J Intell Manuf
DOI 10.1007/s10845-016-1261-7

Assembly sequence optimization using a flower pollination algorithm-based approach

Atul Mishra[1] · **Sankha Deb**[1]

Received: 18 September 2015 / Accepted: 7 September 2016

Abstract One of the important decisions in assembly process planning is determination of assembly sequence. Choice of the optimum sequence is made difficult due to various reasons. There are various precedence constraints and optimization criteria. Moreover, a product may be possible to assemble in many alternative ways following different sequences, thus making assembly sequence optimization a multi-modal optimization problem with multiple optimum solutions. It is necessary to generate as many unique optimum solutions as possible in order to allow the process planner to take a decision. Moreover, with increase in part count, the number of feasible sequences rises staggeringly, thereby making assembly sequence optimization laborious and time consuming. Most conventional mathematical algorithms are known to perform poorly when used to obtain multiple optimum solutions. On the other hand, soft computing based evolutionary optimization algorithms are good candidates for multi-modal optimization. Another challenge is to develop an algorithm that can automatically maintain diversity in the optimum solutions found over the generations (i.e. optimum solutions having the same fitness but unique). Keeping the above in mind, in the present paper, an intelligent assembly sequence optimization methodology based on application of flower pollination algorithm (FPA) has been developed to automatically generate multiple unique optimal assembly sequences, subject to various precedence constraints, based on minimisation of number of orientation changes and tool changes. Since in the present paper, FPA has been applied for the first time to a discrete optimization problem like assembly sequence optimization, the main challenge before us in applying FPA was the continuous nature of the original FPA. Therefore, modifications have been made by us in the rules for local and global pollination of FPA to make it suited for solving the given discrete optimization problem. In order to evaluate the performance of FPA, the results have been compared with two other well-known soft computing techniques namely, Genetic Algorithm (GA) and Ant Colony Optimization (ACO) and also with a recently published soft computing based algorithm, Improved Harmony Search (IHS). It was found that the novelty of the proposed FPA lies in its capability to find multiple unique optimum solutions in one single simulation run and capability to automatically maintain diversity in the optimum solutions found over the generations. On the other hand, in case of GA, ACO and IHS, it is not possible to maintain the diversity in multiple optimum solutions as the complete population finally converges to a few unique optimum solutions. Therefore, it can be concluded that FPA performs better in solving the given multi-modal optimization problem of assembly sequence optimization.

Keywords Assembly sequence optimization · Computer-aided process planning · Evolutionary optimization algorithms · Flower pollination algorithm (FPA)

Introduction

In assembly process planning, one of the important decision making tasks is that of determination of the sequence in which the different parts are to be assembled. It affects not only the manufacturing time, cost and the quality of the product but may directly or indirectly also influence the

✉ Sankha Deb
sankha.deb@mech.iitkgp.ernet.in

[1] FMS and CIM Lab, Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India

Published online: 14 September 2016

level of difficulty in performing the assembly, the need for re-fixturing, need for frequent tool changes between consecutive assembly operations, the likelihood of parts damage during the assembly, the ability to do in-process testing, the need for rework and so on. A product may be possible to assemble in a number of alternative ways by following different assembly sequences. However, choice of the optimum assembly sequence is made difficult due to the following reasons. First of all, there are various constraints that can drive the choice of a feasible assembly sequence, depending on the design requirements, part and tool accessibility, assembly line and work cell layout, requirements of special operations, etc. Secondly, with increase in the part count (i.e. the number of parts in the assembly), the number of feasible assembly sequences possible also rises staggeringly, thereby making the task of determining the optimal assembly plan laborious and time consuming, if it is performed manually. Thirdly, the assembly sequence optimization is a multi-modal optimization problem, i.e. it may possess multiple optimum solutions (having the same fitness function value). The challenge, therefore, is to obtain all or as many multiple optimum solutions as possible. Most conventional mathematical algorithms are known to perform poorly when used to obtain multiple optimum solutions. On the other hand, soft computing based evolutionary optimization algorithms are good candidates for multi-modal optimization. These algorithms are inspired by naturally occurring phenomena, and by starting the search process with a population of solutions, they can find multiple optimum solutions and globally optimum solutions through an evolutionary strategy. Most of the previous research work had been focused on determining the best or optimum solution (which, in this case, is the optimum assembly sequence), while another crucial problem that had received little attention is that of generation of all or as many multiple optimum solutions as possible.

Keeping the above in mind, in the present paper, an intelligent assembly sequence optimization methodology based on application of Flower pollination algorithm (FPA) has been developed to automatically generate multiple optimal assembly sequences, subject to various assembly precedence constraints, based on minimisation of number of orientation changes of the assembly and minimisation of number of tool changes. In order to evaluate the performance of FPA, the results have been compared with those obtained by using two other well-known soft computing techniques namely, Genetic Algorithm (GA) and Ant Colony Optimization (ACO) and also with a recently published soft computing based algorithm, Improved Harmony Search (IHS).

The rest of this paper is organized as follows. Section "Literature review" presents an overview of the previous research work on assembly sequence planning optimization and recent applications of FPA. A section "Proposed FPA based approach for assembly sequence optimization" describes the FPA based methodology developed for generation of multiple optimum assembly sequences. Section "Illustrative examples: Results and Discussions" presents two illustrative examples showing the application of the FPA based approach developed in this paper and also discusses the results of comparison of FPA with two other well-known soft computing techniques namely, GA and ACO and also with a recently published soft computing based algorithm, IHS. Finally, section "Conclusions" presents the important conclusions and scope for future work.

Literature review

General problem description

An assembly sequence is typically represented by an ordered sequence of components that must be assembled together. For example, if we have a n-component assembly, and if the components are to be assembled in the sequential order of say, component 2, followed by component 3, then component 1 and so on, then the assembly sequence is represented by [2, 3, 1,….]. Furthermore, to generate a feasible assembly sequence, it is important to know the assembly precedence constraints, if any. A precedence constraint of a component is represented by a set of components that must be assembled before that component. For example, if we have an assembly comprising of three components i, j and k, and before successfully assembling component k to component i, if it is necessary to assemble component j to component i, then component k is said to have a precedence with component j. The precedence constraints between components are often represented by $n \times n$ matrix, where n is the number of components in the assembly, and each entry x_{ij} 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. This precedence constraint matrix needs to be provided in order to check for the feasibility of the assembly sequences. The assembly sequence must be generated in accordance with the above precedence constraints. It may be possible to assemble a product in a number of alternative ways by following different sequences. An optimum assembly sequence can help to reduce the cost and time of manufacturing.

Assembly sequence planning and optimization problem

For solving the assembly sequence planning and optimization problem, different evolutionary soft computing based optimization algorithms have been used (Jimenez 2013) like Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Memetic Algorithm (MA), and others.

GA based approaches for assembly process plan optimization had been used by Bonneville et al. (1995), Chen and Liu (2001), Marian et al. (2006), Choi et al. (2009), Kashkoush and ElMaraghy (2013) among others. Bonneville et al. (1995) used GA which starts with valid assembly plans provided by experts. Crossover and mutation were applied on the population and resulting offsprings were evaluated and selected. Liaisons and geometric constraints were used to verify the feasibility of operations. The authors reported that the proposed GA could generate all valid and good assembly plans but its performance was slow, and does not necessarily guarantee optimum plans. Chen and Liu (2001) proposed an adaptive genetic algorithm (with genetic-operator probabilities varying according to certain rules) for efficiently finding global-optimal or near-optimal assembly sequences. With the use of Genetic-Operator Probability Settings (GOPS), population is diversified over the time. Marian et al. (2006) also used GA in the area of assembly process planning where the solutions were generated using guided search considering both intrinsic as well as extrinsic precedence relations. Stochastic sampling has been used for selection and crossover (based on guided search with supplementary conditions) and mutation (modified guided search operator). Choi et al. (2009) presented the optimization of assembly sequences using GA in which assembly time (including set up time and actual assembly time) and number of orientation changes were minimized. Initial population had been generated randomly followed by a validity check performed using the precedence matrix (PM). The crossover (PPX) and mutation (swap mutation), elitist approach and roulette wheel had been used in this work. However, in this work it was assumed that the actual assembly time is constant regardless of the assembly sequence. But in practice, there might be other constraints that may be encountered during assembly (Boothroyd 2005) leading to increase in time and cost of the assembly, e.g. resistance during part insertion due to small clearances available for manipulating the part, or due to instability of the part after placement, thereby requiring realignment, holding down by special fixturing, etc. Zhou et al. (2011) combined the Bacterial Chemotaxis with GA to apply in optimisation of assembly sequences. Assembly sequences were encoded as chromosomes, where gene in the chromosome is treated as a bacterium. Fitness function comprised of length of longest feasible sub-sequence, number of orientation changes, number of gripper changes. Kashkoush and ElMaraghy (2013) developed a variant based CAPP approach using GA for generating assembly sequence for sequential, non-linear product assemblies. Assembly representation has been done using partial assembly tree, and tree information (sequence of leaves and topology) had been encoded into matrices. The proposed method had been demonstrated using a real case study involving three different variants of a back-flushing control valve.

Ant Colony Optimization (ACO) based approaches for assembly process plan optimization had been proposed by Wang et al. (2005), Yu and Wang (2013) and Wang et al. (2014) among others. Wang et al. (2005) proposed an ACO based approach to assembly sequence optimization where an approach "assembly by disassembly" had been used. Geometric feasibility was guaranteed with the use of Interference matrix, and fitness function was comprised of number of disassembly direction changes. However, the approach proposed by the authors considered only optimization of number of orientation changes for the assembly. But there may be other criteria like minimization of changes in tools/gripper used, stability issues, etc., which were not considered. Yu and Wang (2013) presented an improved ACO (MMACS) based approach for assembly sequence planning that combines the advantages of ant colony system (ACS) and max-min ant system (MMAS), where a multi-objective heuristic function had been used comprising of reorientation, parallelism, continuity, stability, and auxiliary strokes (i.e. frequent changes of operation station). The advantages of pseudo-random proportional rule and the local updating rule of ACS had been combined with the global updating rule with mixed strategies and max–min pheromone limits of MMAS. Wang et al. (2014) proposed an ACO algorithm for mechanical assembly, where the proposed method focused on the generation of solution space for finding the optimum solution. DFIG (Disassembly Feasibility Information Graph) had been generated and extended to collect the relevant assembly data and information (including constraint relationships) of assembly and disassembly planning.

PSO based approach for assembly process plan optimization was used by Wang and Liu (2010), Lv and Lu (2010), Li et al. (2013) and Zhang et al. (2014) among others. Wang and Liu (2010) developed the PSO based assembly sequence optimisation technique to generate the optimal or near-optimal assembly sequences. Assembly cost is subjected to geometrical constraints and five assembly process constraints namely, local assembly precedence, number of the unstable parts, assembly direction changes, assembly tools changes, connector changes. Lv and Lu (2010) presented the application of discrete PSO in assembly sequence optimisation. They considered total number of tool changes, orientation changes, and operation type changes and interference times in the product assembly. Special operators namely subtraction operator, addition operator, and multiplication operator are used to update the position and velocity of particles. Li et al. (2013) presented a hybrid algorithm of improved discrete particle swarm optimization (IDPSO) with a modified evolutionary direction operator (MEDO) to solve the assembly sequence planning problem, where fitness function includes changes in number of orientations, tools and operation types. The problem of premature and fast convergence of DPSO algorithm had been removed in IDPSO. The inputs in

the algorithm were assembly parts, their interference matrix, and the weight coefficients. Zhang et al. (2014) presented Immune and Particle Swarm Optimization (IPSO) algorithm to solve the assembly sequence of a certain type of product. Geometric feasibility and coherence had been designed as constraint conditions, and their judgment conditions are presented, and these two constrains and processability had been combined with each other as the objective function. It had been tested by two assembly examples.

