AN ENHANCED GENETIC ALGORITHM FOR ASSEMBLY PLANNING

Dev Anand M.[1], Kumanan S.[2], Girish RR.[3], Selvaraj T.[2] and Asokan P.[2]

[1]Noorul Islam Centre for Higher Education, Thuckalay, Tamilnadu, India
[2]National Institute of Technology, Tiruchirappalli - 620 015, Tamilnadu, India
[3]Rajalakshmi College of Engineering, Tamilnadu, India
E-Mail: anandpmt@yahoo.co.in

Abstract

Assembly planning is very important for competitive manufacturing where assemble-to- order of products is in-practice. Assembly planning is a complex task and an optimal assembly plan is detrimental to meet customer demands. This work presents a genetic algorithm for assembly planning. This problem is more difficult than other assembling problems that have already been tackled with success using these approaches, such as the classic Traveling Salesperson Problem (TSP) or the Job Shop Scheduling Problem (JSSP). It not only involves arranging of tasks, as in those problems, but also the selection of them from a set of alternative operations. Random search methods are being attempted for these types of combinatorial problems. Thus, many current research reports describe efforts to develop more efficient planning algorithms. Genetic algorithms show particular promise for assembly planning. As a result, several recent research reports present assembly planners based upon traditional genetic algorithms. Although prior genetic assembly planners find improved assembly plans with some success, they also tend to converge prematurely at local-optimal solutions. Thus, we present an assembly planner, based upon an enhanced genetic algorithm that demonstrates improved searching characteristics over an assembly planner based upon a traditional genetic algorithm. In particular, our planner finds optimal or near-optimal solutions more reliably and more quickly than an assembly planner that uses a traditional genetic algorithm.

Keywords: Genetic algorithm, Assembly planning, Liaisons graph, Task scheduling, Assembly robots, Optimization problem

I. INTRODUCTION

Increasing competition has forced the manufacturers to seek ways and means of cutting down the production costs, improving the product quality, and reducing the manufacturing lead times. Assembly process is significant in manufacturing and its planning is a combinatorial problem: The number of assembly plans is a factorial function of the number of components in the product. The selection of assembly plans has a bearing effect on the final product. The choice of the assembly sequence in which parts or subassemblies are put together in the mechanical assembly of a product can drastically affect the efficiency of the assembly process. Hence an efficient assembly plan, greatly determines lead-time, production cost, and, thus, potential product success. The assembly-planning problem involves the identification, selection and sequencing of assembly operations stated with their effects on the parts [1]. The identification of assembly operations usually leads to the set of all feasible assembly plans. Besides it also relies on other factors like how the single parts are interconnected in the whole assembly, i.e. the structure of the graph of connections. Two kinds of approaches have been used for searching the optimal assembly plan. The qualitative approach uses the rules in order to eliminate assembly plans that include difficult tasks or awkward intermediate sub-assemblies. A quantitative approach uses an evaluation function that computes the merit of assembly plans. The past research reveals two classes of algorithm could be grouped as classical and non-traditional algorithms.

A. Classical algorithm

The need for opportunistic scheduling was first addressed [2] for robotic assembly dealing with the partial order representation for assembly plans. At first, interactive planners queried the user for geometric-reasoning information [3] [4]. Later, planners work automatically from a geometric and relational model of the assembly [5]. The researcher [6] presented And/Or Graph, for a complete representation of all possible assembly plans by decomposition based on cut-set approach. They dealt and showed how to plan repair sequences using the And/Or graph representation of assembly plans [7]. Two criteria was presented by Luiz et al [8] for the selection of assembly plans: Maximizing the flexibility of sequencing the assembly tasks and Minimizing the assembly time

through parallel execution of assembly task which provided a basis to reduce the makespan. They reviewed [9] on representations of mechanical assembly sequences. Randall et al [10] elaborated the problem of generating the assembly sequences from the geometry of a goal assembly with a model called GRASP, stands for Geometric Reasoning Assembly Sequence Planner. Recent research is directed towards non-traditional algorithms like Genetic Algorithm (GA) owing to its simplicity in representation and efficient in finding near optimal solutions.

B. Non-Traditional Algorithm

The Genetic Algorithms (GAs) are one of the most widely used techniques. Basically, GAs is optimization methodologies based on a direct analogy to Darwinian natural selection and genetics in biological systems. They can deal with complex product assembly planning. However, during the process, the neighborhood may converge too fast and limit the search to a local optimum prematurely [11]. Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions that are selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied. Some problem-based heuristics have been used for generating the individuals in the population [12]. The effect of a local search on the performance of the Genetic Algorithm in terms of solution quality, convergence and computation time is also investigated [13].The influence of tolerance and clearance on product assimilability in different assembly sequences is considered and used as a constraint in assembly planning [14].

