An improved hybrid flower pollination algorithm for assembly sequence optimization

Atul Mishra and Sankha Deb
Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Abstract

Purpose – Assembly sequence optimization is a difficult combinatorial optimization problem having to simultaneously satisfy various feasibility constraints and optimization criteria. Applications of evolutionary algorithms have shown a lot of promise in terms of lower computational cost and time. But there remain challenges like achieving global optimum in least number of iterations with fast convergence speed, robustness/consistency in finding global optimum, etc. With the above challenges in mind, this study aims to propose an improved flower pollination algorithm (FPA) and hybrid genetic algorithm (GA)-FPA.

Design/methodology/approach – In view of slower convergence rate and more computational time required by the previous discrete FPA, this paper presents an improved hybrid FPA with different representation scheme, initial population generation strategy and modifications in local and global pollination rules. Different optimization objectives are considered like direction changes, tool changes, assembly stability, base component location and feasibility. The parameter settings of hybrid GA-FPA are also discussed.

Findings – The results, when compared with previous discrete FPA and GA, memetic algorithm (MA), harmony search and improved FPA (IFPA), the proposed hybrid GA-FPA gives promising results with respect to higher global best fitness and higher average fitness, faster convergence (especially from the previously developed variant of FPA) and most importantly improved robustness/consistency in generating global optimum solutions.

Practical implications – It is anticipated that using the proposed approach, assembly sequence planning can be accomplished efficiently and consistently with reduced lead time for process planning, making it cost-effective for industrial applications.

Originality/value – Different representation schemes, initial population generation strategy and modifications in local and global pollination rules are introduced in the IFPA. Moreover, hybridization with GA is proposed to improve convergence speed and robustness/consistency in finding globally optimal solutions.

Keywords Assembly sequence optimization, Evolutionary optimization, Improved flower pollination algorithm

Paper type Research paper

1. Introduction

Because the assembly cost is estimated to be around 10-30 per cent of the total manufacturing cost (Hong and Cho, 1995), proper assembly process planning is of utmost importance. Typically in mechanical product assemblies, the components need to be assembled in a definite order or sequence to make the assembly feasible such that no component interferes with others, implying that they must comply with certain precedence constraints. There may be other constraints affecting the feasibility of a sequence like individual component and tool accessibility, stability of the assembled components. Again a product may be possible to assemble in many alternative ways following different sequences. The optimal assembly sequence takes the least time and thus results in the minimum cost. Assembly sequence optimization can be done considering various criteria like minimizing the number of orientation changes and the tool changes, stability of the assembly, etc. A significant amount of human expertise and experiential knowledge is needed for assembly sequence optimization after considering all above constraints and criteria. Besides manually determining the optimal sequence is also laborious and time-consuming. For automating the generation of optimum assembly sequence, different computer-aided process planning approaches had been developed. They included applications of algorithms and different graph theoretic approaches. With increase in number of components, the number of feasible assembly sequences also rises staggeringly, resulting in an increase in computational complexity. As a result, the assembly sequence optimization becomes more difficult, and the traditional approaches may be often computationally expensive (Gao *et al.*, 2010). Various soft computing-based evolutionary algorithms like genetic algorithm (GA) and swarm intelligence-based algorithms and their hybrid have been also used to deal effectively with this problem of combinatorial optimization. However, with these algorithms, there are still challenges like convergence speed, computational time, maintaining robustness/consistency in finding the global optimal solution, etc. Keeping this in mind, in this paper, we have proposed two soft computing-based approaches for discrete optimization using an improved flower pollination algorithm (FPA) and a

The current issue and full text archive of this journal is available on Emerald Insight at: **www.emeraldinsight.com/0144-5154.htm**

Assembly Automation
39/1 (2019) 165–185

[DOI 10.1108/AA-09-2017-112]

Received 9 June 2017
Revised 18 September 2017
29 November 2017
9 February 2018
1 June 2018
Accepted 22 July 2018

hybrid FPA-based approach by hybridizing with GA, which have shown promising results.

The paper is divided into following sections. A brief review of previous research is presented followed by description of the assembly sequence optimization problem. Next the basic principle of the original FPA is given. Then we have presented our proposed approaches of improved discrete FPA and the improved hybrid GA-FPA. Finally, a case study is provided to demonstrate the application of the proposed approaches and the comparison results with well-known soft computing algorithms.

2. Review of previous research

The following presents a brief review of previous research on different strategies used for assembly data modelling, approaches developed for solving the assembly sequence optimization problem as well as attempts to integrate assembly process planning with assembly product development, scheduling, line balancing and other problems.

Assembly data modelling provides a logical structure for organizing all the information necessary for solving the problem of assembly sequence planning and optimization. This includes input information about the contacts, spatial relationships, precedence and collision interference relationships between the components, geometric constraints, assembly stability, etc. as well as how to represent the assembly sequences. A brief review of previous research on different approaches for the above assembly data modelling and assembly sequence representation is given below.

For assembly data modelling, mainly graph-based and matrix-based methods have been used by researchers. The graph-based methods include mainly liaison graph, directed/precedence graph and disassembly completed graph (DCG). Cao and Xiao (2007) used the liaison graph to represent the contacts between components. To represent spatial and precedence relationships, directed/precedence graph had been used by Laperriere and ElMaraghy (1994), Bonneville *et al.* (1995), Xie *et al.* (2007), Tseng *et al.* (2010), Gao *et al.* (2014). To represent components and their disassembly directions, the DCG had been used by Wang *et al.* (2005). It is, however, important to note that one problem with the graph-based model could be the fact that the nodes of the graph-based model tend to increase exponentially with increased product complexity, resulting in difficulty of constructing the graph automatically and decrease in algorithm efficiency (Yu and Wang, 2013).

Different matrix-based modelling methods that had been used by researchers for assembly data modelling include liaison matrix, interference matrix, disassembly matrix, extended interference matrix, precedence matrix, moving wedge matrix and stability matrix (SM). The liaison/connection matrix had been used to represent contact between components (Cao and Xiao, 2007; Wang and Liu, 2010; Zhou *et al.*, 2011). To represent the collision interference relationships along the axes of Cartesian coordinates ($\pm x$, $\pm y$, $\pm z$), the interference matrix had been used by Cao and Xiao (2007), Lv and Lu (2010a), Gao *et al.* (2010), Zhou *et al.* (2011), Ghandi and Masehian (2015) and Li *et al.* (2016). The disassembly matrix had been used to reduce the complexity by combining the directional information of x-, y-, z- of interference matrix (Wang *et al.*, 2005; Sharma *et al.*, 2008; Zhou *et al.*, 2013). The extended interference matrix had been used to have directional information when the assembly directions are other than the orthogonal Cartesian axes (Yu *et al.*, 2009; Yu and Wang, 2013). The precedence matrix had been used to convey the information about the precedences between parts and determine feasible assembly sequences that satisfy precedence constraints (Tseng *et al.*, 2007; Choi *et al.*, 2009; Tseng *et al.*, 2010, 2011; Mishra and Deb, 2016). The moving wedge matrix had been used to describe geometric constraints between components in an assembly (Chen and Liu, 2001; Shuang *et al.*, 2008). The SM had been used to represent the stability values for two consecutive parts (Li *et al.*, 2016). Since matrix-based assembly models are easier than graph-based models to store and express the static or dynamic relationships between components, they are widely used to solve the problems of assembly sequence planning problems (Yu and Wang, 2013). In the present paper, matrix-based methods have been used for modelling the precedence relationships between components and the assembly stability.

For representation of assembly sequences, different encoding schemes had been used by previous researchers. For example, in the work presented by Cao and Xiao (2007), Choi *et al.* (2009), Wang and Liu (2010), Lv and Lu (2010a), Li *et al.* (2016) and Mishra and Deb (2016), the sequences had been encoded in the form of an ordered list of only component numbers, while in the work reported by Wang *et al.* (2005) and Gao *et al.* (2010), the encoded sequences contained component numbers along with their assembly directions. Tiwari *et al.* (2005) represented the sequences considering the assembly operations and the product variant numbers. In the work presented by Tseng *et al.* (2007), Chang *et al.* (2009) and Gao *et al.* (2014), the sequences were encoded in the form of connectors to reduce the complexity of the problem and enable them to make use of similar data for same type of connectors. However classification of connectors from assembly models itself can be a challenging task. In the present paper, the assembly sequences are encoded as an array of component numbers.

To solve assembly sequence optimization, previous researchers had used various soft computing approaches, many of which are inspired by nature. They include the applications of various evolutionary-based optimization approaches like GA, memetic algorithm (MA) and recently a harmony search algorithm inspired by musical improvisation processes. For example, Bonneville *et al.* (1995) developed a GA-based approach with initial population as valid assembly plans which then underwent crossover and mutation followed by evaluation and selection of offspring based on minimizing the reorientations. For feasibility, liaisons and geometric constraints were used. The GA performance was reported to be slow, not guaranteeing optimum plans. Another approach for assembly sequence optimization was presented by Choi *et al.* (2009), where the assembly time and number of orientation changes were considered. Precedence matrix was used for feasibility of assembly sequences. The algorithm started with randomly generated feasible assembly sequences. Kashkoush and ElMaraghy (2013) proposed a GA approach for generating assembly sequences for sequential, non-linear product

assemblies. Partial assembly tree was used for assembly representation, and to code the tree information, i.e. sequences of leaves and topology, matrices were used. Tseng *et al.* (2007) developed an MA-based approach, where assembly components were categorized into different types of connectors considering fastener types, assembly tools, directions and connector-based precedence graph. Similarity of engineering data of connectors was considered for evaluation of assembly sequences. Partially mapped crossover (PMX) and guided mutation operator followed by a binary search tree were used. It has been reported in Li *et al.* (2003) that although GA can deal with a complex product assembly planning optimization problem, during the process the neighbourhood may converge too fast which can limit the search to a local optimum prematurely. To overcome this drawback of GA, researchers have tried to combine it with other local search optimization techniques. Gao *et al.* (2010) developed an assembly sequence optimization approach based on MA where a chromosome represented an assembly sequence consisting of genes containing component number and direction variable. They considered times of the assembly direction changes and assembly feasibility. PMX and swap mutation, in addition to local search, were used. Wang *et al.* (2013) proposed an enhanced harmony search algorithm for assembly sequence planning. To strengthen the optimization performance and its effectiveness for solving assembly sequence planning (ASP), the method uses the largest position value rule and an efficient switching strategy of local search. The feasibility was ensured by interference matrix. Li *et al.* (2016) proposed an improved harmony search (IHS) algorithm. A discretization encoding technique for ASP was proposed that reduced the searching space followed by an initial harmony using opposition-based learning (OBL) strategy. A way to improvise a new harmony and local search scheme was developed. The comparison with other algorithms like GA and MA was reported on the basis of robustness/consistency of finding global best solutions.

Various swarm intelligence-based approaches like particle swarm optimization (PSO), ant colony optimization (ACO), artificial immune system (AIS) and recently firefly algorithm and FPA have been also explored for solving the problem of assembly sequence optimization. For example, Wang and Liu (2010) developed a PSO-based assembly sequence optimization approach. The assembly cost was subjected to geometrical constraints and five assembly process constraints. However, in this work, the initial assembly sequence should be selected at first which caused a long running time (Li *et al.*, 2013a). Lv and Lu (2010a) also used a PSO approach in assembly sequence optimization. They considered tool changes, orientation changes, operation-type changes and interference times in product assembly. Special operators, namely, subtraction operator, addition operator and multiplication operator, were used to update the position and velocity of particles. But their algorithm was easily affected by the individual optimal fitness value, and the algorithm was easy for convergence in the starting evolution (Li *et al.*, 2013a). Li *et al.* (2013a) developed an improved PSO algorithm for high-speed trains assembly sequence planning. The position, velocity and mathematical operations of the PSO were redefined to solve the ASP problem. A choose strategy of global optimal position was proposed to overcome the local convergence problem of basic PSO in the early iteration. Problem with using original PSO has been reported in the paper by Yu and Wang (2013) that it is not favourable for the discrete problem such as ASP, where the solution is in discrete integer space and another important issue is that PSO is easily trapped in local optimum. To overcome it, Yu and Wang (2013) proposed an improved ACO called MMACS, i.e. max-min ant colony system, in which the multi-objective heuristic function consists of reorientation, parallelism, continuity, stability and auxiliary strokes. The pseudo-random proportional rule and local updating rule of ACS had been combined. Their approach was inferior in running time and occupying space than the priority rule screening. It is worth mentioning that it is desirable to have fast convergence speed of the algorithm; however, it also gives the problem of premature convergence. Hence, a balanced convergence speed should be maintained to achieve the global optimum with robustness which happens when there is a proper balance between exploration and exploitation in the algorithm. This also improves the overall quality of the solutions (Yu and Wang, 2013). In general, this problem of premature convergence and slow convergence of soft computing-based optimization algorithms had also been discussed in the paper by Ting *et al.* (2015). Wang *et al.* (2014) developed an ACO algorithm for assembly sequence optimization. A disassembly feasibility information graph was generated and extended to retrieve the relevant data (including constraint relationships) of assembly and disassembly planning. Tiwari *et al.* (2005) developed an AIS approach for assembly configuration planning using Maslow's need hierarchy theory and theory of clonal selection. They attempted to solve the combinatorial problem using few iterations by making it computationally efficient. Cao and Xiao (2007) used immune optimization algorithm (IOA) to generate optimal assembly plan. It was reported that GA, being a global search method and lacking efficient local search capability, misses the optimal case and hence is not much robust, and because of evolution process of population, the diversity is lost, making it computationally expensive to converge. Assembly sequences were evaluated based on number of components, assembly direction changes, tool changes, etc. Bahubalendruni *et al.* (2016) developed an advanced immune strategy to obtain optimal assembly sequence. Two separate immune models, namely, bone marrow model and negative selection models were developed as part of their work. Zhang *et al.* (2016) developed a discrete double-population firefly algorithm. It guaranteed the population diversity reducing chances of premature convergence and enhanced local and global search capabilities by parallel evolution of solutions. Objectives including assembly stability, assembly polymerization and assembly direction changes had been considered. Mishra and Deb (2016) developed a discrete FPA-based assembly sequence optimization approach for generating not only the global best assembly sequence but as many unique optimum solutions as possible. However, it was reported in the paper that this virtue of maintaining diversity in optimum solutions came at the expense of slow convergence compared to other algorithms like GA, ACO and IHS, which is one of the shortcomings of their approach.

