Optimization of Robotic Assembly Sequence

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

Mechanical Engineering

(Production)

2010-2012

BY

Yogesh Rao

(ROLL: 210ME2244)

Department of Mechanical Engineering
National Institute of Technology Rourkela, 2010-12

Optimization of Robotic Assembly Sequence

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

Mechanical Engineering

(Production)

2010-2012

BY

Yogesh Rao

Under the supervision of

Dr. B.B.Biswal

Department of Mechanical Engineering
National Institute of Technology Rourkela, 2010-12

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **"Optimization of Robotic Assembly Sequence"** submitted by **Yogesh Rao** in partial fulfilment of the requirements for the award of **Master of Technology** Degree during the session of 2010-2012 in the Department of **Mechanical Engineering** with **"Production Engineering"**, National Institute of Technology, Rourkela is a reliable work carried out by him under my supervision and guidance.

To the best of my knowledge, the work reported in this thesis is original and has not been submitted to any other university or institute for the award of any degree or diploma.

He bears a good moral character to the best of my knowledge and belief.

Prof. B.B. Biswal
Dept. of Mechanical Engineering
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENT

I would like to express my heartiest gratitude to my guide and supervisor **Prof. (Dr.) B.B. Biswal**, **Professor, Department of Mechanical Engineering, NIT Rourkela** for his valuable and enthusiastic guidance, who not only guided the academic project work but also stood as a teacher and philosopher in understanding the imagination in pragmatic way, I want to thank him for introducing me for the field of optimization and giving the opportunity to work under him. His presence and optimism have provided an invaluable influence on my career and outlook for the future. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

I express my gratitude to **Dr K.P.Maity**, Professor and Head, Department of Mechanical Engineering, for extending all possible help in carrying out the dissertation work directly or indirectly. He has been great source of inspiration to me and I thank him from bottom of my heart.

I would like to give a special thanks to **Prof. BBVL.Deepak**, Department of Industrial Engineering for his continuous encouragement and support.

I feel pleased and privileged to fulfill my parent's ambition and I am greatly indebted to them for their moral support and continuous encouragement while carrying out this study. This thesis is dedicated to my family.

I would like to thank my friends, Abhishek Tiwari, Sukesh Babu V.S, Suman kumar, who provide me moral support during my work.

Yogesh Rao

210ME2244

ABSTRACT

The assembly process is combination of several products into a single product. The assembly process affects manufacturing processes very great extent because it is very time consuming and expensive process. The cost of assembly can reach up to 30% of the manufacturing cost. Instability and direction change in assembly process increases the cost of assembly thus the total cost of product is increased very great extent. The production rate decreases with increase in time in assembly process, so the correct assembly sequence is needed to reduce the time and cost of assembly. For the given product assembly model, the sequences and paths of parts is determined by assembly sequence planning (ASP) to obtain the assembly with minimum costs and shortest time. Industries are taking interest in automated assembly system; robotic assembly system comes under category of this assembly system which uses robots for performing the required assembly tasks. This system is one of the most flexible assembly systems to assemble various parts into desired assembly. Robotic assembly systems can handle a wide range of styles and products, so that same product can be assembled different ways, and to recover from errors. Robotic assembly has the ability to switch to different products and styles because robotic assembly is programmable assembly and it has advantage of greater process capability. Robotic assembly is faster, more efficient and precise than any conventional process. It is very important to determine the feasible, stable and optimal assembly sequence for an assembly system. An assembly sequence plan is a high-level plan for constructing a product from its component parts. It specifies which sets of parts form subassemblies, the order in which parts and subassemblies are to be inserted into each subassembly, are to be performed. The aim of the present work is to determine stable, feasible and optimal robotic assembly sequence which follows the assembly constraints and reduces the assembly cost. An important feature of this developing process is represented by the need to automatically determine the assembly plan by recognizing the optimum sequence

of operations based upon cost and accuracy. Products with large number of parts have several alternative feasible sequences among which optimal assembly sequence is generated. Traditional methods often generate combinatorial explosions of alternatives, with intolerable computational times. A new methodology has been developed to find out the best robotic assembly sequence among the feasible robotic sequences. The feasible robotic assembly sequences have been generated based on the assembly constraints and later, Artificial Immune System (AIS) and particle swarm optimization with mutation operation has been applied to generate feasible and optimal assembly sequences and result is compared with the previous technique. In AIS Clonal selection and Affinity maturation have been implemented to determine the optimal assembly sequence. During the implementation, each assembly sequence and its energy value have been considered as antibody and the antibody affinity respectively. In PSO, each part of the assembled product is considered as the particle (bird) and mutation operation is performed for selected assembly sequence in each iteration to update the position and velocity of each particle. To generate optimal assembly sequence, a fitness function is generated, which is based on the energy function associated with assembly sequence. The sequence which is having the best fitness value followed by all assembly constraints is treated as the optimal robotic assembly sequence. Present research work has been divided into six chapters. The introduction of the topic and the related matters including the objectives of the work are presented in Chapter 1.The literature reviews on different issues of the topic in Chapter 2. In Chapter 3 Steps of assembly sequence generation, assembly constraints, instability is presented Chapter 4 presents generation of stable assembly sequences using Novel immune approach method and Particle swarm optimization with mutation operation for the generation of robotic assembly sequence. In Chapter 5, Result and discussion obtained from different methods are presented. Finally, Chapter 6 presents the conclusion and future work.

CONTENTS

Chapter 4: Assembly Sequence and Soft Computing Methods

Chapter 5: Results and Discussions

List of Figures

Figure Title Page

List of Tables

Table Title Page

PSO parameters

Nomenclature

L_i	Liaison of the i_{th} and $(i+1)_{th}$ components
N	Number of parts
$l_{\alpha\beta}$	Liaision between part α and β
p_{α}	α_{th} component of assembly product
p_{β}	β_{th} component of assembly product
$C_{\alpha\beta}$	Contact-type connection matrix
$f_{\alpha\beta}$	Fit-type connection matrix
d	Assembly direction
C_d	Directional contact connection
f_d	Fit type element
rc	Real contact in *d* direction between two parts
vc	Virtual contact in *d* direction between two parts
sw	screwing in *d* direction
rf	Round peg-in-hole fit
mp	Multiple round peg-in-hole fit
pf	Polygon fit
vf	Virtual fit
0	No fit
n_p	Set of parts
PC	Precedence Constraints
AM	Assembly matrix

$S(p_k)$	Motion instability matrix
$S\{p_k(l_{jk})\}$	Instability matrices of directional connections established by the liaison l_{jk}
E_{seq}	Energy function associated with ASG
E_J	Energy related with Assembly cost
E_P	Energy related with Precedence constraints
E_C	Energy related with Connectivity constraints
J	Assembly cost
C_P	Energy constant related with Precedence constraints
C_C	Energy constant related with Connectivity constraints
C_J	Assembly constant related with costing
C_{as}	Normalized degree of motion instability
C_{nt}	Normalized number of assembly direction changes
ρ_s	Cost constant related to normalized degree of motion instability
ρ_t	Cost constant related to normalized number of assembly direction changes
μ_i	Precedence index
λ_i	Connectivity index
BA	Base assembly
p	Antibody

Abbreviations

AIS	Artificial Immune System
ASG	Assembly Sequence Generation
GA	Genetic Algorithm
IOA	Immune Optimization Approach
PSO	Particle swarm optimization
AM	Assembly Matrix
EA	Evolutionary Algorithm

Chapter 1

Introduction

1.1 Overview

Most engineered products-from pencil sharpeners to aircraft engines-are assembled units. During product design and development, designers traditionally consider not only functionality but also ease of manufacture of individual components and parts. However, little attention is paid to those aspects of design that will facilitate assembly of parts. Emerging soft-computing technique can enable part design, scheduling, process planning, understanding and analysis and effective sequence estimation for assembling a product. Every one of the products has some definite characteristic in common with every other product. The common characteristic is that, the product itself contains of number of parts that should be joined to form a finished product. Without the ability to assemble products, manufacturing companies could not manufacture, and hence their existence in world would really be difficult. Assembly is the process of joining separate components together to form a single final assembled unit. A single assembly task involves combining two or more components or subassemblies together. It is an important consideration in many cases, the order in which these tasks are performed. Because of physical constraints such as accessibility and stability of assembly many such orders may not be feasible. There can be many feasible sequences exist, but some are more desirable than others according to criteria such as the need for jigs or fixtures, the number of tasks that can be performed simultaneously. Assembly planning is defined as the process of determining an assembly

plan, which defines either a complete or partial sequence in which the assembly tasks can be accomplished. Finding out the choice of assembly sequence is very difficult for two reasons. First, the number of feasible sequences can be large even at a small parts amount and can increase with increasing parts count, and second, seemingly minor design changes can modify the available choices of assembly sequences. Generally, techniques for exploring the choices of assembly sequence are informal and incomplete. A simple way of generating assembly sequences is either to assemble the part by all the way, or disassemble the part by all the way. Apply all the possible assembly or disassembly sequence. The final stage of the technological process is assembly during which previously manufactured machine components are put together into assembly units, or lower-order assembly units are assembled together into higher-order assembly units and finally, into a finished product (machine). Assembly is carried out in accordance with a series of planned actions so that the assembly units and a complete product meet all the specifications imposed by a designer. One of the primary objectives of assembly design is to determine the most correct sequence of assembling the components (units) together. The aim of planning of assembly sequences is to determine different orders ways in which the assembly operations can be performed and to evaluate the orders of determining the optimum sequence. The criterion according to which the assembly plans are evaluated is the total cost of the assembly process. Because of geometric and technological constraints that are imposed it is difficult to develop an optimum assembly sequence. The number of possible assembly sequence depends exponentially on the number of parts the product is made of. A proper assembly plan reduces the number of base part reorientations and by combining manipulations into multi manipulation operations and the simultaneous attachment of several parts eliminates some assembly operations whereby the number of needed assembly tools and hence the production costs are reduced.

1.2 Methods of Generating Assembly Sequences

Among the current methods of generating assembly sequences one can distinguish four basic groups:

1. Methods characterized by a three-stage procedure of building sequences. First, sequential relations between the finished product's components are generated taking all the geometrical and mechanical constraints into account. This can be done by analysing the assembly or the disassembly operations of the finished product. The sequential relations are used to generate assembly sequences. Finally, the best sequence is selected according to the optimization criterion adopted.

2. Methods involve dividing the assembled unit into subunits and generating proper subsequence by applying simple rules.

3. To build expert systems for the assembly of specific, unique units.

4. Methods generating different product assembly sequence variants.

Considering the accuracy of the results obtained and the time, in which they are obtained, the methods can be divided into:

• Algorithmic methods – yielding optimum (according to the criterion adopted) assembly sequences,

• Heuristic methods – yielding good solutions in a reasonable time.

1.3 Classification of Assembly System

The Figure 1.1 given below classifies the assembly system. Assembly system is broadly classified into three categories.

1. Manual assembly.
2. Semi-automatic assembly.
3. Automated assembly.

Automated assembly is further divided into two categories fixed automation and flexible automation.

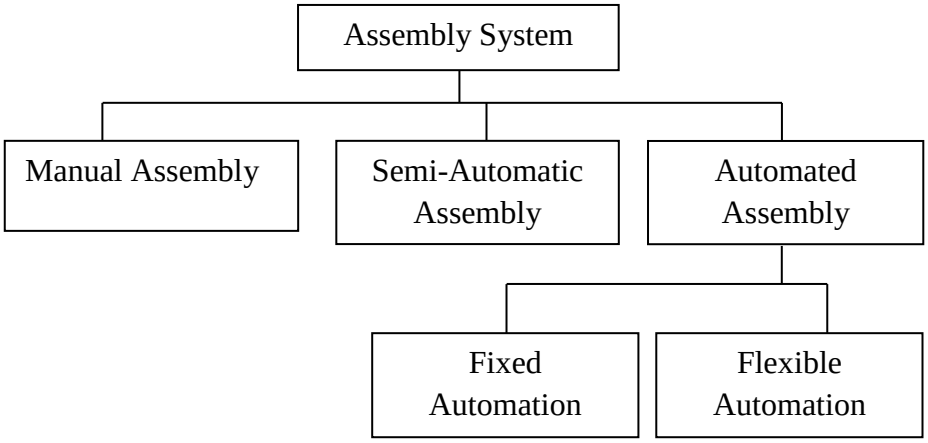

Figure 1.1: Classification of assembly system.

