A GA based parameter meta-optimization of ACO algorithm for solving assembly sequence optimization

Atul Mishra[1] and Sankha Deb[2]
FMS and Computer Integrated Manufacturing Laboratory,
Department of Mechanical Engineering,
Indian Institute of Technology Kharagpur, Kharagpur -721302, India.
[1] atulmishra_chdi@yahoo.in
[2] sankha.deb@mech.iitkgp.ernet.in

ABSTRACT

Assembly sequence planning deals with finding the sequence of operations to assemble the components and sub-assemblies into the final product. With advent of Artificial Intelligence, several soft-computing based evolutionary optimisation algorithms had been used by researchers to solve the problem of finding the best feasible assembly sequence. In the proposed paper, an Ant Colony Optimisation (ACO) algorithm based approach has been used for assembly sequence optimisation based on minimizing the number of direction changes, while conforming to the precedence constraints between components. Because of the graph-based nature of ACO, it is relatively less computationally expensive than many of the other soft computing based approaches reported in the literature, and moreover it is also well suited to the nature of the problem on hand. However, the ACO algorithm has a lot of parameters such as rate of pheromone evaporation, pheromone decay parameter, number of iterations, etc. each of which needs to be varied in order to obtain the best convergence rate of the algorithm, Furthermore, the combination of the parameters also needs to be optimised. In the present paper, a binary coded Genetic Algorithm (GA) has been proposed to optimise the parameters of the ACO algorithm. Some of the reasons for choosing GA include the fact that the number of parameters in GA is less than ACO and thus it takes less time and efforts to optimise the GA than ACO. Moreover the modeling of the solutions (i.e. chromosomes) in the form of binary strings is easier. The optimized ACO parameters have been used for solving a problem of assembly sequence optimization for a sixteen component assembly to demonstrate the effectiveness of our proposed meta-optimization procedure.

Keywords: Computer-Aided Process Planning, Assembly Sequence Optimization, Ant Colony Optimisation, Meta-optimisation by Genetic Algorithm.

1 INTRODUCTION

Assembly sequence planning deals with finding the sequence of operations to assemble the components into final product. Since assembly costs contribute to a significant amount of total manufacturing cost, proper assembly sequence planning plays a crucial role in reducing cost. Significant expertise and knowledge are necessary to determine the optimal assembly sequence. Moreover, with increase in number of components, the number of feasible assembly sequences possible also increases, making manual process planning laborious and time consuming. To overcome these drawbacks, various Computer-Aided Process Planning approaches have been developed. With advent of Artificial Intelligence, several soft-computing based evolutionary optimization algorithms had been used for assembly sequence optimization, based on criteria like changes in orientation, assembly tools, and stability. They possess number of parameters and their values can have significant impact on the effectiveness of an algorithm. Moreover, the parameter values need to be tuned for each new problem instance. Naturally a key challenge is determining the optimum combination of parameters (known as meta-optimization), which if done manually, may involve lot of time and effort. In the present paper, an ACO algorithm has been used for assembly sequence optimization based on number of orientation changes, while conforming to the part precedence constraints. Because of graph-based nature, it is less computationally expensive than other soft computing approaches, and moreover it is also well suited to the nature of the problem on hand. However, ACO has a lot of parameters such as Q value to adjust the increment of pheromone, rate of pheromone evaporation, pheromone decay parameter, number of iterations required for convergence. Each of these needs to be varied to obtain the best algorithm performance. Furthermore, the parameter combination also needs to be optimized. In the present paper, a binary coded GA has been proposed to optimise the ACO parameters. Reasons for choosing GA include the fact that the number of GA parameters is less than ACO and modeling of the solutions in form of binary strings is easier.