Other algorithms such as Psychoclonal Algorithm, Memetic algorithm and recently an Improved Harmony Search algorithm had also been used for assembly sequence optimization. Tiwari et al. (2005) applied the extension of Artificial Immune Systems (AIS) approach, due to its virtue of learning and memory acquisition, in the field of assembly configuration planning. The proposed algorithm applied the concept of Maslow's need hierarchy theory and the theory of clonal selection and was given the name of Psychoclonal Algorithm. The results of Psychoclonal Algorithm were compared with that of Genetic Algorithm and Immune Algorithm. Tseng et al. (2007) used Memetic algorithm for assembly sequence optimization where assembly parts had been categorized into the connectors having different fastener types, assembly tools, directions, and connector based precedence graph. Fitness function had been determined by the similarity of the engineering data of the connector. PMX crossover and Insert mutation and followed by binary tree search had been used to generate the feasible solutions. Later, guided crossover and guided mutation had been used to meet with the constraints of the assembly problem and the local search. Cao and Xiao (2007) explored the use of Immune optimisation Algorithm (IOA) in the problem of assembly planning to generate the optimal assembly plan. The algorithm was based on the bionic principles of AIS, where IOA introduced manifold immune operations including immune selection, clonal selection, inoculation and immune metabolism. Assembly sequences were evaluated based on total number of components, assembly direction changes, assembly tool changes, location of base component in the sequence, and feasibility degree. Gao et al. (2010) developed an assembly sequence optimisation approach based on Memetic Algorithm where a chromosome represented an assembly sequence consisting of genes containing the part number and the direction variable. They considered times of the assembly direction changes and assembly feasibility as the objective function. PMX crossover and swap mutation in addition to local search have been used to generate new offsprings. Xing and Wang (2012) presented the hybrid PSO and GA based assembly sequence optimisation for compliant assemblies based on graph theory. Liaison graph and adjacency matrix were used to describe the geometry of the compliant assemblies and assembly sequence is represented as the string of parts, whose length is equal to number of parts in the assembly. Assembly sequences were evaluated on the basis of assembly variation due to dimensional tolerance. Recently, Li et al. (2016) developed an assembly sequence planning algorithm based on Improved Harmony Search (IHS). They have proposed new aspects like an initial harmony memory (HM) established using the opposition-based learning (OBL) strategy, a way to improvise a new harmony and a local search strategy. They considered changes of assembly direction and assembly tool and stability criteria in the fitness function.

It is evident from the above literature review that many different soft computing based evolutionary optimization approaches have been developed for determining the best or optimum assembly sequence. However, it is to be kept in mind that the assembly sequence optimization is a multimodal optimization problem, i.e. it may possess multiple optimum solutions (having the same fitness function value) e.g. sequences requiring same number of orientation and tool changes. However, other constraints may be encountered in practice such as resistance during part insertion due to small clearances available for manipulating the part with the tool/gripper, or due to instability of the part after placement, thereby requiring realignment, holding down by special fixturing, etc., whose time and cost can also be significant in some cases. Moreover, multiple assembly sequences can be sometimes useful for providing alternative routing opportunities in case of dynamic scheduling of mixed-model assembly lines, in cases where some assembly operations on a product type have to wait due to unavailability of the assembly station as it is occupied in processing another product type. In that case, alternative process routes, if available, can be explored to see if the assembly precedence constraints would permit a subsequent operation to be done before. Therefore, we should aim at finding not only the best or optimum assembly sequence, but all or as many such multiple optimum solutions as possible. It is, however, felt that this is a crucial problem that had received relatively little attention in the research work reported in the literature review.

Flower pollination algorithm based optimization

Recently with the development of the flower pollination algorithm (FPA) by Yang (2012), it has been used for solving economic dispatch problems in electrical power systems (Dubey et al. 2015) and in the field of design by Sabarinath et al. (2015). Dubey et al. (2015) developed a modified flower pollination algorithm (MFPA) for solving small and medium scale economic dispatch problems of power systems by making local pollination operation more effective through a user-controlled scaling factor and through introduction of an additional intensive exploitation phase. Results of modified FPA were compared with FPA, Cuckoo Search, and Gravitational Search Algorithm (GSA). Sabarinath et al.

(2015) developed an FPA based approach for optimizing the design of a flywheel and compared the results with GA. In both the above problems, FPA was reported to give promising results.

Although FPA has been shown to give promising results in solving engineering optimization problems as evident from the above two applications, the main challenge to apply FPA to solve an assembly sequence optimization problem is the continuous nature of the FPA. This is so because the search space of basic FPA is a real space domain, while in assembly sequence optimization, the solution search space is discrete, comprising of discrete assembly sequences, from which the optimum solution must be determined. As a result, the basic FPA equations cannot be used directly to solve the given problem of discrete assembly sequence optimization. Therefore, modifications need to be made in the basic FPA in the rules for local and global pollination to make it suited for solving the problem on hand. Keeping the above in mind, the present paper proposes a modified FPA based approach for automatically generating multiple optimal assembly sequences. A comparison of the results of the proposed FPA with those of GA, ACO and IHS have been also presented, which show that the proposed approach outperforms GA, ACO and IHS in finding multiple optimum assembly sequences.

Proposed approach for assembly sequence optimization

Problem description

In this paper, the purpose is to develop a methodology of determining the optimal assembly sequence, which can help to reduce the time and cost of manufacturing. A change of the assembly orientation incurs time and usually increases the assembly cost. The number of direction/orientation changes may be computed as follows.

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

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

$$dir\_change_{i,i+1} = \begin{cases} 0, & if assembly\_dir_i = assembly\_dir_{i+1} \\ 1, & otherwise \end{cases} \tag{2}$$

Moreover, during parts handling and insertion for assembly, the human worker (in case of manual assembly) or a robot manipulator (in case of automated assembly) grasps and places the parts with the help of assembly tools such as screw driver, wrench, hammer, rivet gun, or sometimes two/three fingered grippers (in case of automated assembly), etc. A change of assembly tool also incurs time and increases the assembly cost. The number of assembly tool changes may be computed as follows.

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

where, n is the number of components in the assembly, $tool\_change_{i,i+1}$ indicates the change in assembly tool for two consecutive assembly operations, $tool\_number_i$ 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),

$$tool\_change_{i,i+1} = \begin{cases} 0, & if tool\_number_i = tool\_number_{i+1} \\ 1, & otherwise \end{cases} \tag{4}$$

Thus in assembly sequence optimization, the above two objectives pertaining to assembly orientation changes and tool changes should be minimized to reduce the time and cost of manufacturing. Accordingly, for evaluating the assembly sequences, a fitness function (FF) has been formulated and given in Eq. 5.

$$\begin{aligned} FF = 1/(&w_x * \text{Total no. of direction changes} \\ &+ w_y * \text{Total no.of tool changes} \\ &- feasibility\ index) \end{aligned} \tag{5}$$

where w_x and w_y are the weights associated with total no. of direction changes and total no. of tool changes respectively.

For pruning out the infeasible assembly sequences while evaluating them, it is necessary to check their feasibility using the precedence matrix. To assign a higher fitness value to feasible sequences, a feasibility index has been introduced as stated in Eq. 6.

$$\text{The feasibility index} = \begin{cases} 2, & if\ the\ assembly\ sequence\ is\ feasible \\ 0, & otherwise \end{cases} \tag{6}$$

Moreover, it is to be kept in mind that there may be many possible alternative assembly sequences all of which are equally optimal i.e. they require same number of orientation changes and tool changes and thus have the same fitness function (FF) value. Therefore, the assembly sequence optimization

is a multi-modal optimization problem. Hence, our objective is to find not only a single optimum assembly sequence but to identify as many optimum assembly sequences as possible.

Basic principle of flower pollination algorithm

This section presents the basic principle of flower pollination algorithm (FPA) or flower algorithm that was originally proposed by Yang (2012). Flowering plants in nature have been evolving for more than 125 million years through the process of evolution. In a plant, a flower ultimately serves the purpose of reproduction via pollination. Flower pollination is typically associated with the transfer of pollen, and such transfer is often performed by pollinators such as insects, birds, bats and other animals. In nature, pollination phenomenon can take two forms: abiotic and biotic. Biotic pollination is estimated to account for about 90 % of flowering plants, with pollens being transferred by insects and/or animals. It is estimated that about 10 % of pollination belongs to abiotic pollination that does not require any pollinators. Wind and diffusion in water assist in pollination of such flowering plants and grass is a good example of a product that is the result of abiotic pollination. Honeybees are another good example of pollinator, which can also develop the flower constancy i.e. tendency of pollinators to visit exclusively certain flower species while avoiding other flower species. Such flower constancy may provide evolutionary advantages as this can maximise the transfer of flower pollen to the same plants, and thus maximising the reproduction and hence survival of the same flower species.

Furthermore, pollination can also take the form of self-pollination or cross-pollination. Cross-pollination, or allogamy, is pollination occurring from pollen of a flower of a different plant, whereas self-pollination is the fertilisation of one flower, from pollen of the same flower or different flowers of the same plant, which often occurs when there is no reliable pollinator available. Biotic, cross-pollination may occur at long distances, and the pollinators such as bees, bats, birds and flies can fly a long distance, thus they can be considered as the global pollination. In addition, bees and birds may behave Levy flight behavior, with jump or fly distance steps that obey a Levy distribution.

The following rules have been laid down for flower pollination algorithm (Yang 2012),

1. Biotic and cross-pollination is considered as global pollination process with pollen carrying pollinators performing Levy flights.
2. Abiotic and self-pollination are considered as local pollination.
3. Flower constancy can be considered as the reproduction probability, which is proportional to the similarity of two flowers involved.
4. Local pollination and global pollination are controlled by a switch probability $p \in [0, 1]$. Due to the physical proximity and other factors such as wind, local pollination can have a significant fraction p in the overall pollination activities.

The global pollination (rule 1) can be represented mathematically as

$$x_i^{t+1} = x_i^t + \gamma L \left(g^* - x_i^t\right) \tag{7}$$

where, x_i^t is the pollen i or solution vector x_i at iteration t, and g* is the current best solution found among all solutions at the current generation/iteration. L is a step-size parameter, more specifically the Levy-flights-based step size that corresponds to the strength of the pollination. Since insects may travel over a long distance with various distance steps, a Levy flight can be used to mimic this characteristic efficiently. That is, $L > 0$ is drawn from a Levy distribution is given in Eq. 8.

$$L = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} * \frac{1}{s^{1+\lambda}} \tag{8}$$

In the above equation, the gamma function Γ, as suggested in the paper by Yang (2012), is the standard gamma function whose value depends on the value of λ. Yang (2012) had suggested the value of λ to be 1.5, for which gamma function value is 0.88 (Abramowitz and Stegun 1964).

The local pollination (rule 2) can be represented as given in Eq. 9 where ∈ is drawn from the uniform distribution in the range of 0 to 1.

$$x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right) \tag{9}$$

Figure 1 shows the pseudo code of the basic flower pollination algorithm developed by Yang (2012).

Proposed flower pollination algorithm for assembly sequence optimization

The main challenge to apply FPA to solve an assembly sequence optimization problem is the continuous nature of the basic FPA presented by Yang (2012). This is so because the search space of basic FPA is a real space domain, while in assembly sequence optimization, the solution search space is discrete. Therefore, the following modifications in basic FPA have been made by us in the rules for local and global pollination to make it suited for solving the problem on hand.

```
Objective min or max f(x), x = (x1, x2, x3,…., xd)
Initialise a population of n flowers/pollens gametes with random solutions
Fins the best solution g* in the initial population
Define a switch probability p ∈ [0, 1]
while (t<MaxGeneration)
        for i=1:n (all n flowers in the population)
                if rand< p
                        Draw a (d-dimensional) step vector L which obeys a Levy distribution
                        Global pollination via x_i^{t+1}=x_i^t + γ L (g*-x_i^t)
                else
                        Draw ∈ from a uniform distribution in [0,1]
                        Randomly choose j and k among all solutions
                        Do local pollination via x_i^{t+1}=x_i^t + ∈ (x_j^t-x_k^t)
                end if
        Evaluate new solutions
        If new solutions are better, update them in the population
        end for
Find the current best solution g*
end while
```

Fig. 1 Pseudo code of the basic flower pollination algorithm

Proposed representation scheme

An assembly sequence has been represented as a string, thus the sequence [1 3 4 5 6 7 8 9 10 15 16 2 12 11 13 14] is represented as "01,03,04,05,06,07,08,09,10,15,16,02,12,11,13, 14", each part number being represented by a two-digit number e.g. part number 2 being represented by 02, part number 13 being represented by 13.