1.3.1 Manual Assembly: The operation in manual assembly is carried out manually with or without the help of general purpose tool like screwdriver and pliers. Individual parts or component are transferred either manually or using mechanical equipment such as transfer lines or parts feeds and then components are manually assembled. This assembly method is very flexible and adaptable. The assembly cost in this method is constant. Manual assembly is independent of production volume.

1.3.2 Semi-Automatic Assembly: In semi-automatic assembly system one and only one specific product is assembled. So in this assembly system machinery needs a huge capital investment. As production volume surges, the capital investment decreases more than total manufacturing cost.

1.3.3 Automatic Assembly: Automated assembly mainly referred to as fixed automation. Either synchronous or non-synchronous indexing machines and automatic feeders where parts are held by a free-transfer device are used. Machines are used for the assembly of a product. These systems lack any flexibility to conciliated changes in the design of the

product. It necessities a huge capital investment, as well as significant time and engineering work before actual production can be started.

1.3.4 Automatic Assembly using Robot (Robotic assembly)

Production volume is greater than that of a manual assembly system but lesser than that of automatic assembly system.

Common forms of Robotic Assembly

1. One arm robot functioning at a single workstation that includes parts feeders, magazines, etc.

2. Two robotic arms operating at a single workstation.

- A programmable controller (PLC) is used to coordinate and control the motions of the two arms.
- It is denoted as a robotic assembly cell and similar to FMS cell.

3. Multi-station robotic assembly system.

- Multi-station robotic assembly system is capable of performing several assembly operations simultaneously.
- It can execute different assembly operations at each station.
- It has great flexibility and adaptability to design changes.

1.4 Comparison of Assembly Methods

- Manual assembly requires the least capital investment followed by the two simplest forms of robotic assembly.
- Multi-station robotic assembly system compares to automatic system with special-purpose machines requires more capital investment for a large production volume but less capital investment for a moderate production volume.

- Assembly cost per product is constant for manual assembly
- Assembly cost per product decreases linearly with increasing production volume for automatic assembly using special-purpose machines.
- In the case of robotic assembly, the assembly cost per product decreases with increasing production volume, but becomes less economical after exceeding the annual production volume at a certain point.

1.5 Product Design

1.5.1 Product Design for Manual Assembly

To design products for manual assembly we need to both the assembly time and the skills essential for assembly workers.

Rules for product design for manual assembly:

- Remove the need for any decision making by the assembly worker, comprising his or her having to make any final changes.
- Ensure availability and discernibility.
- Component should be designed to be self-aligning and self-locating so that it could be removed the need for assembly tools. The types of parts should be minimized by adopting the concept of standardization as a design philosophy.
- Multifunction and flexible components should be used.
- The number of separate parts in an assembly should be minimalized by eradicating excess parts and, whenever possible, mixing two or more parts together, as handling lesser parts are much easier.
- The criteria for eliminating the parts count per assembly is recognized by G. Boothroyd and P Dewhurst comprise negative answers to the following questions :

- Does the part move comparative to all other parts which are already assembled?
- Must the part prepared of a different material?
- Must the part be distinct from all other parts previously assembled because if not assembled, assembly of other parts would be impossible?

- Try to make all motions simple, for example, excluding multi motion insertions.
- Part should be planned, so that it could have maximum symmetry in order to enable easy orientation and holding during assembling.

1.5.2 Product Design for Automatic Assembly

Parts should be: uniform, high quality, have great geometric tolerances, to remove any downtime of the assembly system due to parts incongruity or manufacturing faults. Important factors contain orientation, handling of parts to the assembly machine.

Rules for automatic assembly are:

- Reducing the number of dissimilar components in an assembly by using the three questions listed previously.
- There should be Use of self-aligning and self-locating features like chamfers, guidepins, dimples, and some types of screws.
- Avoid fastening by screws because it is expensive and time-consuming.
- Thus, suggested to design parts that will snap together by a press fit.
- Make the principal and most rigid part of the assembly as a base where other parts are assembled vertically in order to take advantage of gravity.

- Seek the use of standard components and materials to avoid the possibility of parts nesting, or shingling during feeding.
- Avoid flexible, fragile, and abrasive parts and confirm that the parts have sufficient strength to resist the forces exerted on them during feeding and assembly.
- Avoid reorienting assemblies because each reorientation may require a separate station or a machine.
- Design parts by presenting or admitting the parts to the assembly machine in the right orientation to ease automation.

1.5.3 Product Design for Robotic Assembly

The product design rules for robotic assembly are fundamentally the same as those for manual or automatic assembly. Two very important concerns that have to be taken into consideration when designing components for robotic assembly:

1. Design a component so that it can be grasped, and injected by that robot's end effector. Otherwise it will result in the need for an extra robot and, therefore higher assembly cost.
2. Design parts so that they can be held to the robot's arm in an orientation appropriate for grasping.

1.6 Methods for evaluating and improving product (DFA)

Methods are based on evaluating the ease or difficulty with which parts can be handled and assembled together into a given product. An analytical process is followed where the problems related with the components design are detected and quantitatively evaluated.

Most commonly used methods:

- The Boothroyd-Dewhurst DFA Method.
- The Hitachi Assembly Evaluation Method.

- The Lucas DFA Method.
- The Fujitsu Productivity Evaluation System.

1.6.1 Boothroyd-Dewhurst DFA Method: This method is established in the late 1970s by Professor Geoffrey Boothroyd, at the University of Massachusetts, Amherst in cooperation with Salford University of England. Figure 1.2 given below describes the step of this method. First, the suitable assembly method is selected by means of charts then; the analytical procedure corresponding to the assembly method is selected.

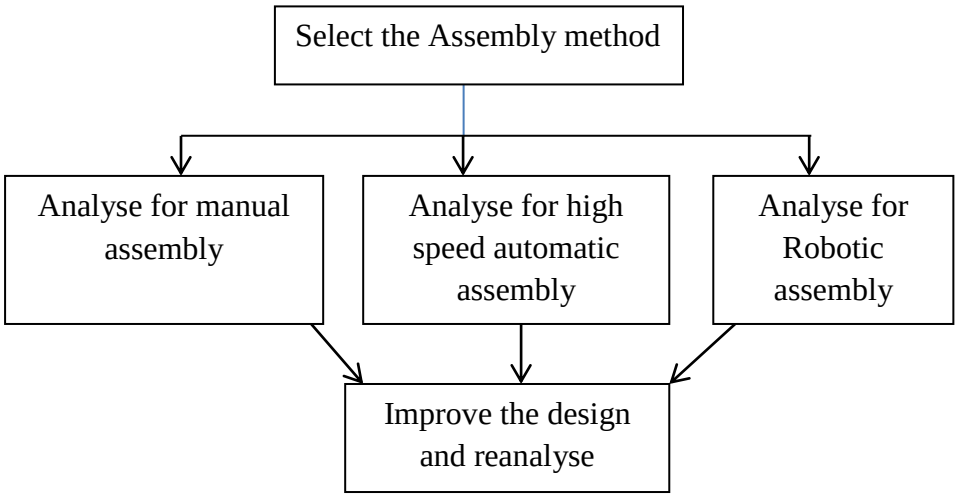

Figure 1.2: Boothroyd-Dewhurst DFA Method.

The assembly time for each component part is then found by addition of the handling time of that part to its insertion time.

- Once the components and the assembly time for each are known, total assembly time and assembly cost for the present design is estimated.
- The next step is to decrease the parts count by eliminating or combining some parts. Therefore finding "theoretically needed" parts.

- Design is improved by studying the worksheet and removing components that have comparatively high handling and insertion times. This process is reiterated until an optimal design is achieved.

Disadvantage: Decreasing the parts count could manufacture and use of complex components. Since assembly cost is 15% of total cost, the final product could be assembled but expensive to manufacture.

1.6.2 Hitachi Assembly Evaluation Method: The method does not correctly differentiate between manual and automatic assembly, this difference is accounted for unescapably within the structured analysis. The Hitachi AEM approach is based on measuring the assemblability of a design based on followings:

- For complex operations, penalty scores that depend upon the difficulty and nature of each operation are allowed.
- The method of estimating the time (and cost) of an operation includes breaking it into its elemental components and allocated time for each elemental motion based on compiled practical observations.
- Any saving in the assembly cost can be accomplished by removing the parts count in a product or simplifying the assembly processes.

1.6.3 Lucas DFA Method: Unlike the previous two methods, the Lucas DFA evaluation is not based on monetary costs, but on three indices that give a virtual measure of assembling difficulty. The goal of decreasing the parts count and the estimation of the insertion operations are shared with the previous two methods. Analysis is carried out in three stages.

- Functional
- Feeding (or handling)
- Fitting analyses.

1.6.4 Fujitsu Productivity Evaluation System: Unlike other DFA methods it is not a improvement procedure after completion of the detailed design. Rather it can be considered as a software package which can be used as a tool to aid in finding a detailed design that is easy to manufacture and assemble with cost efficiency. Limited to bench type manual assembly of comparatively small parts. It consists of four subsystems as shown in Figure 1.3, based upon making full use of an expert system comprising practical manufacturing and design data and rules of thumb gathered from experience.

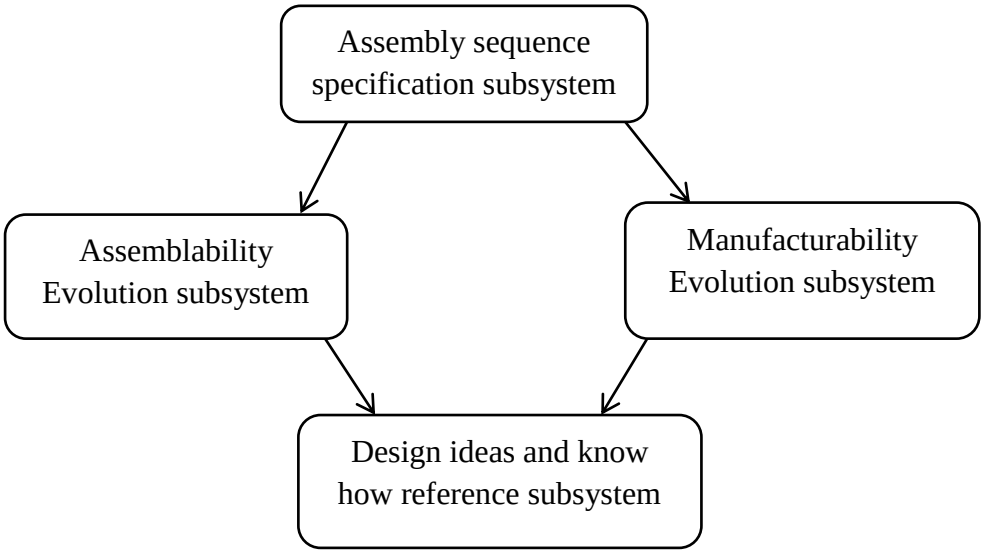

Figure 1.3: Fujitsu Productivity Evaluation Systems.

1.7 Classification of Assembly Sequences

1. Stable Assembly Sequence: The Sequences that sustain the stability of in-process subassembly movement are considered as stable sequences, by means of which the parts can be effectively assembled to form an end-product.

2. Feasible Assembly Sequence: Once the assembly limitations have been inferred, assembly sequences that fulfill the assembly constraints are called the feasible assembly sequences.

3. Optimal Assembly Sequence: An assembly sequence is called optimal assembly sequence when it reduces the assembly cost while satisfying assembly constraints.

1.8 Need of Assembly Sequence Optimization

Assembly of a product is very time consuming process, instability and change in the direction of assembly affects the productivity and also the cost. It is needed to automatically plan the assembly of a product or, in other words, to determine the best sequence of operations and program the machines to perform the required tasks. It means for sequencing the components and programming the machine proper assembly sequence should be known. Determining the choices of assembly sequence is difficult because all possible assembly sequence are very large even small number of parts and slight design changes can extremely modify the available choices of assembly sequences. There is very large no of assembly sequence for a product and all the sequences have large variation in cost, efficiency and time. It is not possible that all the sequences are feasible sequence. Robotic assembly systems are more cost effective and qualitative. Robotic assembly directly affects the productivity of the process, cost of production, and the product quality. It is necessary to generate an appropriate sequence which minimizes the assembly cost and satisfies the assembly constraints. The assembly constraints include the precedence constraints and the connectivity constraints. The precedence constraint is a set of parts that must be joined before a pair of parts is joined. The connectivity constraints, is the connective connections between the two parts. It is necessary that part to be gathered onto an in-process subassembly should have at least one real connection with some part belonging to the in-process subassembly. The feasible sequences do not always promise the parts to fix onto an in-process subassembly; parts may be loosely joined, and may come apart during the handling. Instability in motion and change in direction make the sequence unstable this should be avoided to make the sequence stable, by means of which the parts can be effectively assembled to form an end-product. An assembly sequence

must not have effect of external and internal effects of forces to make the assembly sequence stable. External forces are due to gravitation and internal forces are due to the mutual contact of the objects. So, it is important to find out, whether a configuration of the objects remains stable during or after the assembly process. Products with large number of parts have large alternative feasible sequences and among this sequence feasible sequences are to be found. Determining the best assembly sequence is one of the most critical problems. Robotic assembly sequence is requirement of industries and being a cost intensive process, it is necessary to determine the optimal sequence with the constraints of the process in mind.