An appropriately modified version of the well-known partially matched crossover, and purposely defined mutation operators allow the algorithm to produce near-optimal assembly plans starting from a randomly initialized population of (possibly non-feasible) assembly sequences [15].The use of the Ordering Genetic Algorithms (OGA) such a method avoids the aggregation of several technical criteria into a unique fitness value, and lets us compares the individuals of the population one to each other [16].The feasibility of using a Genetic Algorithm (GA) for generating and evaluating assembly planning was introduced and demonstrated by Ames et al. [17]. They observed a notable comparison between GA with the cut-set approach for generating assembly plans. Bonneville et al applied genetic algorithm to assembly sequence planning with limited resources [18]. The main contribution of the method is the incorporation of the knowledge provided by the specific heuristics of the problem in a local search procedure. An assembly planning based on genetic algorithm was addressed by Bautista et al, incorporating certain criteria to assess the quality of feasible assembly sequences, like minimizing the orientation changes, the gripper changes etc [19]. Del valle et al [1] presented a two algorithmic approach for task scheduling problem: GA intended for the earlier stages and A* algorithm for the final ones. An optimum assembly plan is now sought, selected from the set of all feasible assembly plans. This model proposes a genetic algorithm which is different from the one presented in [20] based on the plans provided by an expert, as it becomes obsolete when comes to new product development. The development of an efficient and robust Genetic algorithm for assembly planning needs attention. This paper proposes a Genetic Algorithm (GA) application for generating optimal assembly plans.

II. PROBLEM DEFINITION

The process of joining parts together to form a unit is known as assembly. The joining process results in the connection of one part with parts already assembled. A sub- assembly is a group of parts having the property of being able to be assembled independently of other parts of the product. An assembly plan is a set of assembly tasks with ordering amongst its elements. Each task consists of joining a set of sub-assemblies to give rise to an ever-larger sub-assembly. An assembly sequence is an ordered sequence of the assembly tasks satisfying all the ordering constraints. Each assembly plan corresponds to one or more assembly sequences. The work is focused on choosing an optimal or near-optimal assembly plan. The geometric constraints imposed by product are represented by liaison graph. A feasible assembly plans are prepared based on precedence matrix. The assumptions involved in this work are: Exactly two parts or sub-assemblies are joined at each time, whenever parts are joined forming a

subassembly, all contacts between parts in that subassembly are established, the feasibility of joining two subassemblies is independent of how those sub-assemblies were built.

III. SOLUTION METHODOLOGY

The proposed methodology is depicted as a flow chart shown in Figure1. The Product data is conceived and a liaison graph is built based on the connection of parts i.e. geometric constraints and their ordering amongst them. The precedence relations are derived from the constraints imposed on a component when located in the final assembly. A component can be either completely constrained or partially constrained, when the component is disassembled along at least one mating direction. For a component that is completely constrained, precedence relations exist such that the component has to be in place before some other set of components. Next, identify the number of possible assembly pair's, which represents assembly task. With this, construct a precedence matrix for generating initial population for the Genetic Algorithm. A purposely-developed crossover and mutation is used to manipulate the chromosome, representing the solution for the formulated problem.

A. Product modeling

The first stage to search the assembly plans for a given product is to have a good representation of the product. This is obtained from liaison graph or graph of connection. It's a relationship between two parts which are touching or effectively touching, whether physically attached or not". In Liaison graph parts are dots and joints are lines. A product '*P*' built from a set of component '*C*' is modeled by its liaisons graph $[C, \Gamma]$, where 'Γ' is the set of the mechanical liaisons that exist between the components. The Figures 2 shows a mechanical product in its exploded form and the liaisons graph. An oil pump assembly constitutes five components (node) linked by seven

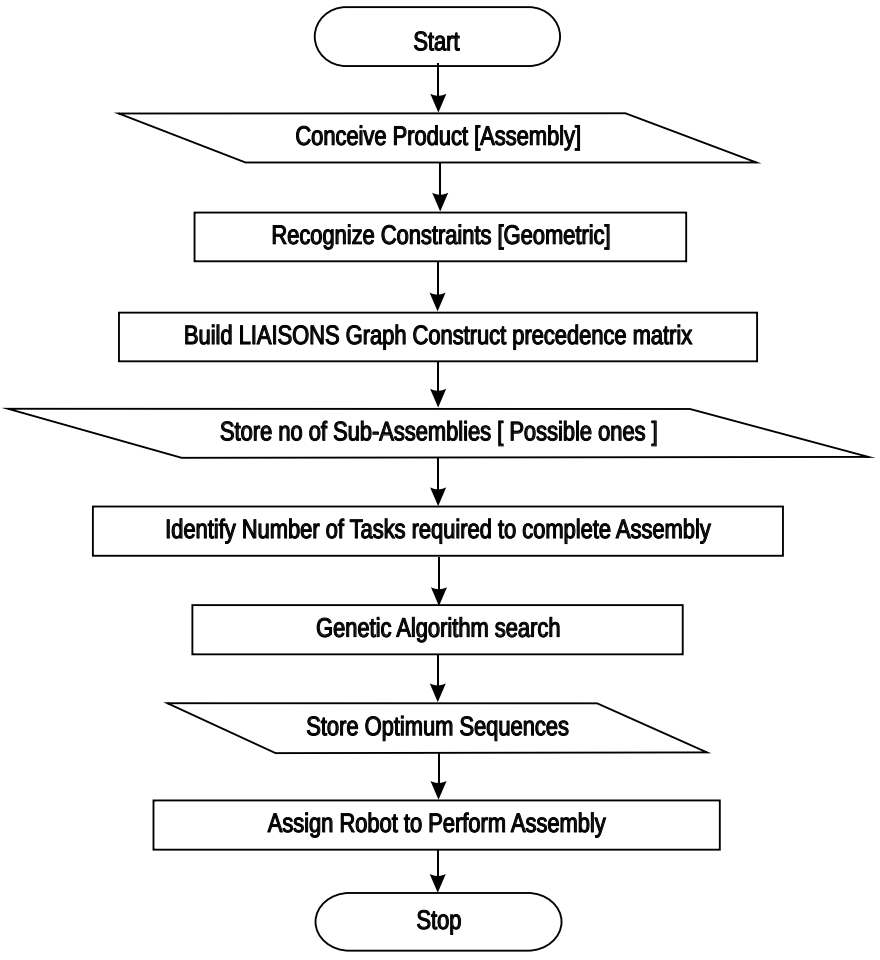
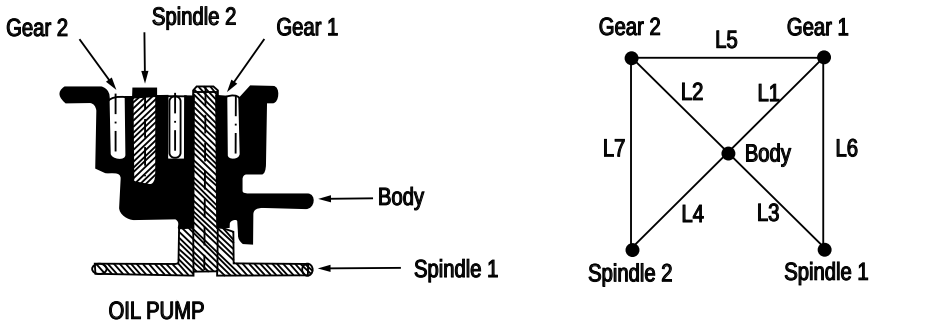


