations and Combined Objective Function. There are certain geometric relations between each part in an assemble. Some intermediate assembly may need reorientation in order to get the parts assembled in a particular sequence. The target here is to minimise the number of reorientations. The total number of reorientations necessary during the assembly of a part is a function of the geometry of the part and the subassembly. Reorientation is defined as follows:

$$R_i = \begin{cases} 1, & \text{reorientation is needed when entering component } i, \text{ for } i = 1, \ldots, n \\ 0, & \text{otherwise, for } i = 1, \ldots, n. \end{cases} \tag{4}$$

The total number of reorientations is given by

$$R = \sum_{j=1}^{n} R_j. \tag{5}$$

Thus, $Z_2 = \min R$.

A multicriteria utility function is used to describe two combined objective functions (Z_1 and Z_2). Thus, the combined objective function (COF) is as follows:

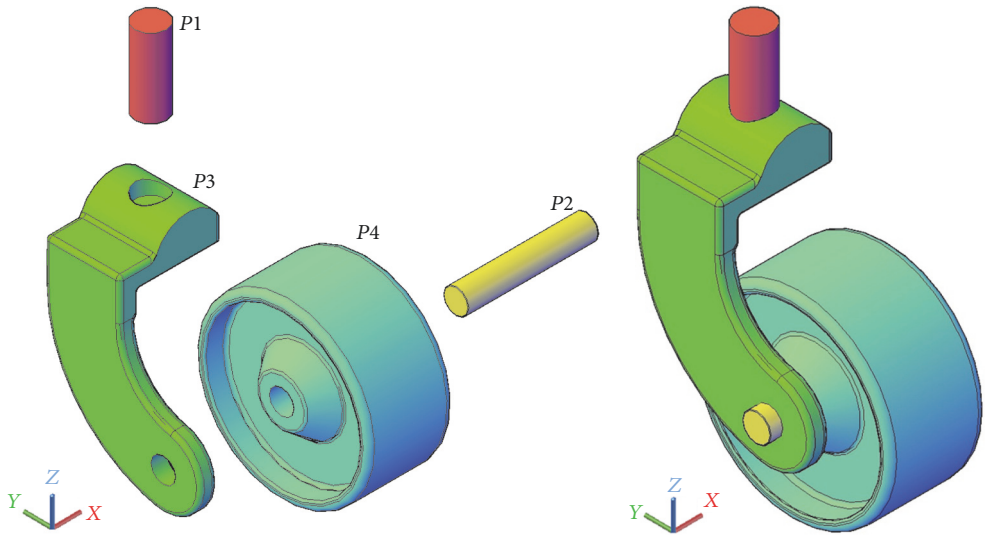
$$\min Z = w_1 \times Z_1 + w_2 \times Z_2, \tag{6}$$

where Z is the combined objective function, w_i is the weight of the individual functions, and i is 1 or 2. The weight of the function depends on the strategy followed and expert opinion. In this study, $w_1 = 0.9$ and $w_2 = 0.1$ were used to evaluate the performance of the DABC algorithm. Further, only the Z_1 function was considered in the APSD program.

3.3. The Matrix-Based ASP Method. The proposed approach for the determination of assembly sequences begins with creation of a CAD assembly model. The assembled parts generated in the CAD medium are sufficient for geometric knowledge but not for assembly planning. The data pertaining to the parts created in the CAD environment are used as input to the assembly planning system. Then, a matrix-based mathematical model is formed and, finally, its assembly sequence is determined by means of Boolean algebra [6, 18]. The theorem is explained below based on the wheel of a shopping cart, as exemplified in Figure 1.

The contacts between two components, a and b, are represented by the 1×6 binary function $C_{ab} = (C_1, C_2, C_3, C_4, C_5, C_6)$. Six elements are used to represent the $+X$, $+Y$, $+Z$, $-X$, $-Y$, and $-Z$ directions of the Cartesian coordinates. A matrix called the contact function matrix is then defined as follows: $C_i = 1$ signifies contact in the direction i; $C_i = 0$ signifies no contact between a and b. The translational motion between parts a and b is represented by a 1×6 binary function called the function of translation, which is defined by $T_{ab} = (T_1, T_2, T_3, T_4, T_5, T_6)$. In this function, $T_i = 1$ signifies that part b can move freely in the direction of i without collision with part a; $T_i = 0$ signifies either that part b hits part a in the direction of i or part a prevents part b from moving freely.

As can be seen in Figure 1, all the values in the P_1P_2 line of the contact matrix have a value of zero, as there is no contact in the six axes between parts P_1 and P_2. The values for the contact and translational functions of the wheel of a shopping cart comprising four parts are presented in Table 1. Because parts P_1P_2 do not prevent each other's movement, a value



(a) Exploded view (b) Assembly view

Figure 1: Three-dimensional view of shopping cart wheels.

Table 1: The contact and translational function of the wheel of a shopping cart.

Pair	C_{+x}	C_{+y}	C_{+z}	C_{-x}	C_{-y}	C_{-z}	T_{+x}	T_{+y}	T_{+z}	T_{-x}	T_{-y}	T_{-z}
P_1P_2	0	0	0	0	0	0	1	1	1	1	1	1
P_1P_3	1	1	0	1	1	1	0	0	1	0	0	0
P_1P_4	0	0	0	0	0	0	1	1	1	1	1	0
P_2P_1	0	0	0	0	0	0	1	1	1	1	1	1
P_2P_3	0	1	1	0	1	1	1	0	0	1	0	0
P_2P_4	0	1	1	0	1	1	1	0	0	1	0	0
P_3P_1	1	1	1	1	1	0	0	0	0	0	0	1
P_3P_2	0	1	1	0	1	1	1	0	0	1	0	0
P_3P_4	0	0	0	0	0	0	0	1	1	1	1	0
P_4P_1	0	0	0	0	0	0	1	1	0	1	1	1
P_4P_2	0	1	1	0	1	1	1	0	0	1	0	0
P_4P_3	0	0	0	0	0	0	1	1	0	0	1	1

Table 2: For the assembly $P_1P_3P_2$, contact and translational functions relation of P_1P_2 and P_1P_3 pairs.