A brief review of different approaches for assembly sequence optimization is given below. Hong and Cho [1] developed an approach for robotic assembly sequence optimization by Hopfield neural network and an expert system to infer the assembly constraints from the liaison data of the product. Bonneville et al. [2] developed a GA based approach. 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 et al. [3] proposed a three-stage integrated approach with heuristic working rules to assist the planner to develop a better assembly plan. Above Graph and transforming rules were used to create a correct Explosion Graph of the assembly models, followed by a three-level relational model to create a complete relational model graph and an incidence matrix, a mathematical model based on a penalty index was formulated, and a revised minimum spanning table method was used to generate and evaluate a feasible assembly sequence. Wang et al. [4] developed an ACO approach for optimisation of assembly sequences. The optimal solution was with respect to the least reorientations during assembly. The concept of assembly by disassembly was adopted in that paper and disassembly matrix was used to guarantee the validity and feasibility of sequences. The parameters of this ACO were chosen after trial computations. Cao and Xiao [5] explored Immune Optimisation Algorithm (IOA). It was based on the bionic principles of Artificial Immune Systems (AIS). Assembly sequences were evaluated on the basis of total number of components, number of changes in assembly directions and tools, base component location, and feasibility degree. Chang et al. [6] showed the application of AIS for assembly sequence planning exploration using connector concepts. Combination property, tool property, direction property, precedence relationships were considered. Wang and Liu [7] developed a PSO based approach. Assembly cost was subjected to geometrical constraints and five assembly process constraints namely, local assembly precedence, number of the unstable parts, assembly direction and tool changes, connector changes. Lv and Lu [8] presented the application of discrete PSO. They considered total number of tool changes, orientation changes, and operation type changes and interference times in the product assembly. Gao et al. [9] developed an 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. Zhou et al. [10] combined the Bacterial Chemotaxis with GA. Assembly sequences were encoded as chromosomes, where gene in the chromosome is treated as a bacterium. Fitness function comprised of length of longest sub-sequence, number of orientation changes, number of gripper changes. Xing and Wang [11] presented a hybrid PSO and GA based optimisation for compliant assemblies based on graph theory. Liaison graph and adjacency matrix were used to describe the geometry of the compliant assemblies. Assembly sequences were evaluated on the basis of assembly variation due to dimensional tolerance. Recently, Li et al. [12] developed an 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.

In the following sections, first the approach of assembly sequence optimisation by using ACO is described, followed by the proposed approach for parameter meta-optimization of ACO by GA. An illustrative example is given to demonstrate the application and the results and discussions are presented. Finally important conclusions and scope for future work are given.

2 APPROACH FOR ASSEMBLY SEQUENCE OPTIMIZATION BY USING ACO

The ACO approach developed by Wang et al. [4] has been applied to determine the best feasible assembly sequence with least number of direction changes. The concept of assembly by disassembly has been adopted, where a disassembly sequence has been represented as an ordered list of disassembly operations (DO). Each DO is denoted as a duple DO = (N, D), where N is the component number and D is its disassembly direction i.e. either of ±X/±Y/±Z. The feasibility of an assembly sequence is ensured using a Disassembly Matrix (DM). The number of rows of DM is equal to number of parts n. The number of columns of DM is equal to 3n, where each column of corresponding component refers to one of the disassembly directions, +X/+Y/+Z. If the component c_i does not pose interference with the component c_j in the direction of +X, then DM (i, j) value of +X column will be equal to 0, otherwise 1. Conversely, if the component c_i does not pose interference with the component c_j in the direction of –X, then DM (j, i) value of +X column will be equal to 0, otherwise 1. A typical Disassembly Matrix (DM) has been presented in Table 1 for a four component assembly, shown in Figure 1 (Wang et al. [4]).