1.9 Objective of the Research

The aim of the present research work is to determine, stable, feasible, and optimal robotic assembly sequence fulfilling the assembly constraints with minimum assembly cost. The present research aims is at developing an approach for generating robotic assembly sequences using the evolutionary technique considering of the degree of freedom, instability of assembly motions and directions. The comprehensive objective of research work is defined as follows.

i) To generate feasible assembly sequences automatically.

ii) To reduce the cost and time of assembly

iii) To apply new methodologies and modify some conventional methodologies for determining optimal assembly sequences for robotic assembly systems in an orderly manner.

iv) To apply suitable new techniques for generation of optimized assembly sequence that would give better or similar result than that of previous techniques.

1.10 Plan of the Thesis

The thesis describing the present research work is divided into six chapters. The subject of the topic and the related matters including the objectives of the work are presented in Chapter 1.The reviews on several diverse streams of literature on different issues of the topic in Chapter 2. In Chapter 3 Steps of assembly sequence generation, assembly constraints, instability is presented Chapter 4 presents generation of stable assembly sequences using Novel immune approach method and Particle swarm optimization with mutation operation for the generation of robotic assembly sequence. In Chapter 5, Result and discussion obtained from different methods are presented. Finally, Chapter 6 presents the conclusion and future scope of the research work.

1.11 Summary

There is large number of sequences for a product and as the number of parts increases; the number of sequences also increases. It is necessary to determine optimized assembly sequence, so that time and cost for assembling a product could be minimized .Different method their significance and disadvantages are presented. Product design required for all the assembly system is presented which is very useful for reducing assembly cost.

Chapter 2

Literature Survey

2.1 Overview

The aim of this literature survey is to find an efficient algorithm for optimizing assembly sequence because earlier methods are non-optimizing and time consuming. Different methods have been studied to assembly sequence problem. The recent interest in robotic assembly and automatic generation of optimized robotic assembly plans has led to research on automatic generation of assembly sequences. For this reason, it is very important to develop new procedures which simultaneously satisfy the constraints and reducing the cost and time. There have been so many research works and experimental extrapolations for the generation of appropriate and correct assembly sequences which is reflected through enormous number of literatures. The preliminary study of the subject necessitates a general review of the work carried out by various researchers. The relevant literatures are studied and conferred in relation to the methodologies and systems of executing the above components or activities towards an integrated environment for supporting the present goal set.

2.2 Important Literatures Related to Present Work.

Table 2.1 presents some of the essential work carried out on assembly sequence generation methods.

Table 2.1: Important literature reviews related to assembly sequence generation

SI	AUTHOR	YEAR	TITLE	REMARK
1.	Mascle and J. Figour	1990	Methodological approach of sequence determination using disassembly method	Constraint method is generated and the least constraint parts are disassembled at each step and obtains the assembly in its reverse
2.	Luiz and Arthur	1991	Representations of mechanical assembly sequences	Analysed mechanical assembly sequences based on directed graphs, establishment conditions and precedence relationships.
3.	Cho and Shin	1994	Automatic inference on stable robotic assembly sequences based upon the evaluation of base assembly motion instability	Develops a graph search method for automatic implication on stable robotic assembly sequences based upon motion instability.
4.	Sugato and Jan	1997	A structure-oriented approach to assembly sequence	Developed an assembly sequence planner which is used as a tool for finding good plans more rapidly

			planning	by using high-level expert advice and reusing sub plans for repeated substructures.
5.	Hong and Cho	1999	A genetic-algorithm-based approach to the generation of robotic assembly sequences	Proposed genetic algorithm to generate robot assembly sequences. Their methodology obtains the optimal assembly sequence by minimizing the assembly cost while satisfying the assembly constraints
6.	Castro and Timmis	2002	Artificial immune system: a new computational intelligence approach	Develops a new computational intelligence approach by using artificial immune system
7.	Schutte, j	2005	Evaluation of a particle swarm algorithm for biomechanical	Implemented PSO algorithm for the biomechanical optimization and conclude that PSO algorithm is easier to be fulfilled than GA algorithm.
8.	Chen and Lin	2007	A particle swarm optimization approach to optimize component placement in printed	A particle swarm optimization approach to optimize component placement in printed circuit board assembly proposed an adaptive particle swarm optimization

			circuit board assembly	approach to solve the problem of minimizing the printed circuit board assembly time simultaneously with optimization of assignment problems for a pick-and-place
9	Surajit Surajit, S., Biswal, B.B., Dash, P. and Choudhury	2008	robotic assembly sequence using ant colony optimization Generation of optimized	implemented ant colony optimization to find out optimized robotic assembly sequence by minimizing iteratively an energy function, which satisfies the conditions: less assembly cost and the process constraints
10.	Hong and Cong	2010	An assembly sequence planning approach with a discrete particle swarm optimization algorithm	used a discrete particle swarm optimization (DPSO) algorithm to solve assembly sequence planning

Luiz and Arthur (1991) have analysed mechanical assembly sequences based on directed graphs, establishment conditions and precedence relationships. There are five representations of assembly sequences. These assembly sequences are based on directed graphs, on AND/OR graphs, on establishment conditions, and on precedence relationships. The latter includes two types: precedence relationships between the establishment of one connection between parts

and the establishment of another connection, and precedence relationships between the establishment of one connection and states of the assembly process.

Baldin, et al. (1991) developed simplified method which can find the optimal solutions, but have a problem of the search space explosion for an increased number of parts. The method built a combined set of user-interactive computer programs that generates all feasible assembly sequences and then aids the user in evaluating their value based on various criteria. The programs use a disassembly analysis for determining sequences and provide on-line visual aids during generation and evaluation. The method generally is a cut-set method for the determination and representation of all ordered and mechanical assembly limitations as precedence relations.

Lee (1989), Shin and Cho (1994) proposed disassembly method. In this method an assembly sequence was determined by the reverse order of disassembly sequence expressed in a list of parts each of which is sequentially chosen to have minimum cost of disassembly. A mathematical approach is proposed to the analysis of disassemblability of a product for determining stable robotic assembly sequences.

Mosemann, H.; Rohrdanz, F.; Wahl, F(1998) discussed assembly stability as a constraint for assembly sequence planning. The analysis of (sub) assembly stability avoids mating parts which tend to disassemble under the influence of gravity. Furthermore, the number of reorientations which gives a good idea on the value of an assembly sequence depends on the stability of the parts to be assembled. Algorithms are given to calculate the set of potentially stable orientations of an (sub) assembly considering static friction under uniform gravity. This set is applied for the evaluation of assembly sequences and the minimization of the number of reorientations during plan execution. Therefore, a new evaluation function based

on the set of potentially stable assembly orientations is proposed and integrated into the assembly cost evaluation of a high level assembly planning system.

Sugato and Jan (1997) have developed an assembly sequence planner which is used as a tool for finding good plans more rapidly by using high-level expert advice and reusing sub plans for repeated substructures.

Lowe, G., and Shirinzadeh, B (2004) proposed a self-learning technique for selecting a sequence and dynamically changing the sequence is presented, selection is based on the history of assemblies. The evaluation is dependent on part properties rather than parts and their relationships, thus no previous knowledge of parts and their interaction is required in the decision making process. Most production engineers apply constraint based evaluation and history to identify the solution sequence. The method assumes assembly is without constraint. This maximises the ability of the algorithm to select sequences for new products and optimise them.

Zhou, X., Du pingan, Zhou, Y (2007) present a systematic approach for automatic assembly sequence planning (ASP) by using an integrated framework of assembly relational model (ARM) and assembly process model (APM), which are established by object oriented method. ARM, consisting of assembly, components and liaisons, is used to describe the geometric relationships between components in terms of contact, constraint and interference matrixes.

Lee, S (1992) presents an assembly planning system that operates based on a recursive decomposition of assembly into subassemblies, and analyses assembly cost in terms of

stability, directionality, and manipulability to guide the generation of preferred assembly plans. Method is established for evaluating assembly cost in terms of the number of fixtures (or holding devices) and reorientations required for assembly, through the analysis of stability, directionality, and manipulability. All these factors are used in defining cost and heuristic functions for an AO* search for an optimal plan.

Wang et al. (1998) have outlined an off-line heuristics to find out sequence plan in order to improve robotic assembly efficiency. Later they applied their algorithm to a Cartesian robot, which can allow dynamic allocation. An off-line heuristics is described to sequence the insertion orders and to assign corresponding components to a magazine so as to improve robotic assembly efficiency. The algorithms are developed for a Cartesian robot, which follows dynamic allocation of pick-and-place locations.

Barnes et al. (2002) have modelled a computer based tool named Design for Assembly (DFA) tool, which is useful for Computer Aided Design (CAD) models (Tiam et al. 1999) development and infer/extract relevant information.

A mathematical approach called disassembly approach (Mascle and Figour, 1990; Shin et al. 1995; Tiam et al. 1999) has been used to generate stable robotic assembly sequences.

Hong and Cho (1993, 1995) have developed a computational scheme based on neural network in order to generate the optimized robotic assembly sequence while satisfying the assembly constraints and minimizing the assembly cost. An assembly sequence is called optimal when it satisfies a number of conditions: it must satisfy assembly constraints, keep the stability of in-process subassembly, and minimize assembly cost. Currently, various

search algorithms have been reported for the purpose, but as the number of the parts increases they often fail to generate assembly sequences due to the explosion of the search space. Based upon the inferred assembly costs obtained from the expert system, evolution equation of the network is derived, and finally obtains an optimal assembly sequence resulting from the evolution of the network.

The most common evolutionary algorithm, Genetic Algorithm (GA) has been used by Hong and Cho (1998), and Shiang and Yong (2001), to generate robot assembly sequences. Their methodology obtains the optimal assembly sequence by minimizing the assembly cost while satisfying the assembly constraints. This method denotes an assembly sequence as an individual, which is assigned a fitness related to the assembly cost.

Galantucci et al. (2004) have implemented a methodology called hybrid Fuzzy Logic-Genetic Algorithm for planning the automatic assembly and disassembly sequence of products.

Wang et al. (2004) have developed an ant colony algorithm-based approach to generate optimized assembly sequence for assembled mechanical products. Their approach generates optimal solutions based on the amount of ants cooperating with the least reorientations during assembly processes. For diverse assemblies, the approach generates different amount of ants collaborating for finding the optimal solutions with the least reorientations during assembly processes.

Schutte et al. (2005) implemented PSO algorithm for the biomechanical optimization and conclude that PSO algorithm is easier to be fulfilled than GA algorithm.

Shen et al. (2006) proposed an improved fuzzy discrete particle swarm optimization method and applied it to traveling salesman problem.

Chen and Lin (2007) proposed an adaptive particle swarm optimization approach to solve the problem of minimizing the printed circuit board assembly time simultaneously with optimization of assignment problems for a pick-and-place Machine. The particle swarm optimization (PSO) approach has been successfully applied in continuous problems in practice. The component assignment sequencing problem in printed circuit board (PCB) has been verified as NP-hard (non-deterministic polynomial time). The objective of the problem is to minimize the total traveling distance (the traveling time).

Cao, P.B and Xiao, R. B (2007) have proposed a novel approach, called the immune optimization approach (IOA), to generate the optimal assembly plan. Inspired by the vertebrate immune system, artificial immune system (AIS) has emerged as a new branch of computational intelligence. Based on the bionic principles of AIS, IOA introduces manifold immune operations including immune selection, clonal selection, inoculation and immune metabolism to derive the optimal assembly sequence.

Liao et al. (2007) resolve the complex job-shop scheduling problem using an improved PSO algorithm in which local heuristic information is introduced.

Surajit et al. (2008) have implemented ant colony optimization to find out optimized robotic assembly sequence by minimizing iteratively an energy function, which satisfies the conditions: less assembly cost and the process constraints. A robotic assembly sequence is called optimal when it reduces assembly cost and satisfy the assembly limitations. The

assembly cost relates to assembly operations, assembly motions and assembly direction changes.