There had been applications of hybrid soft computing approaches as well to enhance the performance of the algorithm

for assembly sequence optimization. For example, Lv and Lu (2010b) improved the DPSO algorithm by combining it with simulated annealing as a multi-objective hybrid evolutionary search algorithm. This improved the search capability and obtained a better effect than DPSO. However, the convergence problem in the initial evolution was not improved notably and more iterations were required than DPSO. Zhou *et al.* (2011) combined bacterial chemotaxis with GA for assembly sequence optimization. Assembly sequences were encoded as chromosomes and a gene treated as bacterium. The fitness function comprised length of longest subsequence, number of orientation changes, gripper changes. Li *et al.* (2013b) proposed a hybrid approach by collaborating discrete PSO with an evolutionary operator to produce improved individuals guided by a series of optimal evolutionary directions belonging to current population. Interference matrix was used for geometric feasibility. In the fitness function, the numbers of components, direction changes, tool changes, and the operation type changes were considered. Zhang *et al.* (2014) presented a hybrid algorithm IPSO, where PSO was combined with immune algorithm. By effective selection mechanism of immune algorithm, the IPSO overcomes premature convergence. The objective function consisted of location of base component in the sequence, tool and direction changes, geometric feasibility and the coherence between successive components.

There had been also some attempts by previous researchers to integrate the assembly sequence planning problem with other problems like assembly line balancing and scheduling, assembly product design, plant assignment. For example, Tiwari *et al.* (2005) coupled the problem of assembly sequence planning with that of assembly line scheduling for different product variants. An attempt to combine the problem of assembly sequence optimization with plant assignment was undertaken by Tseng *et al.* (2009, 2010) for selection of a suitable plant to perform assembly operations at least cost. Tseng *et al.* (2008), Wang *et al.* (2012) and Lu and Yang (2016) attempted to combine assembly sequence planning with line-balancing which helped to ensure better quality of solution for assembly operation as well as minimize lead time to launch the product. Recently, Gruhier *et al.* (2015) presented an integrated approach to assembly product design and assembly sequence planning.

It is worth noting that other than the problem of premature convergence and slow convergence for complex assemblies, the areas which need to be focused on are easy encoding, deviations of the optimal solution after each independent run (least mean square error), how to choose the weight coefficients of the fitness function, etc. From the review of previous research given above, evidently soft computing-based optimization algorithms hold lot of promise in solving the combinatorial assembly sequence optimization problem. The researchers, drawing inspiration from natural phenomena, continue to seek and develop newer and better algorithms to overcome challenges like improving convergence speed without getting into local optima, computational time and robustness/consistency in finding the global optimal solution. With above challenges in mind, we have proposed in this paper two new soft computing-based approaches based on discrete optimization using an improved FPA (IFPA) and a hybrid FPA based approach by hybridizing with GA, which have shown promising results.

3. Description of the problem of assembly sequence optimization

A typical mechanical assembly product is composed of number of distinct components. For an assembly having n individual components, where each component is denoted by p_i, where $i = 1, 2, 3 \ldots n$, the assembly sequences can be presented as the ordered list of components, where the assembly sequence must contain all the parts of the assembly. These assembly sequences can either be feasible or infeasible. Feasible sequences are those sequences that are possible to assemble without violating precedence constraints, otherwise they are referred as infeasible. These infeasible sequences are of no meaning to manufacturing practice. A typical assembly sequence for an assembly consisting of four components can be written as:

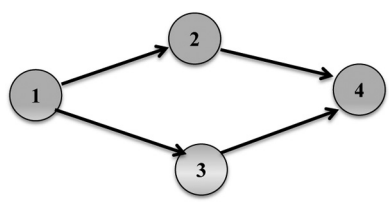
$$S_{typical} = [p_4,\ p_1,\ p_2,\ p_3]$$

This conveys that, at first, the component p_4 is selected to assemble the product and then component p_1, followed by component p_2 and so on. In the present paper for the sake of the algorithm, the assembly sequences have been presented as [4, 1, 2, 3]. If S_i be the set of all infeasible assembly sequences for a product, S_f be the set of all feasible assembly sequences and S be the set of total number of assembly sequences, then:

$$S = S_i \cup S_f \text{ and } S_i \cap S_f = \Phi \quad (1)$$

If a soft computing-based approach is used for assembly sequence planning, the developed algorithm has to find out the optimal assembly sequence out of the total assembly sequences S. This optimal assembly sequence should satisfy all the precedence constraints (i.e. it must be a feasible sequence) and also should take least time to assemble with utmost ease. In a feasible assembly sequence, all the precedence constraints, which represent a set of components that must be assembled before that component, are respected. The precedence constraints between components are often represented by a directed graph (as shown in Figure 1) or by $n \times n$ matrix (precedence matrix or PM), where n is the number of components in the assembly, and each entry $PM_{i,j}$ in the matrix is assigned the value of 1, if a precedence exists between component i and j, otherwise it is assigned the value of 0. It is obvious that the diagonal entries of the matrix would all always be zero as a component cannot have precedence with itself.

Figure 1 A typical assembly precedence diagram



$$PM = \begin{bmatrix} PM_{11} & PM_{12} & \cdots_{\ldots} & PM_{1n} \\ \cdots_{\ldots} & \cdots_{\ldots} & \cdots_{\ldots} & \cdots_{\ldots} \\ \cdots_{\ldots} & \cdots_{\ldots} & \cdots_{\ldots} & \cdots_{\ldots} \\ PM_{n1} & PM_{n2} & \cdots_{\ldots} & PM_{nn} \end{bmatrix}_{n\ x\ n}$$

$$\mathrm{PM}_{i,\ j} = \begin{cases} 1, & \textit{component } i \textit{ should be assemled only after } j \textit{ is assemled} \\ 0, & \textit{no precedence between component } i \textit{ and } j \end{cases}$$

This precedence constraint matrix needs to be provided for assembly sequence planning to check for the feasibility of the assembly sequences. The assembly sequence must be generated in accordance with the above precedence constraints. A typical precedence matrix for an assembly made up of four components i.e. A = {p_1, p_2, p_3, p_4} could be as follows:

$$PM = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

From the above given precedence matrix, it is evident that the component "2" has a precedence constraint with Component "1" as $p.m._{2,1}$ is 1, while it does not have any constraint with Components "3" and "4", as both $p.m._{2,3}$ and $p.m._{2,4}$ are 0. This means that to assemble this product feasibly, Component 2 should be assembled only after the component "1" has been assembled. Likewise, Component "3" has precedence constraint with only component "1" and Component "4" has precedences with Components "1", "2", "3". In the present work, the degree of feasibility of an assembly sequence has been measured in terms of number of feasibility violations (FV) present in the sequence. This can be found from the precedence matrix *PM* using the following equation.

$$FV = \sum_{i=1}^{(n-1)} \sum_{j=i+1}^{n} PM_{(i,j)} \tag{2}$$

where i and j are the locations of the component numbers in the sequence and n is the total number of components in the assembly. This can be understood from the example explained below, say, an assembly has four components and one of the infeasible sequences for this assembly is [p_4, p_1, p_2, p_3] and its precedence matrix is given above. For calculating the number of FVs, for an assembly sequence, the first component is selected and checked to determine with which components it has precedence, and whether it is following the precedence or not. In this example, Component "4" has precedence with Components "1" , "2" and "3" , as the value in the *PM* is 1 here for each of $p.m._{4,1}$, $p.m._{4,2}$ and $p.m._{4,3}$. So because of location of component "4" in the assembly sequence, it has a total of three FVs. Next, we move for the second component of the assembly sequence, i.e. Component "1", and check its precedence. It reveals that that it does not require any other component to be assembled before it, and hence, it has no FV. In a similar manner, each component of the assembly sequence is checked to see if it satisfies the precedence criterion or not. For the solution [p_4, p_1, p_2, p_3], Components "2" and "3" also do not have any FVs. Hence it can be concluded that the number of FVs for the infeasible solution [p_4, p_1, p_2, p_3] is 3. It is obvious that for a feasible assembly sequence, the number of FVs would be zero. Hence, it can be written as:

$$FV = \begin{cases} 0, & \textit{if the sequence is feasible} \\ \geq 1, & \textit{if the sequence is infeasible} \end{cases}$$

The following presents the details about the various criteria that have been considered to obtain the optimal assembly sequence.

3.1 Minimization of direction changes

As each component of the assembly is assembled along a direction, it becomes obvious that changing the direction of assembly of the consecutive components will increase the handling time resulting in increase in overall cost of the assembly. Hence an assembly sequence requiring least direction changes is most preferable to save time. The number of direction changes (n_d) can be mathematically computed as below.

$$\text{Total number of direction changes, } n_d = \sum_{(i=1)}^{(n-1)} (dir\_change_{i,i+1}) \tag{3}$$

where:

$$dir\_change_{i,i+1} = \begin{cases} 0, & \textit{if } assembly\_dir_i = assembly\_dir_{i+1} \\ 1, & \textit{otherwise} \end{cases}$$

$dir\_change_{i,i+1}$ indicates the change in assembly direction for two consecutive assembly operations and n is the number of components in the assembly; $assembly\_dir_i$ indicates the assembly direction of component number i (which may be one or more of the following directions, namely, $\pm x$ or $\pm y$ or $\pm z$).

3.2 Minimization of tool changes

At times, several tools and grippers (in case of robotic assembly) are required for performing the assembly or manipulation. The change in tool/gripper is a nonproductive task, as it consumes handling time leading to increase in overall assembly cost. Hence, minimizing the tool changes is also one of the important criteria to reduce the overall assembly time and cost. The number of tool changes (n_t) can be mathematically computed as below.

$$\textit{Total number of tool changes, } n_t = \sum_{(i=1)}^{(n-1)} (tool\_change_{i,i+1}) \tag{4}$$

where:

$$tool_change_{i,i+1} = \begin{cases} 0, & \textit{if } tool\_number_i = tool\_number_{i+1} \\ 1, & \textit{otherwise} \end{cases}$$

$tool\_change_{i,i+1}$ indicates the change in assembly tool/gripper for two consecutive assembly operations, and n is the number

of components in the assembly, $tool\_number_i$ indicates the tool number necessary for handling/insertion of component number i (each assembly tool is identified by a unique tool number in the tool database).

3.3 Base component location in the assembly

The base component in an assembly sequence should be the first component in the assembly order. This base component information is provided by the user. The location of the base component in the sequence is represented by B, whose value is calculated as follows:

$$\text{Location of base component, } \mathrm{B} = \begin{cases} 1, & \text{if the base component is in first position in the sequence} \\ 0, & \text{otherwise} \end{cases}$$

It may be noted that whether the base component of the product is located in the first position of the assembly sequence is considered as a factor that influences the quality of the assembly sequences (Cao and Xiao, 2007). This decision is left to the user as stated above. The selection of the base component is based on factors like it should be the heaviest and/or largest of all the components, should have most number of mating links to other components, should be easy to fixture and locate in the vise, etc. (Barnes *et al.*, 2004).