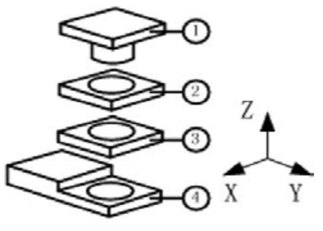


Figure 1: Four component assembly

Table 1: Disassembly Matrix for Four component assembly

Directions➧	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
1	0	0	0	1	1	0	1	1	0	1	1	0
2	1	1	1	0	0	0	0	0	0	0	0	0
3	1	1	1	0	0	1	0	0	0	0	0	0
4	1	1	1	0	0	1	0	0	1	0	0	0

The problem of assembly sequence optimization has been modeled using a graph, with each node representing a component number and its disassembly direction. In ACO, first of all, using the DM it is necessary to identify those components which can be used to start the disassembly operation. The number of these components decides the number of ants for the ACO algorithm. These ants are placed on the starting nodes and from the START node, the ants start moving towards the END node, gradually connecting the intermediate nodes. The next node of the graph is selected by the ants using a probability function given in eq. 1. The probability with which ant k in DO i chooses DO j is calculated as follows.

$$p_k\,(i,j) = \begin{cases} \frac{\tau(i,j)[\eta(i,j)^{\beta}]}{\sum_{u\in C_k}\tau(i,j)[\eta(i,j)^{\beta}]}, if\ j\ \in C_k(i) \\ 0,\ Otherwise \end{cases} \qquad (1)$$

The ants release some pheromone while traversing the path; this information is passed onto subsequent generations using a matrix, namely Pheromone Matrix (PhM). During the course of algorithm, this PhM is updated using global and local updating rule. Gradually, the pheromone on the paths that are less traversed by ants is evaporated because of more distance, and the pheromone on the more traversed paths concentrates since they are traversed by more number of ants due to shorter distance. This is how the shortest path is selected, which refers to the feasible and optimal solution. The ACO algorithm has number of parameters such as Q, ρ (rate of pheromone evaporation), and gamma (pheromone decay parameter), and the number of iterations required for convergence. The values of the above parameters need to be varied in order to obtain the best performance of the algorithm, and furthermore, the combination of the parameters also needs to be optimized, which if done manually, may involve lot of time and efforts. Keeping the above in mind, a binary coded GA has been proposed here to optimize ACO parameters, details of which are given in section 2.1.

3 PROPOSED APPROACH FOR PARAMETER META-OPTIMIZATION OF ACO BY GA

The GA consists of three steps: 1) Selection, 2) Crossover, and 3) Mutation. The first is to encode the solutions in the form of chromosomes. In the present paper, the binary coded GA has been proposed. A typical chromosome of GA is [0 1 0 1 0 0 0 0 1]. The length of each chromosome is the summation of length of each parameter when converted into binary string. The initial population for GA is generated randomly. A typical population of chromosomes of population size "3" is as follows: [0 1 0 1 0 0 1 1 0; 1 0 1 0 0 1 0 1 0; 0 1 1 1 0 0 0 1 0].
The selection operator is used to choose the good quality chromosomes. In the present work, Tournament Selection has been applied to generate the mating pool. It provides selection pressure by holding a tournament among "s" competitors, i.e. tournament size. The winner of the tournament is the individual with the highest fitness. The winner is then inserted into the mating pool. The mating pool, being comprised of tournament winners, has a higher average fitness than the average population fitness. Increased selection pressure can be provided by simply increasing the tournament size "s", as the winner from a larger tournament will, on average, have a higher fitness than the winner of a smaller tournament (Miller and Goldberg [13]). After making the mating pool, the population individual undergoes the crossover operation, based on a probability, known as crossover probability. A simple 2-point crossover has been used to generate the offsprings possessing the mix quality of their parents. Then, based on the mutation probability, the population of GA undergoes the mutation. A simple swap mutation operator has been used. After this process, the offsprings are evaluated again by converting the binary chromosomes into the real parameter values, followed by running the ACO algorithm ten times. The fitnesses of offsprings are compared with their parents, which if improved, are retained and carried over to next generation, otherwise they are discarded. The GA cycle is repeated number of times, until the prespecified number of iterations of GA is completed. For the conversion of binary values of GA to real values of ACO parameters, the entire binary string of GA chromosome is divided into substrings, where each substring represents different ACO parameters namely, Q (a constant number used to adjust the increment of pheromone), ρ (rate of pheromone evaporation), gamma (pheromone decay parameter) and the number of iterations required for convergence. Wang et al. [4] suggests that the Q should be a discrete number and the value of ρ and gamma should lie between 0 and 1. The binary substring is then converted by using the formulae given in eq. 2 to its decimal equivalent which represents the actual parameter value of the ACO algorithm.