Pair	C_{+x}	C_{+y}	C_{+z}	C_{-x}	C_{-y}	C_{-z}	Pair	T_{+x}	T_{+y}	T_{+z}	T_{-x}	T_{-y}	T_{-z}
P_1P_2	0	0	0	0	0	0	P_1P_2	1	1	1	1	1	1
P_3P_2	0	1	1	0	1	1	P_3P_2	1	0	0	1	0	0
TC	0	1	1	0	1	1	TT	1	0	0	1	0	0

of "1" is assigned to all the variables in the first line of the matrix.

After determining the contact (C) and the translational (T) function, to determine the feasibility of the assembly of this couple, the total contact (TC), total translation (TT), total contact result (TCR), and total translation result (TTR) values are calculated. Two types of limiters (connection and priority) have to be considered following procurement of the contact and translational functions for the FAS of the parts to be assembled. These parts depend on the other parts for assembly to be learned from the connection limiter. The priority limiter states the parts that have to be assembled in advance. Not assembling these parts in the correct sequence will prevent assembly of subsequent parts. These limitations are verified through application of the principles of Boolean algebra to the matrix elements of C and T functions (Table 2).

$\mathrm{TC}_i = C(P_1P_2) \vee C(P_3P_2)$, where i denotes that the relevant columns in each of axes 1 to 6 are subject to the "OR" operation. Finally, TCR is obtained.

This does not include

$$\mathrm{TCR} = \mathrm{TC}_1 \vee \mathrm{TC}_2 \vee \mathrm{TC}_3 \vee \mathrm{TC}_4 \vee \mathrm{TC}_5 \vee \mathrm{TC}_6. \quad (7)$$

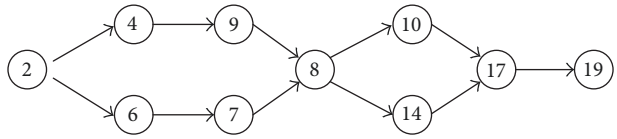


Figure 2: The assembly precedence diagram for performance of DABC [15].

Table 3: Feasible assembly sequences of wheel of a shopping cart.

Feasible assembly sequences	
P_1-P_3-P_2-P_4	P_3-P_1-P_2-P_4
P_2-P_3-P_1-P_4	P_3-P_2-P_1-P_4
P_2-P_3-P_4-P_1	P_3-P_2-P_4-P_1
P_2-P_4-P_3-P_1	P_4-P_2-P_3-P_1

A value of TCR = 1 signifies that connections exist between the parts. In the example given TCR = 1, which means that part P_2 is in contact with the set of subassembly parts P_1P_3. Contact is necessary for the parts to be assembled, but contact alone is not sufficient. In order to obtain priority limiters, an accuracy table is elicited from the motion prevention functions as in the connection limitations. Next, priority limiters are obtained by subjecting the parts to a logical AND ($\wedge$) operation. The results are then subjected to a logical OR ($\vee$) operation to obtain TTR.

$$\mathrm{TTR} = \mathrm{TT}_1 \vee \mathrm{TT}_2 \vee \mathrm{TT}_3 \vee \mathrm{TT}_4 \vee \mathrm{TT}_5 \vee \mathrm{TT}_6. \quad (8)$$

A value of TTR = 1 that there is freedom of movement between the parts in at least one axis, whereas TTR = 0 signifies lack of freedom of movement in all axes. In order for one part to be assembled with another set of parts, it must meet the suitability criteria. The feasibility of assembly (FA) is obtained by subjecting the total movement prevention outcome and the total contact outcomes to a logical AND operation. If the result of this operation is one, then assembly of the part is possible; otherwise, it is impossible to assemble. In the example assembly presented,

$$\mathrm{FA} = \mathrm{TCR} \wedge \mathrm{TTR} = 1 \wedge 1 = 1. \quad (9)$$

This means that it is possible to assemble part P_2 and subset parts P_1P_3. In conclusion, suitable $P_1P_3P_2$ subset assembly parts can be obtained. After this stage, the same theorem can be used to determine whether part P_4 can be assembled with the subassembly of $P_1P_3P_2$. Thus, FAS can be determined by applying the theorem to all the dual parts. FAS for the given parts are presented in Table 3.

4. Artificial Bee Colony (ABC) Algorithm for ASP

Artificial intelligence (AI) is often used to solve problems that do not precisely match mathematical models or to decrease industrial costs. Even low-level computers are capable of solving difficult problems in a relatively short time with the help of AI. Due to the necessity of rapid manufacturing of products in the internationally competitive environment, the assembly sequence is very important in terms of cost [1]. Therefore, it is only natural that a whole new field that focuses especially on ASP optimisation has been developed.

One of the most effective AI models is the ABC algorithm, proposed by Karaboga [19], which models the behaviour of bees in search of food. As the values in the ASP optimisation problem are not continuous, discrete optimisation has to be used to solve the problem. The neighbouring solution creation strategy necessitates new approaches when the problem is recognised as discrete optimisation. Seeking solutions for discrete optimisation problems with the help of the ABC algorithm is an area of interest that is constantly increasing. One such discrete optimisation problem is the workplace flow scheduling problem, for which models for dynamic clustering processes in different types of datasets have been presented [20–25]. In addition, next-generation problems such as cloud service composition, DNA sequencing, and DNA three-dimensional structure prediction problems are present in the literature [26–28].