Hong Guang (2010) has proposed a discrete particle swarm optimization algorithm to solve the assembly sequence planning based on some key technologies including a special coding method. To make the DPSO algorithm effective for solving ASP, some key technologies including a special coding method of the position and velocity of particles and corresponding operators for updating the position and velocity of particles are proposed and defined.

Recntly, Edmunds et al. (2011 implemented a Hierarchical Genetic Algorithm, not only for reducing the problem size but also for gener) have ating optimal disassembly sequence.

2.3 Summary

Several old and new methodologies have been studied for this present work. The above literatures are reviewed which are very helpful to find out the easy way for assembly sequence generation. By the help of this literature reviews the problems occurred during assembly sequence could be found out. Several new and old technique are compared which can help in reducing the error during assembly sequence generation. There are several works remaining to be done for correct assembly sequence generation.

Chapter 3

Assembly Sequence Generation

3.1 Overview

When a product is assembled, a prescribed order to put components into a fixture to complete the final assembly of the product is required. This order is known as assembly sequence of the product. There are many different techniques and methods have been used but the objective is same. All the methods are based on determining correct and stable assembly sequence that would be capable of reducing the cost and time. To determine required sequences, many researchers used assembly constraints and part contact level graph because the explicit acquisition of the assembly constraints has several merits. In some methods liaison matrix and assembly constraint are used to determine correct assembly sequence. One way of finding correct assembly sequence is to apply the sequences one by one on product and then check it's feasibility but it is quite time consuming and costly process. It is difficult to apply all the sequences to assemble a product. Therefore it is very necessary to use correct method and detail study of all the parameters required for assembly sequence.

3.2 Assumptions and Liaison Connectivity

A product is appropriate for robotic assembly when the following situations are satisfied [24].

i. All the individual components are rigid.
ii. Assembly operation can be done in all mutually perpendicular directions excluding Z direction.
iii. Parts can be assembled by simple inclusion or screwing.

A product comprising *n* parts is represented in the following format

N = (P, L), where N is a product having parts

P = $\{p_a \mid \alpha=1, 2 \ldots\ldots n\}$, and organised by the liaisons

L = $\{l_{ab} \mid a, b = 1, 2 \ldots\ldots r.\ a \neq b\}$

The liaison l_{ab} represents the connective relationship between a pair of parts p_a and p_b. The connective relations can be a contact-type or a fit-type connection and is given by:

$$l_{ab} = liaison(p_a, C_{ab}, f_{ab}, p_b) \tag{3.1}$$

Where

C_{ab} = contact-type connection matrix

And

f_{ab} = fit-type connection matrix.

The dimension of each matrix is 2 × 3 elements, and represented by

$$C_{ab} = \begin{pmatrix} C_x & C_y & C_z \\ C_{\bar{x}} & C_{\bar{y}} & C_{\bar{z}} \end{pmatrix}, \quad f_{ab} = \begin{pmatrix} f_x & f_y & f_z \\ f_{\bar{x}} & f_{\bar{y}} & f_{\bar{z}} \end{pmatrix} \tag{3.2}$$

The assembly directions for robotic assembly are given as $d \in \{x, y, \bar{x}, \bar{y}, \bar{z}\}$.

The contact types are defined as follows:

$$C_d = \begin{cases} 0: \text{no contact in the 'd' direction between } p_a \text{ and } p_b \\ rc: \text{real contact in the 'd' direction between } p_a \text{ and } p_b \\ vc: \text{virtual contact in the 'd' direction between } p_a \text{ and } p_b \end{cases}$$

$$f_d = \begin{cases} 0: \text{no fit in the 'd'direction between } p_a \text{ and } p_b \\ sw: \text{screwing in the 'd' direction between } p_a \text{ and } p_b \\ rf: \text{round peg in hole fit in the 'd' direction between } p_a \text{ and } p_b \\ mp: \text{multiple round peg in hole fit in the 'd' direction between } p_a \text{ and } p_b \end{cases}$$

3.3 Assembly Constraints

There are two types of assembly constraints: precedence constraints and connectivity constraints. A precedence constraint of a liaison l_{ab} is considered by a set of n_p parts that must be connected before two parts p_a and p_b are interconnected and is given by

$$PC(l_{ab}) = \{p_\gamma \mid \gamma = \gamma_1, \gamma_2, ..., \gamma_{n_P}\} \tag{3.3}$$

and $$PC(p_f) = \bigcup_{l=l_1}^{l_q} P(l_{ab}) \tag{3.4}$$

3.4 Assembly Motion Instability

Another important factor to be considered in assembly sequence planning is the instability of a base assembly motion during disassembly. This is because, when disassembling a part, the base assembly needs to be fixed without being taken apart. Here, the assembly motion instability means a degree to what extent parts belonging to a base assembly are fixed. In evaluating such instability, the effects of connecting and grasping status with fixture, and gravity can be included. However, this study stresses the gravity effect to establish the basic concept of instability when inferring stable robotic assembly sequence. To evaluate the assembly motion instability, the instability of its individual part should be firstly examined. Usually, downward assembly motion is preferable in robotic assembly.

3.5 Instability Rules

Rule 1: When p_k is assembled to p_j by l_{jk}, a liaison instability of p_k with respect to the fixed part p_j, $S\{p_k(l_{jk})\}$ is obtained by AND operating all the instability matrices of directional connections established by the liaison

$$S\{p_k(l_{jk})\} = \left[\bigwedge_\delta S\{p_k(l_{jk}(c_\delta)\}\right] \wedge \left[\bigwedge_\delta S\{p_k(l_{jk}(f_\delta)\}\right], \text{ for all } \delta \in \{x, y, z, \bar{x}, \bar{y}, \bar{z}\}, \tag{3.5}$$

where, $S\{p_k(l_{jk}(c_\delta))\}$ is an instability matrix for p_k connected with a directional contact connection c_δ, and $S\{p_k(l_{jk}(f_\delta))\}$ indicates that for a directional fit connection f_δ.

Rule 2: When a part is simultaneously assembled with more than one part of a base part, the part instability is obtained depending upon stability of the base part movement.

Case 1: fixed base parts. If p_k is assembled with a set of fixed base parts, $PF = \{p_f \mid f = 1,2,\cdots,m\}$, then $S\{p_k\}$ can be obtained by AND operating all the liaison instabilities, $S\{p_k(l_{fk})\}$, ($f = 1,\ldots,m$)

$$S\{p_k\} = \bigwedge_f S\{p_k(l_{fk})\}, (f = 1,2,\ldots,m) \tag{3.6}$$

Case 2: unstable base parts. If p_k is assembled with a set of unstable base parts, $PU = \{p_u \mid u = m+1,\cdots,q\}$, then $S\{p_k\}$ can be obtained by OR operating.

$$S\{p_k\} = \bigwedge_u [S\{p_k(l_{uk})\} \vee S(p_u)], (u = m+1,\ldots,q). \tag{3.7}$$

Case 3: combination of fixed and unstable base parts. If p_k is assembled with both $PF = \{p_f \mid f = 1,2,\cdots,m\}$ and $PU = \{p_u \mid u = m+1,\cdots,q\}$, then $S\{p_k\}$ can be obtained by AND operating.

$$S\{p_k\} = \bigwedge_f S\{p_k(l_{fk}\} \wedge \left[\bigwedge_u [S\{p_k(l_{uk}\} \vee S(p_u)]\right], \ (f = 1,2,\cdots,m), (u = m+1,\cdots,q) \tag{3.8}$$

3.5.1 Base Assembly Motion Instability

The degree of motion instability of the *lth* base assembly *BA*, can be determined by summing the instabilities of the parts P, is belonging to the *BA*,. The motion instability *E*, of the l^{th} *base*-assembly is defined as:

$$E_l = \sum_{j=1}^{l} R_{ls}(P_j) \tag{3.9}$$

Where *R*, ($\boldsymbol{p_j}$) is the motion instability of a part *p*,

3.6 Steps of Assembly Sequence Generation

Assembly sequence generation is a complicated task, the first and very important step is to generate all possible assembly sequence. Important steps of assembly sequence generation are given in Figure 3.1For generating sequences each part is selected one by one. If there is five parts then the sequences will be very large taking first part fixed. For an example there are five parts a, b, c, d, and e. For these five parts taking a as fixed part the possible sequences will as follows.

abcde, acbde, adbce, aebcd, abdce, abdec, adbce, adbec, adebc, adecb, aebdc, acdbe, adcbe, adceb, aedbc, acebd, aecbd.

In this way sequence are generated fixing parts one by one. Once the all possible sequence are generated liaison matrix between the parts are created. The connection between parts is determined by the help of liaison matrix. The next step in assembly sequence generation is to determine feasible assembly sequence. The feasible assembly sequence is determined by applying assembly constraints. The assembly constraints include precedence constraints and connectivity constraints. The sequence which follows the both precedence constraints and connectivity constraints is called feasible assembly sequence.

The objective function is given in terms of Energy sequence. Energy sequence includes energy related with assembly cost, precedence constraints, and connectivity constraints. It is assumed that all the sequence generated is stable. To determine energy related with cost degree of freedom between two mating parts and change in direction is determined. The assembly constraints are determined with the help of assembly matrix and liaison diagram.

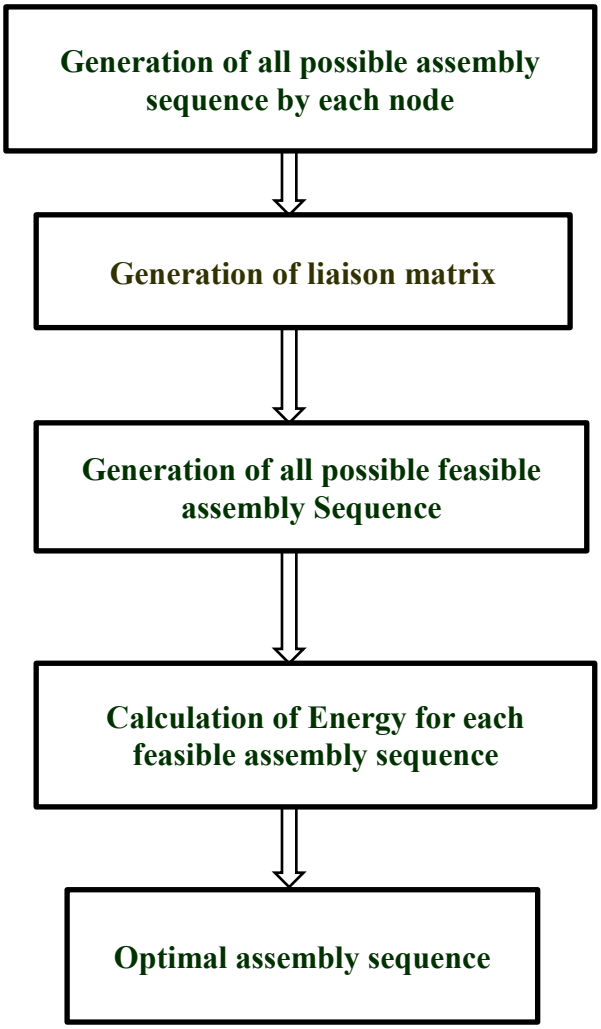

Figure 3.1: Flow diagram of assembly sequence generation

3.7 Assembly Matrix for Constraint Evaluation

Assembly constraints are determined by using assembly matrix for an assembly consisting of q parts can be represented as follows:

$$AM= \begin{array}{c} \\ A_1 \\ A_2 \\ . \\ . \\ A_n \end{array} \begin{array}{c} \begin{array}{|c|c|c|c|} \hline A_1 & A_2 & \ldots & A_n \\ \hline \end{array} \\ \begin{bmatrix} A_{11} & A_{12} & \cdots\cdots & A_{1n} \\ A_{12} & A_{22} & \cdots\cdots & A_{2n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ A_{n1} & A_{n2} & \cdots\cdots & A_{nn} \end{bmatrix} \end{array}$$

Where A_1, A_2,…,A_n represent the n parts in the assembly respectively.

A_{ij}=1 if assembly is possible in between part i and j part and direction of assembly is in positive direction otherwise

A_{ij}=0 and A_{ii}=0 because the part cannot be assembled with itself.

3.8 Objective Function of Assembly Sequence

Energy function, E_{sequence} , is associated with assembly sequence can be represented as:

$$E_{\text{sequence}} = E_J + E_P + E_C \tag{3.10}$$

Where,

E_{sequence} = Energy function related with ASG

E_J, E_P and E_C = Energy related to Assembly cost, Precedence constraints and Connectivity

where,

C_J = an energy constant associated to assembly sequence cost J.