$$\text{actual number of iteratons for ACO} = \text{round}\left(\text{minimum number of iteratons} + \frac{(\text{maximum number of iteratons} - \text{minimum number of iteratons})}{\left(2^{length\ of\ substring\ for\ number\ of\ \text{iteratons}}-1\right)}\right) \quad \mathbf{(2)}$$

Likewise, the actual value of Q, ρ, and gamma can also be determined. These values are then used as parameters of the ACO algorithm. The following fitness function of GA has been taken into account considering the average fitness provided by all the ants in the last iteration. The fitness function, FF can be thus given as follows:

$$FF = mean\, fitness\, of\, all\, simulation\, runs\, of\, ACO$$

The flow chart of the GA based parameter meta-optimization of ACO algorithm is given in Figure 2.

4 RESULTS AND DISCUSSIONS

An example of a punching machine assembly consisting of 16 components as shown in Figure 3, has been considered here to demonstrate the application of the proposed approach for parameter meta-optimization of ACO algorithm. Figure 4 presents an extract of the disassembly matrix, which is a 16 x 48 matrix for the given 16 component assembly. Here, the GA is used to determine the best parameter combination for ACO algorithm for solving the given assembly sequence optimization problem. The main parameters which affect the performance of GA are population size, crossover and mutation probability. A sensitivity analysis of the three GA parameters was performed to determine the influence of these parameters on the optimal solution given by GA. The following ranges of variation of different ACO algorithm parameters have been assumed: Q value is varied between 1 and 8, ρ value is varied between 0.1 and 0.8, gamma value is varied between 0.1 and 0.8 and the number of iterations is varied between 50 and 190. To study the influence of GA crossover probability, it was varied from 0.5 to 0.95 in steps of 0.05 keeping the mutation probability fixed at 0.1 and population size fixed at 5. Each binary chromosome of GA yields a particular combination of real parameters of the ACO algorithm. Using this parameter combination, the ACO algorithm has been run ten times to evaluate the fitness each GA chromosome, and then their average fitness value over these ten runs has been calculated. This average indicates the actual performance of the ACO algorithm for a certain parameter combination. Each GA simulation is carried out for 50 iterations. Figure 5 shows the variation of fitness value of GA with the crossover probability. It is observed that a crossover probability of 0.7 gives the optimal fitness value. Next, to study the influence of mutation probability, it was varied from 0.05 to 0.15 in steps of 0.025 keeping the crossover probability fixed at 0.7 and population size fixed at 5. Figure 6 shows the variation of fitness value with the mutation probability. It is observed that an optimal fitness value is obtained for mutation probability of 0.1. We further increased the population size of GA from 5 to 10. However, there was no improvement of the optimal solution given by GA. Therefore from the above results, it may be concluded that the optimum GA parameters are crossover probability of 0.7 and mutation probability of 0.1, when we use a GA population size of 5 and the corresponding optimum parameters of the ACO algorithm are as follows: Q = 7, ρ = 0.1, gamma = 0.2, number of iterations = 170. Figure 7 shows the convergence plot of the GA when it was run with the above set of optimal parameters. Figure 8 shows the convergence plot of the ACO algorithm for the given assembly sequence optimization problem when the ACO algorithm was run with the above set of optimal parameters recommended by the proposed GA based meta-optimization approach. The optimal disassembly sequences along with their directions given by the meta-optimized ACO algorithm are as follows. [(12, -Y) (11, -Y) (14, Y) (13, Y) (2, -X) (9, Z) (15, Z) (8, Z) (7, Z) (5, Z) (6, Z) (10, Z) (16, Z) (4, Z) (3, Z) (1, Z)] and [(13, Y) (14, Y) (12, -Y) (11, -Y) (2, -X) (9, Z) (6, Z) (7, Z) (5, Z) (8, Z) (10, Z) (16, Z) (15, Z) (3, Z) (4, Z) (1, Z)]. Each of these sequences requires three direction changes.