4.1. Description and Initialisation of the ABC Algorithm for ASP. In the ABC algorithm, bees' gathering nectar correspond to the feasible solution of the problem, while the quality of the nectar in the source corresponds to the fitness value. In our study, the feasible solutions of the problem correspond to the sequence of the parts to be assembled. Further, the fitness values are in the form of the values scaled with (10), calculated in (6) on the basis of the assembly time and knowledge of the reorientation, if any.

$$\mathrm{fitness}_i = \begin{cases} \dfrac{1}{1+f_i}, & f_i \geq 0 \\ 1 + \left|(f_i)\right|, & f_i < 0. \end{cases} \quad (10)$$

Half of the initially formed population is regarded as employed and the other half as onlooker bees. While initial solutions are being formulated, a random source is determined, as defined in (11), in the whole solution range for each employed bee. Rand(0, 1) in the equation denotes a value varying between zero and one. The fitness value of the source visited by the employed bees can be obtained by means of a problem-related function replacing it with f_i in (10). To assess the performance of DABC, the priority graph (Figure 2) composed of 19 parts in the reference article and the priority matrix obtained from them are used for random sequences initial solutions, and FAS values (which will be

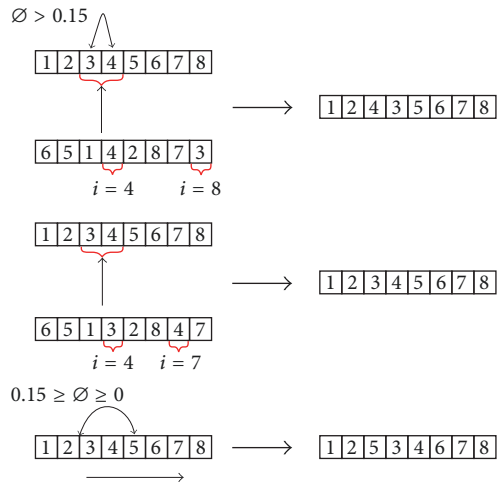


Figure 3: Proposed neighbouring solutions generation mechanism.

explained in the next section) are used for initial solutions in the assessment of APSD.

$$x_{ij} = x_j^{\min} + \text{rand}\,(0,1)\left(x_j^{\max} - x_j^{\min}\right). \tag{11}$$

$i = 1 \cdots SN$ source, $j = 1 \cdots D$, the employed bee assigned to turn to the source, SN is total number of sources, D is total number of parameters, $x_j^{\max}$ is upper limit of parameter j, and $x_j^{\min}$ is lower limit of parameter j

4.2. Employed Bee Phase. The bees assigned to make honey are divided into three different categories in the algorithm formulated: scout bees assigned to locate utterly new sources; employed bees bringing nectar to the hive from the newly discovered source and nearby sources as well as describing the location and quality of the source to other bees waiting in the hive; and onlooker bees waiting in the hive to go to the sources they determine according to the descriptions they have received of the location of the source. Honey bees associate the location of sources with the position of the sun and impart this to the other bees, together with the knowledge about the nectar, by means of their dances in the hive. During these activities, employed bees continue their search of other sources nearby. As is seen in (12), on their way to a newly discovered source, they make use of either the source they are in or one of the sources from which other employed bees collect nectar.

$$v_{ij} = x_{ij} + \emptyset_{ij}\left(x_{ij} - x_{kj}\right). \tag{12}$$

The new source is derived by changing the parameter from the source, where $\overrightarrow{x}_i$ denotes the change from $\overrightarrow{v}_i$ and $\emptyset_{ij}$ is a randomly selected number in the range $[-1, 1]$. However, because the values found are discrete, the newly obtained solution will recur in all stages in the manner presented below. In this random process, the source k, which is one of the sources selected randomly from among other sources, is used. Thus, as the sources approach each other, the size of the steps will decrease, resulting in a more intensive search. The solutions exceeding the limits are fixed to the limits, as in

$$v_{ij} = \begin{cases} x_j^{\min}, & v_{ij} < x_j^{\min} \\ v_{ij}, & x_j^{\min} \le v_{ij} \le x_j^{\max} \\ x_j^{\max}, & v_{ij} > x_j^{\max}. \end{cases} \tag{13}$$

4.3. Generation of Neighbouring Solutions. As the values are discrete, new solutions cannot be realised with numerical changes. Equation (11) is not used directly to produce neighbouring solutions. However, as was noted previously, it is necessary to take advantage of the algorithm in the process of obtaining neighbouring solutions, sticking to the foundations of the algorithm. Inspired by the discrete optimisation approaches in the literature, the method shown in Figure 3 was devised.

In this strategy, the $\emptyset_{ij}$ component used by the ABC algorithm represents the random selection of the function which yields the new solution from the former one taken suitably for its purpose. This process is achieved as follows:

(i) A randomly selected different solution $(\overrightarrow{x}_k)$ is used for 85% of the neighbouring solutions. The components determined in terms of random numbers not exceeding 30% of the solution dimension are selected from $\overrightarrow{x}_i$, beginning from the random position. The neighbouring solution is generated by sequencing the parts in this section and arranging their location in other solutions. Thus, the tendency to turn to a different solution is realised.

(ii) In the remaining solutions, a randomly selected solution is brought to its previous location to reduce the rate of assimilation, and the components between the replaced component and its own location are shifted to the next location. This process is called insert.

Thus, while new solutions are being obtained by means of different solutions, early convergence is prevented with

random solutions. Addressing only the solutions appropriate for the existing conditions when generating neighbouring solutions would give rise to early convergence and result in the algorithm becoming stuck in local minima. Penalising the fitness value of unfit solutions with certain coefficients not only raises the probability of unfit solutions, if any, but also eliminates the possibility of getting stuck in local minima.