The value of J is expressed as:

$$J = \begin{cases} 1: if\ assembly\ sequence\ is\ unstable \\ \rho_s C_{as} + \rho_t C_{nt} : otherwise \end{cases} \tag{3.11}$$

The energy linked with precedence constraints is:

$$E_P = C_P \sum_{i=1}^{n} \mu_i \tag{3.12}$$

where,

C_P = positive constant and

μ_i = precedence index which is allocated to 0, if it fulfils the precedence constraints, otherwise 1.

The energy associated with connectivity is: $E_C = C_C \sum_{i=1}^{n} \lambda_i$ (3.13)

In a similar manner connectivity index λ_i is enumerated on the basis of liaison relationships.

The objective of the present work is to determine stable, feasible, and optimal robotic assembly sequence with minimum assembly cost. The objective function for the robotic assembly is given by [24].

$$E_{seq} = C_J\ J + \sum_{i=1}^{n} (C_P \mu_i + C_C \lambda_i) \tag{3.14}$$

3.9 Motion Instability and Assembly Direction Changes

The probable direction of assembly sets $DS^k_{cdabe}s(k = c,b,a,d,e)$ for each part of a sequence and the ordered lists $DL_i^{cbade}(i = 1,2,.....,m)$ of possible assembly directions corresponding to the assembly sequence are given by:

$$\begin{aligned}
&p_c:\ DS^c_{cbade} = \{d^c_j \in D \mid j = 1,2,.....\} && DL_1^{cbade} = \left\{d^c_1, d^b_1, d^a_1, d^d_1, d^e_1\right\}\\
&p_b:\ DS^b_{cbade} = \{d^b_j \in D \mid j = 1,2,.....\} && DL_2^{cbade} = \left\{d^c_2, d^b_2, d^a_2, d^d_2, d^e_2\right\}\\
&p_a:\ DS^a_{cbade} = \{d^a_j \in D \mid j = 1,2,.....\} && \cdots\\
&p_d:\ DS^d_{cbade} = \{d^d_j \in D \mid j = 1,2,.....\} && \cdots\\
&p_e:\ DS^e_{cbade} = \{d^e_j \in D \mid j = 1,2,.....\} && DL_m^{cbade} = \left\{d^c_m, d^b_m, d^a_m, d^d_m, d^e_m\right\}
\end{aligned}$$

Standardized degree of motion instability based upon the above equation, and the number of assembly direction changes are estimated [24]. The formula for standardized motion instability C_{as} is:

$$C_{as} = \frac{1}{m}\sum_{i=1}^{m}\left\{\frac{1}{12\times i}\sum_{j=1}^{i}\left(S\{BA_j\})_i\right)\right\} \tag{3.15}$$

Where $BA_j\,(j = 1,2,3,4,5)$ is the in-subassembly generated at the j^{th} assembly step, and $S\{BA_j\}$ is the degree of motion instability of the j^{th} subassembly. Similarly, the number of direction changes C_{nt} can be given as:

$$C_{nt} = \frac{1}{m}\sum_{i=1}^{m}\left\{\frac{1}{i}\sum_{j=1}^{i}(NT_j)_i\right\} \tag{3.16}$$

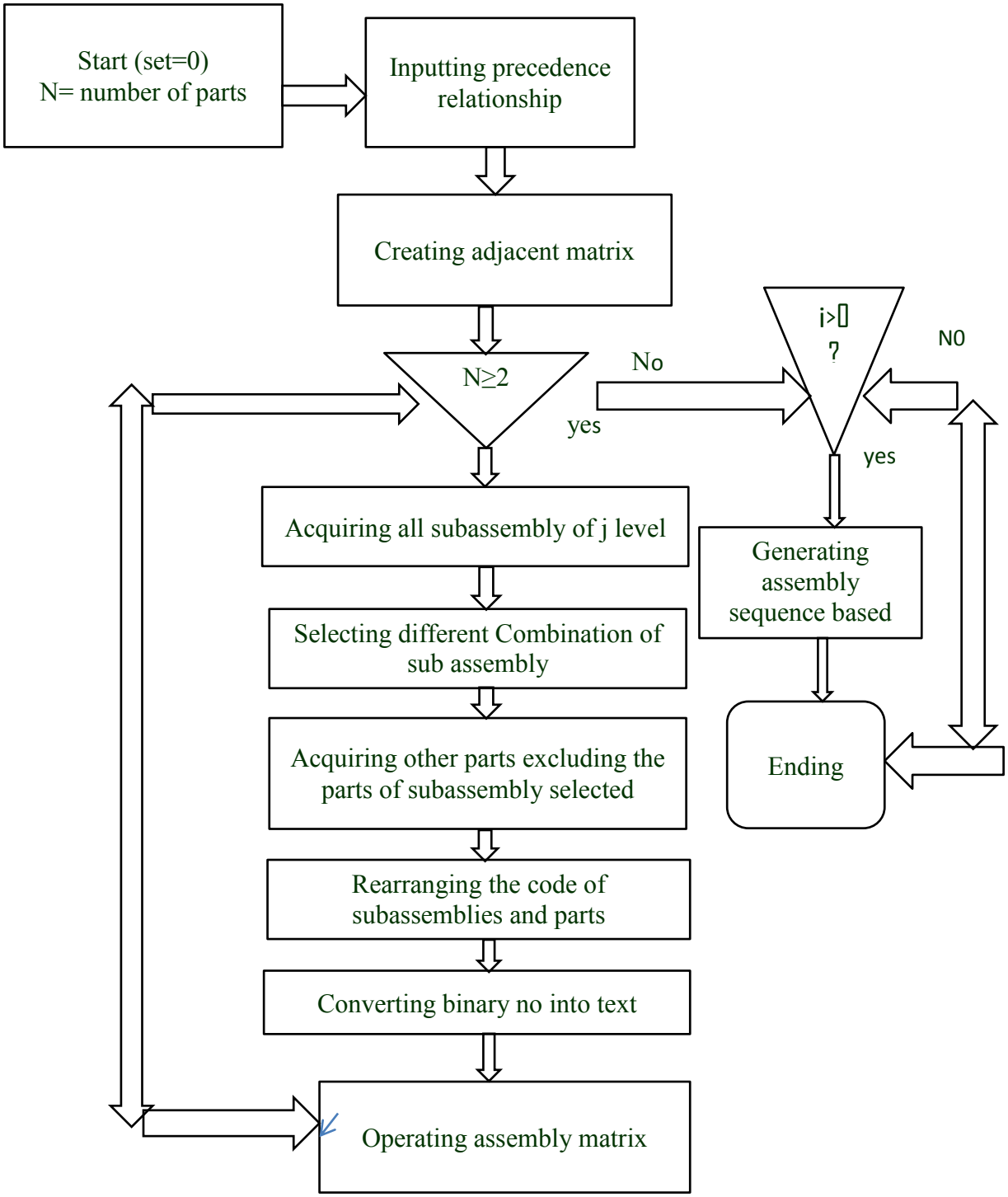

Figure 3.2: Algorithm for generating feasible assembly sequences.

3.10 Summary

A systematic way of assembly sequence generation is presented. This method is based upon getting stable and feasible assembly sequence. The basic concept is based upon the observation that, when a part should be assembled there should be no direction change. The assembly sequence should follow the precedence constraints and connectivity constraints. The assembly sequence thus obtained is called stable and feasible assembly sequence. Evolution of assembly matrix and instability of base assembly motion is presented in this chapter which is initial step of assembly sequence generation. The objective function for robotic assembly prepared based upon cost precedence constraint and connectivity constraints.

Chapter 4

Methods for Assembly Sequence Generation

4.1 Overview

A variety of optimization tools are existing for application to the problem of optimizing assembly sequence, but their appropriateness and usefulness are also under scanner. Finding the best sequence generation comprises the conventional or soft-computing methods based upon following the procedures of search algorithms. Examples of such techniques are: Simulated Annealing, Tabu Search, Neural network, Evolutionary Computation, Ant Colony Optimization, Particle swarm optimization and Novel Immune System. Study of various optimization methods reveals that Particle swarm optimization and Novel Immune System (AIS) technique can be a conveniently used to solve such kind of problems. Earlier numbers of methods have been proposed on robotic assembly sequence generation, but all of them have the problem of search space explosions. To overcome such situation a method has been modified to generate optimal assembly sequence using Particle swarm optimization and Novel Immune System. It is best suitable for combinatorial optimization problems.

4.2 Comparison of Evolutionary Techniques

4.2.1 Comparisons between Genetic Algorithm and PSO

Most of evolutionary techniques have the following procedure:

1. To generate an initial population randomly

2. Reckoning of a fitness value.

3. Regeneration of the population which based on fitness values.

4. If criteria are fulfilled, then stop, otherwise go back to 2.

From the procedure, we know that PSO has many common points as GA. Both algorithms start from a group of a population generated randomly, to evaluate the population both have fitness values. In both the technique population is updated and search for the optimum with random techniques.

However, PSO does not have operators like mutation and crossover. Particles are update with the internal velocity. They have memory that is important to the algorithm. The information sharing mechanism in PSO is significantly different compared with genetic algorithms (GAs). In GAs, chromosomes relate each other. So the entire population moves like a single group towards an optimal search area. In PSO, only g_{best} gives out the information to others. It is a one way information distribution mechanism. The estimation only looks for the best solution. All the particles tend to unite to the best solution quickly compared with GA.

4.2.2 Artificial Neural Network and PSO

An artificial neural network (ANN) is an analysis paradigm. Recently there have been substantial research efforts to apply evolutionary computation (EC) techniques for the purposes of estimating one or more phases of artificial neural networks.

Evolutionary computation methodologies have been applied to three main aspects of neural networks: network architecture, network connection weights and network learning algorithms. Most of the work including the estimation of ANN has focused on the network topological structure and weights. Usually the weights and/or topological structure are encoded as a chromosome in GA. The selection of fitness function depends on the research goals.

The benefit of the EC is that EC can be used in cases with non-differentiable PE transfer purposes and no gradient information available.

The disadvantages are

1. The performance is not modest in some problems.

2. Representation of the weights is hard and the genetic operators have to be prudently selected or developed.

There are numerous papers stated using PSO to substitute the back-propagation learning algorithm in ANN in the past several years. It presented PSO is a auspicious method to train ANN. It is faster and gets better results in most cases. It also avoids some of the problems GA encountered.

4.2.3 Comparison between IOA and GA

First, GA selects the individuals of the next generation only based on fitness level of the individuals; hence the search of optimal individual is limited in the direction of good quality individuals, making the algorithm tend to converge prematurely at local optimal solutions. Unlike GA, IOA introduces an immune selection operation to take into account the fitness, the concentration and the affinity of the antibody when choosing the individuals of the next generation. Accordingly, maintenance of population diversity can be achieved, which helps to avoid premature convergence and increase the opportunity of global optimization.

Secondly, GA lacks the capability of local search, thus it usually misses the optimal individual. However, in IOA, the clonal selection operation is employed to enhance the local search by intensifying the exploitation of known space, which helps the algorithm converge rapidly.

Finally, assembly planning is a problem with intensive constraints, and in GA, usually a large number of low fitness level, even infeasible, solutions are generated; furthermore, population degradation cannot be avoided in the evolution process of the population. All of these seriously influence the efficiency of the search for the optimal assembly sequence. By introduction of the immune operation of inoculation, IOA improves the quality of solution

candidates based on the heuristic knowledge implied in the vaccines. By doing so, the validity of solution candidates is improved. Therefore, the search for the optimal assembly sequence is accelerated and improvement in efficiency of the algorithm is achieved.

4.3 Immune Optimization Concept

Artificial Immune Systems (AIS) are computational paradigms that belong to the computational intelligence family and are inspired by the biological immune system. During the past decade, they have attracted a lot of interest from researchers aiming to develop immune-based models and techniques to solve complex computational or engineering problems. This work presents a survey of existing AIS models and algorithms with a focus on the last five years. The Clonal Selection principle is the whole process of antigen recognition, cell proliferation and differentiation into memory. Several artificial immune algorithms have been developed imitating the clonal selection theory. In the artificial immune system a population of N antibodies is generated, each specifying a random solution for the optimization process. During each iteration, some of the best existing antibodies are selected, cloned and mutated in order to construct a new candidate population. New antibodies are then evaluated and certain percentage of the best antibodies is added to the original population. Finally a percentage of worst antibodies of previous generation are replaced with new randomly create ones.

AIS is a computational intelligence approach with powerful problem- solving capability. Basically, the following bionic principles of AIS lay the foundation for IOA is proposed in this paper.