Proposed method of implementing global pollination rules

Let us suppose that we have two assembly sequences represented by the following two strings:

String 1: "01,04,08,10,16,07,03,09,05,15,06,02,12,13, 14,11"

String 2: "01,04,07,08,10,16,03,09,15,06,05,02,13,14, 12,11"

Suppose that string 1 is the global best (g*) found among all solutions at the current iteration and the string 2 (x_i^t) is the string under consideration on which the global pollination rule is to be applied to generate a new string (x_{ii}^{t+1}). As per the global updating rule, the difference ($g^* - x_i^t$) between the two strings is first computed as follows.

$(g^* - x_i^t)$ = "0000010205909999900900999899020 0"

The difference ($g^* - x_i^t$) is then multiplied by the value of L computed as per the equation 7 and a scaling factor (γ) which yields the following result. The value of γ has been taken as 10^{-10} for the given problem to control the step size. The following shows the method of computing the product, $\gamma * L * (g^* - x_i^t)$ for a step length of s = 5.

$$\gamma * L * (g^* - x_i^t) = 10^{-10} * 0.0053 * 0000010205909999900900999899020 0 = 54, 12, 18, 22, 00, 04, 75$$

The above result is then added into the string x_i^t as per the global pollination rule to yield a new assembly sequence as explained below.

$$\begin{array}{r} 01, 04, 07, 08, 10, 16, 03, 09, 15, 06, 05, 02, 13, 14, \underline{12, 11} \\ + \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 54, 12, 18, 22, 00, \underline{04, 75} \\ \hline 01, 04, 07, 08, 10, \underline{16,}03, 09, 15, 60, 17, 20, 35, 14, \underline{16, 86} \end{array}$$

As shown above, addition is performed on each component of the assembly from the right side. It may be, however, necessary to repair the string following the addition operation as explained below. If after addition, the value becomes more than the maximum number of components in the assembly, the value is reverted to its previous value. For example, if 75 (last two digits of the $\gamma * L * (g^* - x_i^t)$) is added to last two digits of string 2 (x_i^t), the value becomes 86. Since the maximum number of components in the assembly is 16, hence this value is reverted to 11 only. Next, the last but one digit of the string 2 (x_i^t) correspond to 12. When the last but one digit of the $\gamma * L * (g^* - x_i^t)$) i.e. 04 is added, the resultant becomes 16. But there is already a component 16 in the sequence. As it is not permitted to have the same component number repeated more than once in a sequence, it is necessary to repair the above string through a rearrangement of the some of the string elements so as to make sure that no number is repeated in the same assembly sequence. In the

above string, it is observed that the component number 12 is missing and therefore the component 16 is replaced with the component 12. Thus the new string resulting from the global pollination is as given below.

New string
= "01, 04, 07, 08, 10, 12,03, 09, 15, 06, 05, 02, 13, 14, 16,11"

Proposed method of implementing local pollination rules

The local pollination equation takes the difference $(x_j^t - x_k^t)$ between two strings x_j^t and x_k^t, which are selected from the same population. These two strings are selected from the population by generating two random numbers within the population size. Let us assume that the strings are as given below:

x_i^t = "01, 04, 10, 07, 08, 16, 03, 05, 06, 09, 15, 02, 12, 13, 14, 11"
x_j^t = "01, 04, 03, 06, 10, 16, 05, 07, 08, 09, 15, 02, 11, 13, 14, 12"
x_k^t = "01, 03, 04, 08, 05, 06, 07, 10, 09, 16, 15, 02, 14, 11, 13, 12"

The difference between $(x_j^t - x_k^t)$ gives the following value

$(x_j^t - x_k^t)$ = "0000989805099796989299999702010 0"

A random number $(\in)$ is drawn from a uniform distribution in the range of 0 to 1, which is then multiplied with the difference between the two strings $(x_j^t - x_k^t)$ along with a scaling factor (k) that we have introduced in the modified local pollination rule. The resulting product is next added to the selected string (x_i^t) on which local pollination is being performed. After taking the scaling factor (k) into consideration, the equation for local pollination given in Eq. 9 is modified as follows.

$$x_i^{t+1} = x_i^t + k * \in * \left(x_j^t - x_k^t\right) \tag{10}$$

The reason for introducing the scaling factor (k) is to shorten the length of the resulting string to be added so as to make sure that after the addition, only a portion of the original string (x_i^t) gets modified, while some portion of the original string may remain intact. This may help to ensure that some property of the original string is inherited by the new offspring.

For example, if the random number generated is 0.03069, then we obtain the following

$k * \varepsilon * (x_j^t - x_k^t)$
$= 10^{-20} * 0.03069 * 0000989805099796989299999702010 0$
= "03, 03, 80, 04"

The above result is then added into the string x_i^t as per the local pollination rule given in Eq. 6 to yield a new assembly sequence in a similar manner as done for global pollination rule.

01, 04, 10, 07, 08, 16, 03, 05, 06, 09, 15, 02, 12, 13, 14, 11	
+	03, 03, 80, 04
01, 04, 10, 07, 08, 16,03, 05, 06, 09, 15,02, 15, 16,94, 15	

Next it is necessary to repair above string which is the result of addition. For example, the bits which are repeated in the assembly sequence e.g. 15 and 16 need to be rearranged in such a way that no component number is repeated in the same assembly sequence as already explained above in case of our proposed scheme of global pollination rule. The new sequence resulting from the local pollination obtained after the rearrangement as well as the old sequence is as given below.

Old sequence = "01, 04, 10, 07, 08, 16, 03, 05, 06, 09, 15, 02, 12, 13, 14, 11"
New sequence = "01, 04, 10, 07, 08, 11, 03, 05, 06, 09, 12, 02, 15, 16, 14, 13"

While implementing the local pollination rule given in Eq. 10, the string representing the result of $k * \varepsilon * (x_j^t - x_k^t)$ was initially added to the string x_i^t at the end. However, it was found that the FPA algorithm required large number of iterations for convergence to the optimum and often it got stuck in local optimum. It was then decided to add string representing the result of $k * \varepsilon * (x_j^t - x_k^t)$ at different positions in the string x_i^t, the position being chosen randomly. It was found after the above modification that the number of iterations required for convergence to the optimum reduced and the average fitness of the solutions also improved by as much as 40 %. It has further been observed because of the use of randomly selected location for the addition of multiplied value with the original string, the effect of not having same-number conflict has minimized and in turn, the performance of the algorithm is immune to this effect.

To investigate the effect of the parameter scaling factor k on the performance of the algorithm, the value of k was varied from 10^{-30} to 10^{-10} by keeping the population size fixed at 20, the step length 7, switch probability (p) 0.5. It was observed that the algorithm performs best for the value of $k = 10^{-20}$ in terms of highest average fitness and largest number of unique optimum solutions. However, lower values of k ($<10^{-20}$) adversely affected the performance of the

algorithm in the form of slower convergence, lower average fitness and lesser number of unique optimum solutions. Further for higher values of k ($> 10^{-20}$), the run time of the algorithm was found to be longer. This behavior is perhaps due to the fact that for higher values of k, the resulting string to be added is longer, which explains why it will take longer computational time.

The modified FPA proposed by us has been implemented in MATLAB for obtaining the optimal assembly sequences and the results are presented in the next section. For comparing the results of our proposed FPA with those of GA, ACO and IHS algorithms, the GA given in Choi et al. (2009), the ACO algorithm given in Wang et al. (2005) and the IHS algorithm given in Li et al. (2016) have also been implemented in MATLAB and their results have been also presented. However, in the GA approach adopted in the present paper, the crossover operator that has been used is partially matched crossover (PMX), whereas in the GA approach by Choi et al. (2009), precedence preservative crossover (PPX) was used. The reason for selecting PMX over PPX is that the results obtained by us using the PMX crossover operator were found to give slight improvement in maintaining the diversity of solutions than those obtained using the PPX.

Illustrative examples: results and discussions

Two example products taken from literature are considered here to demonstrate the working of the proposed FPA and also to evaluate its performance and compare it with those of GA, ACO and IHS.

Example 1: Motor drive assembly

The first example is a motor drive assembly with a total of 12 different parts adapted from Design for Assembly User Guide (2010). Figure 2 shows the assembly with all its parts and their assembly directions. Table 1 lists the tools and grippers for performing the assembly. The above information must be provided by the user as input. In addition to the above tool and gripper information and the assembly directions, the user also inputs the precedence relationships between the parts as shown in Table 2. The component number 1 has been selected as the base component.

Table 1 Tools and grippers for performing the motor drive assembly shown in Fig. 2

Part no.	Part name	Tool/gripper name
1	Motor base	Two-finger parallel gripper
2	Bushing_1	Two-finger parallel gripper
3	Motor	Two-finger parallel gripper
4	End plate	Two-finger adaptive gripper
5	Sensor	Two-finger adaptive gripper
6	Cover	Two-finger parallel gripper
7	Motor screw_1	Slotted screw driver no. 1
8	Set screw	Allen key
9	End plate screw_1	Slotted screw driver no. 2
10	Cover screw_1	Philips screw driver
11	Stand off_1	Open end wrench
12	Grommet	Hammer

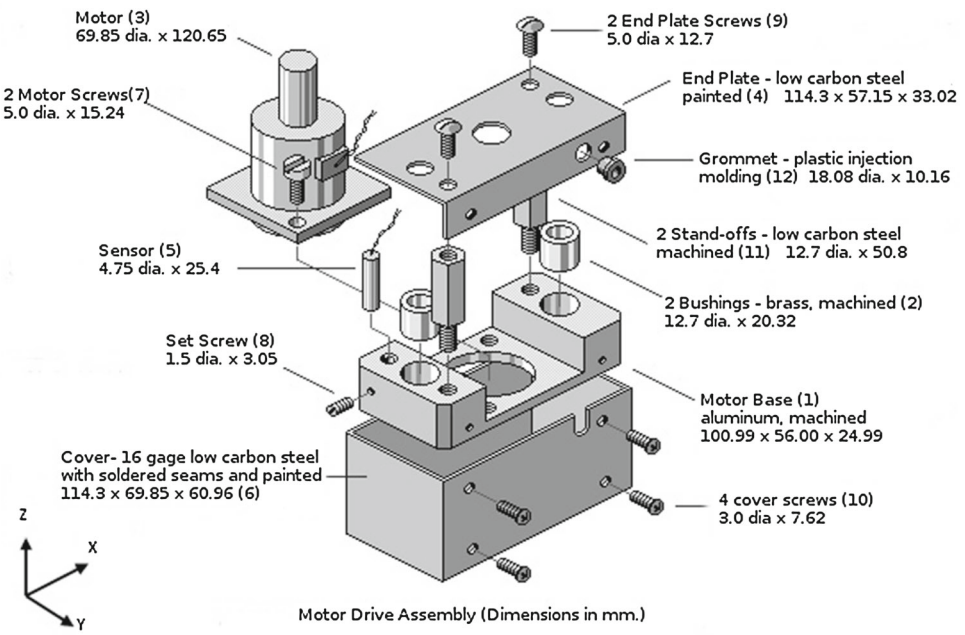


Fig. 2 A motor drive assembly

Table 2 Precedence relationships between the parts of the motor drive assembly

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	0	1	0	1	1	0	0	1	0
5	1	0	0	0	0	0	0	0	0	0	0	0
6	1	1	1	1	1	0	1	1	1	0	1	1
7	1	0	1	0	0	0	0	0	0	0	0	0
8	1	0	0	0	1	0	0	0	0	0	0	0
9	1	1	1	1	1	0	1	1	0	0	1	0
10	1	1	1	1	1	1	1	1	1	0	1	1
11	1	0	0	0	0	0	0	0	0	0	0	0
12	1	1	1	1	1	0	1	1	1	0	1	0
Assembly directions	$-z$	$-z$	$-z$	$-z$	$-z$	$+z$	$-z$	$+x$	$-z$	$-y$	$-z$	$-y$