Fig. 1. Flow Chart Showing the Complete Process Description.



L1 = Body - Gear 1
L2 = Body - Gear 2
L3 = Body - Spindle 1
L4 = Body - Spindle 2
L5 = Gear 1 - Gear 2
L6 = Gear 1 - Spindle 1
L7 = Gear 2 - Spindle 2

LAISONS GRAPH [C,]
c= {Body,Spindle 1,Spindle 2, Gear 1, Gear 2 }
= { L1,L2,L3,L4,L5,L6,L7 }

Fig. 2. A Mechanical Product and its Liaisons Graph.

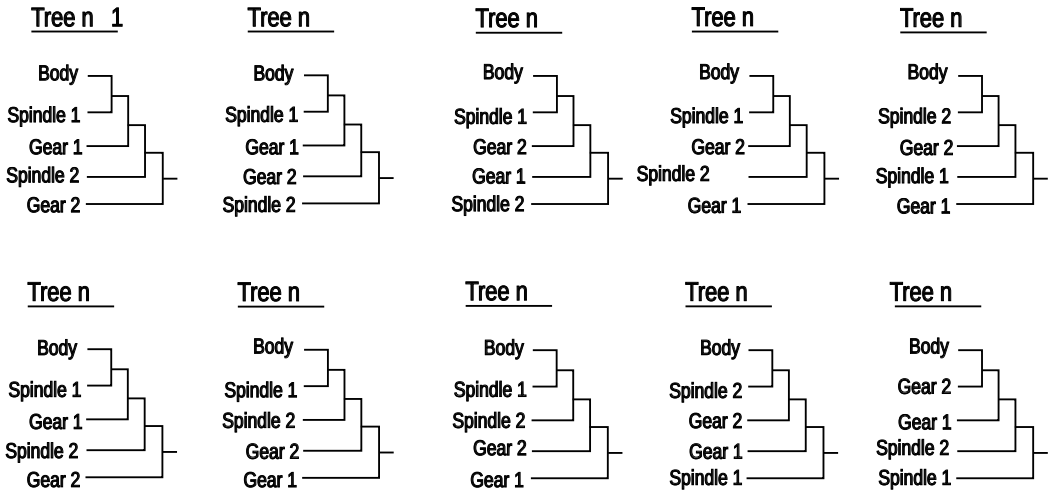


Fig. 3. Some Assembly Trees of the Oil Pump.

liaisons (connections). Once the liaisons graph is built, possible assembly pairs are derived from the graph. These pairs represents task, which are required to generate feasible assembly plans. Prior to this, it is necessary to become aware of how many levels are required to generate plans or needed to complete final product. At each level an assembly task is performed to carry on further level. An assembly tree for a product '*P*' is a directed tree whose root represents the product '*P*' whose leaves represent the components of '*P*', and whose intermediate nodes represent the intervening subassemblies produced by the process. The Figure 3 depicts 10 of the 40 valid assembly plans of the oil pump assembly.

B. Proposed genetic algorithm

The description of the genetic algorithm concepts for the assembly-planning problem is presented in this section. A flow chart is outlined in the figure 4 depicts an overall idea of the steps needed to design such an algorithm for the generation of assembly plans.