4.4. Onlooker Bee Phase. Onlooker bees waiting in the hive act on the information supplied by employed bees. Equipped with this information, they choose one of the sources and go to that source to collect nectar. Which source the onlooker bees will go to is determined in accordance with the feasibility value obtained with the ratio of the fitness value of that source to the total value of all the sources from which all the employed bees collect nectar. The feasibility value is calculated using (14). The onlooker bees going to the randomly selected source in view of this feasibility value enable the local search to intensify around the sources with greater quality, and they take nectar from the neighbouring sources located with the neighbouring solutions generation mechanism.

$$p_i = \frac{\text{fitness}_i}{\sum_{j=1}^{SN} \text{fitness}_j}. \tag{14}$$

4.5. Scout Bee Phase. In the ABC algorithm, sources with depleted nectar are abandoned by the honey bees, thereby preventing cessation of the algorithm. In this process, employed bees resume searching for new sources while onlooker bees return to the hive. The failure counter, which determines when each source is deemed depleted, checks whether the limit parameter defined before each iteration is exceeded. Failure_i starts as zero when initial solutions are generated. In both the employed and onlooker bee phases, if the former value of the source is replaced with the value produced in the phase, it is reset to zero. Otherwise, the value on the counter is increased by one. In each generation, only one source can remain. Usually, scout bees determine a source using the same method as their initial solutions. However, in our study, a different method is employed as the development of discrete values incurs significant computation time. This method is realised, without comparing fitness value, with the insert operation explained in the generating neighbouring solutions section above, and is repeated up to three times until similarity is sufficiently reduced. Each time these three phases are completed, the favourable solution in the population is compared with the previous favourable solution and the superior one is selected. The general DABC algorithm proposed is presented in Figure 4.

5. Assembly Planning System Design (APSD)

As more than one assembly sequence of the products is usually used in industry, and only one of these sequences is chosen by various criteria for the realisation of assembly following acquisition of all the assembly sequences, these criteria are determined by an expert assembly sequence planner for any product, and only one of them is sufficient to complete the assembly planning.

In this study, the software APSD was devised in AutoCAD using the algorithms explained above and the VBA module. APSD consists of four main tabs: (1) Parts List, (2) *C*-*T* Functions, (3) Feasible Assembly Sequences, and (4) Assembly Tree. These tabs help users carry out the four main procedures shown in Figure 5, specifically, identification of parts, determination of *C* and *T* functions, generation of all feasible assemblies, and displaying of the assembly tree.

In the software, FAS are first found and then the optimisation algorithms are executed. First, the contact and translational functions are described as arrays. A procedure called the assembly test function (ATF) was coded to achieve these calculations and to test the feasibility of the sequences using the matrix-based assembly sequence methodology explained in the previous section. With the implementation of recursive programming, a procedure called the proper part finding procedure (PPFP) was coded to accomplish part selection from the parts list before the feasibility of the assembly is checked. The flowchart of this algorithm is shown in Figure 6 [29].

In this section, determination and optimisation of the FAS of the connection frame assembly system shown in Figure 7 are realised with the software developed.

When the "Identification of Parts" button is pressed, the program automatically names the parts (*P*1-*P*2-*P*3-*P*4-*P*5-*P*6-*P*7-*P*8-*P*9). Figure 5 shows the interface of the program. In addition, the names assigned by AutoCAD are shown next to the part names. As seen in the figure, there are nine parts on the connection frame assembly. To determine all the feasible sequences of the parts, first contact and translational matrices have to be found. This is achieved by pressing the "Calculate the *C*-*T* Functions" button in the interface of the program. While moving each part along the six axes to determine its interaction with the other parts, the program prints the matrix in a section of the *C*-*T* Function window. As there are nine parts, as is seen in Figure 8, in the connection frame (9!) 362880 sequences are possible. However, not all of these sequences are significant for assembly. To obtain the feasible sequences in the assembly it is necessary to press the "Feasible Assembly Sequences" button. This returns all the assembly sequences obtained using the matrix-based assembly sequence theorem presented in the previous section. Thus, 4530 FAS are presented, as shown in Figure 8.

As shown in the figure, 4530 FAS are obtained in the assembly system. However, the designer cannot possibly assess all these sequences one by one. Therefore, the found FAS are transferred to a text file by means of "Write Results to Text File" function and input into MATLAB. Unlike the assessment of the performance of the DABC algorithm, FAS are taken as entry in place of the priority matrix. Next, optimisation is realised from FAS in view of the assembly time, for which Table 5 is used according to the number of components. For example, for a nine-component assembly, the 9×9 section of the table is used.

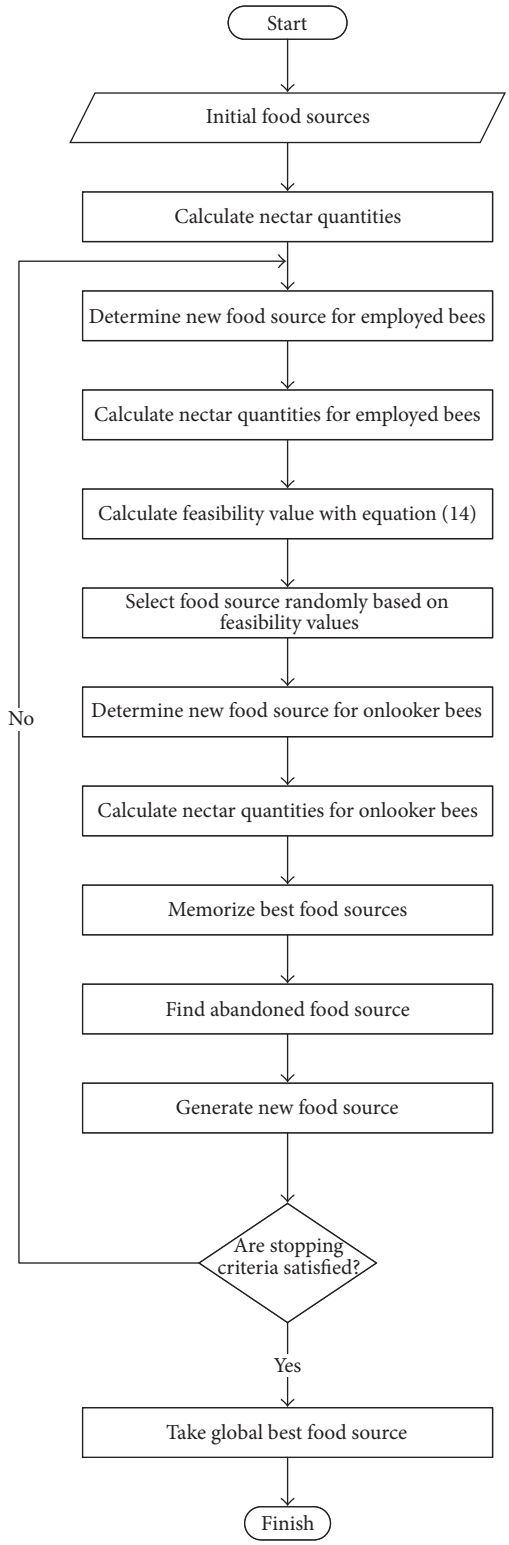


Figure 4: Flowchart of the proposed discrete artificial bee colony (DABC) algorithm.