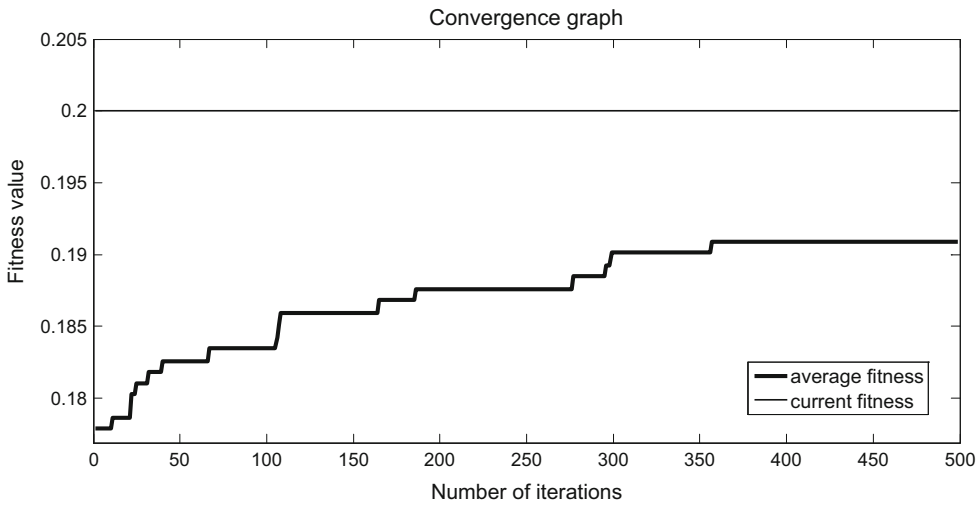


Fig. 3 Convergence graph of FPA for the motor drive assembly for 500 iterations

Results of FPA simulation runs for motor drive assembly

The basic FPA developed by Yang (2012) has five main control parameters. These include population size, maximum number of iterations, switch probability (between 0 and 1), a factor by which step size is controlled i.e. γ and the step size s used to calculate the value of L in the global pollination rule. In the modified FPA proposed by us, there is an additional parameter namely the scaling factor (k) used in the local pollination rule. Initially in our FPA simulation runs, the population size was fixed at 6 and the step length was fixed at 5, switch probability (p) was increased from 0.5 to 0.7 in steps of 0.1 keeping maximum number of iterations constant at 500. The population size was increased from 6 to 25 in steps of 5. The step length was increased from 5 to 7 in steps of 1. The maximum fitness obtained is 0.2 (assuming the weights $w_x = w_y = 0.5$ in the fitness function given in equation 5). The maximum number of unique optimum solutions obtained after 500 iterations is 9. For 500 iterations, the optimum set of FPA parameters for which the maximum number of unique optimum solutions is obtained is population size of 20, step length of 7 and switch probability of 0.5. Figure 3 shows the convergence graph of FPA for 500 iterations. Table 3 shows the list of optimum assembly sequences corresponding to above set of FPA parameters.

Using the above set of parameters, the number of iterations was increased to 2000 to see if we obtain more number of unique optimum solutions. Figure 4 shows the convergence graph of FPA for 2000 iterations. It is observed that the maximum number of optimum solutions was found to increase to 15. Table 4 shows the list of optimum assembly sequences obtained after 2000 iterations.

Table 3 List of unique optimum feasible assembly sequences for motor drive assembly generated by FPA for population size of 20 after 500 iterations

Sl. No.	Optimal assembly sequence	No. of orientation changes	No. of tool changes	Fitness value
1.	[1 2 3 5 8 11 7 4 9 12 6 10]	5	9	0.2000
2.	[1 2 3 7 5 11 8 4 9 12 6 10]	5	9	0.2000
3.	[1 2 3 11 5 7 8 4 9 12 6 10]	5	9	0.2000
4.	[1 3 2 5 7 8 11 4 9 12 6 10]	5	9	0.2000
5.	[1 3 2 5 7 11 8 4 9 12 6 10]	5	9	0.2000
6.	[1 3 2 5 8 7 11 4 9 12 6 10]	5	9	0.2000
7.	[1 3 2 5 11 8 7 4 9 12 6 10]	5	9	0.2000
8.	[1 3 2 11 5 8 7 4 9 12 6 10]	5	9	0.2000
9.	[1 3 2 11 7 5 8 4 9 12 6 10]	5	9	0.2000

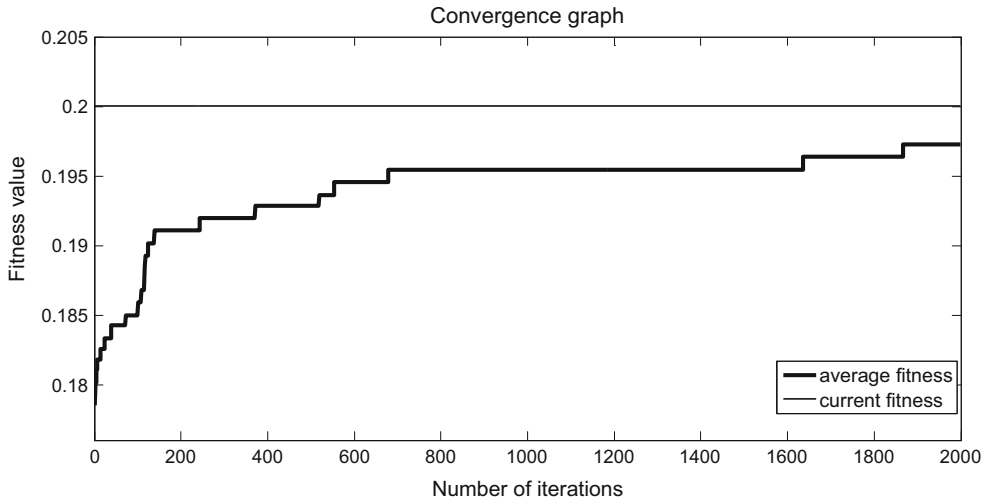


Fig. 4 Convergence graph of FPA for the motor drive assembly for 2000 iterations

Table 4 List of unique optimum assembly sequences for motor drive assembly generated by FPA for population size of 20 after 2000 iterations

Sl. No.	Optimal assembly sequence	No. of orientation changes	No. of tool changes	Fitness value
1.	[1 2 3 5 7 8 11 4 9 12 6 10]	5	9	0.2000
2.	[1 2 3 5 8 11 7 4 9 12 6 10]	5	9	0.2000
3.	[1 2 3 5 11 7 8 4 9 12 6 10]	5	9	0.2000
4.	[1 2 3 5 11 8 7 4 9 12 6 10]	5	9	0.2000
5.	[1 2 3 11 5 7 8 4 9 12 6 10]	5	9	0.2000
6.	[1 2 3 11 5 8 7 4 9 12 6 10]	5	9	0.2000
7.	[1 2 3 11 7 5 8 4 9 12 6 10]	5	9	0.2000
8.	[1 3 2 5 7 11 8 4 9 12 6 10]	5	9	0.2000
9.	[1 3 2 5 8 7 11 4 9 12 6 10]	5	9	0.2000
10.	[1 3 2 5 8 11 7 4 9 12 6 10]	5	9	0.2000
11.	[1 3 2 5 11 7 8 4 9 12 6 10]	5	9	0.2000
12.	[1 3 2 5 11 8 7 4 9 12 6 10]	5	9	0.2000
13.	[1 3 2 7 11 5 8 4 9 12 6 10]	5	9	0.2000
14.	[1 3 2 11 5 7 8 4 9 12 6 10]	5	9	0.2000
15.	[1 3 2 11 7 5 8 4 9 12 6 10]	5	9	0.2000

Table 5 Comparison of results of GA, ACO, IHS and FPA for motor drive assembly after 500 iterations

	GA	ACO	IHS	Proposed FPA
Optimum fitness value	0.2000	0.2000	0.2000	*0.2000*
Optimum number of tool changes	9	9	9	*9*
Optimum number of direction changes	5	5	5	*5*
Average fitness value	0.2000	0.1991	0.1891	0.1909
Number of unique optimum assembly sequences	4	9	8	*9*

Table 6 Comparison of results of GA, ACO, IHS and FPA for motor drive assembly after 2,000 iterations

	GA	ACO	IHS	Proposed FPA
Optimum fitness value	0.2000	0.2000	0.2000	*0.2000*
Optimum number of tool changes	9	9	9	*9*
Optimum number of direction changes	5	5	5	*5*
Average fitness value	0.2000	0.1952	0.1894	0.1973
Number of unique optimum assembly sequences	4	9	12	*15*

Comparison between the results of FPA, GA, ACO and IHS algorithms for motor drive assembly

The GA simulation runs were carried out for 500 iterations and population size of 20. Using the same fitness function given in Eq. 5, the maximum fitness obtained was 0.2. The maximum number of unique optimum solutions obtained after 500 iterations is 4. For 500 iterations, the optimum set of GA parameters for which the maximum number of unique optimum solutions was obtained is crossover probability of 0.90 and mutation probability of 0.05. Using the above set of parameters, as the number of iterations was increased to 2000, the maximum number of optimum solutions was still found to be 4 i.e. the same as that obtained after 500 iterations.

Similarly, the simulation runs of the ACO algorithm were also carried out for 500 iterations and population size of 20. Using the same fitness function given in Eq. 5, the maximum fitness obtained was 0.2. The maximum number of unique optimum solutions obtained after 500 iterations is 9. For 500 iterations, the optimum set of ACO parameters for which the maximum number of unique optimum solutions was obtained is rate of pheromone evaporation (ρ) 0.1, pheromone decay parameter (γ)0.1, relative importance of pheromone (β)0.8, and initial value of pheromone trail (τ_0)1. Using the above set of parameters, as the number of iterations was increased to 2000, the maximum number of optimum solutions was still found to be 9 i.e. the same as that obtained after 500 iterations.

Similarly, the simulation runs of the IHS algorithm were also carried out for 500 iterations and population size of 20. Using the same fitness function given in equation 5, the maximum fitness obtained was 0.2. The maximum number of unique optimum solutions obtained after 500 iterations is 8. For 500 iterations, the optimum set of IHS parameters for which the maximum number of unique optimum solutions was obtained is Harmony Memory Considering Rate (HMCR) 0.9 and Pitch Adjustment Rate (PAR) 0.1. Using the above set of parameters, as the number of iterations was increased to 2000, the maximum number of optimum solutions was found to increase to 12.