Generation of an initial population: The first step in developing a genetic algorithm for assembly planning is to map the problem solutions (assembly sequences) to chromosomes. The chromosome must be encoded in such a way that it contains the information about the solution which it represents. The initial population is generated randomly from the group of precedence

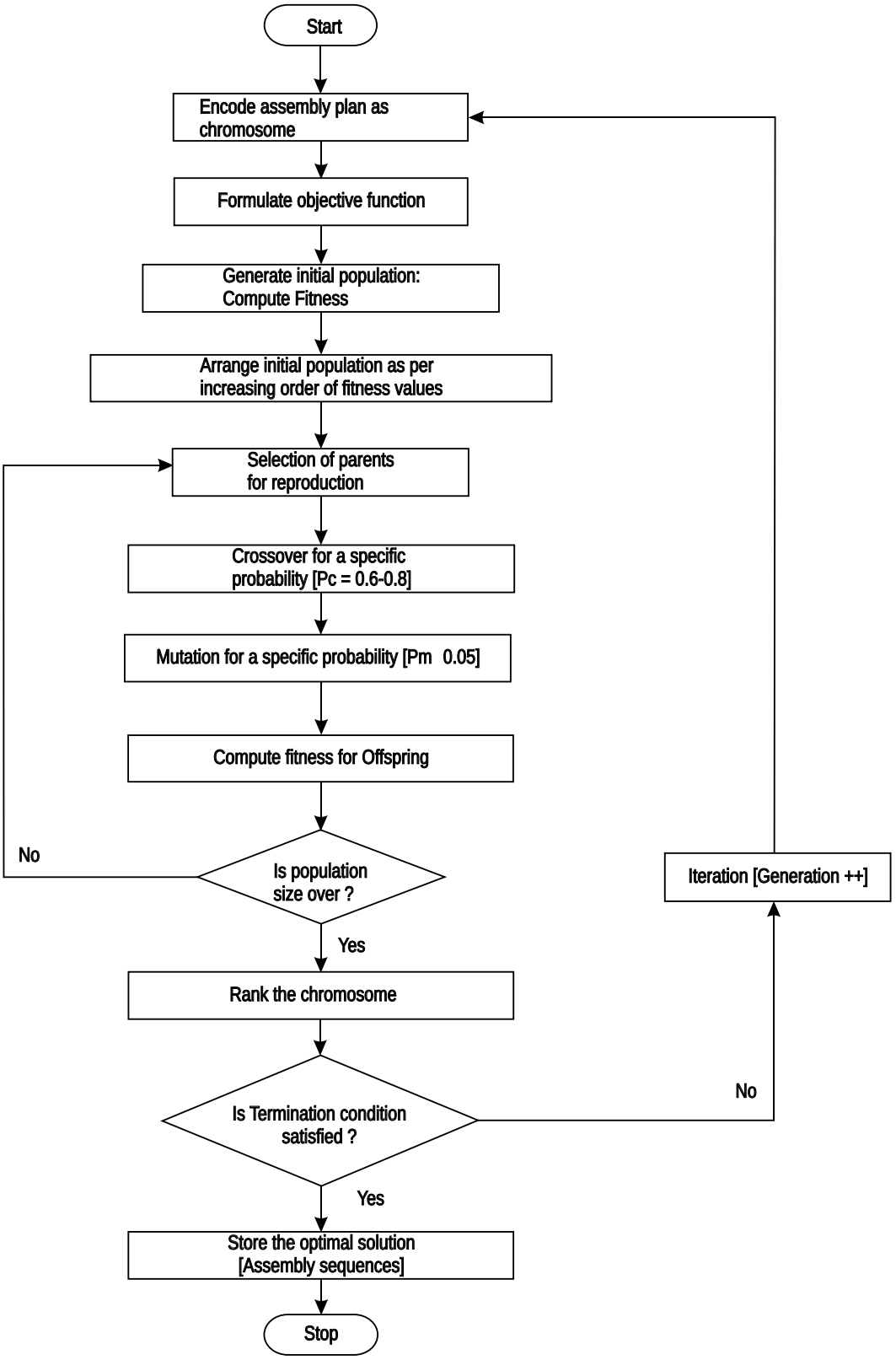


Fig. 4. Flow Chart Proposed GA for the Assembly Task Scheduling.

matrix and the population size is set to '*n*'. Hence '*n*' strings will be created randomly. Phenotypic coding is used to represent the chromosome as shown in figure 5. The each node corresponds to a gene that represents tasks number in a particular level, required to build an assembly along with other tasks in that particular levels of the plan.

The chromosomes, which represent the assembly plans, are checked for feasibility from the precedence matrix before undergoing genetic operation. In the figure, genes 3, 14, 25, 27, 28 and 31 are required to complete final assembly. All the chromosomes have the same length. It means that the number of levels is same for building a complete final product, such that in each level one task will be completed.

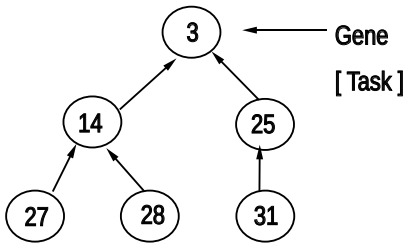


Fig. 5. A Typical Encoded Chromosome [Assembly Tree].

Objective function: The objective function is used to provide a measure of how individuals have performed in the problem domain. The objective function for the problem is to obtain an optimum or near-optimal assembly plans based on the minimization of the makespan for a given product. The objective function is defined as follows

Where

$$m = P - 1$$

$$P_t(n) = OC(n) + DoF(n) \quad ...(1)$$

$$C_t(AP) = \sum_{i}^{m} P_t(i) \quad ...(2)$$

$$MS = Min\,[C_t]_j \quad ...(3)$$

M : Number of tasks in each assembly Tree

P : Number of parts

$P_t(n)$: Processing time for the task '**n**' as *i* denote number of task required to complete a final product.

n : Number of tasks represents an assembly pair obtained from the precedence relations of liaisons graph.