6. Results and Discussion

The program code of the DABC algorithm was developed in MATLAB and executed on a desktop computer with an i7-4790 3.60 GHz CPU, 16 GB RAM, and 1 TB memory. The parameters used for the DABC algorithm for the purpose of comparison with other studies were as follows: limit value, 100; colony size (NP) values, 20, 40, 100; and number of generations, 100. For the solutions applied in APSD, the size of the population was doubled; the population was 10 times

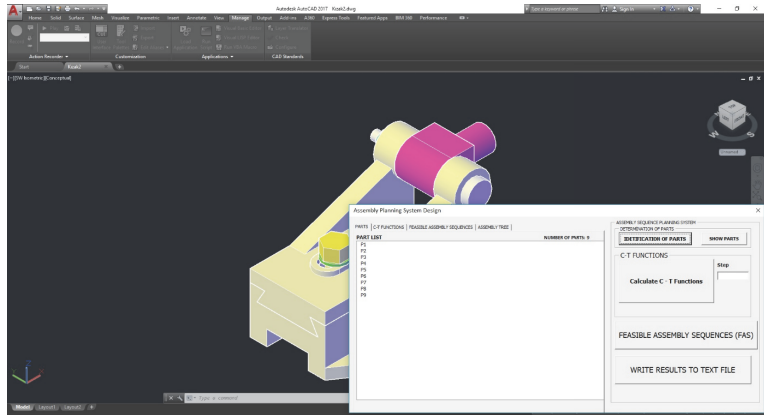


Figure 5: Program interface is shown in AutoCAD visual basic for application module.

Table 4: Parameters of optimisation techniques.

	DABC	SA	CS-GA	GA
Number of iteration	100	500	1000	1000
Colony size	20, 40, 100	-	15	15
Limit	100	-	-	-
Run	10	-	100	100
Cross over rate	-	-	0.8	0.8
Mutation rate	-	-	0.2	0.2
Initial temperature	-	100	-	-
Cooling rate	-	0.95	-	-

the number of features; and the limit value was one-half the size of the population. Thus, efforts were made to enable the system to respond to assemblies involving different features. The selected parameters yielded values far below those in the literature. In addition, a penalty value of five was used for each parameter in the compared systems, but, for APSD, the number 50 was used only once to make it conform to the rules and optimum results obtained. The parameters in the article taken as reference to assess the performance of DABC are presented in Table 4. The assembly duration and reorientation of the parts used in the studies taken as reference and in the present study are presented in Tables 5 and 6, respectively. Further, Table 7 shows the assembly duration of the sequences in the reference articles according to (6), compared with the assembly duration obtained from DABC.

As the population size suggested in the DABC algorithm has been run 10 times and the minimum value of results has been taken into consideration. However, in each of the ten operations, the minimum value was obtained at least once (Figure 9).

The assembly with nine components in Figure 7 was tested in APSD and 4530 feasible solutions were obtained. Following optimisation in DABC, a total assembly time of 173.800 and the assembly sequence 1-9-8-5-4-3-6-2-7 were obtained.

7. Conclusion

In this study, a program for finding the optimum sequence among the FAS of an assembly system designed with computer assistance as a 3D solid object based on assembly time using DABC was developed. The program automatically realises the background within the framework of algorithms determined at every step by minimising utilisation and interaction. Examination of the results obtained reveals that the optimum sequence was selected from among hundreds of available assemble sequences, indicating that the program is effective. The program has been tested on assembly systems consisting of parts of different numbers and shapes. In addition, the performance of the DABC algorithm was compared with the results of other methods proposed in the literature with superior results being obtained.

All the possible assembly sequences modelled as 3D solid in AutoCAD were generated automatically. Further, the developed program produced consistent results over numerous runs. Hundreds of FAS can be generated even in cases in which the number of parts in an assembly is small. In the developed software, selection is made from among the available sequences according to predetermined criteria. Despite these results, the method we have proposed has one constraint: the efficacy of the method suggested to obtain assembly sequence is predicated on 3D CAD data