3.4 Maximization of stability of the assembly sequences

Stability of the components during the assembly is also very important, which refers to the stability among mating components during the assembly process. The unstable situation will require additional fixture to maintain stability, which might reduce efficiency and increase costs. In an assembly, the connection types, for defining stability, can be categorized into stable connection (SC), conditional stable connection (CSC) and unstable connection (USC). As the name suggests, SC signifies inseparable contact between components. Sometimes between components, there is CSC, which signifies that the connections are not autonomously stable, but components may separate from each other spontaneously. For example, in Figure 2, the components will separate when it is turned upside down. For unstable connections (USC), fixtures are necessary to hold the components in place and maintain contact. This classification scheme and its details can be found in the paper by Li *et al.* (2016). Like precedence matrix, the above information can also be modelled in the form of a matrix, having size of $n \times n$, where n is the number of components in the assembly and each entry $l_{i,j}$ in the matrix is assigned a value depending upon the connection type. A typical SM for n-component assembly could be represented as:

$$SM = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}_{n \times n}$$

where l_{ij} represents the stability between component i and component j. The value of each element of the SM can be 0, 1 or 2 depending on whether the connection type is unstable, conditionally stable or stable, respectively. Thus, it can be written as follows:

$$l_{ij} = \begin{cases} 2, & \text{If the connection type between component } i \text{ and component } j \text{ is SC or CSC. It is better to use a fixture to clamp component } i \text{ than clamp component } j. \\ 1, & \text{If the connection type between component } i \text{ and component } j \text{ is CSC. It is better to use a fixture to clamp component } j \text{ than clamp component } i. \\ 0, & \text{if there is no connection between component } i \text{ and component } j, \text{ or the connection type between component } i \text{ and component } j \text{ is USC. Or the connection type is SC or CSC, and it is desirable to use a fixture to clamp component } j, \text{ but not feasible to clamp component } i. \end{cases}$$

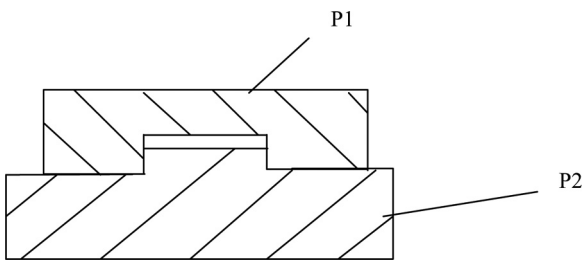
The stability of the assembly sequences is measured in terms of stability index (SI). Larger the value of SI, more stable the assembly sequence is. The value of SI can be found out from the SM using equation (5) (Li *et al.*, 2016):

$$SI = \sum_{i=2}^{n} S_i \mid 0 \le SI \le 2n - 2 \tag{5}$$

where S_i is the stability of the component "i" and n is the number of components in the assembly. If there is any element among l_{ij} $(1 \le j \le i - 1)$ equal to 2, $S_i = 2$; otherwise, if any element among l_{ij} $(1 \le j \le i - 1)$ equal to 1, $S_i = 1$. If all the elements among l_{ij} $(1 \le j \le i - 1)$ are 0, $S_i = 0$.

The problem of assembly sequence optimization involves determining the best feasible assembly sequence based on one or more criteria such as minimization of number of direction changes and tool changes, maximizing the stability of the components/sub-assemblies while performing the assembly, avoiding FVs in the sequence, location of base component in the sequence, etc. Figure 3 shows the different assembly process constraints and the optimization criteria in assembly sequence planning. Several evolutionary soft computing-based approaches were used by previous researchers to solve the above problem. A new approach based on pollination of flowers, which was originally developed by Yang (2012) for continuous optimization, is applied by us after introducing some modifications to solve the current discrete optimization problem that we have on hand.

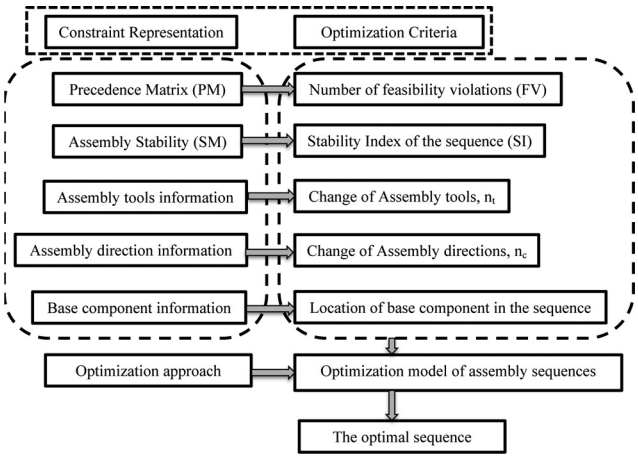
Figure 2 Example of conditional stable connection



4. Proposed approach for assembly sequence optimization

The FPA was developed by Yang (2012) for solving continuous optimization problems. It was inspired by the pollination process of flowering plants. Through biology, it is established that the pollination can be abiotic or biotic. In case of biotic pollination group (accounting for almost 90 per cent of flowering plants), pollens are transferred by pollinators such as insects and animals. On the other hand, in case of abiotic

Figure 3 Assembly process constraints and optimization criteria related to assembly sequence planning



pollination group (accounting for almost 10 per cent of flowering plants) pollens are transferred through the help of wind and diffusion. A good example of abiotic pollination is that of grass. Pollinations can be achieved by self-pollination or cross-pollination. In case of cross-pollination, pollination occurs from the pollen of a flower of a different plant, while self-pollination is the fertilization of one flower from pollen of the same flower or different flowers of the same plant. Biotic cross-pollination may occur over long distances; pollination performed by bees, bats, birds and flies, which can fly over long distances, can be considered as global pollination. Moreover, bees and birds may exhibit Levy flight behaviour with jump or fly distance steps following a Levy distribution.

The FPA consists of two pollination rules, namely, global pollination and local pollination. Biotic and cross-pollination can be considered processes of global pollination and pollen-carrying pollinators move in a way that obeys Levy flights. This process can be mathematically represented as:

$$x_i^{t+1} = x_i^t + \gamma L\ \left(g^* - x_i^t\right) \tag{6}$$

where x_i^t is the pollen i or solution vector x_i at iteration t, x_i^{t+1} is the pollen i or solution vector x_i at iteration $(t + 1)$, $g^\star$ is the current best solution found among all solutions at the current generation/iteration. γ is a scaling factor to control the step size. Here, the Levy distribution equation is represented as:

$$L = \left(\lambda\,\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)/\pi\right) * \left(1/s^{1+\lambda}\right) \tag{7}$$

where L is step-size parameter, Γ is the standard gamma function. It is to be noted that in this work, the value of γ is taken as 1.

On the other hand, abiotic and self-pollination make up for the local pollination process that is mathematically represented as:

$$x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right) \tag{8}$$

where x_i^t is the pollen i or solution vector x_i at iteration t, $\in$ is a random number using uniform distribution, x_j^t is a pollen from a different flowers of the same plant species at iteration t. x_k^t is a pollen from another different flower of the same plant species at iteration t.

Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to similarity of two flowers involved. The selection of type of pollination process (i.e. global or local) is decided using a probability termed as switch probability.

The FPA was shown to give promising results in solving continuous optimization problems like economic dispatch problems in electrical power systems (Dubey *et al.*, 2015) and in the field of design (Sabarinath *et al.*, 2015). However, the main challenge to apply FPA to solve the given discrete assembly sequence optimization is the continuous nature of basic FPA, whereas in assembly sequence optimization, the solution search space is discrete, comprising discrete assembly sequences. A discrete FPA approach was proposed by Mishra and Deb (2016) for assembly sequence optimization, but it was found to suffer from slower rate of convergence and hence required more computational time compared to other algorithms like GA, ACO and IHS.

Keeping the above in mind, in the present paper, we have proposed an improved discrete FPA-based approach for assembly sequence optimization, and to further improve the performance of this FPA, we have proposed a hybrid GA-FPA–based approach. These approaches have been implemented for solving the problem of assembly sequence optimization and the results have been compared with the previous version of discrete FPA and also with well-known soft computing algorithms like GA, MA and IHS. The hybrid GA-FPA proposed by us has shown promising results in solving the given discrete optimization problem on hand in terms of higher global fitness and higher average fitness, improved consistency in generating global optimum solutions and faster convergence (especially when compared with the previously developed variant of FPA and IHS).

4.1 Improved discrete FPA for assembly sequence optimization

First of all, we shall present the modifications to the basic FPA in order to develop the improved discrete FPA approach proposed in this paper. The following are the new data representation scheme of FPA and new rules for local and global pollination being proposed by us.

4.1.1 Proposed assembly sequence representation scheme and initial population generation

In the improved discrete FPA-based approach, an assembly sequence is encoded as an array of component numbers. Thus, [1, 2, 3, 4, 5, 6, 7, 8] denotes an assembly sequence of eight components, where 1, 2, 3 and so on are unique component numbers and the order in which they appear is the sequence in which they have to be assembled. It is to be noted that each component is identified by a unique number and can appear only once in an assembly sequence. This implies that repetitions of component numbers in a sequence are not allowed.

The population of flowers (i.e. assembly sequences) in the FPA is generated randomly without considering the feasibility of these sequences. This gives the advantage of maintaining diversity in flowers during the course of the algorithm and helping to avoid the local optima. For cases like assembly sequence planning, the infeasible sequences nearer to the

optimal solution space can help identify the optimal solution (Zhang *et al.*, 2016). To expand the field of initial solutions (i.e. assembly sequences) in the search space, an emerging method, namely, OBL is used, which was originally proposed by Tizhoosh (2005). Recently, this method was used in the area of assembly sequence planning by Li *et al.* (2016).

After the generation of flowers using the OBL method, they are evaluated using the fitness function and then based on the population size of FPA, the best flowers, which will undergo pollination, are selected. It is observed that the performance of FPA increases significantly if the initial population is generated using the OBL method. Suppose that $P(x_1, x_2, x_3, \ldots, x_d)$ is a solution vector of D dimensions and $x_i \in [x_i^L, x_i^U]$, $i = 1,2,3,\ldots D$, x_i^L and x_i^U are the lower and upper limits of the variable x_i, respectively. In the OBL method, the opposing solution vector OP $(x_1', x_2', x_3', \ldots, x_d')$ can be defined as follows:

$$x_i' = x_i^L + x_i^U - x_i \tag{9}$$

4.1.2 Modified global pollination rule

In case of global pollination of original FPA (Yang, 2012), a global best flower ($g^\star$) found so far is selected from the population based on its fitness value and thereafter subtractions of individual flowers (sequences) x_i^t are performed from the aforementioned global best. The resultant $(g^\star - x_i^t)$ is multiplied with the number L generated by Levy flight. In the present problem of assembly sequence optimization, the solution search space is discrete and repetitions of component numbers in a sequence are not allowed. Therefore, the following modifications in global pollination are made by us to make it suited for solving the given problem on hand.

Each flower is, in this case, an assembly sequence x_i^t represented by an array of component numbers that must be unique. The global best flower (or sequence) $g^\star$ is selected and then element-wise subtractions from the array representing individual flowers (or sequences) are performed from the global best flower. It may so happen at this stage that some of the elements of the resulting array $(g^\star - x_i^t)$ are negative numbers, in which case the absolute value is considered. This array is then multiplied by an array L of randomly generated numbers following the Levy distribution. The values of L are generated from equation (7), which gives an array of whole numbers. The property of the Levy flight is that it generates most of the time a small number, and suddenly it jumps to a very large number randomly thus mimicking the Levy flight behaviour of birds. The resulting array $L^\star(g^\star - x_i^t)$ is then added to the flower (or sequence) under consideration x_i^t using element-wise additions to obtain a new sequence x_i^{t+1}. These steps lead the sequence under consideration to move towards the global best found so far. The example given below illustrating the modified global pollination rule illustrates the above steps:

$$\begin{aligned}
g^* &= [4\ 1\ 6\ 3\ 5\ 7\ 2\ 8] \\
x_i^t &= [3\ 2\ 4\ 1\ 6\ 7\ 5\ 8] \\
(g * -x_i^t) = [1\ -1\ 2\ 2\ -1\ 0\ -3\ 0] &\sim [1\ 1\ 2\ 2\ 1\ 0\ 3\ 0] \\
L &= [1\ 1\ 8\ 1\ 1\ 1\ 1\ 4] \\
L * (g * -x_i^t) &= [1\ 1\ 16\ 2\ 1\ 0\ 3\ 0] \\
x_i^t &[3\ 2\ 4\ 1\ 6\ 7\ 5\ 8] \\
x_i^{t+1} = x_i^t + L * (g * -x_i^t) &= [4\ 3\ 20\ 3\ 7\ 7\ 8\ 8]
\end{aligned}$$

It is to be noted that in the resulting array x_i^{t+1}, the values of those elements, which exceed the maximum number of components, are replaced following the normal distribution by randomly generated numbers whose values are less than the maximum number of components. In the example given above, the third number in x_i^{t+1} is "20", which is more than 8, i.e. the maximum number of components present in the assembly, and hence this number is replaced with a uniformly distributed random value, say "2" here. The assembly sequence solution will become:

$$x_i^{t+1} = [4\ 3\ 2\ 3\ 7\ 7\ 8\ 8]$$

A repetition of any component number will also render a sequence infeasible. To make the sequence feasible, a repair strategy is adopted to replace the repeated component number with the missing component number based on precedence constraints. First, those elements that are missing and the elements that are repeated in the resulting array are identified; then their number of precedences is calculated from the precedence matrix (PM). Now, the components which have fewer precedences are moved forward in the order so that the resulting sequence tends to be feasible after repair. This helps in generating a feasible array, i.e. sequence.