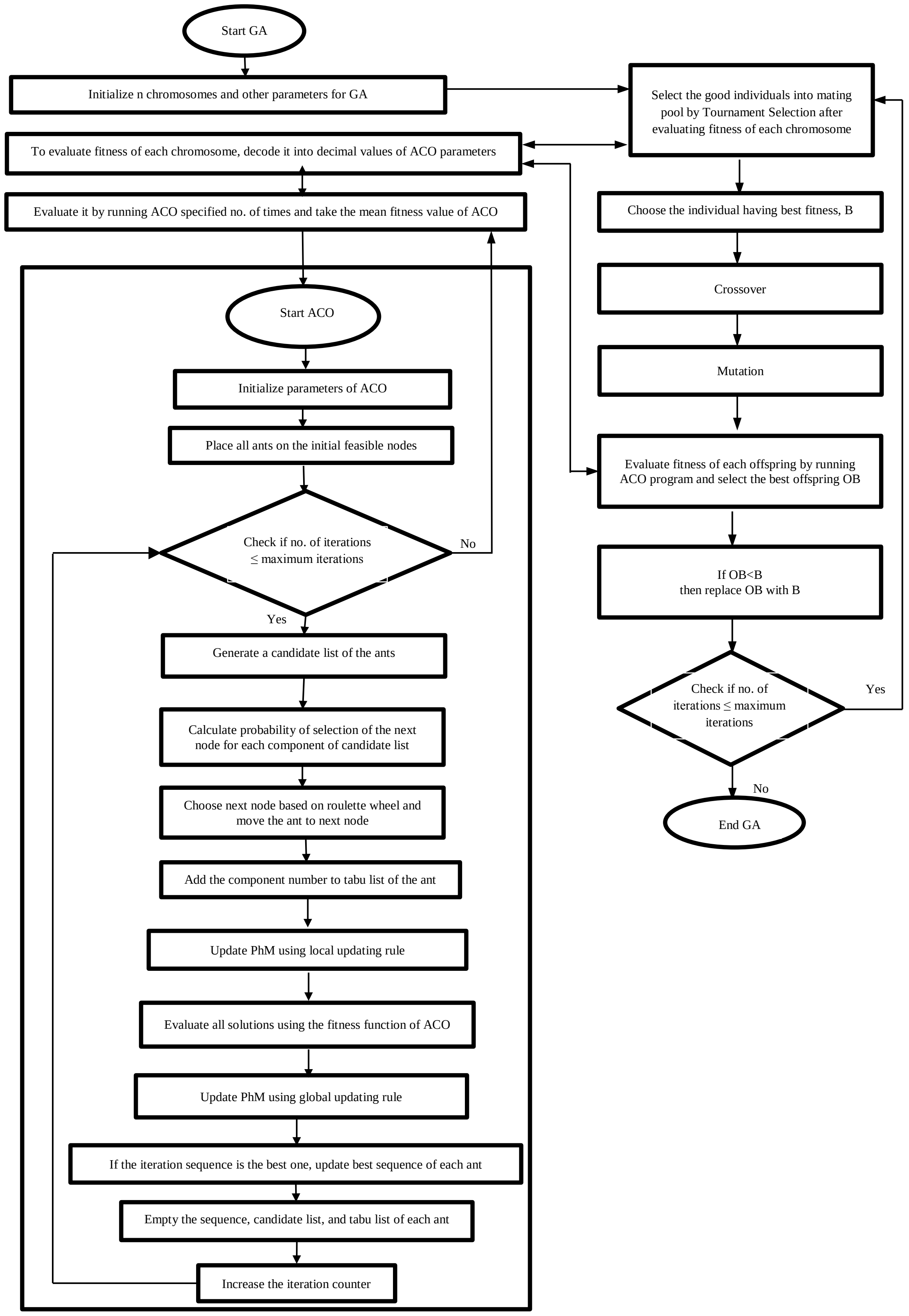

Start GA
Initialize n chromosomes and other parameters for GA
To evaluate fitness of each chromosome, decode it into decimal values of ACO parameters
Evaluate it by running ACO specified no. of times and take the mean fitness value of ACO
Start ACO
Initialize parameters of ACO
Place all ants on the initial feasible nodes
Check if no. of iterations ≤ maximum iterations
No
Yes
Generate a candidate list of the ants
Calculate probability of selection of the next node for each component of candidate list
Choose next node based on roulette wheel and move the ant to next node
Add the component number to tabu list of the ant
Update PhM using local updating rule
Evaluate all solutions using the fitness function of ACO