Table 5 presents the comparison between the results of simulation runs of GA, ACO, IHS and FPA for 500 iterations for the given example of motor drive assembly keeping population size 20 for all the four algorithms. Table 6 presents the comparison between the results of simulation runs of the four algorithms for 2000 iterations. The numbers indicated in italics in Tables 5 and 6 are the results obtained by the proposed FPA, which are better than or as good as the results obtained by GA, ACO and IHS. From the results of Table 5, after 500 iterations, the proposed FPA is found to give optimum fitness value, which is comparable with those of GA, ACO and IHS. In terms of capability to generate multiple unique optimum solutions, both the proposed FPA and ACO could generate 9 unique optimum solutions, as compared to GA which could give only 4 unique optimum solutions and IHS which could give only 8 unique optimum solutions, and therefore after 500 iterations, both FPA and ACO were found to be better than GA and IHS. It is to be further noted that the GA population at the end of all the simulations runs was found to be less diverse (i.e. having lesser number of unique optimum solutions) compared to FPA, ACO and IHS, and it comprised of multiple copies of one or more optimum assembly sequences. Therefore from Tables 5 and 6, although average fitness of GA is higher than FPA, the number of unique optimum solutions generated by GA is far less compared to FPA. Figure 5 shows the comparison between the convergence graphs of FPA, GA, ACO and IHS after 500 iterations. From a comparison of the above graphs, it is found that convergence

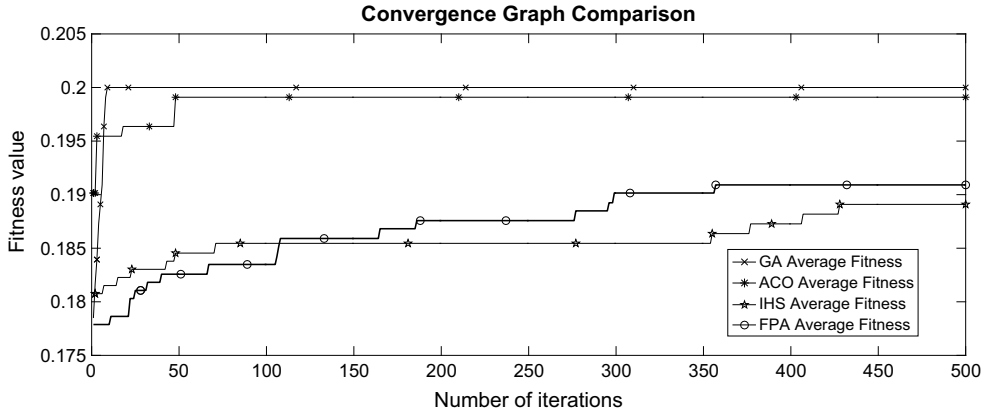
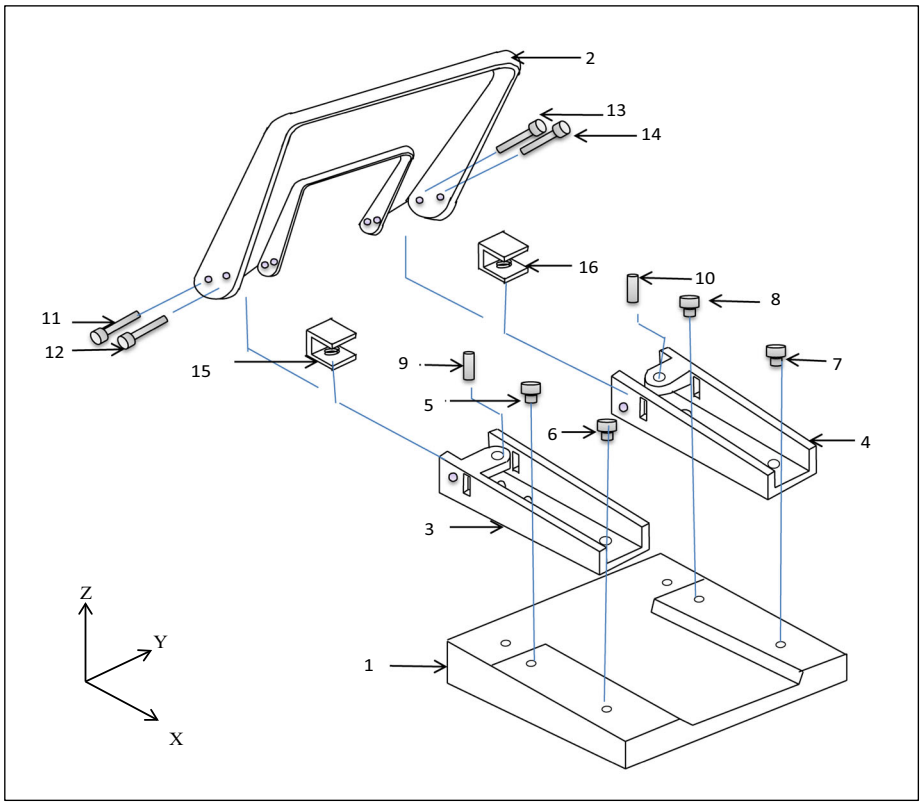


Fig. 5 Comparison of convergence of the four algorithms (i.e. GA, ACO, IHS and FPA) for motor drive assembly after 500 iterations



Part no.	Part name
1	Base
2	Top Cover
3	U-shaped bracket a
4	U-shaped bracket b
5	Rivet a
6	Rivet b
7	Rivet c
8	Rivet d
9	Cutter a
10	Cutter b
11	Rivet e
12	Rivet f
13	Rivet g
14	Rivet h
15	Clip a
16	Clip b

Fig. 6 Punching machine assembly

of FPA is slower compared to GA and ACO, and faster than IHS in general. However, as the number of iterations was increased to 2000, FPA is found to clearly outperform GA, ACO and IHS in terms of its capability to generate more number of unique optimum assembly sequences i.e. 15 compared to only 4, 9, and 12 sequences that could be generated by GA, ACO, and IHS respectively. Furthermore, as the simulation run is continued for 2000 iterations, it is observed that the average fitness of FPA tends to catch up with the average fitness of GA, ACO and IHS. In conclusion, FPA is clearly found to be better in solving the given multi-modal optimization problem of assembly sequence optimization.

Example 2: Punching machine assembly

The second example that has been considered is that of punching machine assembly. This assembly contains 16 components. The assembly with all the parts and their assembly directions have been shown in Fig. 6. Table 7 lists the tools and grippers for performing the assembly. The above information must be given by the user as input. In addition to

the above information, the user also inputs the precedence relationships between the parts as shown in Table 8. The component number 1 has been selected as the base component.

Table 7 Tools and grippers for performing the punching machine assembly shown in Fig. 5

Part no.	Part name	Tool/gripper name
1	Base	Two-finger adaptive gripper
2	Top cover	Three-finger adaptive gripper no. 1
3	*U*-shaped bracket a	Two-finger parallel gripper
4	*U*-shaped bracket b	Two-finger parallel gripper
5	Rivet a	Rivet gun no. 1
6	Rivet b	Rivet gun no. 1
7	Rivet c	Rivet gun no. 1
8	Rivet *d*	Rivet gun no. 1
9	Cutter a	Rivet gun no. 2
10	Cutter b	Rivet gun no. 2
11	Rivet e	Rivet gun no. 2
12	Rivet f	Rivet gun no. 2
13	Rivet g	Rivet gun no. 2
14	Rivet h	Rivet gun no. 2
15	Clip a	Three-finger adaptive gripper no. 2
16	Clip b	Three-finger adaptive gripper no. 2

Results of FPA simulation runs for punching machine assembly

Initially in our FPA simulation runs, the population size was fixed at 6 and the step length was fixed at 5, switch probability (p) was increased from 0.5 to 0.7 in steps of 0.1 keeping maximum number of iterations constant at 700. The population size was increased from 6 to 25 in steps of 5. The step length was increased from 5 to 7 in steps of 1. The maximum fitness obtained is 0.4 (assuming the weights $w_x = w_y = 0.5$ in the fitness function given in equation 5). The maximum number of unique optimum solutions obtained after 700 iterations is 9. For 700 iterations, the optimum set of FPA parameters for which the maximum number of unique optimum solutions is obtained is population size of 20, step length of 7 and switch probability of 0.5. Figure 7 shows the convergence graph of FPA for 700 iterations. Table 9 shows the list of optimum assembly sequences corresponding to above set of FPA parameters. All the above assembly sequences have the same fitness value.

Using the above set of parameters, the number of iterations was increased to 10,000 to see if we obtain more number of unique optimum solutions. It is further observed that the maximum number of optimum solutions was found to increase to 18. Figure 8 shows the convergence graph of FPA for 10,000 iterations. Table 10 shows the list of unique optimum assembly sequences obtained after 10,000 iterations.

Table 8 Precedence matrix for punching machine assembly

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
11	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0
12	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0
13	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1
14	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1
15	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
16	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
Assembly directions	$-z$	$+x$	$-z$	$-z$	$-z$	$-z$	$-z$	$-z$	$-z$	$-z$	$+y$	$+y$	$-y$	$-y$	$+x$	$+x$

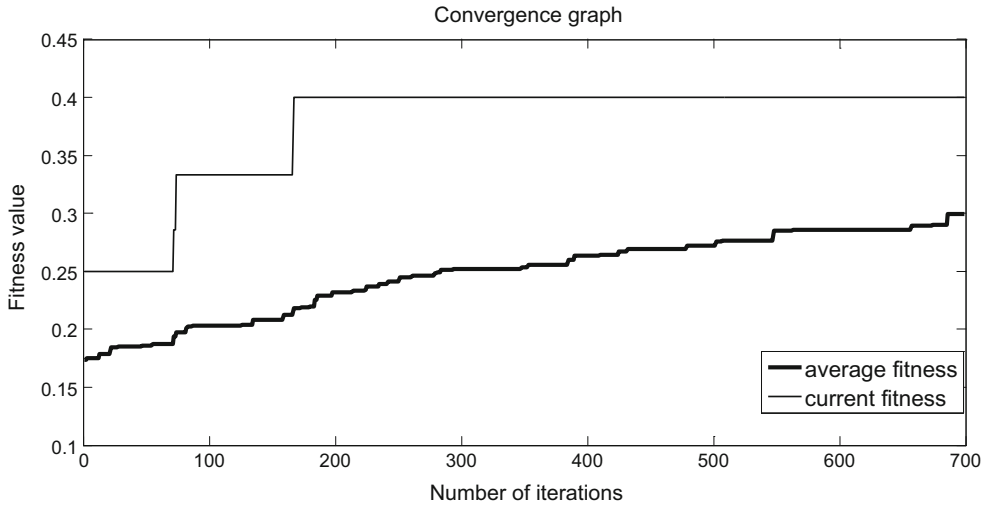


Fig. 7 Convergence graph of FPA for the punching machine assembly for 700 iterations

Table 9 List of unique optimum assembly sequences for punching machine generated by FPA for population size of 20, and maximum number of iterations 700

Sl. No.	Optimal assembly sequence	No. of orientation changes	No. of tool changes	Fitness value
1.	[1 3 4 6 8 5 7 9 10 15 16 2 13 14 11 12]	3	6	0.4000
2.	[1 3 4 10 9 7 5 6 8 15 16 2 13 14 11 12]	3	6	0.4000
3.	[1 4 3 10 9 8 6 7 5 16 15 2 13 14 11 12]	3	6	0.4000
4.	[1 3 4 9 10 6 8 7 5 15 16 2 11 12 14 13]	3	6	0.4000
5.	[1 3 4 5 6 7 8 10 9 15 16 2 11 12 14 13]	3	6	0.4000
6.	[1 4 3 10 9 6 5 8 7 16 15 2 14 13 11 12]	3	6	0.4000
7.	[1 4 3 5 8 6 7 10 9 16 15 2 12 11 14 13]	3	6	0.4000
8.	[1 4 3 6 5 8 7 10 9 16 15 2 12 11 13 14]	3	6	0.4000
9.	[1 3 4 7 5 8 6 10 9 16 15 2 12 11 14 13]	3	6	0.4000

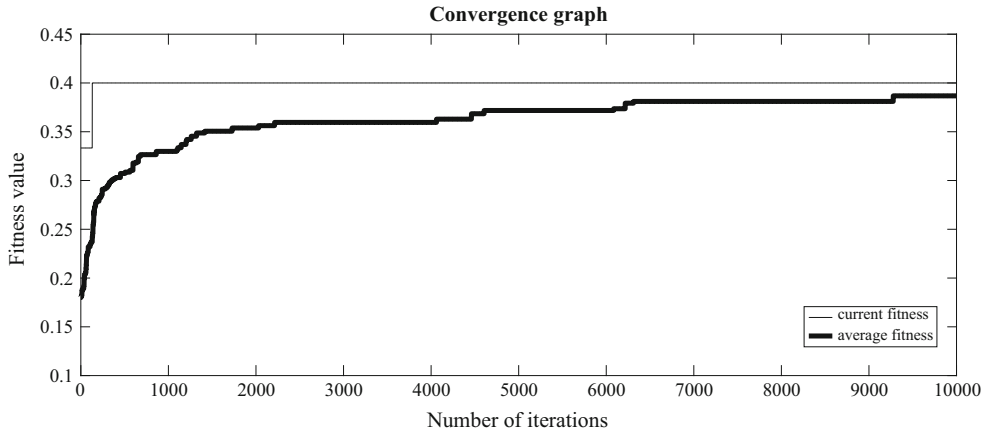


Fig. 8 Convergence graph of FPA for the punching machine assembly for 10,000 iterations