In the above example, if we take the sequence x_i^{t+1} = [4 3 2 3 7 7 8 8] after global pollination for repair strategy, we find that component numbers "3, 7, and 8" are each appearing more than once and the component numbers "1, 5 and 6" are missing from the resulting sequence. Now, from the PM, the number of precedences for each component is calculated and whichever component has the least precedence, is moved forward in the order. Now, if Component 3 has equal number of precedences as that of Component "1", then Component "1" will be placed in place of first "3" in the sequence, and the resulting sequence becomes x_i^{t+1} = [4 1 2 3 7 7 8 8]. Now, the component numbers "7 and 8" are appearing more than once in the solution, and component numbers "5 and 6" are missing from the sequence. Hence, we check the PM and find that Component "5" has lesser number of precedences than that of Component "7"; hence, the first component number "7" is replaced with Component "5". The resulting solution becomes x_i^{t+1} = [4 1 2 3 5 7 8 8]. Thereafter, it is found that only component number "8" is appearing more than once, while component number "6" is missing which has lesser precedence than Component "8". Hence, the location where the Component "8" is appearing first time is replaced with Component "6". The process goes on until there are no missing component numbers left. Finally, we find that the assembly sequence becomes x_i^{t+1} = [4 1 2 3 5 7 6 8]. With the help of this mechanism, the infeasible assembly sequences start moving towards the feasible solution space and may randomly visit the infeasible solution space, neighbouring some good quality solutions.

4.1.3 Modified local pollination rule

In case of local pollination of original FPA (Yang, 2012), two flowers (or sequences), say x_j^t and x_k^t are picked up from the same population to perform the local pollination and then element-wise subtraction between the arrays representing the two flowers (or sequences) is performed. The resulting array

$(x_j^t - x_k^t)$ is then multiplied by $\in$, which is an array of randomly generated numbers 0 and 1. This resulting array is further added to the sequence under consideration x_i^t to obtain a new sequence x_i^{t+1}. Because two sequences are randomly picked up from the same population as that of solution under consideration, this emulates the local pollination in the algorithm. The following example illustrating the modified local pollination rule illustrates the above steps.

$$\begin{aligned}
x_i^t &= [4\ 2\ 3\ 1\ 5\ 8\ 6\ 7]\\
x_j^t &= [2\ 3\ 4\ 1\ 5\ 7\ 6\ 8]\\
x_k^t &= [3\ 2\ 5\ 4\ 6\ 7\ 1\ 8]\\
(x_j^t - x_k^t) = [-1\ \ 1\ \ -1\ \ -3\ \ -1\ \ 0\ \ 5\ \ 0] &\sim [0\ 1\ 0\ 0\ 0\ 0\ 5\ 0]\\
\in &= [0\ 1\ 1\ 0\ 0\ 1\ 0\ 0]\\
\in * (x_j^t - x_k^t) &= [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]\\
x_i^{t+1} = x_i^t + \in * (x_j^t - x_k^t) &= [4\ 3\ 3\ 1\ 5\ 8\ 6\ 7]
\end{aligned}$$

It is to be noted that in the resulting assembly sequence x_i^{t+1}, some of the component numbers may be repeated. These repeated numbers are replaced with the missing component numbers using the same repair strategy based on the precedence matrix (PM) as already explained in section "Modified global pollination rule".

4.1.4 Precedence constraints and evaluation criteria

The fitness function (FF) used here comprises number of tool changes, number of direction changes, the location of base component in the assembly sequence, stability of the assembly and number of FVs. The purpose of considering FVs is to differentiate the feasible solutions from the infeasible solutions. This helps in pruning of the infeasible solutions from the population. The FF for evaluating the sequences (i.e. flowers) is given in equation (10):

$$FF = w_1 \frac{(n-1-n_d)}{(n-1)} + w_2 \frac{(n-1-n_t)}{(n-1)} + w_3 * B + w_4 * \frac{SI}{(2n-2)} + \frac{1}{\left(\frac{FV}{n}+1\right)} \quad (10)$$

where w_1 is the weight coefficient for the direction changes, w_2 is the weight coefficient for the tool changes, w_3 is the weight coefficient for the base component location, w_4 is the weight coefficient for stability and FV is the number of FVs in an assembly sequence.

4.1.5 Steps to obtain the optimal sequence by the improved discrete FPA

The steps for determining the optimal assembly sequence by the improved discrete FPA are summarized in steps to obtain the optimal sequence by the IFPA.

Steps to obtain the optimal sequence by the IFPA

Step 1: Enter as input the following information about the components of the assembly: component numbers, their assembly directions (±X/±Y/±Z), their tool and gripper requirements, the base component, the Precedence Matrix (PM) and the Stability Matrix (SM).
Step 2: Initialize the FPA parameters, namely, population size, switch probability, step size and maximum number of iterations.
Step 3: Generate randomly a population of flowers, apply opposition-based learning (OBL) method to generate opposite individuals and evaluate them using the fitness function given in equation (9).
Step 4: Store the best assembly sequence and its fitness value.
Step 5: Perform the global or local pollination on the population individuals based on switch probability.
Step 6: Use the proposed repair strategy to avoid repetitions of component numbers in the resulting sequences.
Step 7: Evaluate the population again and retain those individuals having improved fitness value; otherwise discard them.
Step 8: Update the best sequence and its fitness value; if found.
Step 9: Increment the iteration counter by one.
Step 10: Repeat steps 5 to 8, until the maximum number of iterations is reached.

The above improved discrete FPA has been implemented for solving various assembly sequence optimization problems and found to give promising results as given in section "Case studies: results and discussions" that are better than the discrete FPA proposed by Mishra and Deb (2016) and also better than some of the well-known soft computing algorithms like GA, MA and IHS in terms of higher fitness, improved consistency in generating global optimum solutions, faster convergence and reduced computational time.

4.2 Hybrid GA-FPA–based approach for assembly sequence optimization

We have further proposed a hybrid GA-FPA–based approach, where GA is used ahead of FPA to explore the search space and then FPA is used to exploit the available good candidates (solutions) as well as explore the search space further. The hybrid GA-FPA has been implemented for solving various assembly sequence optimization problems and was found to give even better results as given in section "Case studies: results and discussions" when compared to the improved discrete FPA that is discussed in the previous section.

4.2.1 Generation of the initial population

The population individuals (or sequences) are generated randomly without considering their feasibilities and then OBL method is applied to generate the opposite individuals. The generated population individuals are evaluated using the fitness function and the good quality solutions are selected.

4.2.2 Steps to obtain the optimal sequence by the hybrid GA-FPA

The steps for determining the optimal assembly sequence by the hybrid GA-FPA are summarized to obtain the optimal sequence by the hybrid GA-FPA.

Steps to obtain the optimal sequence by the hybrid GA-FPA

Steps 1: Generate the initial population randomly.
Step 2: Use the OBL to generate the opposite individuals.
Step 3: Evaluate the population using the

fitness function and store the good solutions.
Step 4: Run the improved hybrid GA-FPA.
Step 4.1: Run the GA.
Step 4.1.1: Perform the roulette wheel selection on the population.
Step 4.1.2: Apply the Partially Mapped Crossover (PMX) operator to generate new offsprings.
Step 4.1.3: Apply the Swap mutation operator.
Step 4.1.4: Evaluate the resulting population and compare the offsprings with their parents; and update the best population individuals.
Step 4.2: Store the best candidate found so far and its fitness value.
Step 4.3: Run the improved discrete FPA.
Step 4.3.1: Generate a uniformly distributed random number and compare with the switch probability.
Step 4.3.2: Perform global or local pollination on the population individuals based on switch probability.
Step 4.3.3: Evaluate the population and replace the parent flowers with those flowers that are found to be better.
Step 4.4: Store the best candidate found so far and its fitness function and the average fitness of the population.
Step 5: Increment the iteration counter and go to step 3 until maximum number of iterations are reached.

5. Case studies: results and discussions

Two case studies involving a 15-component generator assembly (Wang and Liu, 2010) and 26-component worm gear reducer assembly (Xu *et al.*, 2012) are provided in this section to demonstrate the applications of the improved discrete FPA and the hybrid GA-FPA approaches for assembly sequence optimization. Further their results have been compared with those of the original FPA earlier proposed by Mishra and Deb (2016) and also with well-known soft computing algorithms like GA, MA and IHS.

5.1 Case study I

The generator assembly (adapted from the paper by Wang and Liu, 2010) consists of 15 components as shown in Figure 4. The PM and the SM are also shown that must be provided by the user. The information on assembly directions and tools/grippers is given in Table I that must also be provided by the user. The base component for the assembly is Component 1.

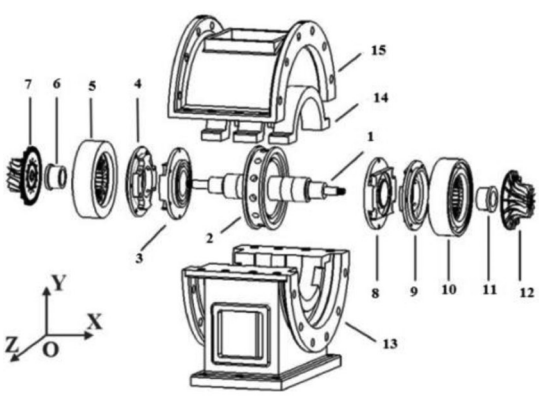
5.1.1 Computational results of application of the improved discrete FPA

We shall first present the computational results of application of the IFPA. There are three main control parameters used in the proposed FPA, namely, population size, switch probability (between 0 and 1) and the step size s used to calculate the value of L in the global pollination rule. Initially in our FPA simulation runs for the given case study, the population size was fixed at 5, and the step size was fixed at 1, switch probability (p) was increased from 0.1 to 0.9 in steps of 0.4, keeping maximum number of iterations constant at 500. It may be noted that the value of γ is 1. The population size was increased from 5 to 25 in steps of 5. The step size was increased from 1 to 15. The best sequence obtained by the FPA is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], which has least number of direction changes (i.e. 3) and tool changes (i.e. 8), no FVs in this assembly sequence, the base component provided by the user (in this case component number 1) at the first location in the assembly and the most stable configuration in accordance with SM. The maximum fitness (corresponding to the best sequence) obtained is 1.5607 [assuming the weights $w_1 = w_2 = 0.25$, $w_3 = 0.1$, $w_4 = 0.4$ in the fitness function given in equation (10)]. For 500 iterations, the set of FPA parameters that gave the optimum performance of the IFPA in terms of its consistency in finding the global best solution is as follows: population size of 20, step size of 15 and switch probability of 0.1. For the above set of parameters, the consistency of finding the global best sequence was found to be 50 per cent (based on 20 independent runs of the FPA simulations). Figure 5 shows the convergence graph of the IFPA for 500 iterations.

5.1.2 Computational results of application of the hybrid GA-FPA

In this section, we shall present the results of application of the hybrid GA-FPA. For the GA-FPA simulation runs, initially the

Figure 4 (a) 15-component generator assembly from Wang and Liu (2010). Copyright © 2010 Elsevier Ltd.; (b) its precedence matrix (PM); and (c) its SM.



(a)

```
PM=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 0 0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 1 0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 1 1 0 0 0 0 0 0 0 0 0;
    1 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    1 1 0 0 0 0 0 1 0 0 0 0 0 0 0;
    1 1 0 0 0 0 0 1 1 0 0 0 0 0 0;
    1 1 0 0 0 0 0 1 1 1 0 0 0 0 0;
    1 1 0 0 0 0 0 1 1 1 1 0 0 0 0;
    1 1 1 1 1 1 1 1 1 1 1 1 1 0 0;
    1 1 1 1 1 1 1 1 1 1 1 1 1 0 0;
    1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0];
```

(b)

```
SM = [0 2 0 0 1 1 0 0 0 0 1 0 1 0 0;
      1 0 2 0 0 0 0 0 0 0 0 0 1 0 0;
      1 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      2 0 0 0 0 0 2 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      2 0 0 0 0 0 0 0 0 0 0 2 0 0 0;
      0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
      0 2 0 0 0 0 0 0 0 0 0 0 0 2 2;
      0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
      0 0 0 0 0 0 0 0 0 0 0 0 1 2 0];
```

(c)

Table I Information on the assembly directions and tools/grippers required for the 15-component assembly

Component no.	Assembly direction	Tool/Gripper no.
1	X+	1
2	X+	1
3	X+	2
4	X+	2
5	X+	2
6	X+	2
7	X+	2
8	X−	1
9	X−	4
10	X−	2
11	X−	1
12	X−	2
13	Y+	1
14	Y−	1
15	Y−	3

population size was fixed at 5, the GA crossover probability fixed at 0.9 and the GA mutation probability fixed at 0.1, the FPA step size fixed at 1 and the FPA switch probability (p) was increased from 0.1 to 0.9 in steps of 0.4, keeping maximum number of iterations constant at 500. The population size was increased from 5 to 25 in steps of 5. The FPA step size was increased from 1 to 15. The best sequence obtained by the hybrid GA-FPA is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], which has the least number of direction changes (i.e. 3) and tool changes (i.e. 8), no FVs in this assembly sequence, the base component provided by the user (in this case Component 1) at the first location in the assembly and the most stable configuration in accordance with SM. The fitness value of this sequence is 1.5607, which is calculated using equation (10). The maximum fitness obtained is 1.5607 [assuming the weights $w_1 = w_2 = 0.25$, $w_3 = 0.1$, $w_4 = 0.4$ in the fitness function given in equation (10)]. For 500 iterations, the set of parameters that gave the optimum performance of the hybrid GA-FPA in terms of its consistency in finding the global best solution is as follows: population size of 20, the GA crossover probability fixed at 0.9, the GA mutation probability fixed at 0.1, the FPA step size of 7 and the FPA switch probability of 0.1. For the above set of parameters, the consistency of finding the global best sequence was found to be 65 per cent (based on 20 independent runs of the GA-FPA simulations). Figure 6 shows the convergence graph of the hybrid GA-FPA for 500 iterations.