Update PhM using global updating rule
If the iteration sequence is the best one, update best sequence of each ant
Empty the sequence, candidate list, and tabu list of each ant
Increase the iteration counter
Select the good individuals into mating pool by Tournament Selection after evaluating fitness of each chromosome
Choose the individual having best fitness, B
Crossover
Mutation
Evaluate fitness of each offspring by running ACO program and select the best offspring OB
If OB<B then replace OB with B
Check if no. of iterations ≤ maximum iterations
Yes
No
End GA

Figure 2: Flowchart of the proposed Meta-optimization algorithm

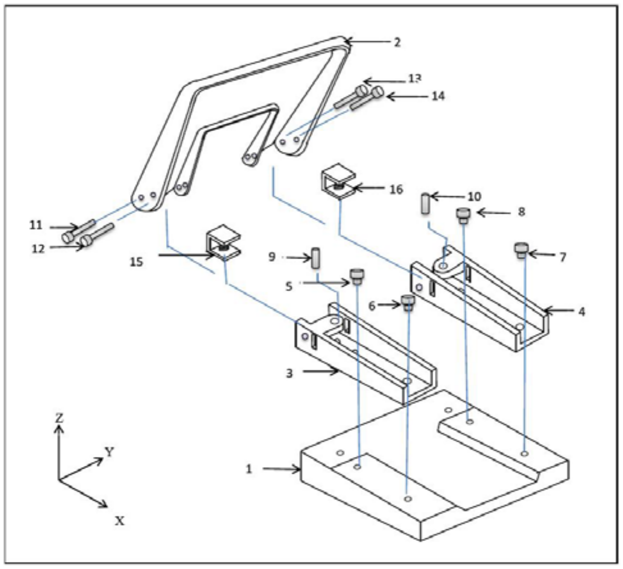
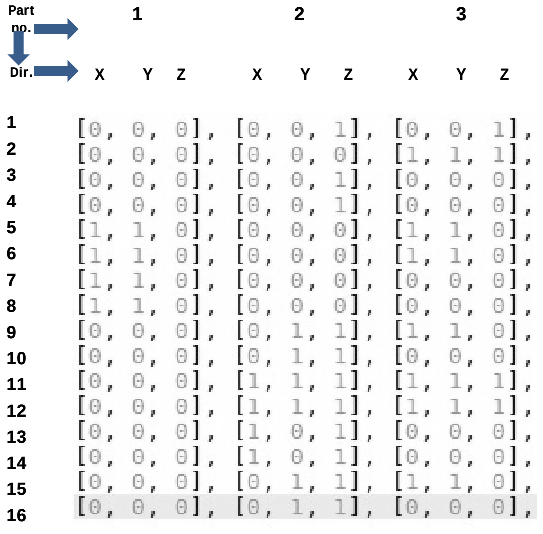
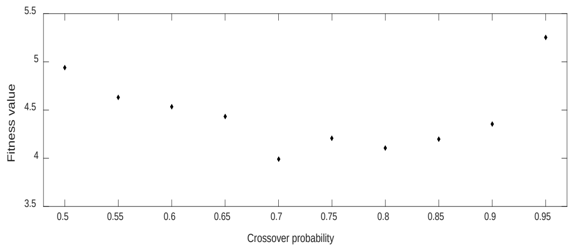