$OC(n)$: Operational complexity

$DoF(n)$: Subassembly Degrees of freedom

C_{tj} : Completion time for each Assembly plan as 'j' denotes population size usually 20.

MS : Makespan for each number of Assembly plans

The makespan for this purpose is taken similarly as in [4] consists of two factors:

(i) Operational complexity: It takes in to consideration of the type of assembly operation with the Screw operation weighting as 4, Insertion as 2 and Placement as 1, in accord with typical time, fixturing and manipulation requirements.

(ii) Subassembly degrees of freedom: It takes in to account the difficulty in handling the participating subassemblies and it proportional to their number of degrees of freedom. Subassemblies with more degrees of freedom are unstable and therefore more difficult to handle.

Evaluate the fitness function: The fitness function is normally used to transform the objective function value in to a measure of relative fitness. Since the objective is a minimization problem, the objective function value itself is used as fitness function. The fitness function must be calculated for the initial set population and after performing genetic operation so as to evaluate the fitness for each string. The significance of evaluating the fitness value for each string provides information to decide whether the strings are carried for next iteration (generation) to obtain the desired result.

Selection for reproduction: There are various methods to select the chromosomes for reproduction. Some are Roulette wheel selection, Boltzman selection, Tournament selection, Rank selection, Steady state selection etc. But in this work the Rank selection method is utilized.

Rank selection: Once after an initial population is generated, fitness value is computed for each string in that population. This leads the chromosomes having an equal chance to be selected. But, this method leads to slower convergence, because; the best chromosomes do not differ much from the other ones. Selection is the process of determining the number of times, or trials, a particular individual are chosen for reproduction and, thus, the number of offspring that an individual will produce. The selection of individuals can be viewed as two separate processes:

1. Determination of the number of trials an individual can expect to receive.
2. Conversion of the expected number of trials into a discrete number of offspring.

Genetic operator Two families of genetic operators purposely have been developed for searching the whole solution space. The first includes operators that search locally for new sequences from parent chromosomes. The other family of operators is intended to search for sequences in assembly plans of parents as well as the crossbreed chromosomes. This is basically made by introducing a new task in a solution, and substituting certain tasks in order to maintain the validity of the chromosomes.

Crossover: The basic operator for producing new chromosomes in the GA is that of crossover. Like its counterpart in nature, crossover produces new individuals that have some parts of both parent's genetic material. The simplest form of crossover is that of single-point crossover. In this approach strings are subjected to crossover based on the specified crossover probability. The probability of crossover is usually high and ranges from 0.6 to 0.8 and this has been considered in the problem. Once the string is chosen for crossover, its mate and crossover site are selected randomly. But in this problem, the crossover point is fixed to 2, as to maintain the precedence relationship among the task. This results in the formation of offspring retaining its feasibility. Hence in this assembly problem a '*Selective Crossover*' is utilized to perform the crossover operation between the particular group of strings i.e. assembly trees.

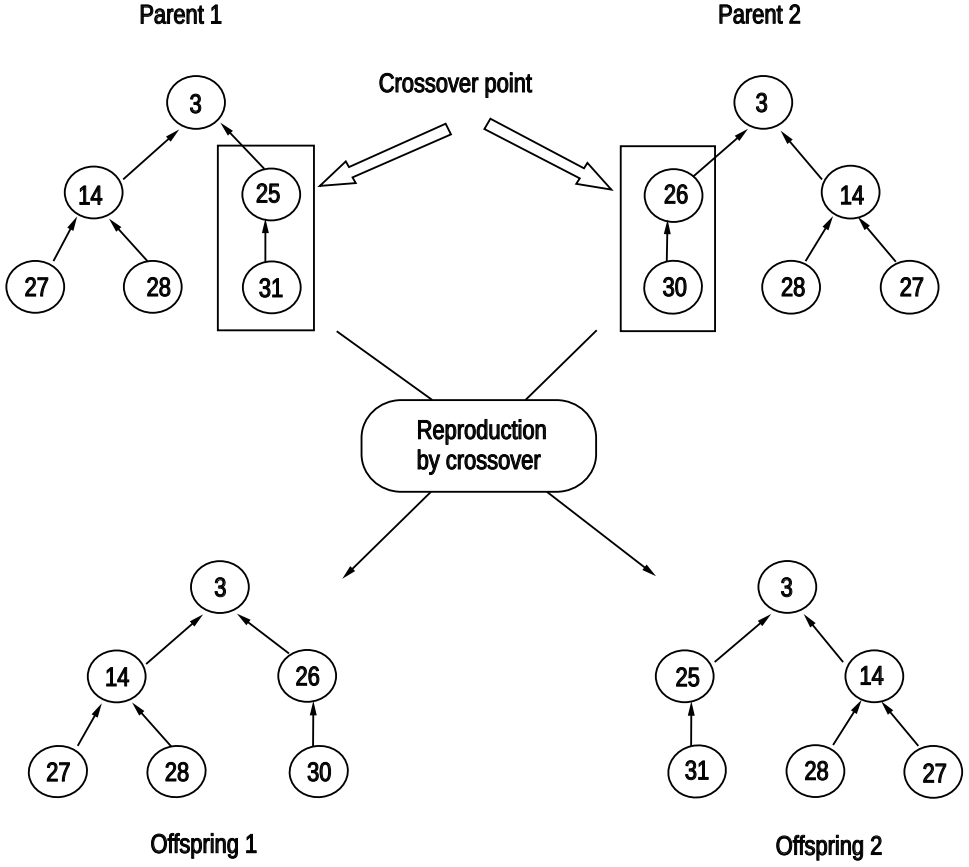


Fig. 6. Proposed Crossover for the Encoded String.