- Immune regulation
- Clonal selection principle
- Vaccines and inoculation
- Immune metabolism

4.4 Applying Immune Optimization Concept to Assembly Sequence Generation

Here two immune based algorithms namely Clonal selection and Affinity maturation principles are used to find out the best assembly sequence from the possible assembly sequences. When antigens are entered into human body, antibodies are released from the immune component known as B-cells. Then the produced antibodies interact with the recognized foreign invaders and reduce their effect on the human body (Castro and Zuben, 1999). In this way, the immune system protects human body from the wide variety of harmful foreign agents. Similarly, each assembly sequence is considered here as an antibody and each antibody is produced according to the affinity maturation principle. In Immune optimization approach two phase mutation has been taken for generating assembly sequence (antibody). In the first phase, two positions are selected randomly and are called pair-wise interchange mutation. In the second phase, the considered positions are inversed and are called inverse mutation. After generating each antibody, the next step is to calculate its affinity strength in order to select suitable antibody. Since each assembly (antibody) has some energy value corresponding to the affinity value of that particular antibody, the affinity value of each antibody can be calculated as follows:

$$\text{Affinity}\ (p) = 1 / E_{Seq} \tag{4.1}$$

Where, $E_{Sequence}$ is the energy value of an individual assembly. For stable sequence the energy element is low and for unstable sequence, it is high. Lower the energy element greater is the affinity value. The cloning of antibodies is directly proportional to the affinity function. More clones are produced on higher affinity values or lower energy values.

While generating each antibody using maturation principle, the antibodies are stored in the sequential order, if the affinity value is higher than the original value; otherwise, it stores the

original value. In receptor editing, poorest percentage of antibodies are eliminated and randomly created antibodies are replaced. This mechanism causes to new search regions in the total search space.

AIS is realized by the

Following steps:

(1) Recognition of antigens;

(2) Generation of initial antibodies.

(3) Evaluation of antibodies, i.e. calculations of the fitness, the affinity and the concentration of the antibodies.

(4) Proliferation and suppression of antibodies, i.e. conducting the immune selection operation to proliferate high fitness level antibodies and suppress high concentration level.

(5) Generation of new antibodies, i.e. conducting the crossover and the clonal selection operation to generate the next generation antibodies.

(6) Improvement of antibodies, i.e. partially adjusting solution candidates with vaccines to make the candidates approach the optimal solution.

Steps 3–6 will be repeated until convergence criteria are satisfied. Basically, AIS is a kind of general optimization approach that can be applied to solve many problems. The research work utilizes the AIS which introduce an immune selection operation to take into account the fitness/energy function, the concentration and the affinity of the antibody/stable sequences when choosing the individuals of the next generation. Simultaneously, maintenance of population diversity can be achieved, which helps to avoid premature convergence and increase the opportunity of global optimization. By introducing immune operation of inoculation in the form of stability conditions of the robotic assembly, the validity of solution candidates/sequences is improved. Therefore, the search for the optimal assembly sequence is accelerated and improvement in efficiency of the algorithm is achieved. In AIS, the clonal

selection operation is employed to enhance the local search by intensifying the exploitation of known space, which helps the algorithm converge rapidly. AIS is found to be interesting and suitable for such kind of formulations and hence it has been chosen for being applied to obtain optimized assembly sequence with additional constraints such as precedence and connectivity constraints. Figure 4.1 shows the algorithm of process of assembly planning based on immune optimization approach.

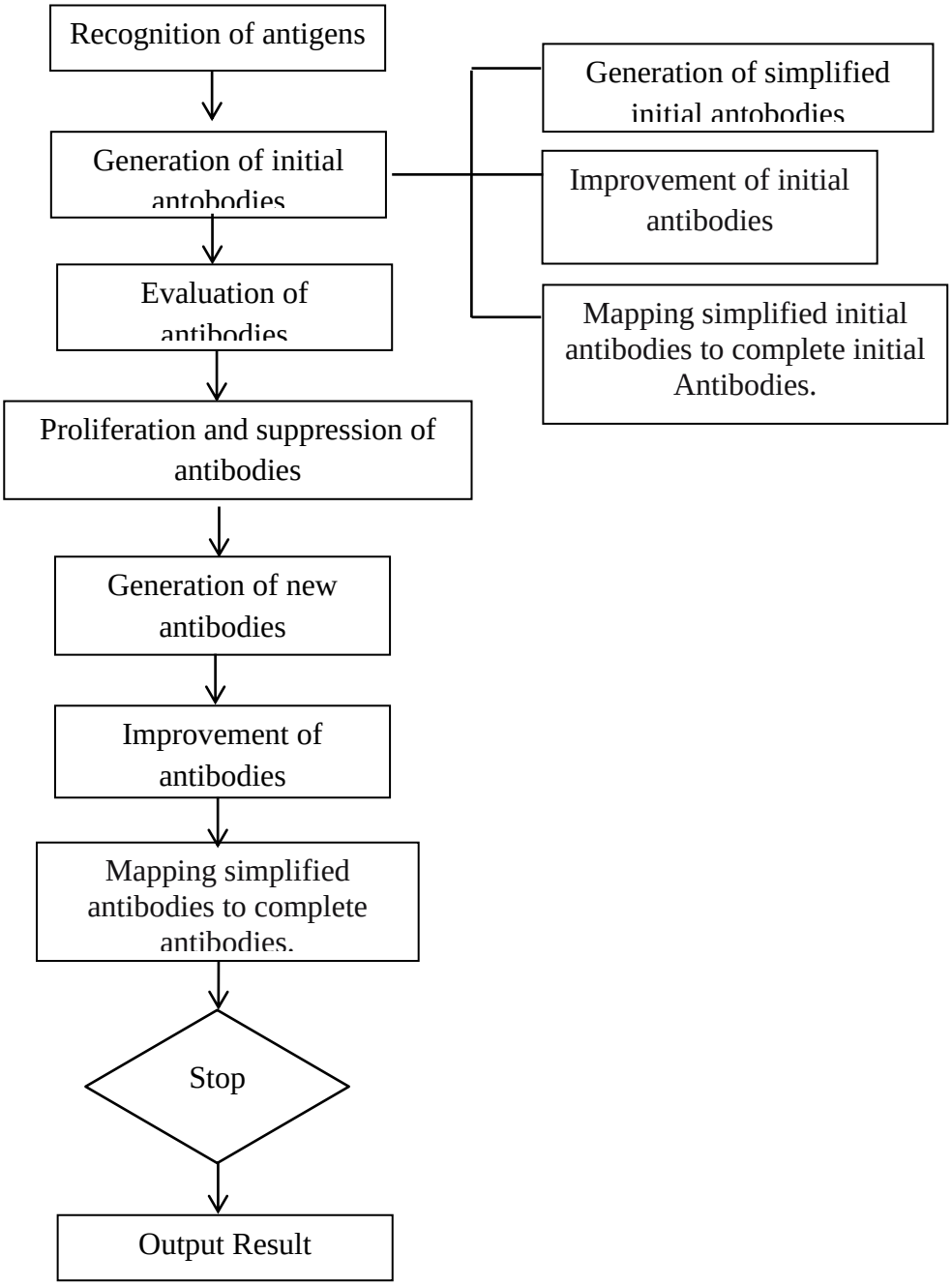

Figure 4.1: Process of assembly planning based on IOA.

4.5 Case Study

Example problem 1: A Gear train assembly as shown in Fig 4.2(a) is considered for determining the assembly sequence and authorising the proposed method. Fig 4.2(b) shows the directions for assembly or disassembly and Fig 4.2(c) represents the liaison diagram of the components

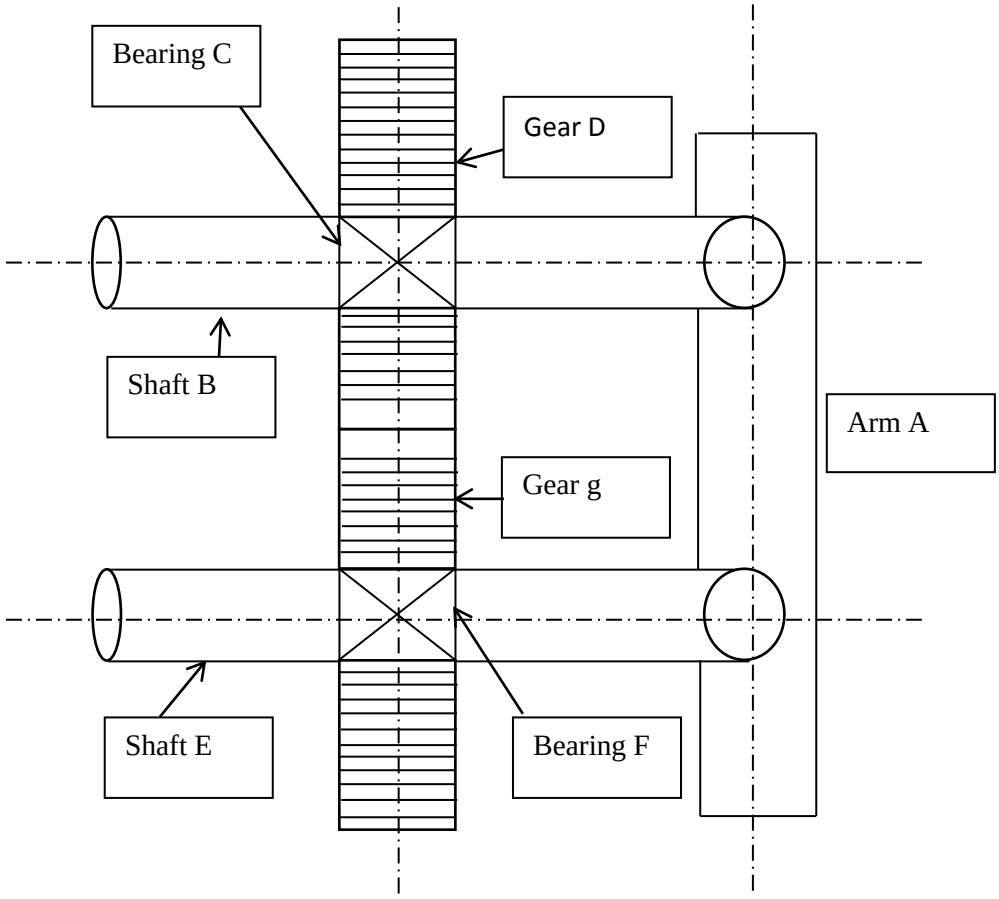

Figure 4.2 (a): Gear train assembly.

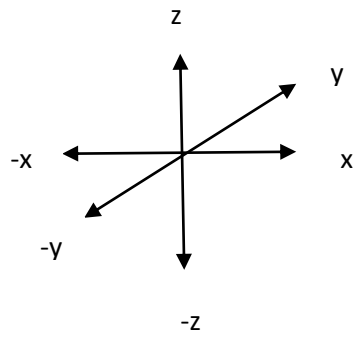

Figure 4.2 (b): Directions for assembly or disassembly.

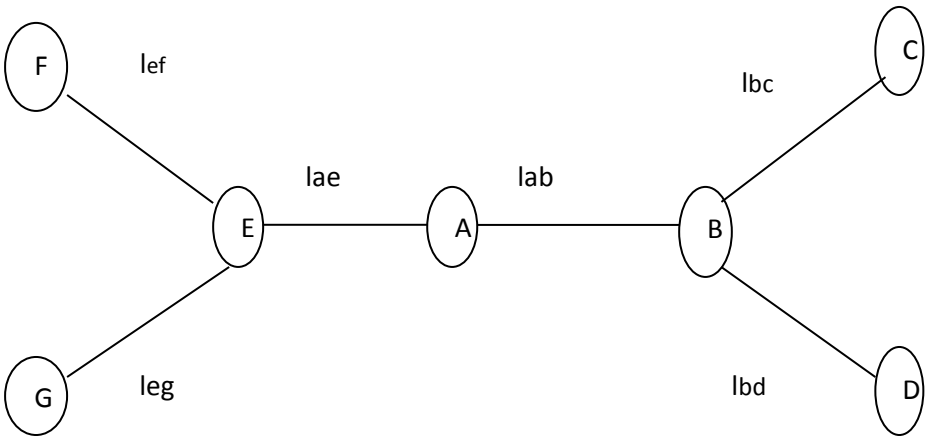

Figure 4.2 (c): Liaison graph model.