5.1.3 Parameter settings of the hybrid GA-FPA

The three main parameters which affect the performance of the hybrid GA-FPA are found to be population size, FPA step size and FPA switch probability. To check the influence of the above parameters on the performance of the hybrid GA-FPA, the above three parameters are varied, and each simulation run is repeated 100 times, and their results shown in the Table II along with the average of global best solution (or sequence) over 100 independent runs, consistency in finding the global best in terms of number of runs out of 100 that could find the global best and the mean square error. It is to be noted that for all the above simulation runs of the hybrid GA-FPA, the GA crossover and mutation probabilities were kept fixed at 0.9 and 0.1, respectively, as this parameter combination was found to give optimum results. From the above results, it is observed that the optimum set of hybrid GA-FPA parameters is population size of 20, FPA step size of 7 and FPA switch probability of 0.1.

5.1.4 Comparison between the results of the GA, IHS, MA, original FPA, improved FPA and the hybrid GA-FPA approaches

To compare the results of the improved discrete FPA and the hybrid GA-FPA with those of GA, IHS, MA and original FPA, we have performed 20 independent simulation runs of each of the above six algorithms for 500 iterations. For the GA simulation runs, the set of parameters that gave the optimum performance in terms of best global fitness that it could achieve is crossover probability of 0.90 and mutation probability of 0.05. For the IHS simulation runs, the set of parameters that gave the optimum performance in terms of best global fitness that it could achieve is harmony memory considering rate (HMCR) 0.9 and pitch

Figure 5 Convergence graph of IFPA for the 15-component assembly

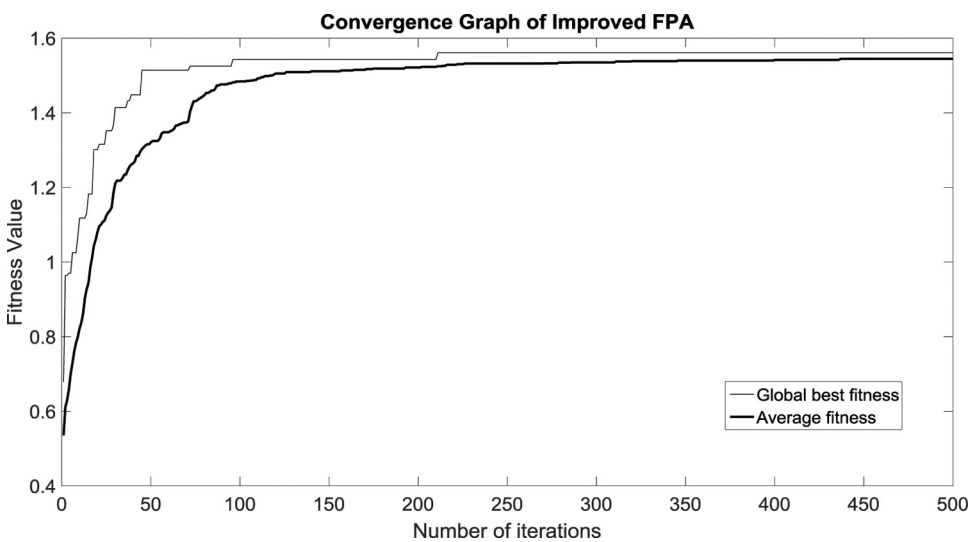
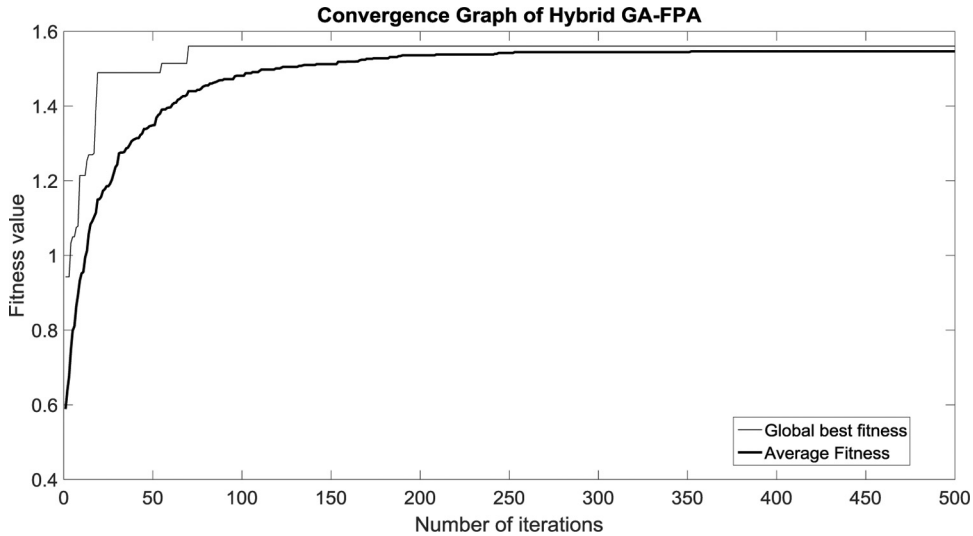


Figure 6 Convergence graph of hybrid GA-FPA for the 15-component assembly



adjustment rate (PAR) 0.1. For the MA simulation runs, the set of parameters that gave the optimum performance in terms of best global fitness that it could achieve is crossover probability of 0.80 and mutation probability of 0.02. For the original FPA simulation runs, the set of parameters that gave the optimum performance in terms of best global fitness that it could achieve is step size of 20 and switch probability of 0.2. For simulation runs of the improved discrete FPA, we have used the optimum set of parameters given in section "Computational results of application of improved discrete FPA", i.e. FPA step size of 15 and FPA switch probability of 0.1. For simulation runs of the hybrid GA-FPA, we have used the optimum set of parameters given in section "Computational results of application of the hybrid GA-FPA", i.e. the GA crossover probability fixed at 0.9, the GA mutation probability fixed at 0.1, the FPA step size of 7 and the FPA switch probability of 0.1. To maintain fairness in comparison of the results, the population size and maximum number of iterations are kept fixed at 20 and 500, respectively, for the simulation runs of all the algorithms. Moreover, the same fitness function given in equation (10) is used in all the algorithms.

Table III presents a summary of the results of comparison of all the six algorithms, namely, GA, IHS, MA, original FPA, IFPA, hybrid GA-FPA for the 15-component example. For each algorithm, the best fitness value, mean of the average fitness values, mean of the best fitness values, number of runs out of 20 independent runs in which convergence to the best assembly sequence fitness was achieved and the probability of obtaining the best assembly sequence (per cent) are given. On comparison of the above results, it is found that out of the six algorithms, three algorithms including the hybrid GA-FPA and IFPA proposed by us as well as the MA could find the optimal or the best assembly sequence, with a fitness of 1.5607 requiring least number of direction changes (i.e. 3), and also least number of tool changes (i.e. 8). However, it is important to note that the hybrid GA-FPA could find the best assembly sequence 13 times out of 20 runs, which gives a 65 per cent probability (or success rate) of finding the best sequence in 500 iterations. In contrast, the success rate of finding the best sequence by the MA is found to be only 45 per cent and the success rate for the same in case of IFPA is found to be 50 per cent, while the success rate of finding the best sequence by GA, IHS and original FPA is each equal to zero (it is to be noted that the fitness values corresponding to best sequence found by GA, IHS and original FPA are 1.5429, 1.2869 and 1.5429, respectively, which are suboptimal). Further it is noteworthy to mention that only the hybrid GA-FPA could reach the best fitness value of 1.5607 in the least number of iterations. Table IV shows a comparison of the computational time of GA-FPA and IFPA proposed by us with that of the original FPA (Mishra and Deb, 2016) for 15-components assembly. These simulations were run on a PC having Intel Core i5-3470S CPU 2.90 GHz, 8 GB RAM. The table clearly shows that the proposed hybrid GA-FPA and IFPA take much less time than the original FPA due to different representation scheme and operators used.

Table V lists the best sequences for the 15-component example that have been obtained by all the six algorithms. It further gives the corresponding fitness values, number of direction changes, tool changes, SI, location of base component in the sequence and number of FVs. On comparison of the above results, it is found that the best assembly sequence provided by proposed GA-FPA requires the least number of direction changes (i.e. 3), and also least number of tool changes (i.e. 8). Further, the stability of the assembly is found to be the highest (as indicated by SI 11) in case of the best sequence provided by GA-FPA. As evident from the comparison results given in Table III, for the 15-component example, the hybrid GA-FPA could find the optimal solution 13 times out of 20 runs, which gives a 65 per cent probability of finding the best assembly sequence in 500 iterations. In contrast, the probability of finding the best solution by the MA is only 45 per cent.

Furthermore, Figure 7 shows the convergence plots of global best fitness of the six algorithms, while Figure 8 shows the convergence plots of average fitness. On comparing the convergence plots of global best fitness for all the algorithms, it is found that the hybrid GA-FPA clearly outperforms the other five algorithms, as it is able to reach the global best fitness value in least number of iterations (i.e. only 65 iterations), as

Table II Effects of different parameter combinations on performance of the proposed hybrid GA-FPA

Population	Switch probability	Step size	Average of global best over 100 runs	Consistency in finding the global best (the global best fitness value)	Mean square error
5	0.1	1	1.4073	1 (1.5607)	0.0111
		7	1.4828	4 (1.5607)	0.0031
		15	1.4838	5 (1.5607)	0.0029
	0.5	1	1.1435	1 (1.5250)	0.0164
		7	1.4348	1 (1.5607)	0.0046
		15	1.4265	2 (1.5321)	0.0069
	0.9	1	1.0164	1 (1.3216)	0.0183
		7	1.3893	2 (1.5250)	0.0067
		15	1.3960	1 (1.5429)	0.0053
10	0.1	1	1.5142	17 (1.5607)	0.0012
		7	1.5373	26 (1.5607)	0.0004
		15	1.5288	21 (1.5607)	0.0007
	0.5	1	1.3405	1 (1.5250)	0.0112
		7	1.5067	13 (1.5607)	0.0014
		15	1.4969	6 (1.5607)	0.0016
	0.9	1	1.2084	1 (1.4982)	0.0157
		7	1.4612	1 (1.5429)	0.0022
		15	1.4595	1 (1.5607)	0.0023
15	0.1	1	1.5346	26 (1.5607)	0.0003
		7	1.5446	45 (1.5607)	0.0003
		15	1.5481	56 (1.5607)	0.0002
	0.5	1	1.4245	3 (1.5607)	0.0073
		7	1.5149	9 (1.5607)	0.0006
		15	1.5119	5 (1.5607)	0.0005
	0.9	1	1.2689	1 (1.5071)	0.0170
		7	1.4685	2 (1.5607)	0.0022
		15	1.4801	2 (1.5607)	0.0014
20*	*0.1**	1	1.5374	31 (1.5607)	0.0003
		*7**	*1.5504*	*59 (1.5607)*	*0.0002*
		15	1.5458	41 (1.5607)	0.0002
	0.5	1	1.4432	3 (1.5607)	0.0069
		7	1.5298	31 (1.5607)	0.0006
		15	1.5246	19 (1.5607)	0.0006
	0.9	1	1.3152	2 (1.5607)	0.0153
		7	1.4924	5 (1.5607)	0.0013
		15	1.4952	4 (1.5607)	0.0009
25	0.1	1	1.5438	46 (1.5607)	0.0003
		7	1.5530	59 (1.5607)	0.0001
		15	1.5509	57 (1.5607)	0.0001
	0.5	1	1.4647	2 (1.5607)	0.0043
		7	1.5228	16 (1.5607)	0.00006
		15	1.5187	9 (1.5607)	0.0006
	0.9	1	1.3644	1 (1.5607)	0.0099
		7	1.5039	8 (1.5607)	0.0012
		15	1.5039	3 (1.5607)	0.0007

Notes: *Hybrid GA-FPA parameters that gave optimum results; italic values indicates that the optimal parameters and their corresponding values for which the proposed GA-FP algorithm provides the best result

compared to GA, IHS, original FPA, IFPA and MA. IFPA and MA could reach to that solution in 210 and 365 iterations, respectively, while GA, original FPA and IHS algorithm could not attain the global best solution out of given number of iterations. This again shows that the performance of the developed GA-FPA is found to be certainly better than that of other algorithms. In short, the developed GA-FPA could achieve the global best solution in least number of iterations with faster convergence (especially when compared with IHS and original FPA) and with more robustness.