Comparison between the results of FPA, GA, ACO and IHS algorithms for punching machine assembly

The GA simulation runs were carried out for 700 iterations and population size of 20. Using the same fitness function given in Eq. 5, the maximum fitness obtained was 0.4. The maximum number of unique optimum solutions obtained after 700 iterations is 4. For 700 iterations, the optimum set of GA parameters for which the maximum number of unique optimum solutions was obtained is crossover probability of

Table 10 List of unique optimum assembly sequences for punching machine generated by FPA for population size of 20, and maximum number of iterations 10,000

Sl. No.	Optimal assembly sequence																No. of orientation changes	No. of tool changes	Fitness value
1.	1	4	3	9	10	8	7	6	5	15	16	2	14	13	12	11	3	6	0.4000
2.	1	3	4	9	10	6	8	7	5	15	16	2	13	14	12	11	3	6	0.4000
3.	1	3	4	6	7	5	8	9	10	15	16	2	13	14	11	12	3	6	0.4000
4.	1	4	3	9	10	5	8	7	6	16	15	2	13	14	11	12	3	6	0.4000
5.	1	3	4	9	10	5	8	7	6	15	16	2	12	11	14	13	3	6	0.4000
6.	1	4	3	10	9	8	6	7	5	15	16	2	14	13	11	12	3	6	0.4000
7.	1	4	3	8	7	6	5	9	10	15	16	2	13	14	12	11	3	6	0.4000
8.	1	3	4	9	10	5	6	7	8	16	15	2	14	13	12	11	3	6	0.4000
9.	1	3	4	10	9	8	6	5	7	16	15	2	14	13	11	12	3	6	0.4000
10.	1	3	4	9	10	5	6	7	8	16	15	2	12	11	14	13	3	6	0.4000
11.	1	3	4	5	7	6	8	10	9	15	16	2	13	14	12	11	3	6	0.4000
12.	1	4	3	6	5	8	7	10	9	15	16	2	14	13	11	12	3	6	0.4000
13.	1	3	4	6	8	5	7	9	10	15	16	2	13	14	12	11	3	6	0.4000
14.	1	3	4	6	5	8	7	9	10	15	16	2	12	11	13	14	3	6	0.4000
15.	1	4	3	9	10	8	7	5	6	16	15	2	14	13	11	12	3	6	0.4000
16.	1	3	4	5	6	8	7	9	10	16	15	2	11	12	14	13	3	6	0.4000
17.	1	3	4	9	10	8	5	7	6	16	15	2	13	14	11	12	3	6	0.4000
18.	1	3	4	9	10	5	6	8	7	15	16	2	14	13	12	11	3	6	0.4000

Table 11 Comparison of the results of GA, ACO, IHS and FPA for punching machine assembly after 700 iterations

	GA	ACO	IHS	FPA
Optimum fitness value	0.4000	0.4000	0.4000	*0.4000*
Optimum number of tool changes	6	6	6	*6*
Optimum number of direction changes	3	3	3	*3*
Average fitness value	0.4000	0.3091	0.2474	0.2941
Number of optimum unique assembly sequences	4	9	3	*9*

0.90 and mutation probability of 0.05. Using the above set of parameters, as the number of iterations was increased to 10,000, the maximum number of optimum solutions was found to increase to 8.

Similarly, the simulation runs of the ACO algorithm were also carried out for 700 iterations and population size of 20. Using the same fitness function given in Eq. 5, the maximum fitness obtained was 0.4. The maximum number of unique optimum solutions obtained after 700 iterations is 9. For 700 iterations, the optimum set of ACO parameters for which the maximum number of unique optimum solutions was obtained is rate of pheromone evaporation (ρ) 0.1, pheromone decay parameter (γ) 0.1, relative importance of pheromone (β) 0.8, and initial value of pheromone trail (τ_0) 1. Using the above set of parameters, as the number of iterations was increased to 10000, the maximum number of unique optimum solutions was still found to be 9 i.e. the same as that obtained after 700 iterations.

Similarly, the simulation runs of the IHS algorithm were also carried out for 700 iterations and population size of 20. Using the same fitness function given in Eq. 5, the maximum fitness obtained was 0.4. The maximum number of unique optimum solutions obtained after 700 iterations is 3. For 700 iterations, the optimum set of IHS parameters for which the maximum number of unique optimum solutions was obtained is Harmony Memory Considering Rate (HMCR) 0.9 and Pitch Adjustment Rate (PAR) 0.1. Using the above set of parameters, as the number of iterations was increased to 10000, the maximum number of optimum solutions was found to increase to 11.

Table 11 presents the comparison between the results of simulation runs of GA, ACO, IHS and FPA for 700 iterations for the given example of punching machine assembly keeping population size 20 for all the four algorithms. Table 12 presents the comparison between the results of simulation runs of the four algorithms for 10,000 iterations. The numbers indicated in italics in Tables 11 and 12 are the results obtained by the proposed FPA, which are better than or as good as the results obtained by GA, ACO and IHS. From the results of Table 11, after 700 iterations, the proposed FPA is found to give optimum fitness value, which is comparable with those of GA, ACO and IHS. In terms of capability to generate multiple unique optimum solutions, both the proposed FPA and ACO could generate 9 unique

Table 12 Comparison of the results of GA, ACO, IHS and FPA for punching machine assembly after 10000 iterations

	GA	ACO	IHS	FPA
Optimum fitness value	0.4000	0.4000	0.4000	*0.4000*
Optimum number of tool changes	6	6	6	*6*
Optimum number of direction changes	3	3	3	*3*
Average fitness value	0.4000	0.3543	0.4000	0.3868
Number of optimum unique assembly sequences	8	9	11	*18*

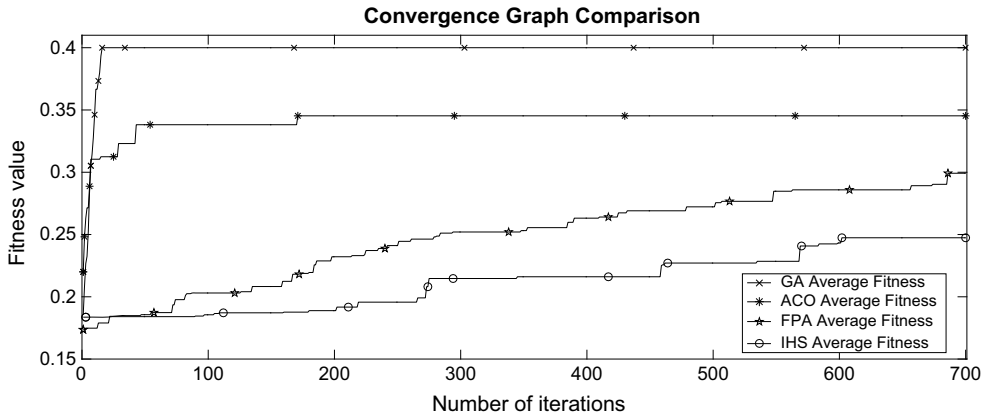


Fig. 9 Comparison of convergence of the four algorithms (i.e. GA, ACO, IHS and FPA) for punching machine example for 700 iterations

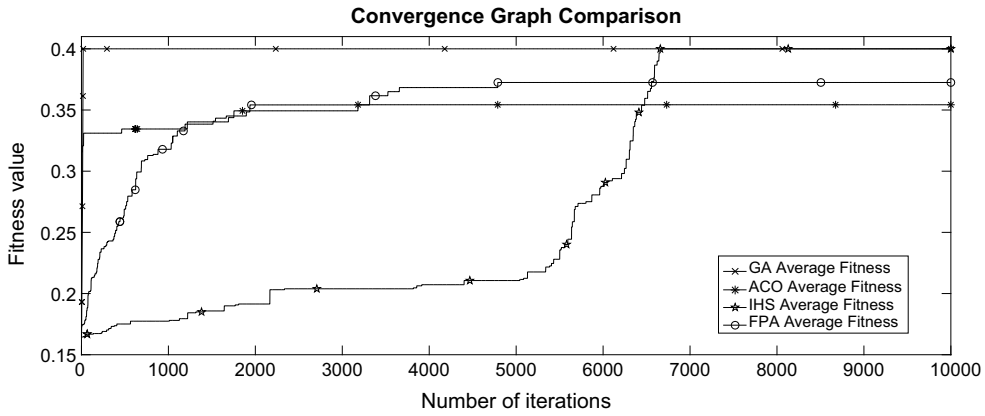


Fig. 10 Comparison of convergence of the four algorithms (i.e. GA, ACO, IHS and FPA) for punching machine example for 10,000 iterations

optimum solutions, as compared to GA which could give only 4 unique optimum solutions and IHS which could give only 3 unique optimum solutions, and therefore after 700 iterations, both FPA and ACO were found to be better than GA and IHS. Figure 9 shows the comparison between the convergence graphs of GA, ACO, IHS and FPA after 700 iterations. From a comparison of the above graphs, it is found that convergence of FPA is initially slower compared to GA and ACO and faster than IHS in general. However, as the number of iterations was increased to 10,000, FPA is found to clearly outperform GA, ACO and IHS in terms of its capability to generate more number of unique optimum assembly sequences i.e. 18 compared to only 8, 9 and 11 sequences that could be generated by GA, ACO and IHS respectively. Furthermore, as the simulation run is continued for 10,000 iterations, it is observed that the average fitness of FPA tends to catch up with the average fitnesses of GA and IHS, and surpasses the average fitness of ACO as shown in Fig. 10 and Table 12. It is to be further noted that in case of the GA and IHS, the population at the end of all the simulation runs was found to be less diverse (i.e. having lesser number of unique solutions) compared to FPA and it comprised of multiple copies of one or more assembly sequences as can be seen in Tables 14 and 16. Therefore, in Fig. 10 and Table 12 we see that the average fitnesses of GA and IHS are higher than FPA, even though the number of unique optimum solutions

Table 13 List of all the assembly sequences found in the population after 10,000 iterations of FPA for the punching machine assembly with population size 20 (Note: 18 out of 20 solutions shown in bold are found to be unique optimum solutions with fitness value 0.4)