The liaisons of the components are represented as:

$$l_{ab} = liaison\left(a, \begin{pmatrix} 0 & c & c \\ c & c & c \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ s & 0 & 0 \end{pmatrix}, b\right) \tag{4.2}$$

$$l_{bc} = liaison\left(b, \begin{pmatrix} 0 & c & c \\ c & c & c \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ p & 0 & 0 \end{pmatrix}, c\right) \tag{4.3}$$

$$l_{bd} = liaison\left(b, \begin{pmatrix} 0 & v & v \\ v & v & v \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ p & 0 & 0 \end{pmatrix}, d\right) \quad (4.4)$$

$$l_{ae} = liaison\left(a, \begin{pmatrix} 0 & c & c \\ c & c & c \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ s & 0 & 0 \end{pmatrix}, b\right) \quad (4.5)$$

$$l_{ef} = liaison\left(b, \begin{pmatrix} 0 & c & c \\ c & c & c \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ p & 0 & 0 \end{pmatrix}, c\right) \quad (4.6)$$

$$l_{eg} = liaison\left(b, \begin{pmatrix} 0 & v & v \\ v & v & v \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ p & 0 & 0 \end{pmatrix}, d\right) \quad (4.7)$$

Example problem 2: A grinder assembly as shown in Fig 4.3(a) is considered as a case study for determination of the assembly sequence and authorizing the proposed method. Fig 4.3(b) shows the directions for assembly or disassembly operations whereas Fig 4.3(c) represents the liaison diagram of product. Since the product is having five parts, the number of possible sequences is 120.

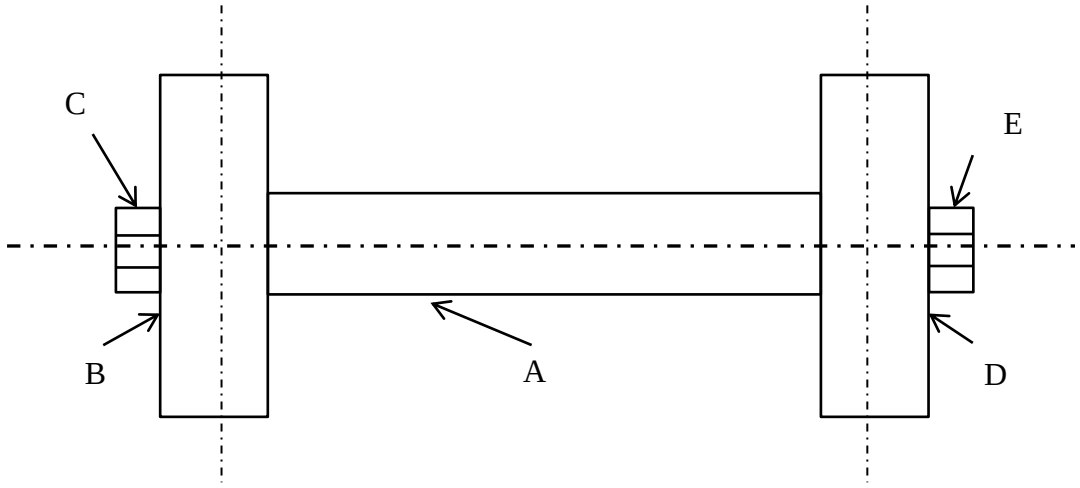

Figure 4.3 (a): An example of a product (Grinder assembly) [24].

Table 4.1: Part description

Part symbol	Part name
A	Shaft
B	Blade
C	Nut
D	Blade
E	Nut

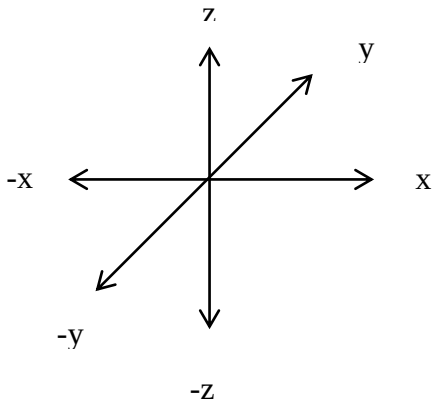

Figure 4.3 (b): Directions for assembly or disassembly.

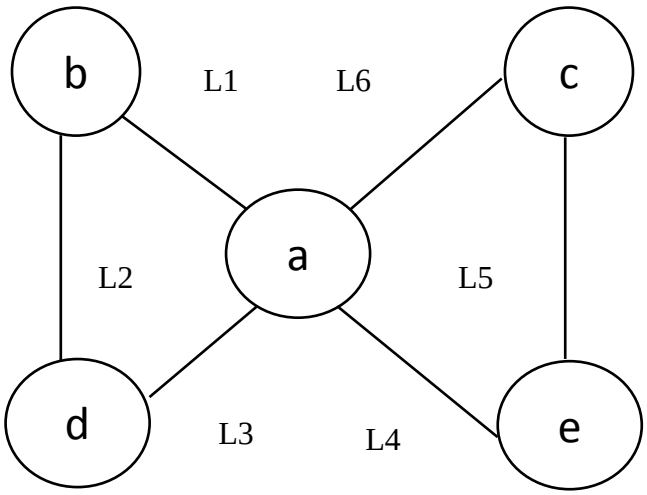

Figure 4.3(c): Liaison graph model of grinder. [a- shaft; b-blade; c-nut; d-blade, and e-nut].

The liaisons of the components are represented as follows:

$$l_{ab} = liaison\left(a, \begin{pmatrix} o & rc & rc \\ rc & rc & rc \end{pmatrix}, \begin{pmatrix} o & o & o \\ rf & o & o \end{pmatrix}, b\right); \quad l_{ac} = liaison\left(a, \begin{pmatrix} o & o & o \\ vc & o & o \end{pmatrix}, \begin{pmatrix} o & o & o \\ sw & o & o \end{pmatrix}, c\right)$$

Similarly the liaisons for connections can be obtained for a-d, a-e, b-c and d-e.

The system will check whether the generated sequence is obeying all assembly constraints or not while generating antibodies (sequences) according to maturation principle. If the generated sequence is not satisfying the assembly constraints then its affinity strength is assigned to be '0'.

4.6 Assembly Constraints for Product

Among the five parts, part a (shaft) is considered as the base part or initial part. According to this consideration, from the different 120 number of possible sequences, only 24 (=4!) sequences are having the affinity value greater than '0' and remaining are assigned to the affinity score of '0'. The precedence constraints have to be applied for the parts 'c' and 'e'. It means 'c' can assembled to the structure only after 'b' assembled, similarly 'e' can assembled to the structure only after'd' assembled. Applying this, the number of sequences is then reduced from 24 to 6. It means 18 sequences are having the affinity strength of '0' and remaining 6 sequences are having affinity value more than '0'.

Therefore the feasible sequences are: i) a-b-c-d-e, ii) a-b-d-c-e, iii) a-b-d-e-c, iv) a-d-b-c-e, v) a-d-b-e-c, and vi) a-d-e-b-c.

4.7 Calculation for E_{seq}

The total energy level for a complete sequence need to be calculated and for that, it is necessary to first find out the energy level of each partial sequence. Let the partial sequence be considered here as 'a-b'.

a) Value of E_P for 'a-b':

$$E_P = C_P \sum_{i=1}^{n} \mu_i = C_P(\mu_a + \mu_b)$$

μ_a= 1 since part-a is not following any constraint

μ_b= 0 since part-b is following by constraint (part-a)

And C_p=35 is a constant chosen arbitrary.

$\therefore E_p = 35(1+0)=35$

b) Value of E_c for 'a-b':

$$E_c = C_c \sum_{i=1}^{n} \lambda_i = C_c(\lambda_a + \lambda_b)$$

$\lambda_a = \lambda_b$ =0; since the generated sequence is stable

and C_c=35 is a constant chosen arbitrary.

$\therefore E_c = 35(0+0)=0$

c) Value of E_j for 'a-b':

$E_J = C_J J$

And $J = \rho_s C_{as} + \rho_t C_{nt}$ since the generated sequence is stable

Here C_{nt} = 0 because there is no change in assembling direction.

And $C_{as} = \frac{1}{m}\sum_{i=1}^{m}\left\{\frac{1}{12\times i}\sum_{j=1}^{i}\left(S\{BA_j\}\right)_i\right\} = \frac{1}{m}\left\{\frac{1}{12\times i}\left(dof(a)+dof(b)\right)\right\}$

for the considering sequence part-a is fixed so its degrees of freedom is '0' and for part-b, 'dof' is '2'.

Therefore, $C_{as} = \frac{1}{1}\left(\frac{1}{12}(0+2)\right) = \frac{1}{6}$; and

$E_J = C_J \mathrm{x}\, J = 45 * 0.5 * \frac{1}{6} = 3.75$ (Here C_J=45 and ρ_s=0.5 values have been considered)

Hence the total energy level for partial sequence 'a-b' is:

$$E_{seq} = C_J\ J + \sum_{i=1}^{n}(C_P\mu_i + C_C\lambda_i) = 3.75 + 35 + 0 = 38.75$$

In similar manner, total energy level for each generated assembly sequence (antibody) can be calculated. The next step is to find out each antibody's affinity strength in terms of sequence energy level. Table 2 represents the feasible assembly sequences and their affinity strengths generated by artificial immune system algorithm.

4.8 Particle Swarm Optimization

A basic alternative of the PSO algorithm works by taking a population (called a swarm) of candidate solutions (called particles). These particles are stimulated around in the search-space according to some simple formulae. PSO simulates the actions of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is, but they know how far they have to go to search the food in each iteration. The operative one is to follow the bird which is adjacent to the food. All of particles have some fitness values which are estimated by the fitness function to be optimized. The particles have to fly through the problem space and follow the current optimum particles. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has been attained so far. (The fitness value is also stored.) This value is considered as pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value. This best value is a global best and called gbest. When a particle precedes part of the population as its topological neighbours, the best value is a local best and is called lbest.

Particle swarm optimization algorithm has two primary operators: Velocity update and Position update. During each generation, each particle is enhanced toward the particles previous best position and the global best position. At each iteration, a new velocity value for each particle is evaluated based on its current velocity, the distance from its previous best position, and the distance from the global best position. The new velocity value is then used to find out the next position of the particle in the search space.

4.9 PSO Algorithm

- For each particle $i = 1, ..., S$ do:
  - Initialize the position of particle
  - Initialize the velocity of particle
  - Initialize the best known position of particle to its initial position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  - If ($f(\mathbf{p}_i) < f(\mathbf{g})$) update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
- Until a termination criterion is satisfied (e.g. number of iterations performed, or adequate fitness reached), repeat:
  - For each particle $i = 1, ..., S$ do:
    - For each dimension $d = 1, ..., n$ do:
      - Pick random numbers within the range [0,1]
      - Update the particle's velocity
      - Update the particle's position
      - If ($f(\mathbf{x}_i) < f(\mathbf{p}_i)$) do:
      - Update the particle's best known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
      - If ($f(\mathbf{p}_i) < f(\mathbf{g})$) update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
- Now **g** holds the best found solution

Figure 4.4 shows the algorithm of process of assembly planning by Particle swarm optimization with mutation operation.