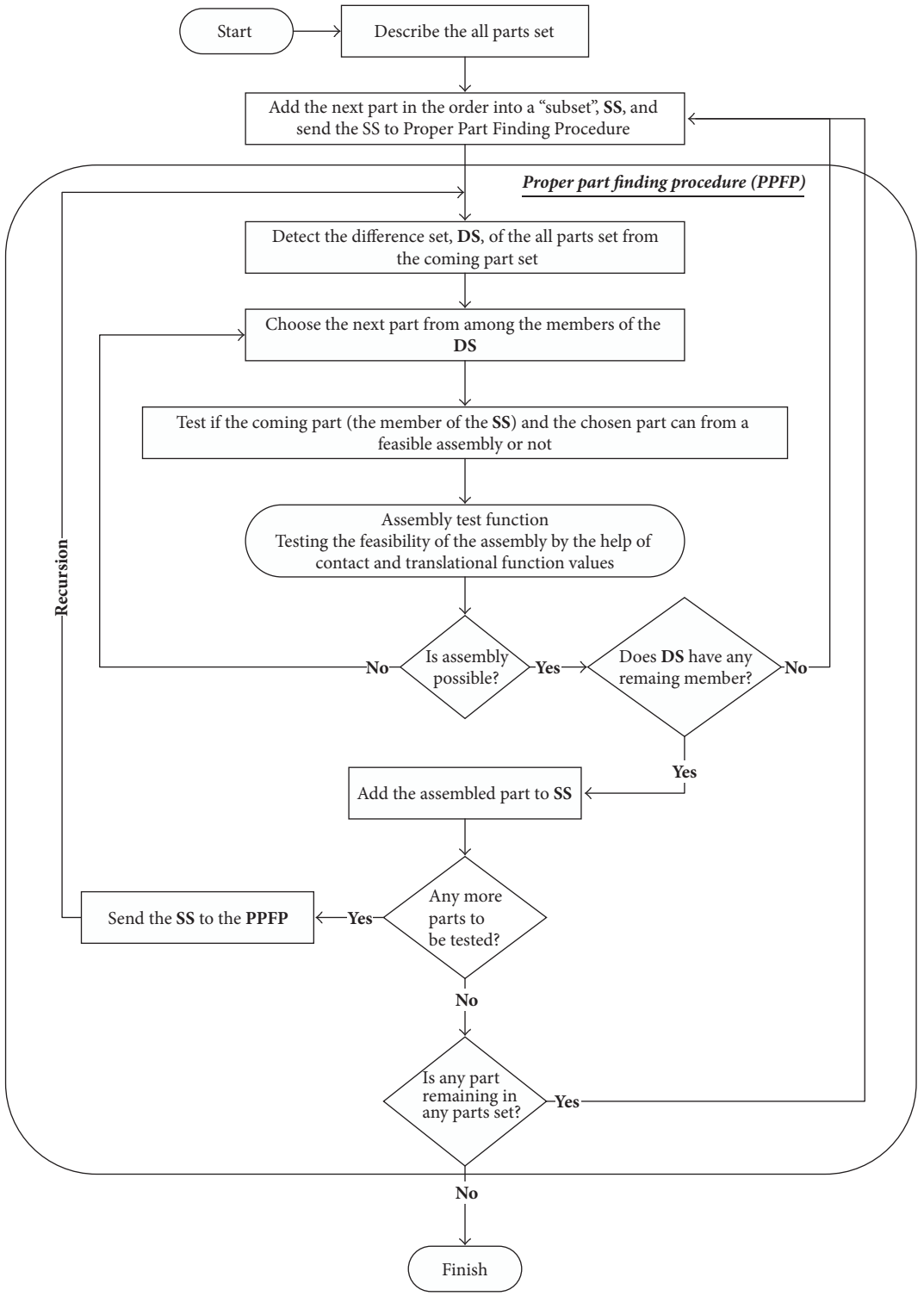
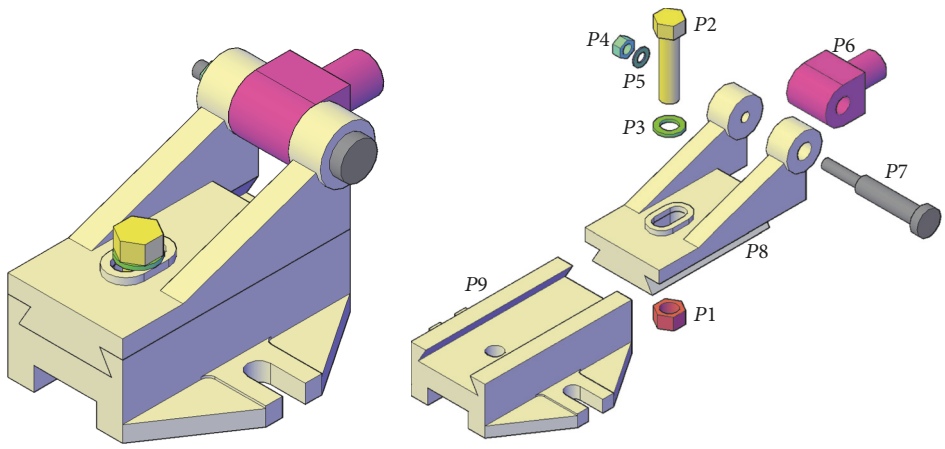


Figure 6: Recursive procedure for testing the feasibility of the parts to be assembled.

fitting in with Cartesian coordinates. On the other hand, the development of algorithm suggested for curved components is one of the topics of future studies we plan to do to assess existing assembly systems by considering both mechanical criteria such as weight and degree of subassembly freedom and by using real assembly durations together with expert assembly sequence planners and enlarging the scope of optimisation with the proposed algorithm.



(a) Assembly view (b) Exploded view

Figure 7: Three-dimensional connection frames in AutoCAD.

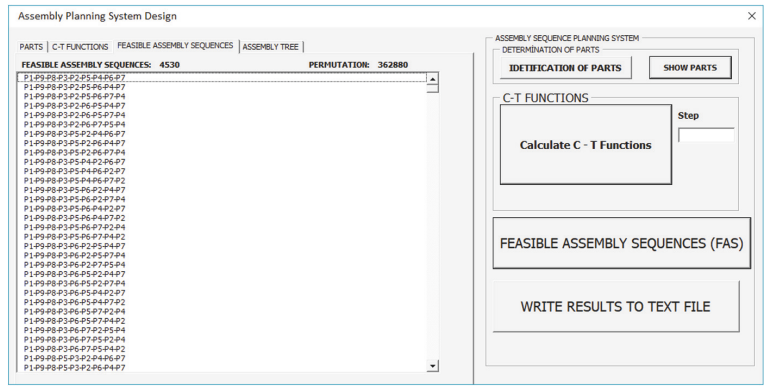


Figure 8: The program interface of the tab of FAS.

Nomenclature

DABC: Discrete artificial bee colony
3D: 3-dimensional
CAD: Computer-aided design
APSD: Assembly planning system design
FAS: Feasible assembly sequences
ASP: Assembly sequence planning
VBA: Visual basic for application module
SA: Simulated annealing
GA: Genetic algorithm
CS-GA: Cuckoo-search genetic algorithm
BPSA: Binary particle swarm algorithm
PSA: Particle swarm algorithm
COF: Combined objective function
C: Contact
T: Translational
TC: Total contact
TT: Total translation
TCR: Total contact result
TTR: Total translation result
FA: Feasibility of assembling
AI: Artificial intelligence
ABC: Artificial bee colony
ATF: Assembly test function
PPFP: Proper part finding procedure
SS: Subset
DS: Difference set.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author would like to acknowledge Editage for English language editing. Also this work was supported by Erciyes University Scientific Research Projects Coordination Unit with Project no. FBY-12-3987.