5.2 Case study II

The worm gear assembly (adapted from the paper by Xu *et al.*, 2012) consists of 26 components as shown in Figure 9(a). The

Table III Results of comparison between GA, IHS, MA, original FPA, IFPA and hybrid GA-FPA for 15-component assembly

Method	GA	IHS	Original FPA (Mishra and Deb, 2016)	MA	IFPA	Hybrid GA-FPA
Best fitness value (no. of iterations to reach the global best fitness)	1.5429	1.2869	1.5429	1.5607 (360)	1.5607 (211)	*1.5607* (65*)*
Mean of the best fitness values	1.5243	1.1834	1.5001	1.5414	1.5441	*1.5527**
Mean of the average fitness values	1.5234	1.0407	1.3037	1.5389	1.5321	1.5368
Number of runs out of 20 independent runs in which convergence to best assembly sequence fitness was achieved	0	0	0	9	10	*13**
Probability of obtaining the best assembly sequence (%)	0	0	0	45	50	*65**

Notes: *Best results obtained; italic values indicate that they are the optimum values given by the proposed GA-FP algorithm

Table IV Computational time of 20 independent runs of algorithms for 15-components assembly

Algorithm	Improved FPA	Hybrid GA-FPA	Original FPA (Mishra and Deb, 2016)
Time (s)	2.884	4.864	24.503

Table V Comparison of best sequences obtained by GA, IHS, MA, original FPA, IFPA and hybrid GA-FPA for 15-component assembly

Algorithm	Best assembly sequence	Fitness value (Success rate of obtaining the best assembly sequence out of 20 runs)	No. of direction changes	No. of tool changes	SI	Base component at the first location	FV
GA-FPA	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];	*1.5607* (65%*)*	*3**	*8**	*11**	*1**	*0**
IFPA	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];	1.5607 (50%)	3	8	11	1	0
MA	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];	1.5607 (45%)	3	8	11	1	0
Original FPA (Mishra and Deb, 2016)	[1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 6, 7, 13, 14, 15];	1.5429 (0%)	4	8	11	1	0
GA	[1, 2, 3, 8, 9, 10, 11, 12, 4, 5, 6, 7, 13, 14, 15];	1.5429 (0%)	4	8	11	1	0
IHS	[1, 2, 3, 4, 5, 6, 8, 9, 10,7, 12, 15, 11, 13, 14];	1.2869 (0%)	7	6	6	1	3

Notes: *Best results obtained; italic values indicate that they are the optimum values given by the proposed GA-FP algorithm

Figure 7 Comparison of global best fitness of GA, IHS, MA original FPA, IFPA and hybrid GA-FPA for 15-component assembly

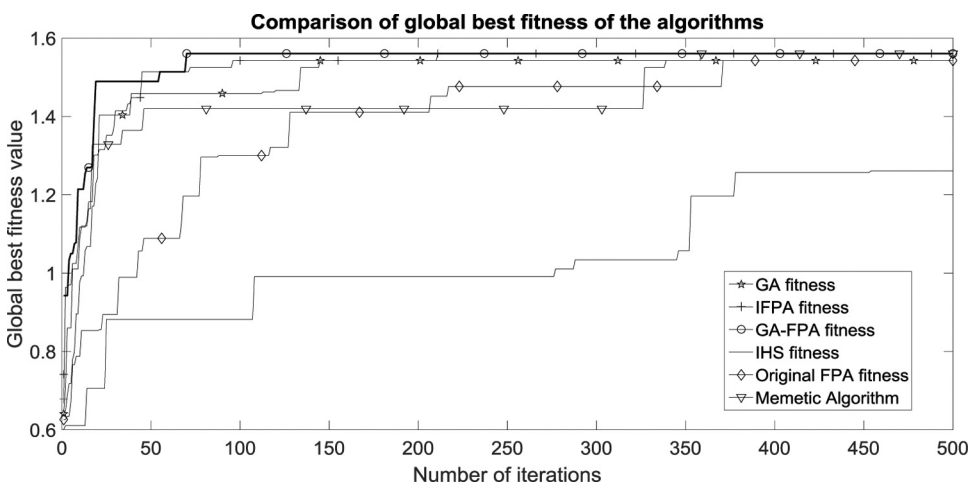


Figure 8 Comparison of average fitness of GA, IHS, MA, original FPA, IFPA and hybrid GA-FPA for 15-component assembly

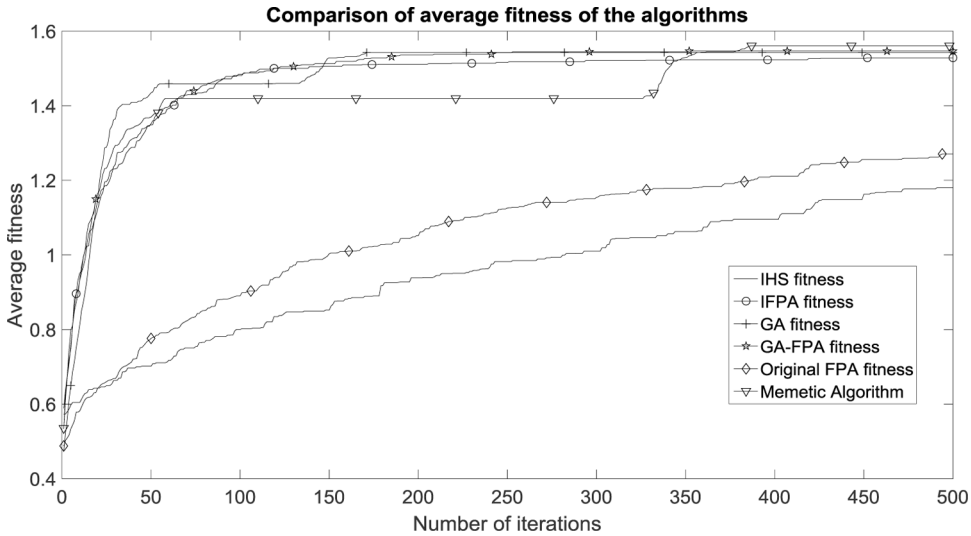
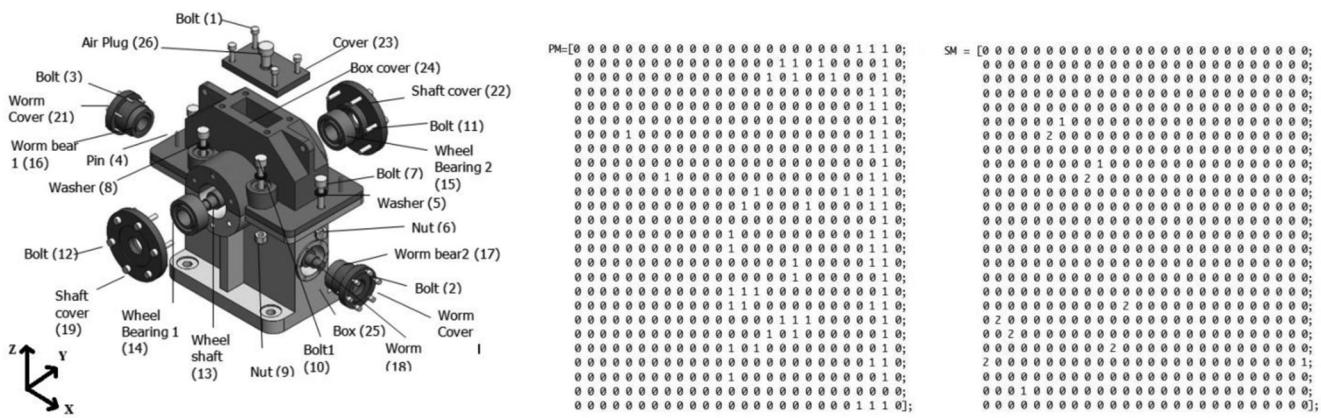


Figure 9 (a) 26-component worm gear reducer assembly (adapted from Xu *et al.*, 2012); (b) its precedence matrix (PM); and (c) its SM.



(a) (b) (c)

PM and the SM are also shown in Figure 9(b) and (c), respectively, which must be provided by the user. The information on assembly directions and tools/grippers is given in Table VI that must also be provided by the user. The base component for the assembly is Component 25.

5.2.1 Computational results of application of the IFPA

The best sequence obtained by the IFPA is [25, 6, 9, 13, 24, 8, 5, 4, 23, 1, 10, 7, 26, 15, 14, 19, 12, 18, 1, 6, 21, 3, 17, 20, 2, 22, 11] which has the least number of direction changes (i.e. 7) and tool changes (i.e. 16), no FVs in this assembly sequence, the base component provided by the user (in this case, Component 25) at the first location in the assembly and with a SI of 10. The fitness corresponding to the best sequence obtained is 1.4500 [assuming the weights $w_1 = w_2 = 0.25$, $w_3 = 0.1$, $w_4 = 0.4$ in the fitness function given in equation (10)]. For 5,000 iterations, the set of FPA parameters that gave the optimum performance of the IFPA in terms of its consistency in finding the global best solution is as follows: population size of 35, step size of 26 and switch probability of 0.1. For the above set of parameters, the consistency of finding the best sequence was found to be 0 per cent (based on 20 independent runs of the IFPA simulations). Figure 10 shows the convergence graph of the IFPA for 5,000 iterations.

5.2.2 Computational results of application of the hybrid GA-FPA

The best sequence obtained by the hybrid GA-FPA is [25, 13, 24, 23, 26, 1, 8, 5, 4, 10, 7, 6, 9, 14, 19, 12, 15, 22, 11, 18, 16, 21, 3, 17, 20, 2] which has the least number of direction changes (i.e. 5) and tool changes (i.e. 15), no FVs in this assembly sequence, the base component provided by the user (in this case, Component 25) at the first location in the assembly, and most stable configuration in accordance with SM having a SI of 11. The fitness value of this sequence is 1.4880, which is calculated using the equation (10). The maximum fitness obtained is 1.4880 [assuming the weights $w_1 = w_2 = 0.25$, $w_3 = 0.1$, $w_4 = 0.4$ in the fitness function given in equation (10)]. For 5,000 iterations, the set of parameters that gave the optimum performance of the hybrid GA-FPA in terms of its consistency in finding the global best solution is as follows: population size of 35, the GA crossover probability

Table VI Information on the assembly directions and tools/grippers required for 26-component worm gear reducer assembly

Component no.	Assembly direction	Tool/Gripper no.
1	Z−	14
2	Y−	4
3	X+	3
4	Z−	5
5	Z−	6
6	Z+	8
7	Z−	7
8	Z−	6
9	Z+	8
10	Z−	7
11	Y−	13
12	Y+	12
13	Z−	2
14	Y+	1
15	Y−	1
16	X+	1
17	X−	1
18	X+	1
19	Y+	1
20	X−	1
21	X+	1
22	Y−	1
23	Z−	1
24	Z−	1
25	Z−	1
26	Z−	1

fixed at 0.9, the GA mutation probability fixed at 0.1, the FPA step size of 26 and the FPA switch probability of 0.1. For the above set of parameters, the consistency of finding the global best sequence was found to be 10 per cent (based on 20 independent runs of the GA-FPA simulations). Figure 11 shows the convergence graph of the hybrid GA-FPA for 5,000 iterations.

5.2.3 Comparison between the results of GA, IHS, MA, original FPA, IFPA and the hybrid GA-FPA approaches

To compare the results of the improved discrete FPA and the hybrid GA-FPA with those of GA, IHS, MA and original FPA, we have performed 20 independent simulation runs of each of the above six algorithms for 5,000 iterations. For the GA simulation runs, the set of parameters that gave the optimum performance in terms of best fitness that it could achieve is crossover probability of 0.90 and mutation probability of 0.05. For the IHS simulation runs, the set of parameters that gave the optimum performance in terms of best fitness that it could achieve is HMCR 0.9 and pitch adjustment rate (PAR) 0.1. For the MA simulation runs, the set of parameters that gave the optimum performance in terms of best fitness that it could achieve is crossover probability of 0.80 and mutation probability of 0.02. For the original FPA simulation runs, the set of parameters that gave the optimum performance in terms of best fitness that it could achieve is step size of 26 and switch probability of 0.2. For simulation runs of the improved discrete FPA, we have used the optimum set of parameters given in section "Computational results of application of improved discrete FPA", i.e. FPA step size of 15 and FPA switch probability of 0.1. For simulation runs of the hybrid GA-FPA, we have used the optimum set of parameters given in section "Computational results of application of the hybrid GA-FPA", i.e. the GA crossover probability fixed at 0.9, the GA mutation probability fixed at 0.1, the FPA step size of 26 and the FPA switch probability of 0.1. To maintain fairness in comparison of the results, the population size and maximum number of iterations are kept fixed at 35 and 5,000, respectively, for the simulation runs of all the algorithms. Moreover, the same fitness function given in equation (10) is used in all the algorithms.