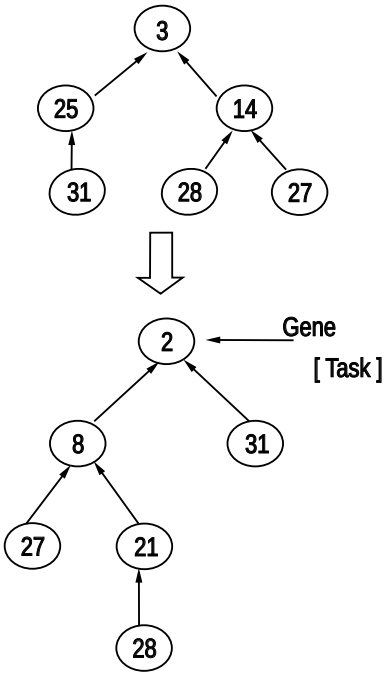


Fig. 7. Proposed Mutation for the Encoded String

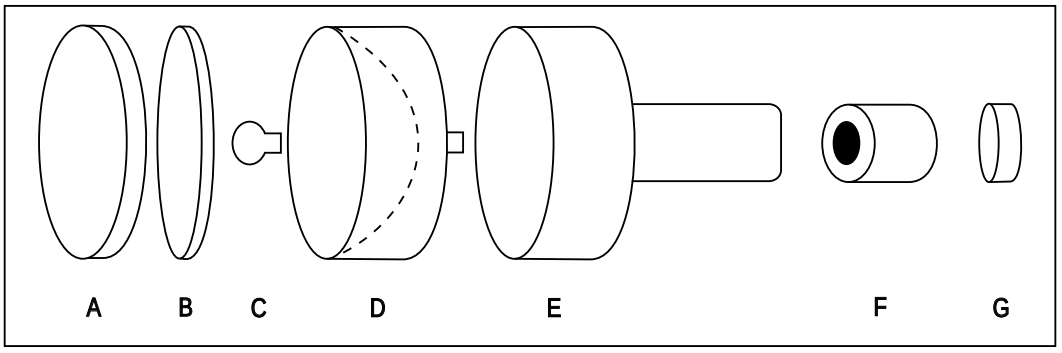
This can be easily understandable by the figure 6. In the parent 1 the task 25 is preceded by task 31 can be replaced by the adjacent parent 2 with the task 26 and task 30, as task 30 precedes task 26. The significance of 'P_c' is the amount of information transmuted from the parent to offspring during mating.

Mutation: Mutation is a random process where another to produce a new genetic structure forcibly replaces one allele of a gene. In GA, mutation is randomly applied with low probability 'P_m', typically in the range 0.001 and 0.05, and modifies elements with in the chromosomes. Once the generated probability for mutation is with in the specified probability, then the string can be mutated. For this problem the mutation site is fixed. As a result a gene is forcibly introduced in the selected string at particular level and certain tasks are replaced so as to maintain the precedence that results in retaining feasibility. This results in generation of offspring. This can be easily understandable by the Figure 7. As shown in the figure the mutation site is level 1 and hence the task 3 is forcibly replaced by another task 2. Since mutating randomly makes the sequence infeasible. These are possibilities such that interchange of tasks lends to an infeasible string. As shown, after forcibly replacing the task 2, the other task, which precedes this task, are to be inserted to maintain the feasibility. Hence a new offspring is created in which 2 is preceded by 8, 31, 27, 21, 28. Once mutation is completed, the strings are arranged in the ascending order of fitness value. After sorting the strings, the new population is cut down to the old population size.

Thus one generation of the genetic process has been completed. To maintain the size of the original population, the new individuals have to be reinserted into the old population. This process continues till a termination criterion is reached i.e. an optimum or near optimal assembly plan with minimum makes pan is found.

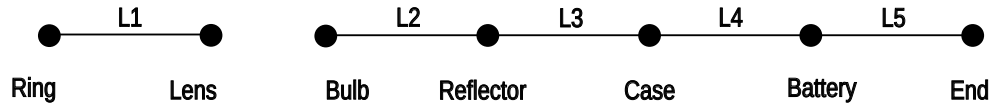
IV. CASE EXAMPLE

To validate the proposed methodology a simple product is considered. The product is a flash light assembly (Torch) shown in the Figure 8, consists of 7 parts. For which liaisons graph is shown in Figure 9. A flash light assembly constitutes seven components (node), linked by five liaisons (connections). Once the liaisons graph is built, possible assembly pairs are derived from the graph. These pairs represents task, which are required to generate feasible assembly plans. Prior to this, it is necessary to become aware of how



A - Ring: B -Lens: C - Bulb: D - Reflector: E - Case: F - Battery: G - End

Fig. 8. Exploded View of a Flash Light Assembly.