Sl. No.	Assembly sequences																No. of orientation changes	No. of tool changes	Fitness value
1.	**1**	**4**	**3**	**9**	**10**	**8**	**7**	**6**	**5**	**15**	**16**	**2**	**14**	**13**	**12**	**11**	3	6	0.4000
2.	**1**	**3**	**4**	**9**	**10**	**6**	**8**	**7**	**5**	**15**	**16**	**2**	**13**	**14**	**12**	**11**	3	6	0.4000
3.	**1**	**3**	**4**	**6**	**7**	**5**	**8**	**9**	**10**	**15**	**16**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
4.	**1**	**4**	**3**	**9**	**10**	**5**	**8**	**7**	**6**	**16**	**15**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
5.	**1**	**3**	**4**	**9**	**10**	**5**	**8**	**7**	**6**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
6.	**1**	**4**	**3**	**10**	**9**	**8**	**6**	**7**	**5**	**15**	**16**	**2**	**14**	**13**	**11**	**12**	3	6	0.4000
7.	**1**	**4**	**3**	**8**	**7**	**6**	**5**	**9**	**10**	**15**	**16**	**2**	**13**	**14**	**12**	**11**	3	6	0.4000
8.	**1**	**3**	**4**	**9**	**10**	**5**	**6**	**7**	**8**	**16**	**15**	**2**	**14**	**13**	**12**	**11**	3	6	0.4000
9.	**1**	**3**	**4**	**10**	**9**	**8**	**6**	**5**	**7**	**16**	**15**	**2**	**14**	**13**	**11**	**12**	3	6	0.4000
10.	**1**	**3**	**4**	**9**	**10**	**5**	**6**	**7**	**8**	**16**	**15**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
11.	**1**	**3**	**4**	**5**	**7**	**6**	**8**	**10**	**9**	**15**	**16**	**2**	**13**	**14**	**12**	**11**	3	6	0.4000
12.	**1**	**4**	**3**	**6**	**5**	**8**	**7**	**10**	**9**	**15**	**16**	**2**	**14**	**13**	**11**	**12**	3	6	0.4000
13.	**1**	**3**	**4**	**6**	**8**	**5**	**7**	**9**	**10**	**15**	**16**	**2**	**13**	**14**	**12**	**11**	3	6	0.4000
14.	**1**	**3**	**4**	**6**	**5**	**8**	**7**	**9**	**10**	**15**	**16**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
15.	1	3	4	9	10	5	8	15	16	7	6	2	11	12	14	13	5	7	0.2500
16.	**1**	**4**	**3**	**9**	**10**	**8**	**7**	**5**	**6**	**16**	**15**	**2**	**14**	**13**	**11**	**12**	3	6	0.4000
17.	**1**	**3**	**4**	**5**	**6**	**8**	**7**	**9**	**10**	**16**	**15**	**2**	**11**	**12**	**14**	**13**	3	6	0.4000
18.	**1**	**3**	**4**	**9**	**10**	**8**	**5**	**7**	**6**	**16**	**15**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
19.	1	4	3	10	9	15	16	5	7	6	8	2	13	14	11	12	5	6	0.2857
20.	**1**	**3**	**4**	**9**	**10**	**5**	**6**	**8**	**7**	**15**	**16**	**2**	**14**	**13**	**12**	**11**	3	6	0.4000

Table 14 List of all the assembly sequences found in the population after 10,000 iterations of GA for the punching machine assembly with population size 20 (Note: 8 out of 20 solutions shown in bold are found to be unique optimum solutions with fitness value 0.4)

Sl. No.	Assembly sequences																No. of orientation changes	No. of tool changes	Fitness value
1.	**1**	**3**	**4**	**10**	**9**	**8**	**5**	**7**	**6**	**16**	**15**	**2**	**11**	**12**	**14**	**13**	3	6	0.4000
2.	1	3	4	10	9	8	5	7	6	16	15	2	11	12	14	13	3	6	0.4000
3.	1	3	4	10	9	8	5	7	6	16	15	2	11	12	14	13	3	6	0.4000
4.	1	3	4	10	9	8	5	7	6	16	15	2	11	12	14	13	3	6	0.4000
5.	**1**	**3**	**4**	**7**	**8**	**6**	**5**	**10**	**9**	**16**	**15**	**2**	**11**	**12**	**14**	**13**	3	6	0.4000
6.	**1**	**3**	**4**	**10**	**9**	**8**	**5**	**7**	**6**	**16**	**15**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
7.	**1**	**3**	**4**	**6**	**5**	**8**	**7**	**10**	**9**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
8.	1	3	4	7	8	6	5	10	9	16	15	2	11	12	14	13	3	6	0.4000
9.	**1**	**3**	**4**	**7**	**8**	**6**	**5**	**10**	**9**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
10.	**1**	**3**	**4**	**10**	**9**	**6**	**5**	**8**	**7**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
11.	**1**	**3**	**4**	**8**	**7**	**6**	**5**	**10**	**9**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
12.	1	3	4	7	8	6	5	10	9	16	15	2	11	12	14	13	3	6	0.4000
13.	1	3	4	10	9	6	5	8	7	15	16	2	12	11	14	13	3	6	0.4000
14.	1	3	4	10	9	8	5	7	6	16	15	2	11	12	14	13	3	6	0.4000
15.	1	3	4	6	5	8	7	10	9	15	16	2	12	11	14	13	3	6	0.4000
16.	1	3	4	7	8	6	5	10	9	15	16	2	12	11	14	13	3	6	0.4000
17.	**1**	**3**	**4**	**6**	**5**	**8**	**7**	**10**	**9**	**15**	**16**	**2**	**11**	**12**	**14**	**13**	3	6	0.4000
18.	1	3	4	10	9	8	5	7	6	16	15	2	12	11	14	13	3	6	0.4000
19.	1	3	4	8	7	6	5	10	9	15	16	2	12	11	14	13	3	6	0.4000
20.	1	3	4	8	7	6	5	10	9	15	16	2	12	11	14	13	3	6	0.4000

generated by GA and IHS is found to be far less compared to FPA.

The proposed FPA is able to maintain diversity in the optimum solutions that are found over the generations, in other words, most of the multiple optimum solutions are found to be unique i.e. 18 out of 20 solutions in the population are found to be unique optimum solutions after 10,000 iterations of FPA as given in Table 13 (note that unique optimum solutions are marked in bold). On the other hand, it was observed that, in case of GA, ACO and IHS, it is not possible to maintain the diversity in multiple optimum solutions as the complete population finally converges to a few unique

Table 15 List of all the assembly sequences found in the population after 10,000 iterations of ACO for the punching machine assembly with population size 20 (Note: 9 out of 20 solutions shown in bold are found to be unique optimum solutions with fitness value 0.4)

Sl. No.	Assembly sequences																No. of orientation changes	No. of tool changes	Fitness value
1.	1	3	4	8	10	9	6	5	7	16	15	2	11	12	13	14	3	7	0.3333
2.	1	4	3	6	8	5	10	9	16	15	7	2	12	14	13	11	6	7	0.2222
3.	**1**	**4**	**3**	**5**	**8**	**6**	**7**	**10**	**9**	**16**	**15**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
4.	1	4	3	9	10	7	16	15	6	5	8	2	12	11	13	14	5	7	0.2500
5.	**1**	**4**	**3**	**7**	**8**	**5**	**6**	**9**	**10**	**15**	**16**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
6.	1	4	3	6	8	10	9	15	16	7	5	2	11	12	14	13	5	7	0.2500
7.	1	3	6	5	9	15	4	8	7	10	16	2	14	13	12	11	5	10	0.1818
8.	**1**	**4**	**3**	**6**	**7**	**8**	**5**	**10**	**9**	**16**	**15**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
9.	**1**	**3**	**4**	**8**	**6**	**5**	**7**	**10**	**9**	**16**	**15**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
10.	1	4	10	7	8	16	3	9	6	5	15	2	11	13	12	14	7	10	0.1538
11.	**1**	**4**	**3**	**7**	**6**	**8**	**5**	**10**	**9**	**16**	**15**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
12.	1	3	6	5	9	4	8	7	10	15	16	2	14	13	11	12	3	9	0.2500
13.	1	4	8	7	10	16	3	6	5	9	15	2	11	12	13	14	5	10	0.1818
14.	**1**	**4**	**3**	**5**	**8**	**6**	**7**	**10**	**9**	**15**	**16**	**2**	**11**	**12**	**13**	**14**	3	6	0.4000
15.	1	3	4	8	7	10	9	16	15	5	6	2	13	14	11	12	5	7	0.2500
16.	1	3	5	6	9	4	7	8	10	15	16	2	13	14	11	12	3	9	0.2500
17.	**1**	**4**	**3**	**9**	**10**	**7**	**6**	**8**	**5**	**16**	**15**	**2**	**11**	**12**	**14**	**13**	3	6	0.4000
18.	**1**	**3**	**4**	**8**	**6**	**5**	**7**	**10**	**9**	**16**	**15**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
19.	1	4	8	7	10	3	6	5	9	16	15	2	14	13	12	11	3	9	0.2500
20.	**1**	**4**	**3**	**7**	**8**	**5**	**6**	**10**	**9**	**15**	**16**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000

Table 16 List of all the assembly sequences found in the population after 10,000 iterations of IHS for the punching machine assembly with population size 20 (Note: 11 out of 20 solutions shown in bold are found to be unique optimum solutions with fitness value 0.4)

Sl. No.	Assembly sequences																No. of orientation changes	No. of tool changes	Fitness value
1.	**1**	**3**	**4**	**8**	**6**	**7**	**5**	**9**	**10**	**15**	**16**	**2**	**11**	**12**	**14**	**13**	3	6	0.4000
2.	**1**	**3**	**4**	**9**	**10**	**7**	**5**	**6**	**8**	**15**	**16**	**2**	**13**	**14**	**11**	**12**	3	6	0.4000
3.	**1**	**4**	**3**	**9**	**10**	**7**	**5**	**6**	**8**	**15**	**16**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
4.	**1**	**3**	**4**	**9**	**10**	**8**	**5**	**6**	**7**	**15**	**16**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
5.	**1**	**3**	**4**	**9**	**10**	**7**	**5**	**6**	**8**	**15**	**16**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
6.	1	4	3	9	10	7	5	6	8	15	16	2	12	11	13	14	3	6	0.4000
7.	1	3	4	9	10	8	5	6	7	15	16	2	12	11	13	14	3	6	0.4000
8.	1	4	3	9	10	7	5	6	8	15	16	2	12	11	13	14	3	6	0.4000
9.	**1**	**4**	**3**	**9**	**10**	**8**	**5**	**6**	**7**	**15**	**16**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
10.	**1**	**4**	**3**	**9**	**10**	**8**	**5**	**6**	**7**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
11.	1	4	3	9	10	7	5	6	8	15	16	2	12	11	13	14	3	6	0.4000
12.	**1**	**3**	**4**	**9**	**10**	**7**	**5**	**6**	**8**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
13.	1	3	4	9	10	8	5	6	7	15	16	2	12	11	13	14	3	6	0.4000
14.	1	3	4	9	10	8	5	6	7	15	16	2	12	11	13	14	3	6	0.4000
15.	**1**	**3**	**4**	**10**	**9**	**7**	**5**	**6**	**8**	**15**	**16**	**2**	**12**	**11**	**13**	**14**	3	6	0.4000
16.	1	3	4	9	10	7	5	6	8	15	16	2	12	11	13	14	3	6	0.4000
17.	1	4	3	9	10	7	5	6	8	15	16	2	12	11	13	14	3	6	0.4000
18.	**1**	**4**	**3**	**9**	**10**	**7**	**5**	**6**	**8**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000
19.	1	4	3	9	10	7	5	6	8	15	16	2	12	11	14	13	3	6	0.4000
20.	**1**	**4**	**3**	**9**	**10**	**5**	**8**	**6**	**7**	**15**	**16**	**2**	**12**	**11**	**14**	**13**	3	6	0.4000

optimum solutions after 10,000 iterations. This is evident from the Tables 14, 15 and 16 (note that unique optimum solutions are marked in bold). In case of GA, only 8 unique optimums out of 20 solutions in the population are found (as given in Table 14), and in case of ACO, only 9 unique optimums out of 20 solutions in the population are found (as given in Table 15) after 10,000 iterations and in case of IHS, only 11 unique optimums out of 20 solutions in the population are found (as given in Table 16) after 10,000 iterations.

In conclusion, the above results once again demonstrate that FPA clearly outperforms GA, ACO and IHS in terms of

its capability to generate multiple unique optimum assembly sequences.

Discussions

It has been stated earlier that the main aim of the present paper is not only to obtain the global optimum sequences, but also to get as many unique optimum sequences as possible. From the results of comparison between FPA, GA, ACO and IHS presented in sections "Comparison between the results of FPA, GA, ACO and IHS algorithms for motor drive assembly" and "Comparison between the results of FPA, GA, ACO and IHS algorithms for punching machine assembly", it can be concluded that the proposed FPA is not only able to find out the global optimum solutions just like well-known algorithms of GA, ACO and recently published IHS, but it is also capable of generating more number of unique optimum solutions than GA, ACO and IHS. The proposed FPA is able to maintain diversity in the optimum solutions found over the generations. On the other hand, in case of GA, ACO and IHS, it is not possible to maintain the diversity in multiple optimum solutions as the complete population finally converges to a few unique optimum solutions.