Table 5: Coefficient of various components in the assembly.

Component to be assembled	Component I																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	10	1	2	3	4	5	6	7	8	9	3.2	4.3	7	6.1	1.2	3.4	0	0	7.4
2	1.5	10	2	2	2	2	2	2	2	2	0	3.1	6	4.3	2.7	4.8	0	3	0.5
3	1	2.3	10	0	4	5	0	4	2.3	4.3	9.8	2.4	5	1.2	3.4	4.5	5.6	3.4	3.1
4	0	2	3.4	10	4.5	0	4	0	8	0	3.4	5.6	5	0	0	3.4	0	0	9.8
5	1.2	1	2	3	10	7.9	8.9	0	1.2	2	2.3	0	3	0	3.6	0	2.8	9.8	0
6	9.8	4.5	0	1.2	3.6	10	3.4	4	0	2.3	4.6	5.6	0	4	3	2	0	0.4	3.2
7	0.5	1.4	2.3	0.5	1.9	1	10	13.4	1.2	4	2.3	0	3	5.7	8.30	2	0.1	0	0.5
8	0	0	0	0	0	1.8	9.8	10	2.3	3	8.9	2.3	0	0	2.3	0.5	9.8	0	2.3
9	1	3	4.5	2.3	4.6	9.8	7.5	6.8	10	6	2.3	3.4	5	12.3	3.4	5.61	1	0	0
10	2.3	4.5	2.3	0	2.3	0	2.1	0	4.5	10	1.1	2.3	2	0	0	2.1	1.2	5.4	9.2
11	1	1	2	3	4	5	6	7	8	9	10	4.5	3	6.1	1.2	3.4	0.3	0	1.3
12	1.5	0	2	2	2	2	2	1	2	2	11.2	10	6	4.3	2.7	4.8	0	3	0.5
13	1	2.3	0	0	4	5	0	4	2.3	4.3	9.8	2.4	10	1.2	2.4	4.5	1.6	2.4	3.1
14	0	2	3.4	0	4.5	0	4	0	8	0	3.4	5.6	5	10	2.1	1.4	1	0	2.8
15	1.2	1	2	3	0	7.9	8.9	0	1.2	2	1.3	4	3	1.4	10	1.3	9.8	9.8	2
16	9.8	4.5	0	1.2	3.6	0	3.4	4	0	2.3	4.6	3.6	0	4	3	10	1.5	0	3.2
17	1	3	4	5	0	5	4	3.4	1.2	4	1.3	0	2	3.7	4.3	2.3	10	3.8	10
18	0.6	0.5	3.4	1.2	3	2	9.8	2	2.3	3	5.9	2.3	0	1.0	2.3	0.5	9.8	10	2.3
19	1	3	4.5	2.3	4.6	9.8	7.5	6.8	0	6	3.3	3	2	3.3	4.4	2.6	0.3	2.5	10

Table 6: Set for reorientations.

Part	Set for reorientation
1	{4, 9} {9, 11} {4, 6, 11} {11, 7} {7, 9} {6, 5} {13, 12, 18}
3	{5, 2} {1, 4, 5} {16, 18} {2, 1, 6} {2, 1, 4, 5} {13, 9, 4} {9, 6, 2} {6, 2, 10} {18, 17}
7	{5, 12, 6} {12, 6, 1} {9, 6, 16} {19, 2, 4, 9, 8} {12, 6} {7, 5} {9, 5} {10, 8, 6, 4}
8	{6, 7, 16} {9, 6, 5, 7} {12, 7} {15, 2} {16, 15, 5} {9, 5, 1}
10	{16, 8} {6, 7, 8} {15, 1} {7, 3} {9, 4} {19, 17, 3}
11	{4, 9, 16} {12, 13} {12, 5, 9} {13, 6, 16} {13, 12} {16, 18, 3}
16	{6, 7} {12, 6} {11, 9, 6} {5, 3, 18, 11, 13} {13, 12, 4, 19, 18, 6, 5, 3, 8}
17	{14, 15} {5, 2} {10, 14, 15} {13, 12, 9} {15, 13, 5} {19, 7}

Table 7: Comparison of optimisation techniques.

Opt. tech.	Assembly sequence	Assembly time	Reorientation
SA	2-1-4-9-3-12-13-16-5-15-18-6-11-7-8-10-14-17-19	528.6600	0
CS-GA*	2-1-4-9-3-12-13-16-5-15-18-11-6-7-8-10-14-17-19	528.4000	1 (6-7-8-10)
GA*	2-1-4-9-3-12-13-16-5-15-18-11-6-7-8-10-14-17-19	528.4000	1 (6-7-8-10)
DABC	2-1-4-9-3-12-13-16-5-15-18-11-6-7-8-14-10-17-19	528.3000	0

*In these studies, sequencing starts with zero.

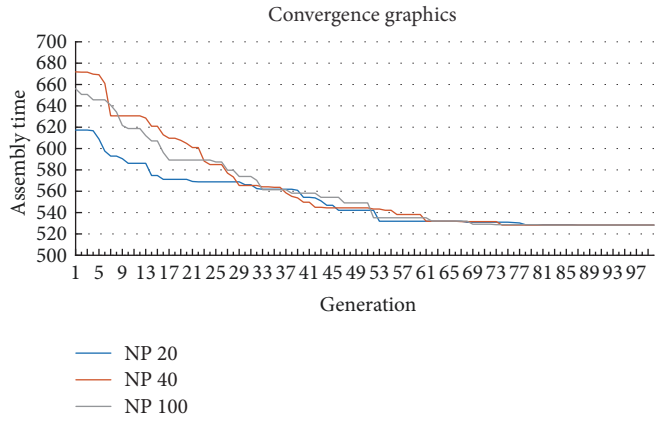


Figure 9: The result of DABC.