The comparison results for the 26-component assembly, presented in Table VII, show that the hybrid GA-FPA could find the optimum solution two times out of 20 runs, which gives a 10 per cent probability of finding the best assembly sequence in 5,000 iterations, while the probability of finding the optimum solution for all the other algorithms including MA is

Figure 10 Convergence graph of IFPA for the 26-component assembly

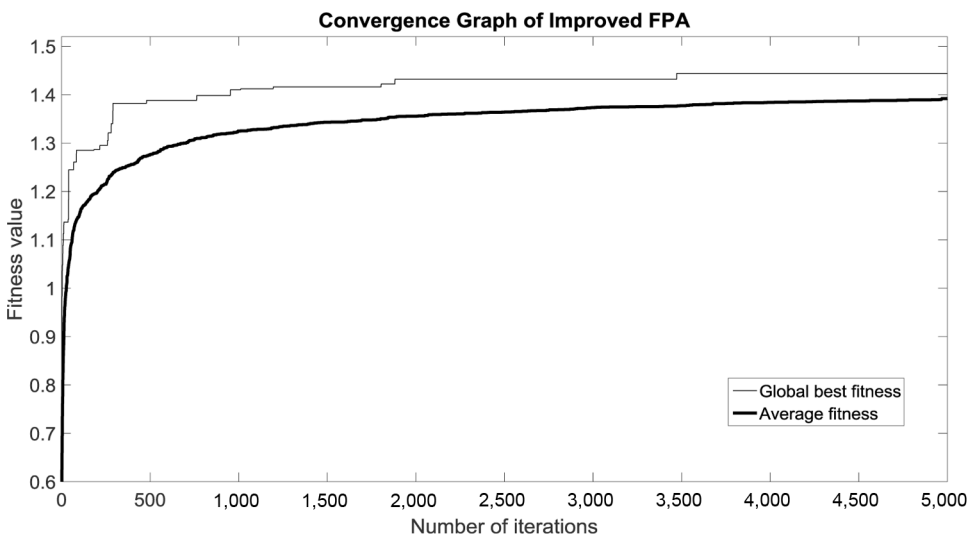


Figure 11 Convergence graph of hybrid GA-FPA for the 26-component assembly

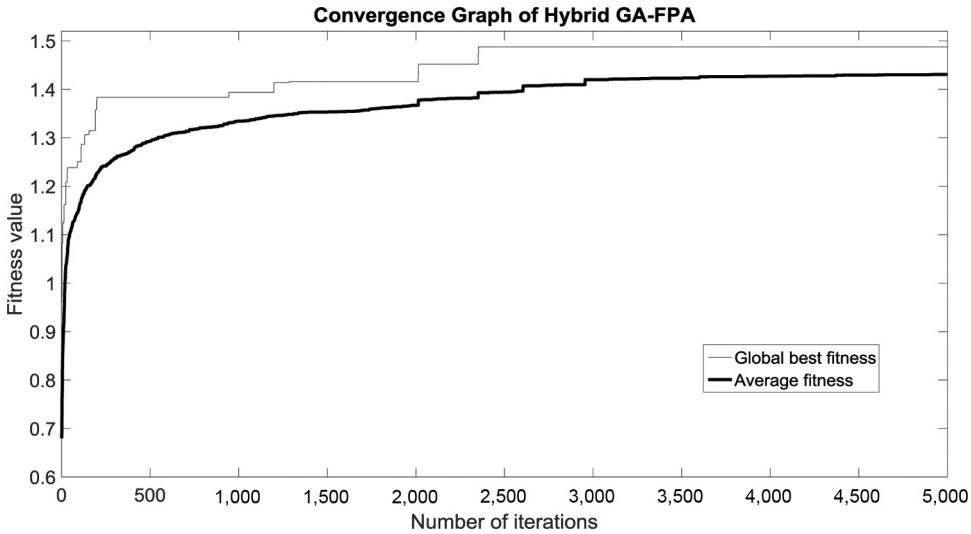


Table VII Results of comparison between GA, IHS, MA, original FPA, improved FPA and hybrid GA-FPA for 26-component worm gear reducer assembly

Method	GA	IHS	MA	Original FPA (Mishra and Deb, 2016)	IFPA	Hybrid GA-FPA
Best fitness value (no. of iterations to reach the global best fitness)	1.4400	1.3780	1.4600	1.4460	1.4500	*1.4880* (2352*)*
Mean of the average fitness values	1.4141	1.2152	1.4255	1.3803	1.3904	1.4152
Mean of the best fitness values	1.4141	1.2730	1.4255	1.4325	1.4349	*1.4503**
Number of runs out of 20 independent runs in which convergence to best assembly sequence fitness was achieved	0	0	0	0	0	*2**
Probability of obtaining the best assembly sequence (%)	0	0	0	0	0	*10**

Notes: *Best results obtained; italic values indicate that they are the optimum values given by the proposed GA-FP algorithm

equal to zero. This once again clearly demonstrates the superiority in performance of the proposed GA-FPA in assembly sequence optimization over the other five algorithms used in this comparison. Table VIII shows a comparison of the computational time of GA-FPA and IFPA proposed by us with that of the original FPA (Mishra and Deb, 2016) for 26-components assembly. These simulations were run on a PC having Intel Core i5-3470S CPU 2.90GHz, 8GB RAM. The table clearly shows that the proposed hybrid GA-FPA and IFPA take much less time than the original FPA due to different representation scheme and operators used.

Table IX shows the best sequences for the 26-component example that have been obtained by all the six algorithms. It further gives the corresponding fitness values as well as number of direction changes, tool changes, SI, location of base component in the sequence, and number of FVs. On comparison of the above results, it is found that the best assembly sequence provided by proposed GA-FPA requires the least number of direction changes (i.e. 5), and also least number of tool changes (i.e. 15). Further, the stability of the assembly is found to be the highest (as indicated by SI 11) in case of the best sequence provided by GA-FPA.

Furthermore, Figure 12 shows the convergence plots of global best fitness of the six algorithms, while Figure 13 shows the convergence plots of average fitness. The comparison of the convergence plots of global best fitness for all the algorithms shows that the hybrid GA-FPA clearly outperforms the other five algorithms in terms of convergence speed, as it is able to reach the global best fitness value in least number of iterations (i.e. only 2352 iterations), as compared to GA, IHS, original FPA, IFPA and MA. All the other algorithms other than the developed algorithms could not reach the global best out of 5,000 iterations. This again shows that the performance of the developed GA-FPA is found to be certainly better than that of other algorithms. In short, the developed GA-FPA achieves global best solution in least number of iterations with faster

Table VIII Computational time of 20 independent runs of algorithms for 26-components assembly

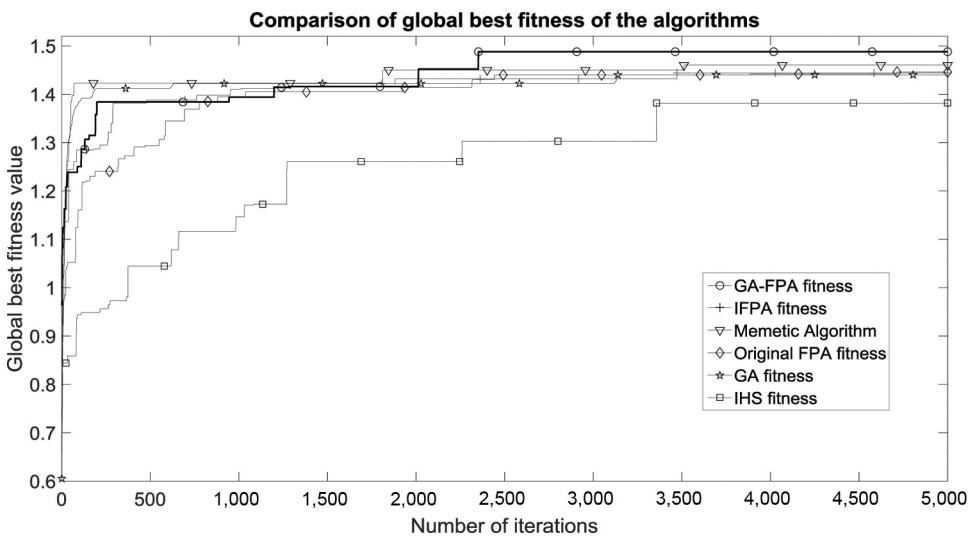
Algorithm	Improved FPA	Hybrid GA-FPA	Original FPA (Mishra and Deb, 2016)
Time (s)	178.046	216.225	840.30

Table IX Comparison of best sequences obtained by GA, IHS, MA, original FPA, IFPA and hybrid GA-FPA for 26-component worm gear reducer assembly

Algorithm	Best assembly sequence	Fitness value (Success rate of obtaining the best assembly sequence out of 20 runs)	No. of direction changes	No. of tool changes	SI	Base component at the first location	FV
GA-FPA	[25 13 24 23 26 1 8 5 4 10 7 6 9 14 19 12 15 22 11 18 16 21 3 17 20 2];	*1.4880* (10%*)*	*5**	*15**	*11**	*1**	*0**
MA	[25 13 24 23 1 5 8 26 4 7 10 9 6 14 15 22 18 16 21 3 19 12 17 20 2 11];	1.4600 (0%)	7	15	10	1	0
IFPA	[25 6 9 13 24 8 5 4 23 1 10 7 26 15 14 19 12 18 1 6 21 3 17 20 2 22 11];	1.4500 (0%)	7	16	10	1	0
Original FPA (Mishra and Deb, 2016)	[25 13 24 23 26 1 5 14 19 18 16 21 3 9 6 17 20 2 22 11 4 8 5 1 10 7 12];	1.4460 (0%)	8	13	7	1	0
IHS	[25 13 24 14 15 18 16 19 17 20 2 22 23 8 5 10 7 6 9 4 11 21 3 12 26 1];	1.3780 (0%)	13	14	6	1	0
GA	[25 13 24 23 1 4 5 8 26 14 15 22 11 6 9 18 17 16 21 3 19 12 20 2 10 7];	1.4400 (0%)	9	15	10	1	0

Notes: *Best results obtained; italic values indicate that they are the optimum values given by the proposed GA-FP algorithm

Figure 12 Comparison of global best fitness of GA, IHS, MA, original FPA IFPA and hybrid GA-FPA for 26-component assembly.

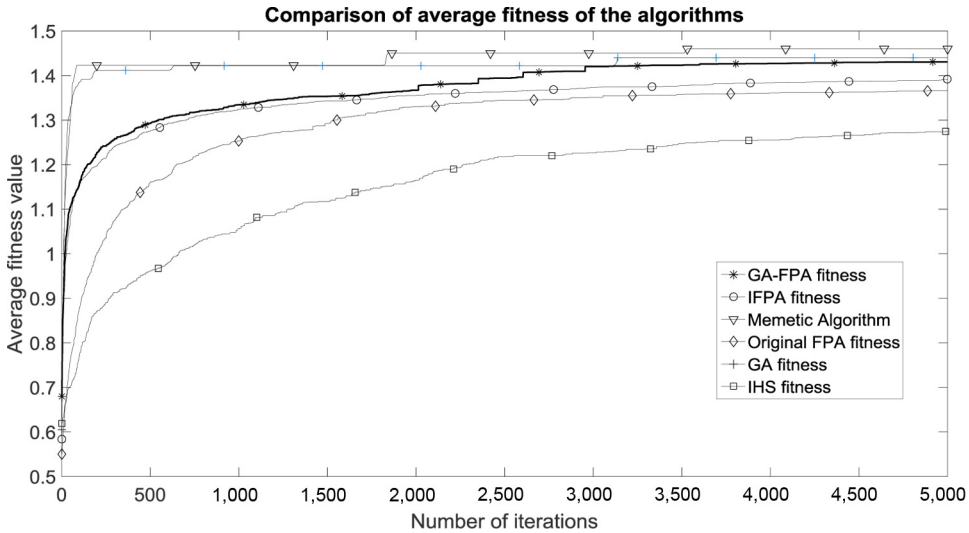


convergence (especially when compared with IHS, original FPA, IFPA) and with more robustness.

6. Conclusions

Because the assembly cost is estimated to be around 10-30 per cent of the total manufacturing cost, proper assembly process planning is of utmost importance. A product may be possible to assemble in many alternative ways following different sequences. The optimal assembly sequence takes the least time and thus results in the minimum cost. However, the assembly sequence optimization is a difficult optimization problem in process planning, as it has to simultaneously satisfy various types of feasibility constraints such as stability, assembly precedence and accessibility, as well as various types of optimization criteria such as minimizing number of changes in orientations, tools and grippers used in assembly, handling and insertion times, etc. Although a number of different soft computing-based evolutionary algorithms had been proposed by previous researchers to solve this problem, there are still challenges like achieving global optimum in least number of iterations with fast convergence speed, maintaining robustness/

Figure 13 Comparison of average fitness of GA, IHS, MA, original FPA IFPA and hybrid GA-FPA for the 26-component assembly



consistency in finding the global optimal solution, etc. which are crucial for solving problems using meta-heuristic approaches. Keeping the above in mind, in the present paper, we have proposed an improved discrete FPA-based approach and to further improve the performance of this FPA, we have proposed a hybrid GA-FPA–based approach. Two case studies involving a 15-component assembly and another more complex 26-component assembly are provided to demonstrate the application of the proposed approaches as well as the results of comparison with well-known soft computing algorithms. The following are some of the important contributions of this paper:

- For solving assembly sequence optimization by using the IFPA and the hybrid GA-FPA, we have proposed an assembly sequence representation scheme using an array of unique numbers, each number representing a unique component in the sequence. The initial population of flowers (i.e. assembly sequences) in the FPA is generated randomly without considering their feasibility, which gives the advantage of maintaining diversity in flowers during the course of the algorithm and helping to avoid local optima. To expand the field of initial solutions (i.e. assembly sequences) in the search space, the method of OBL that was used earlier by Li *et al.* (2016) is used. It is observed that the performance of FPA increases significantly if the initial population is generated using the OBL method.
- In the improved discrete FPA proposed by us, modifications have been proposed in the rules for local and global pollination of the FPA proposed by Yang (2012) that was meant for continuous optimization. It was necessary to make the proposed FPA suited for solving the given discrete optimization problem of assembly sequencing.
- The parameter settings of the hybrid GA-FPA proposed by us are discussed. To check the influence of various parameters on the performance of the hybrid GA-FPA, the parameters have been varied and each simulation run is repeated 100 times. A method is discussed for determining the optimum set of hybrid GA-FPA parameters based on the average value of global best solutions (or sequences) obtained over 100 runs, consistency in finding the global best, and the mean square error.
- To compare the results of the improved discrete FPA and the hybrid GA-FPA with those of GA, MA, IHS and the original FPA (Mishra and Deb, 2016), we have performed 20 independent simulation runs of each of the above algorithms for both the case studies involving the 15-component assembly and the 26-component assembly. On comparing the results, it is found that for both the examples, the proposed hybrid GA-FPA clearly performs better than the algorithms GA, MA, IHS, the original FPA and the improved discrete FPA in terms of its ability to reach the global best fitness value in least number of iterations.
- Further it can be found that in case of the hybrid GA-FPA for the 15-component assembly, in 13 times out of 20 runs, it could find the optimal solution, which gives a 65% probability of finding the best solution or assembly sequence in 500 iterations. In contrast, the probabilities of finding the best solution by the improved discrete FPA proposed by us and MA are 50% and 45%, respectively, while the probability of finding the best solution by GA, IHS and the original FPA is each equal to zero. For the more complex 26-component assembly, out of the six algorithms, only the hybrid GA-FPA could find the optimum solution in 5,000 iterations with more consistency, while none of the other five algorithms could find the optimum.
- In the light of the above comparison results, it can be concluded that for problems involving higher complexity in terms of number of components, the hybrid GA-FPA proposed in this paper clearly outperforms the other five algorithms in terms of achieving global best solution in least number of iterations with fast convergence speed and consistency (or higher probability) in finding the best solution. It should also be noted that the proposed hybrid

GA-FPA and IFPA algorithms take lesser computational time than the original FPA.