LIAISONS GRAPH $[C, \lceil]$
C = { Ring, Lens, Bulb, Reflector, Case, Battery, End }
$\lceil$ = { L1, L2, L3, L4, L5 }

Where
L1: graph of connection between Ring and Lens
L2: graph of connection between Bulb and Reflector
L3: graph of connection between Reflector and Case
L4: graph of connection between Case and Battery
L5: graph of connection between Battery and End

Fig. 9. Liaisons Graph for the Flash Light Assembly

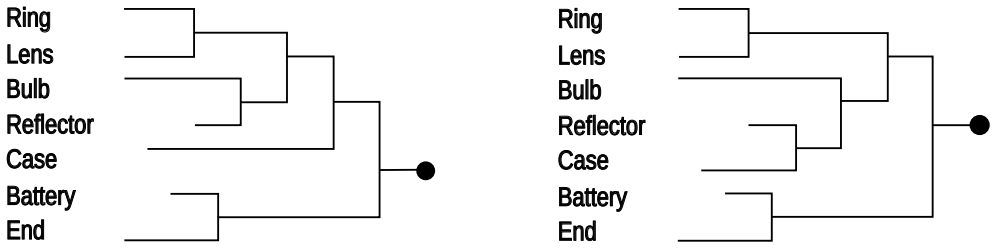


Fig. 10. Some Assembly Tree for the Flash Light Product

many levels are required to generate plans or needed to complete final product. At each level an assembly task is performed to carry on further level. Each assembly plan can be represented by an assembly tree as shown in Figure 10. The best plan corresponds to the tree that has the minimum time An assembly tree for a product '***P***' **is a directed tree whose root represents the product '*P*'** whose leaves represent the components of '***P***'**,** and whose intermediate nodes represent the intervening subassemblies produced by the process. With this information a precedence matrix is constructed. The solution is represented in semantic as well as in and/Or graph form, can be seen from Figure 11.

Number of feasible subassembly pairs represents task and processing times for evaluation for the flash light product: is shown in Table 1.For this type of evaluation function, the search for the best plan can be conducted using genetic algorithm. Thus, the proposed genetic algorithm is applied for the flash light product. The table 2 shows a portion of feasible initial population with its objective function generated by the

(a) In Semantic form:

((Ring, Lens), (Bulb (((Reflector, Case), Battery), End)))

(b) In and/Or graph form:

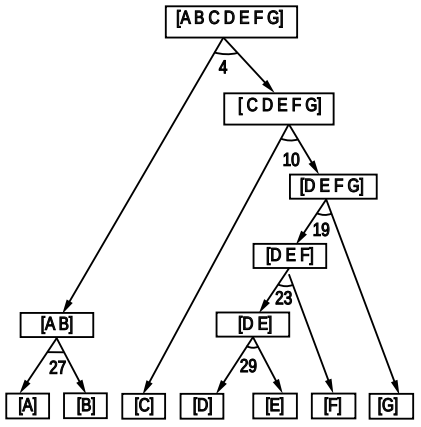


Fig. 11. Representation of Solution in Semantic and And/Or Graph

GA. The table 3 shows a portion of feasible final population with its objective function generated by the GA. The terminal condition is that when there is no improvement in the objective function the iteration stops and the assembly sequence is forwarded for scheduling.

Table 1. Set of Feasible Sub-Assembly Sets and Processing Times for Evaluation for the Flash Light Product

Assembly Task	Sub-Assembly Pairs [Notation]	Processing Time(Min)
1	A B C D E F – G	5
2	A B C D E – F G	7
3	A B C D – E F G	7
4	A B – C D E F G	6
5	A B C D – E F	7
6	A B – C D E F	6
7	A B C D E – F	3
8	A B C D – E	6
9	A B – C D E	5
10	C – D E F G	5
11	C D – E F G	7
12	C D E F – G	5
13	C D E – F G	7
14	A B – C D	7
15	C D E – F	2
16	C D – E F	7
17	C – D E F	5
18	D – E F G	7
19	D E F – G	5
20	D E – F G	7
21	C D – E	6
22	C – D E	6
23	D E – F	2
24	D – E F	6
25	E – FG	7
26	E F – G	6
27	A – B	1
28	C – D	4
29	D – E	4
30	E – F	2
31	F – G	1

Table 2. A Portion of Feasible Initial Population with its Objective Function Generated by the GA

String Number	Assembly Sequence						Processing Time (Min)
1	3	14	25	31	27	28	27
2	3	14	26	30	27	28	27
3	1	5	14	27	28	30	26
4	2	9	14	27	28	31	25
5	3	14	25	31	27	28	27
6	1	6	16	27	28	30	25
7	3	14	26	30	27	28	27
8	1	6	15	21	27	28	24
9	2	9	14	27	28	31	25

Table 3. A Portion of Feasible Final Population with its Objective Function Generated by the GA

String Number	Assembly Sequence						Processing Time (Min)
2	4	27	10	19	23	29	23
2	4	27	10	19	23	29	23
3	4	27	10	19	23	29	23
4	4	27	10	19	23	29	23
5	4	27	10	19	23	29	23
6	4	27	10	19	23	29	23
7	4	27	10	19	23	29	23
8	4	27	10	19	23	29	23
9	4	27	10	19	23	29	23

V. CONCLUSION

In this work, a Genetic Algorithm is proposed for scheduling an assembly task, with the objective as the minimization of makespan. It starts from the liaisons graph, which forms the basis for constructing precedence matrix. Further this precedence matrix and geometric constraints are used to generate a feasible assembly plans (or) to verify for feasibility, represents an initial set of population to carry out genetic operation. The assembly plans are evaluated based on make span. A purposely-developed genetic operator is used to perform crossbreed and mutation function. Thus a Genetic Algorithm is proposed to solve an assembly sequence problem, a much more difficult problem than other sequencing problems. The proposed methodology can be adapted to assemblies where computational time is significant.