Figure 3: Punching Machine assembly



Part no.	1			2			3		
Dir.	X	Y	Z	X	Y	Z	X	Y	Z
1	[0,	0,	0],	[0,	0,	1],	[0,	0,	1],
2	[0,	0,	0],	[0,	0,	0],	[1,	1,	1],
3	[0,	0,	0],	[0,	0,	1],	[0,	0,	0],
4	[0,	0,	0],	[0,	0,	1],	[0,	0,	0],
5	[1,	1,	0],	[0,	0,	0],	[1,	1,	0],
6	[1,	1,	0],	[0,	0,	0],	[1,	1,	0],
7	[1,	1,	0],	[0,	0,	0],	[0,	0,	0],
8	[1,	1,	0],	[0,	0,	0],	[0,	0,	0],
9	[0,	0,	0],	[0,	1,	1],	[1,	1,	0],
10	[0,	0,	0],	[0,	1,	1],	[0,	0,	0],
11	[0,	0,	0],	[1,	1,	1],	[1,	1,	1],
12	[0,	0,	0],	[1,	1,	1],	[1,	1,	1],
13	[0,	0,	0],	[1,	0,	1],	[0,	0,	0],
14	[0,	0,	0],	[1,	0,	1],	[0,	0,	0],
15	[0,	0,	0],	[0,	1,	1],	[1,	1,	0],
16	[0,	0,	0],	[0,	1,	1],	[0,	0,	0],

Figure 4: An extract of the Disassembly Matrix for the Punching Machine assembly



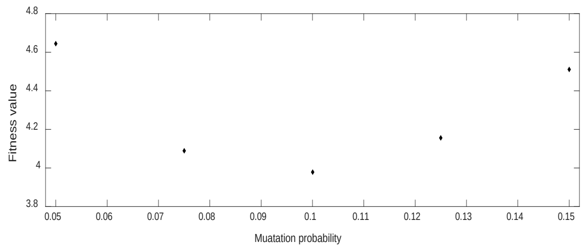
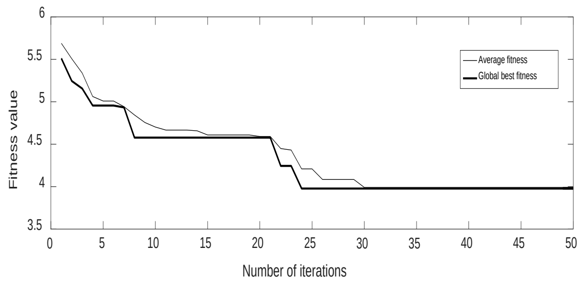