In the proposed FPA, global search is ensured by the global pollination rule which mimics the biotic and cross-pollination i.e. fertilization of one flower occurring from pollen of a flower of a different plant with pollen carrying pollinators like birds and insects flying over long distances by performing Levy flights. The diversity of solutions is maintained as the pollen (solutions in the population) is allowed to explore the search space during global pollination so as to promote exploration, thereby leading the algorithm explore a widespread search space. Local search to explore the solution space around certain near optimal solutions is ensured by the local pollination rule, which mimics the process of abiotic and self-pollination i.e. fertilisation of one flower, from pollen of the same flower or different flowers of the same plant, which often occurs when there is no reliable pollinator available. The local pollination can be thus understood to promote exploitation. In FPA, the switch probability to switch between global and local pollination rules is used to balance between explorations and exploitations in order to find the optimum solution.

However, the results also show that to maintain diversity in the optimum solutions in the population by the proposed FPA comes at the expense of slower rate of convergence. In fact, it is interesting to note that the very term "convergence" refers to a population's decrease in diversity over time (Baeck et al. 2000). So it is perhaps not totally surprising that while trying to maintain diversity in the optimum solutions generated, the convergence rate of the FPA is affected. In fact, it has been earlier reported in some papers (Chen and Montgomery 2011) that maintaining diversity tends to reduce the rate of convergence. The reason for higher average fitness value in the initial iterations of the evolving populations of GA and ACO is due to fast convergence to the optimum solution and presence of multiple copies of one or more optimum solutions in the population, while due to the slower convergence rate of the proposed FPA, the population as it is still evolving has a lower average fitness. But after continuing the simulation run for 10000 iterations in the punching machine example, it is observed that the average fitnesses of FPA and IHS tend to catch up with the average fitness of GA and surpasses the average fitness of ACO. It is further found that although after 10000 iterations, the average fitness of IHS is higher than that of FPA, the population at the end of all the simulation runs of IHS is found to be less diverse (i.e. having lesser number of unique solutions) compared to FPA and it comprised of multiple copies of one or more assembly sequences.

To the best of our knowledge, most of the previous work on assembly sequence optimization focused on determining a single feasible and optimal assembly sequence based on minimization of tool changes and/or orientation changes as they account for a significant proportion of the handling cost during assembly. However, there might be many candidates for optimal sequences. In practice, there might be other constraints that may be encountered during assembly (Boothroyd 2005) leading to increase in time and cost. For example, there may be resistance during part insertion due to small clearances available for manipulating the part with the tool/gripper. Sometimes, the part may be unstable after placement or during subsequent operations and will require realignment until it is finally secured or require holding down by special fixturing. While the cost for special fixtures will be significant for small batch production, it may be, on the other hand, less significant for mass production and therefore should be left to the user's discretion to include it in the list of criteria for optimization. Therefore, our strategy in this paper is to generate as many unique optimum solutions as possible based on optimality criteria of minimizing tool and orientation changes and leave it to the user to take a call on the best sequence according to other softer optimization criteria.

Let us consider the nine optimal assembly sequences for the motor drive example given in Table 3. The optimality criteria used was minimization of tool and orientation changes. The sequences were further analyzed using Design for Assembly software tool kit (Boothroyd and Dewhurst Inc.) and it was found that out of the above nine sequences, the sequence nos. 2, 4, 5 and 6 give the least cost. Out of the five other sequences, sequence nos. 1 and 7 give 8 % more cost due to the fact that in the presence of the standoffs, there will be some resistance to insertion of motor screw due to small clearance available for manipulating the screw driver. As a result, assembly will take longer time. Sequence nos. 3, 8 and 9 give 11 % more cost for the difficulty in insertion of

both the sensor and the motor screw for again the same reason as above i.e. due to small clearance available for manipulating the gripper for the sensor and for manipulating the screw driver for the motor screw.

Likewise, if we subject the nine optimal assembly sequences for the punching machine example given in Table 9 to further analysis by Design for Assembly software, it was found that the sequence nos. 1, 5, 7, 8, 9 give the least cost. The remaining four sequence nos. 2, 3, 4, and 6 result in 2 % more cost. This is due to the fact that in these sequences, the U-shaped brackets are placed onto the base without first securing them by rivets, and therefore the brackets would be unstable during subsequent insertion of the cutters. Hence the U-shaped brackets would require holding down by special fixturing which will result in additional assembly time and fixturing cost.

Multiple assembly sequences can be sometimes useful for providing alternative routing opportunities in case of dynamic scheduling of mixed-model assembly lines, where multiple product varieties are processed simultaneously in the same line. During scheduling, if it is found that one of the assembly operations on a product type has to wait due to unavailability of the assembly station as it is occupied in processing another product type, then alternative process routes can be explored to see if the assembly precedence constraints would permit a subsequent assembly operation to be done before. In this way, not only the manufacturing resources can be optimally utilized but also the total assembly time can be reduced by cutting down the waiting time and work-in-process can be reduced. However, this is only possible when there are multiple ways/sequences of assembling the parts together for a given assembly.

Conclusions

In this paper, an intelligent assembly sequence optimization methodology based on application of FPA has been developed to automatically generate multiple optimal assembly sequences, subject to various assembly precedence constraints, based on minimisation of number of orientation changes and tool changes. The detailed description of the developed methodology are given including the proposed representation scheme as well as the strategies for implementing local and global pollination rules for the given discrete search space while ensuring that no component number is repeated in a generated assembly sequence and further the introduction of a scaling factor in the equation for local pollination, etc. The potential and feasibility for application of the developed methodology has been illustrated with the help of two example products and the results of FPA have been compared with two well-known soft computing based optimization algorithms namely, GA and ACO and also with a recently published soft computing based algorithm, IHS. The following are some of the important contributions of the research work reported in this paper:

1. FPA is, for the first time, being applied to the assembly sequence optimization problem.
2. The main challenge in applying FPA for assembly sequence optimization problem was the continuous nature of the basic FPA that was first developed by Yang (2012). This is so because the search space of basic FPA is a real space domain, while in assembly sequence optimization, the solution search space is discrete. Therefore, in the proposed FPA, modifications have been made by us in the rules for local and global pollination to make it suited for solving the given discrete optimization problem.
3. It is to be kept in mind that the assembly sequence optimization is a multi-modal optimization problem, i.e. it may possess multiple optimum solutions (having the same fitness function value). Although from the literature review it is evident that many different soft computing based optimization approaches have been developed for determining the optimum assembly sequence, none of the papers have addressed the need for generating multiple optimum solutions. Keeping the above in mind, the present paper is aimed at not only obtaining the global optimum solution, but also as many unique optimum solutions as possible.
4. A comparison of the results of FPA simulations with those of two other well-known algorithms of GA and ACO and a recently published soft computing based algorithm, IHS demonstrate that FPA gives optimum number of tool changes and direction changes, which are as good as those obtained by GA, ACO and IHS. The novelty of the proposed FPA lies in its capability to find multiple unique optimum solutions in one single simulation run and capability to automatically maintain diversity in the optimum solutions found over the generations. On the other hand, in case of GA, ACO and IHS, it is not possible to maintain the diversity in multiple optimum solutions as the complete population finally converges to a few unique optimum solutions. However, the results also show that to maintain diversity in the optimum solutions in the population by the proposed FPA comes at the expense of slower rate of convergence. Infact, it has been earlier reported in some papers that maintaining diversity tends to reduce the rate of convergence. Nevertheless, the proposed FPA clearly outperforms GA, ACO and IHS in terms of its capability to generate more number of unique optimum solutions. Therefore, it can be concluded that FPA performs better in solving the given multi-modal optimization problem of assembly sequence optimization.

A future research direction could be application of the proposed FPA to solve other discrete optimization problems in manufacturing such as machining sequence planning, disassembly sequence planning, scheduling of manufacturing systems, and so on.

References

Abramowitz, M., & Stegun, I. A. (1964). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. USA: National Bureau of Standards Applied Mathematics Series.

Baeck, T., Fogel, D. B., & Michalewicz, Z. (2000). *Evolutionary computation 1: Basic algorithms and operators*. London: Taylor and Francis.

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

Boothroyd, G. (2005). *Assembly automation and product design* (2nd ed.). Florida: Taylor and Francis, CRC Press.

Cao, P. B., & Xiao, R. B. (2007). Assembly planning using a novel immune approach. *International Journal of Advanced Manufacturing Technology*, *31*, 770–782.

Chen, S., & Liu, Y. J. (2001). An adaptive genetic assembly sequence planner. *International Journal of Computer Integrated Manufacturing*, *14*(5), 489–500.

Chen, S., & Montgomery, J. (2011). *A simple strategy to maintain diversity and reduce crowding in particle swarm optimization, vol. 7106 of the series lecture notes in computer science*, pp. 281–290.

Choi, Y. K., Lee, D. M., & Cho, Y. B. (2009). An approach to multi-criteria assembly sequence planning using genetic algorithms. *International Journal of Advanced Manufacturing Technology*, *42*, 180–188.

Design for Assembly Tool kit, Release 9.4. (2010). Boothroyd Dewhurst Inc., Wakefield, Rhode Island, USA.

Dubey, H. M., Pandit, M., & 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., & Wang, J. F. (2010). Application of memetic algorithm in assembly sequence planning. *International Journal of Advanced Manufacturing Technology*, *49*, 1175–1184.

Jimenez, P. (2013). Survey on assembly sequencing: A combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, *24*(2), 235–250.

Kashkoush, M., & ElMaraghy, H. (2013). Consensus tree method for generating master assembly sequence. *Production Engineering Research Development*, *8*, 233–242.

Li, M., Wu, B., Hu, Y., Jin, C., & Shi, T. (2013). A hybrid assembly sequence planning approach based on discrete particle swarm optimization and evolutionary direction operation. *International Journal of Advanced Manufacturing Technology*, *68*, 617–630.

Li, X., Qin, K., Zeng, B., Gao, L., & Su, J. (2016). Assembly sequence planning based on an improved harmony search algorithm. *International Journal of Advanced Manufacturing Technology*, *84*, 2367–2380.

Lv, H. G., & Lu, C. (2010). An assembly sequence planning approach with a discrete particle swarm optimization algorithm. *International Journal of Advanced Manufacturing Technology*, *50*, 761–770.

Marian, R. M., Luong, L. H., & Abhary, K. (2006). A genetic algorithm for the optimization of assembly sequences. *Computers & Industrial Engineering*, *50*, 503–527.

Sabarinath, P., Karthick, R., Thansekhar, M. R., & 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*, pp. 166–171, Chennai, India.

Tiwari, M. K., Prakash, A., & 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*, *219*, 137–149.

Tseng, H. E., Wang, W. P., & Shih, H. Y. (2007). Using memetic algorithms with guided local search to solve assembly sequence planning. *Expert Systems with Applications*, *33*, 451–467.

Wang, J. F., Liu, J. H., & Zhong, Y. F. (2005). A novel ant colony algorithm for assembly sequence planning. *International Journal of Advanced Manufacturing Technology*, *25*, 1137–1143.

Wang, Y., & Liu, J. H. (2010). Chaotic particle swarm optimization for assembly sequence planning. *Robotics and Computer-Integrated Manufacturing*, *26*, 212–222.

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

Xing, Y. F., & Wang, Y. S. (2012). Assembly sequence planning based on a hybrid particle swarm optimisation and genetic algorithm. *International Journal of Production Research*, *50*(24), 7303–7312.

Yang, X.-S. (2012). Flower pollination algorithms. In X.-S. Yang (Ed.), *Nature-inspired optimization algorithms* (pp. 155–173). London: Elsevier.

Yu, J., & Wang, C. (2013). A max-min ant colony system for assembly sequence planning. *International Journal of Advanced Manufacturing Technology*, *67*, 2819–2835.

Zhang, H., Liu, H., & Li, L. (2014). Research on a kind of assembly sequence planning based on immune algorithm and particle swarm optimization algorithm. *International Journal of Advanced Manufacturing Technology*, *71*, 795–808.

Zhou, W., Zheng, J. R., Yan, J. J., & Wang, J. F. (2011). A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm. *International Journal of Advanced Manufacturing Technology*, *52*, 715–724.