REFERENCES

[1] Del Valle, C. and Camacho. E.F. 1996, Automatic Assembly Task Assignment for a Multi Robot Environment. Control Engineering. Practice, Vol.4, No. 7, pp. 915-921.

[2] Fox B. R. and Kempf K.G. 1985, Opportunistic Scheduling for Robotic Assembly, Proceedings of IEEE International Conference on Robotics and Automation, pp 880- 889.

[3] De Fazio, T.L. and Whitney. D.E. 1987, Simplified Generation of All Mechanical Assembly Sequences, IEEE Journal of Robotics and Automation Vol. 3, No. 6, pp. 640-658.

[4] Thomas L. De F, Thomas E. A, Guillaume P. A, Daniel E. W. 1990, Computer-Aided Assembly Sequence Editing and Choice: Editing Criteria, Bases, Rules, and Technique. Proceedings IEEE International Conference System Engineering. pp. 416- 422.

[5] Homem De M, L.S. and Sanderson. A.C. 1991, A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences, IEEE Transaction on Robotics and Automat ion Vol. 7, No. 2, pp.228-240.

[6] Homem De M, L.S. and Sanderson. A.C. 1990, And/Or Graph Representation of Assembly Plans, IEEE Transaction on Robotics and Automat ion Vol. 6, No. 2, pp. 188-199

[7] Homem De M, L.S. and Sanderson. A.C. 1988, Planning Repair Sequences Using the And/Or Graph Representation of Assembly Plans, Proceedings of IEEE International Conference Robotics and Automat ion, pp.1861-1862.

[8] Homem De M, L.S. and Sanderson. A.C. 1991, Two Criteria for the Selection of Assembly Plans: Maximizing the Flexibility of Sequencing the Assembly Tasks and Minimizing the Assembly Time Through Parallel Execution of Assembly Tasks, IEEE Transactions on Robotics and Automation, Vol. 7, No. 6, and pp.626-633.

[9] Homem De M, L.S. and Sanderson. A.C. 1990, Representations of Mechanical Assembly Sequences IEEE Transactions on Robotics and Automation, Vol. 7, No. 2, pp.211-227.

[10] Randall H. W and Jean-Francois R. 1990, Maintaining Geometric Dependencies in an Assembly Planner. IEEE, pp. 890- 895.

[11] Del Valle C, Rafael M. G, Miguel T and Eduardo F. C., A Genetic Algorithm for Assembly Sequence Planning, Artificial Neural Nets Problem Solving Methods, Springer Berlin / Heidelberg, ISBN 978-3-540-40211, pp 1044.

[12] Li J. R., Khoo L. P and Tor S. B., 2003, A Tabu-Enhanced Genetic Algorithm Approach for Assembly Process Planning, Journal of Intelligent Manufacturing, Vol. 14, No 2, pp197-208.

[13] Faicel H., Xavier D and Alexandre D. 2009, Genetic Algorithm for Supply Planning in Two-Level Assembly Systems with Random Lead Times, Engineering Applications of Artificial Intelligence, Vol. 22 ,Issue 6, pp 906-915.

[14] Lu C., Wong Y S and Fuh J Y H. 2006, An Enhanced Assembly Planning Approach Using a Multi-Objective Genetic Algorithm, Journal of Manufacture, Professional Engineering Publishing, Vol. 220, No.2, pp 255-272.

[15] Lazzerini B and Marcelloni F, 2000, "A Genetic Algorithm For Generating Optimal Assembly Plans, Artificial Intelligence in Engineering, Vol. 14, No 4, pp 319-329.

[16] De Lit P, Latinne P and Rekiek B. 2001, Assembly Planning With An Ordering Genetic Algorithm, International Journal of Production Research, Volume 39, Number 16, pp 3623-3640.

[17] Ames, A.L., T.L. Calton, R.E. Jones, S.G. Kaufman, C.A. Laguna and R.H. Wilson. 1995, Lessons Learned from a Second Generation Assembly Planning System, Proceedings IEEE International Symposium on Assembly and Task Planning, pp. 41-47.

[18] Bonneville, F., Perrard, C., and Henrioud, J. M. 1995, A Genetic Algorithm to Generate and Evaluate Assembly Plans. Proceeding of IEEE Symposium on Emerging Technologies and Factory Automation, Paris, pp. 231-239.

[19] Bautista, J., Lusa, A., Suárez, R., Mateo, M., Pastor, R., and Corominas, A. 1999, Application of Genetic Algorithms to Assembly Sequence Planning with Limited Resources. Proceedings of International Symposium on Assembly and Task Planning, pp.411- 416.

[20] Beatrice, L., and Gino D. 1999, Assembly Planning Based on Genetic Algorithms. IEEE, pp. 482- 486.

[21] Carmelo, D.V., Miguel Toro, Eduard F. Camacho, Rafael M.G. 2003 A Scheduling Approach to Assembly Sequence Planning, Proceedings of the 5th International Symposium on Assembly and Task Planning, pp.103-108.