Figure 5: Variation of fitness value of GA with the crossover probability

Figure 6: Variation of fitness value of GA with the mutation probability



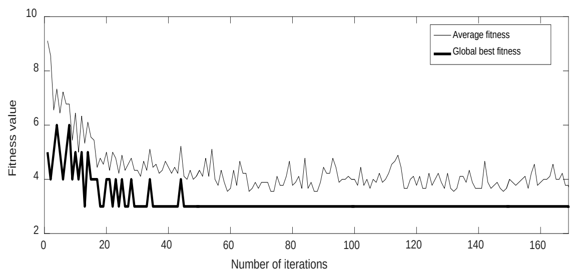


Figure 7: Convergence plot of GA used for meta-optimization of ACO

Figure 8: Convergence plot of ACO for assembly sequence optimization

The corresponding assembly sequences can be obtained by reversing the disassembly sequences, which are as follows. [(1,-Z) (3, -Z) (4, -Z) (16, -Z) (10, -Z) (6, -Z) (5, -Z) (7, -Z) (8, -Z) (15, -Z) (9, -Z) (2, X) (13, -Y) (14, -Y) (11, Y) (12, Y)] and [(1, -Z) (4, -Z) (3, -Z) (15, -Z) (16, -Z) (10, -Z) (8, -Z) (5, -Z) (7, -Z) (6, -Z) (9, -Z) (2, X) (11, Y) (12, Y) (14, -Y) (13, -Y)]. It should be noted that the directions of the components would have to be reversed. The above assembly sequences are found to be feasible as they are in accordance with the precedence constraints. The global best fitness found at the end of all iterations of ACO is 3 i.e. the minimum number of direction changes required by optimal assembly sequence is 3. The average fitness value of all ants at the end of all iterations of ACO was found to be 3.9911.

5 CONCLUSIONS

In the present paper, an ACO algorithm based approach has been used for assembly sequence optimisation based on minimizing the number of direction changes, while conforming to the precedence constraints between components. Further, a binary coded GA has been proposed for meta-optimization of ACO in order to optimize the different ACO parameters such as rate of pheromone evaporation, pheromone decay parameter, number of iterations, etc. Some of the reasons for choosing GA include the fact that the number of parameters in GA is less than ACO and thus it takes less time and efforts to optimise the GA than the ACO. Moreover the modeling of the solutions (i.e. chromosomes) in the form of binary strings is easier. The optimized ACO parameters have been used for solving a problem of assembly sequence optimization for a sixteen component assembly to demonstrate the effectiveness of our proposed meta-optimization procedure. The future scope of work will include considering some other criteria for optimization such as stability, minimizing assembly tool changes to reduce handling time, etc.

REFERENCES

[1] Hong, D S. Cho, H S. 1993. Optimization of robotic assembly sequences using neural network, IEEE/RSJ Int Conf on Intelligent Robots and Systems, Japan, July 26-30, pp. 232-239.
[2] Bonneville, F. Perrard, C. Henrioud, J M. 1995. A genetic algorithm to generate and evaluate assembly plans, INRIA/IEEE Symposium on Emerging Technologies & Factory Automation, France, October 10-13, pp 231-239.
[3] Chen, R S. Lu, K Y. Tai, P H. 2004. Optimizing assembly planning through a three-stage integrated approach, *International Journal of Production Economics*, 88, pp 243–256.
[4] 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, pp 1137–1143.
[5] Cao, P B. Xiao, R B. 2007. Assembly planning using a novel immune approach. *Int J Adv Manuf Technol*, 31(7), pp 770–782.
[6] Chang, CC. Tseng, HE. Meng, LP. 2009. Artificial immune systems for assembly sequence planning exploration, *Engineering Applications of Artificial Intelligence*, 22, pp 1218–1232.
[7] Wang, Y. Liu, JH. 2010. Chaotic particle swarm optimization for assembly sequence planning. *Robot Computer Integrated Manufacturing*, 26(2), pp 212–222.
[8] Lv, HG. Lu, C. 2010. An assembly sequence planning approach with a discrete particle swarm optimization algorithm, *Int J Adv Manuf Technol* 50(5–8), pp 761–770.
[9] Gao, L. Qian, W R. Li, X Y. Wang, J F. 2010. Application of memetic algorithm in assembly sequence planning, *Int J Adv Manuf Technol* 49(9–12), pp 1175–1184.
[10] 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, *Int J Adv Manuf Technol*, 52(5–8), pp 715–724.
[11] Xing, Y F. Wang, Y S. 2012. Assembly sequence planning based on a hybrid particle swarm optimisation and genetic algorithm, *Int Journal of Prod Research*, 50(24), pp 7303–7312.
[12] Li, X. Qin, K. Zeng, B. Gao, L. Su, J. 2016. Assembly sequence planning based on an improved harmony search algorithm, *Int J Adv Manuf Technol*, 84, pp 2367-2380.
[13] Miller, B L. Goldberg, D E. 1995. Genetic Algorithms, Tournament Selection, and the Effects of Noise, *Complex Systems*, 9, pp 193- 212.