It is anticipated that using the proposed approach, assembly sequence planning can be accomplished efficiently and consistently with reduced lead time for process planning, making it cost-effective for industrial applications.

References

Bahubalendruni, M.V.A.R., Deepak, B.B.V.L. and Biswal, B.B. (2016), "An advanced immune based strategy to obtain an optimal feasible assembly sequence", *Assembly Automation*, Vol. 36 No. 2, pp. 127-137.

Barnes, C.J., Jared, G.E.M. and Swift, K.G. (2004), "Decision support for sequence generation in an assembly oriented design environment", *Robotics and Computer-Integrated Manufacturing*, Vol. 20 No. 4, pp. 289-300.

Bonneville, F., Perrard, C. and Henrioud, J.M. (1995), "A genetic algorithm to generate and evaluate assembly plans", *Proceedings of INRIA/IEEE symposium on emerging technologies and factory automation, Paris*, Vol. 2, pp. 231-239.

Cao, P.B. and Xiao, R.B. (2007), "Assembly planning using a novel immune approach", *The International Journal of Advanced Manufacturing Technology*, Vol. 31 Nos 7/8, pp. 770-782.

Chang, C.C., Tseng, H.E. and Meng, L.P. (2009), "Artificial immune systems for assembly sequence planning exploration", *Engineering Applications of Artificial Intelligence*, Vol. 22 No. 8, pp. 1218-1232.

Chen, S.F. and Liu, Y.J. (2001), "An adaptive genetic assembly-sequence planner", *International Journal of Computer Integrated Manufacturing*, Vol. 14 No. 5, pp. 489-500.

Choi, Y.K., Lee, D.M. and Cho, Y.B. (2009), "An approach to multi criteria assembly sequence planning using genetic algorithms", *The International Journal of Advanced Manufacturing Technology*, Vol. 42 Nos 1/2, pp. 180-188.

Dubey, H.M., Pandit, M. and Panigrahi, B.K. (2015), "A biologically inspired modified flower pollination algorithm for solving economic dispatch problems in modern power systems", *Cognitive Computation*, doi: 10.1007/s12559-015-9324-1.

Gao, L., Qian, W.R., Li, X.Y. and Wang, J.F. (2010), "Application of memetic algorithm in assembly sequence planning", *The International Journal of Advanced Manufacturing Technology*, Vol. 49 Nos 9/12, pp. 1175-1184.

Gao, L., Zhang, C., Li, X. and Wang, L. (2014), "Discrete electromagnetism-like mechanism algorithm for assembly sequences planning", *International Journal of Production Research*, Vol. 52 No. 12, pp. 3485-3503, available at: http://dx.doi.org/10.1080/00207543.2013.867087.

Ghandi, S. and Masehian, E. (2015), "A breakout local search (BLS) method for solving the assembly sequence planning problem", *Engineering Applications of Artificial Intelligence*, Vol. 39, pp. 245-266.

Gruhier, E., Demoly, F., Dutartre, O., Abboudi, S. and Gomes, S. (2015), "A formal ontology-based spatiotemporal mereotopology for integrated product design and assembly sequence planning", *Advanced Engineering Informatics*, Vol. 29 No. 3, pp. 495-512.

Hong, D.S. and Cho, H.S. (1995), "A neural-network-based computational scheme for generating optimized robotic assembly sequences", *Engineering Applications of Artificial Intelligence*, Vol. 8 No. 2, pp. 129-145.

Kashkoush, M. and ElMaraghy, H. (2013), "Consensus tree method for generating master assembly sequence", *Production Engineering*, Vol. 8 Nos 1/2, pp. 233-242.

Laperriere, L. and ElMaraghy, H.A. (1994), "Assembly sequences planning for simultaneous engineering applications", *The International Journal of Advanced Manufacturing Technology*, Vol. 9 No. 4, pp. 231-244.

Li, J.R., Khoo, L.P. and Tor, S.B. (2003), "A tabu enhanced genetic algorithm approach for assembly process planning", *Journal of Intelligent Manufacturing*, Vol. 14 No. 2, pp. 197-208.

Li, M., Wu, B., Yi, P., Jin, C., Hu, Y. and Shi, T. (2013a), "An improved discrete particle swarm optimization algorithm for high-speed trains assembly sequence planning", *Assembly Automation*, Vol. 33 No. 4, pp. 360-373.

Li, M., Wu, B., Hu, Y., Jin, C. and Shi, T. (2013b), "A hybrid assembly sequence planning approach based on discrete particle swarm optimization and evolutionary direction operation", *The International Journal of Advanced Manufacturing Technology*, Vol. 68 Nos 1/4, pp. 617-630.

Li, X., Qin, K., Zeng, B., Gao, L. and Su, J. (2016), "Assembly sequence planning based on an improved harmony search algorithm", *The International Journal of Advanced Manufacturing Technology*, Vol. 84 Nos 9/12, pp. 2367-2380.

Lu, C. and Yang, Z. (2016), "Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach", *The International Journal of Advanced Manufacturing Technology*, Vol. 83 Nos 1/4, pp. 243-256.

Lv, H.G. and Lu, C. (2010a), "An assembly sequence planning approach with a discrete particle swarm optimization algorithm", *The International Journal of Advanced Manufacturing Technology*, Vol. 50 Nos 5/8, pp. 761-770.

Lv, H.G. and Lu, C. (2010b), "A hybrid DPSO-SA approach to assembly sequence planning", *Proceedings of the IEEE International Conference on Mechatronics and Automation, Xi'an*, pp. 1998-2003.

Mishra, A. and Deb, S. (2016), "Assembly sequence optimization using a flower pollination algorithm-based approach", *Journal of Intelligent Manufacturing*, doi: 10.1007/s10845-016-1261-7.

Sabarinath, P., Karthick, R., Thansekhar, M.R. and Saravanan, R. (2015), "Energy conservation by design optimization of flywheel using flower pollination algorithm", *Proceedings of national conference on recent trends and developments in sustainable green technologies, Chennai*, pp. 166-171.

Sharma, S., Biswal, B.B., Dash, P. and Choudhury, B.B. (2008), "Generation of optimized robotic assembly sequence using ant colony optimization", *IEEE Automation Science and Engineering (CASE) Conference, Arlington*, pp. 23-26.

Shuang, B., Chen, J.P. and Li, Z.B. (2008), "Microrobot based micro-assembly sequence planning with hybrid ant colony algorithm", *The International Journal of Advanced*

Manufacturing Technology, Vol. 38 Nos 11/12, pp. 1227-1235.

Ting, T.O., Yang, X.S., Cheng, S. and Huang, K. (2015), "Hybrid metaheuristic algorithms: past, present, and future", in Yang, XS. (Ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation. Studies in Computational Intelligence*, Vol. 585, Springer, Cham.

Tiwari, M.K., Prakash, A. and Mileham, A.R. (2005), "Determination of an optimal assembly sequence using the psychoclonal algorithm", *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 219, pp. 137-149.

Tizhoosh, H.R. (2005), "Opposition-based learning: a new scheme for machine intelligence", *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, *Vienna*, pp. 695-701.

Tseng, H.E., Wang, W.P. and Shih, H.Y. (2007), "Using memetic algorithms with guided local search to solve assembly sequence planning", *Expert Systems with Applications*, Vol. 33 No. 2, pp. 451-467.

Tseng, H.E., Chen, M.H., Chang, C.C. and Wang, W.P. (2008), "Hybrid evolutionary multi-objective algorithms for integrating assembly sequence planning and assembly line balancing", *International Journal of Production Research*, Vol. 46 No. 21, pp. 5951-5977.

Tseng, Y.J., Chen, J.Y. and Huang, F.Y. (2009), "An integrated assembly sequence planning and plant assignment model for products assembled in a multi-plant system", *IEEE International Conference on Industrial Engineering and Engineering Management*, *Hong Kong*, pp. 144-148.

Tseng, Y.J., Chen, J.Y. and Huang, F.Y. (2010), "A particle swarm optimisation algorithm for multi-plant assembly sequence planning with integrated assembly sequence planning and plant assignment", *International Journal of Production Research*, Vol. 48 No. 10, pp. 2765-2791.

Tseng, Y.J., Yu, F.Y. and Huang, F.Y. (2011), "A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method", *The International Journal of Advanced Manufacturing Technology*, Vol. 57 Nos 9/12, pp. 1183-1197.

Wang, H., Rong, Y. and Xiang, D. (2014), "Mechanical assembly planning using ant colony optimization", *Computer-Aided Design*, Vol. 47, pp. 59-71.

Wang, H.S., Che, Z.H. and Chiang, C.J. (2012), "A hybrid genetic algorithm for multi-objective product plan selection problem with ASP and ALB", *Expert Systems with Applications*, Vol. 39 No. 5, pp. 5440-5450.

Wang, J.F., Liu, J.H. and Zhong, Y.F. (2005), "A novel ant colony algorithm for assembly sequence planning", *The International Journal of Advanced Manufacturing Technology*, Vol. 25 Nos 11/12, pp. 1137-1143.

Wang, L., Hou, Y., Li, X. and Sun, S. (2013), "An enhanced harmony search algorithm for assembly sequence planning", *International Journal of Modelling, Identification and Control*, Vol. 18 No. 1, pp. 18-25.

Wang, Y. and Liu, J.H. (2010), "Chaotic particle swarm optimization for assembly sequence planning", *Robotics and Computer Integrated Manufacturing*, Vol. 26 No. 2, pp. 212-222.

Xie, L., Fu, Y. and Ma, Y. (2007), "Tool-operation-space oriented strategy for generating assembly sequence plans", *Chinese Journal of Mechanical Engineering*, Vol. 41 No. 10, pp. 215-220.

Xu, L.D., Wang, C., Bi, Z. and Yu, J. (2012), "AutoAssem: an automated assembly planning system for complex products", *IEEE Transactions on Industrial Informatics*, Vol. 8 No. 3, pp. 669-678.

Yang, X.S. (Ed.) (2012), "Flower pollination algorithms", *Nature-Inspired Optimization Algorithms*, Elsevier, London, pp. 155-173.

Yu, J. and Wang, C. (2013), "A max–min ant colony system for assembly sequence planning", *The International Journal of Advanced Manufacturing Technology*, Vol. 67 Nos 9/12, pp. 2819-2835.

Yu, J., Wang, C., Yu, H. and Zhang, W. (2009), "Generation of optimized assembly sequences based on priority rules screening", *Applied Mechanics and Materials*, Vols 16/19, pp. 130-134.

Zhang, H., Liu, H. and Li, L. (2014), "Research on a kind of assembly sequence planning based on immune algorithm and particle swarm optimization algorithm", *The International Journal of Advanced Manufacturing Technology*, Vol. 71 Nos 5/8, pp. 795-808.

Zhang, Z., Yuan, B. and Zhang, Z. (2016), "A new discrete double-population firefly algorithm for assembly sequence planning", *Proceedings of IMechE Part B: Journal of Engineering Manufacture*, Vol. 230 No. 12, pp. 2229-2238.

Zhou, W., Yan, J., Li, Y., Xia, C. and Zheng, J. (2013), "Imperialist competitive algorithm for assembly sequence planning", *The International Journal of Advanced Manufacturing Technology*, Vol. 67 Nos 9/12, pp. 2207, available at: https://doi.org/10.1007/s00170-012-4641-y

Zhou, W., Zheng, J.R., Yan, J.J. and Wang, J.F. (2011), "A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm", *The International Journal of Advanced Manufacturing Technology*, Vol. 52 Nos 5/8, pp. 715-724.

Corresponding author

Sankha Deb can be contacted at: sankha.deb@mech.iitkgp.ernet.in