References

[1] C. Sinanoğlu and H. Rıza Börklü, "An assembly sequence-planning system for mechanical parts using neural network," *Assembly Automation*, vol. 25, no. 1, pp. 38–52, 2005.

[2] H. Dilipak and A. Özdemir, "Generation of Assembly And Subassembly Sequences Of Industrial Products And The Optimization With Constraints," *Gazi University Journal of Science*, vol. 16, no. 4, pp. 797–810, 2003.

[3] C. J. M. Heemskerk, L. N. Reijers, and H. J. J. Kals, "A Concept for Computer-Aided Process Planning of Flexible Assembly," *CIRP Annals - Manufacturing Technology*, vol. 39, no. 1, pp. 25–28, 1990.

[4] T. H. Eng, Z. K. Ling, W. Olson, and C. McLean, "Feature-based assembly modeling and sequence generation," *Computers & Industrial Engineering*, vol. 36, no. 1, pp. 17–33, 1999.

[5] X. F. Zha and H. Du, "A PDES/STEP-based model and system for concurrent integrated design and assembly planning," *Computer-Aided Design*, vol. 34, no. 14, pp. 1087–1110, 2002.

[6] R. B. Gottipolu and K. Ghosh, "Representation and selection of assembly sequences in computer-aided assembly process planning," *International Journal of Production Research*, vol. 35, no. 12, pp. 3447–3466, 1997.

[7] R. B. Gottipolu and K. Ghosh, "A simplified and efficient representation for evaluation and selection of assembly sequences," *Computers in Industry*, vol. 50, no. 3, pp. 251–264, 2003.

[8] Y.-Q. Lee and S. R. Kumara, "A scheme for mechanical assembly design and assembly line layout conceptualization," *Computers & Industrial Engineering*, vol. 27, no. 1, pp. 261–264, 1994.

[9] G. Dini and M. Santochi, "Automated sequencing and subassembly detection in assembly planning," *CIRP Annals - Manufacturing Technology*, vol. 41, no. 1, pp. 1–4, 1992.

[10] Y. Z. Zhang, J. Ni, Z. Q. Lin, and X. M. Lai, "Automated sequencing and sub-assembly detection in automobile body assembly planning," *Journal of Materials Processing Technology*, vol. 129, no. 1-3, pp. 490–494, 2002.

[11] O. Ciszak, "Computer aided determination of the assembly sequence of machine parts and sets," *Advances in Engineering Software*, vol. 48, no. 1, pp. 17–26, 2012.

[12] M. Wu, Y. Zhao, and C. Wang, "Knowledge-based approach to assembly sequence planning for wind-driven generator," *Mathematical Problems in Engineering*, 2013.

[13] J. Guo, H. Tang, Z. Sun et al., "An improved shuffled frog leaping algorithm for assembly sequence planning of remote handling maintenance in radioactive environment," *Science and Technology of Nuclear Installations*, vol. 2015, Article ID 516470, 16 pages, 2015.

[14] S. Motavalli and A.-U. Islam, "Multi-criteria assembly sequencing," *Computers & Industrial Engineering*, vol. 32, no. 4, pp. 743–751, 1997.

[15] Y. K. Choi, D. M. Lee, and Y. B. Cho, "An approach to multi-criteria assembly sequence planning using genetic algorithms," *The International Journal of Advanced Manufacturing Technology*, vol. 42, no. 1, pp. 180–188, 2009.

[16] G. Karthik and S. Deb, "A Methodology for assembly sequence optimization by hybrid cuckoo-search genetic algorithm," *Journal of Advanced Manufacturing Systems*, vol. 17, no. 1, pp. 47–59, 2018.

[17] J. A. A. Mukred, Z. Ibrahim, I. Ibrahim et al., "A binary particle swarm optimization approach to optimize assembly sequence planning," *Advanced Science Letters*, vol. 13, no. 1, pp. 732–738, 2012.

[18] R. B. Gottipolu and K. Ghosh, "Integrated approach to the generation of assembly sequences," *International Journal of Computer Applications in Technology*, vol. 8, no. 3-4, pp. 125–138, 1995.

[19] D. Karaboga, "An idea based on Honey Bee Swarm for Numerical Optimization," Technical Report-TR06 TR06, Erciyes University, Turkey, 2005.

[20] D. Ye and Z. Chen, "A new approach to minimum attribute reduction based on discrete artificial bee colony," *Soft Computing*, vol. 19, no. 7, pp. 1893–1903, 2015.

[21] I. Ribas, R. Companys, and X. Tort-Martorell, "An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization," *Expert Systems with Applications*, vol. 42, no. 15-16, pp. 6155–6167, 2015.

[22] C. Ozturk, E. Hancer, and D. Karaboga, "Dynamic clustering with improved binary artificial bee colony algorithm," *Applied Soft Computing*, vol. 28, pp. 69–80, 2015.

[23] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," *Information Sciences*, vol. 297, pp. 154–170, 2015.

[24] Y.-Y. Han, D. Gong, and X. Sun, "A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking," *Engineering Optimization*, vol. 47, no. 7, pp. 927–946, 2015.

[25] Z. Cui and X. Gu, "An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems," *Neurocomputing*, vol. 148, pp. 248–259, 2015.

[26] D. Karaboga and S. Aslan, "A discrete artificial bee colony algorithm for detecting transcription factor binding sites in DNA sequences," *Genetics and Molecular Research*, vol. 15, no. 2, Article ID 15028645, 2016.

[27] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Applied Intelligence*, vol. 42, no. 4, pp. 661–678, 2015.

[28] T. Batbat and C. Öztürk, "Protein structure prediction with discrete artificial bee colony algorithm," *International Journal of Informatics Technologies*, vol. 9, no. 3, pp. 263–274, 2016.

[29] Ö. Özmen, T. Özen, and C. Sinanoğlu, "Hacim Kriterine Göre Matris Temelli Montaj Sırası Planlama Sistem Tasarımı," in *Proceedings of the The First International Symposium on Industrial Design Engineering (ISIDE14)*, pp. 278–284, Karabük, Turkey, May 2014.