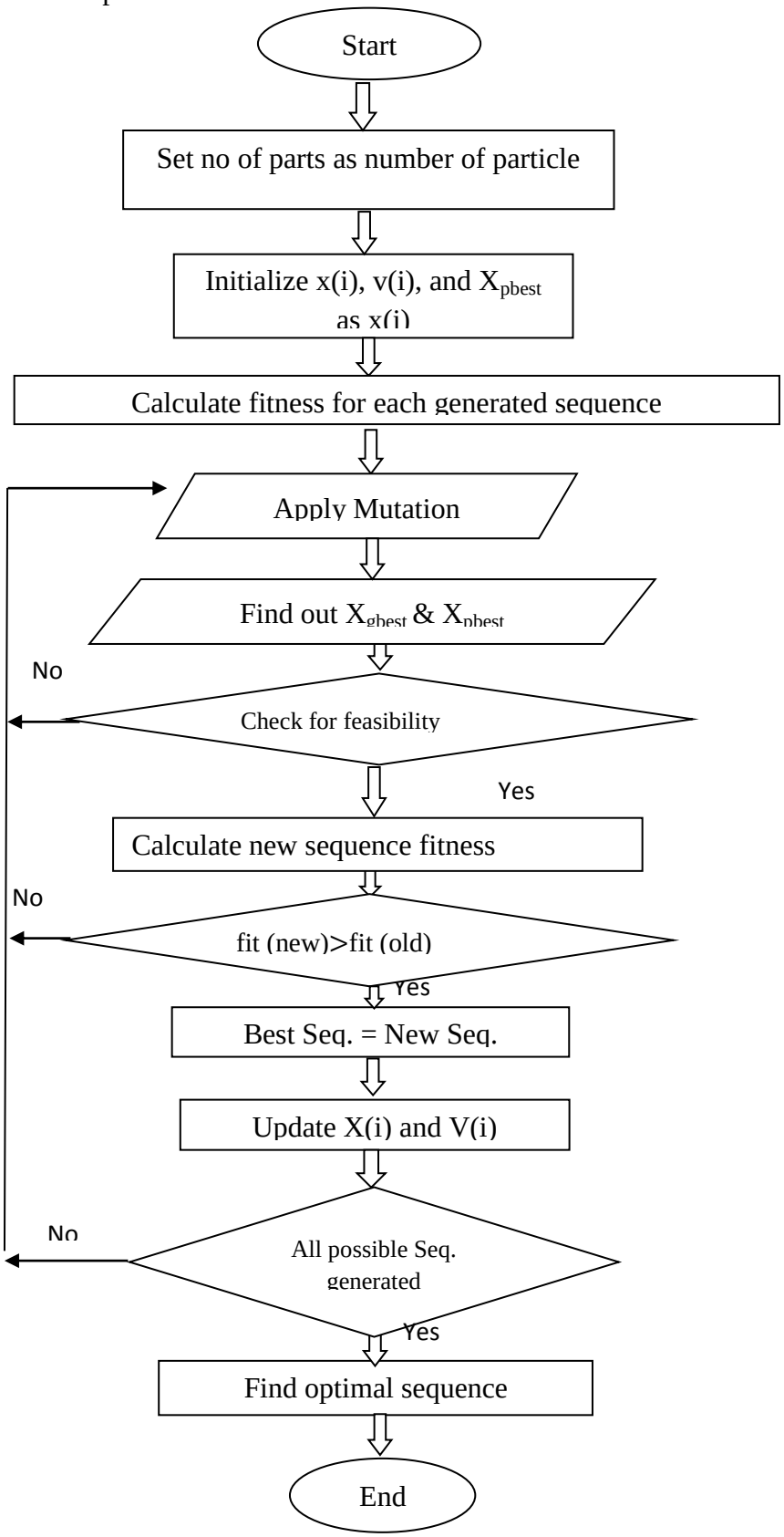

Figure 4.4: Flow chart for the PSO methodology

4.10 Formulation of the Fitness Function

The objective of the optimization of an assembly planning is to minimize the assembly costs by satisfying the assembly constraint .The fitness function is given in terms of energy sequence which depends on assembly direction, motion instability, precedence constraint and connectivity constraint. For an assembly product, if a given assembly sequence is feasible, and then the fitness function of the assembly sequence can be given as follows:

$$F_i = \frac{(1/E_{seq\_i})}{\sum_{j=1}^{n_{pop}}(1/E_{seq\_j})} \tag{4.8}$$

Where $E_{seq} = E_J + E_P + E_C$

and E_{seqi} = Energy function associated with i^{th} sequence.

E_{seqj} = Energy function associated with j^{th} sequence.

E_{seq} = Energy function associated with ASG.

4.11 Applying Particle Swarm Optimization for Assembly Sequence Generation

Three assumptions are taken into consideration while simplifying the optimization model of the assembly sequences.

1. All the parts are rigid, which means that the parts are not deformable in the assembly process. The effect of the assembly tolerance is also neglected.
2. The assembly directions are restricted to $\pm X, \pm Y, \pm Z$ direction in the three orthogonal coordinate axes.
3. Only one part is assembled along one direction.

In this paper PSO with mutation operation is used to determine the optimal assembly sequence. Optimal assembly sequence is generated by following steps:

Step1: consider each part (a,b,c…) as the each individual in the swarm. And initialize each position and velocity values for each individual randomly for 1 to n (number of parts) as illustrated in Table 4.2.

Table 4.2: Initial position and velocity of each individual (part)

Individual Part name	**a**	**b**	**c**	**. . .**	**p**
Position x(i)	1	2	3	. . .	n
Velocity v(i)	1	2	3	. . .	n

Step2: apply mutation operation for two parts by keeping one fixed part with respect to all other parts. For example, a product is constructed with three individual parts say a,b and c and allocate it position values randomly a to 1, b to 2 and c to 3 as illustrated in Table 4.3. Then apply mutation operation to the primary sequence 'a-b-c' such that 'a' (fixed part) with respect to 'b' and then 'c'. So the generated sequence in the second and third iterations will be b-a-c and c-b-a respectively.

Table 4.3: Position value of each part during Mutation operation

Position x(i)	**1**	**2**	**3**
Random sequence	a	b	c
2nd iterative sequence	**b**	**a**	c
3rd iterative sequence	**c**	b	**a**

Step3: updating position and velocity of each individual (part).

To update position and velocity a new parameter is introduced here named as 'position shift'. Since there are two parts while the mutation operation is applying, so the parameter, position shift can be defined as follows:

Position shift = position value (second part) - position value (first part) (4.9)

Position update: Position of particle is updated according to the equation as follows.

$$x(t+1) = x(i) + v(t+1) \quad (4.10)$$

But in the current methodology we are updating position as represented by Eq.(4.11):

$$x(t+1) = x(i) + \text{position shift} \quad (4.11)$$

Velocity update: For updating velocity of particle it is necessary to find out X_{gbest} and X_{pbest} according to the equation (4).

$$v_{i,j}(t+1) = v_i(t) + c_1 * r_1 * [x_{pbest} - x(i)] + c_2 * r_2 * [x_{gbest} - x(i)] \quad (4.12)$$

Where c_1 and c_2 are acceleration coefficients, t represents the iteration number, r_1 and r_2 are random numbers between [0, 1], i (i=1,2,…, n) is the index representing the particles in the swarm.

Finding X_{gbest} & X_{pbest}: X_{gbest} can be obtained after applying mutation operations to one fixed part with respect to all other parts. During the change in position of fixed part, which sequence is giving the optimal fitness value followed by assembly constraints, the position of the fixed part in that sequence is treated as the X_{gbest}. Representation of X_{pbest} & X_{gbest} is given below in Table 4.4.

Table 4.4: Representation of X_{pbest} & X_{gbest}

Parts	**X(i)**	**X(i+1)**	**Position shift**	**X_{pbest}**	**X_{gbest}**
a	1	2	1	2	
b	2	1	-1	1	For 1st iteration '1'
c	3	3	0	3	
					For 2nd iteration '2'

In this table 4.4 the initial position of a is 1 but after mutation operation the position of a is 2. Let us consider b-a-c is having optimal fitness value. So X_{gbest} for this

sequence will be 2 because the position of ‘a’ is 2. X_{pbest} is the initial best position of particle so X_{pbest} for ‘a’ is 1.

X_{pbest} of each part is nothing but the new position of the corresponding part and the cycles will be processed as follows:

1st cycle:

a. Consider initial position of each particle in swarm as its position best.
b. Find X_{gbest} by calculating fitness of each sequence using Eq.(4.8).
c. Calculate new velocities of particle using Eq. (4.12).
d. Calculate new positions of particle using Eq. (4.11).

2nd cycle:

a. Load updated positions & velocities of each particle from first cycle.
b. Consider new positions of particles are their position bests.
c. Find X_{gbest} by calculating fitness of each sequence using Eq.(4.8).
d. Calculate new velocities of particle using Eq. (4.12).
e. Calculate new positions of particle using Eq. (4.11).

3rd cycle:

a. Load updated positions & velocities of each particle from second cycle.
b. Consider new positions from 2nd cycle as their position bests.
c. Find X_{gbest} by calculating fitness of each sequence using Eq.(4.8).
d. Calculate new velocities of particle using Eq. (4.12). ($0<v_i<5$)
e. Calculate new positions of particle using Eq. (4.11).

.

.

. and so on

Step4: once generating robotic sequence in each iteration, feasibility of the sequence is to be checked. If the generated sequence is feasible, the next step is to find out its fitness value using eq. (4.8). Later the fitness of the updated sequence is to be compared with previous sequence fitness. If the updated sequence is giving the best fitness value then PSO (with mutation) iterations will be continued with new sequence, otherwise cycles will be continued with old sequence.

4.12 Case Study

A gear assembly as shown in Fig 1.is considered for generating the assembly sequence and validating the proposed method.

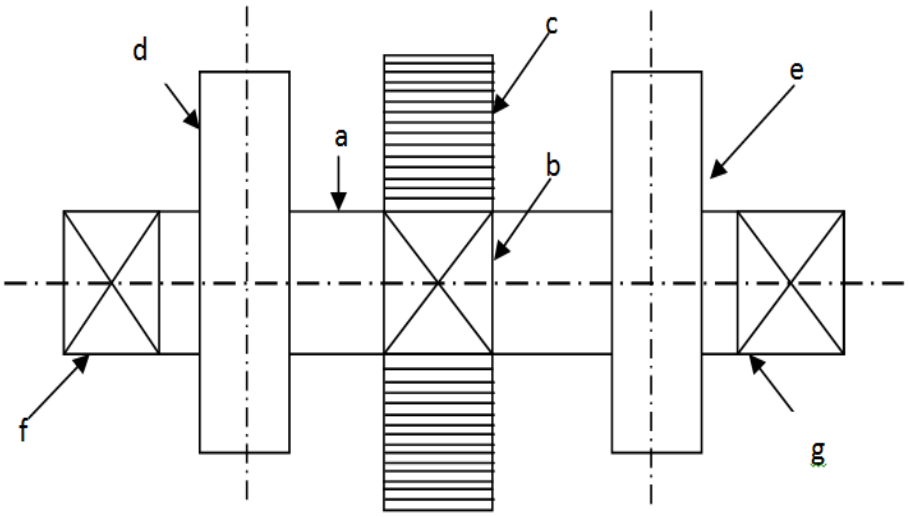

Figure 4.5 (a): A simple example of a product (Gear assembly);

[a-shaft; b-bearing; c-gear, d-pulley, and e-pulley, f-nut, g-nut]

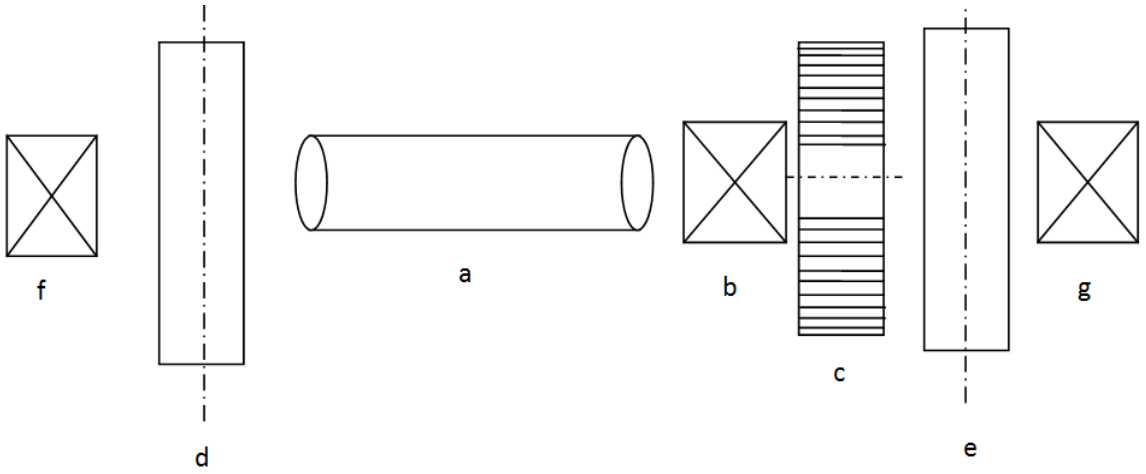

Figure 4.5(b): Blowout diagram of gear assembly;

[a-shaft; b-bearing; c-gear, d-pulley, and e- pulley, f-bearing, g- bearing]

The assembly matrix for the product is given as:

	a	b	c	d	e	f	g
a	0	+1	0	+1	-1	0	0
b	+1	0	+1	0	0	0	0
c	0	+1	0	0	0	0	0
d	0	0	0	0	0	+1	0
e	0	0	0	0	0	0	-1
f	0	0	0	+1	0	0	0
g	0	0	0	0	-1	0	0

With the help of this assembly matrix constraint are determined as:

For [c] = [b] is constraint.

Similarly [f] = [d]

[a] = [d, e]

[g] = [e]

[b] = [a]

This constraint helps in finding the feasible assembly sequences.

For each particle initially positions are assigned randomly as:

Position	1	2	3	4	5	6	7
Particle	a	f	d	b	e	g	c

Initially each particle has velocity v(i)=1. Table.5 to Table.10 represents how the position and velocity of each particle are updating and based on the position best and swarm global best values.

Initially the PSO parameters are considered as C_1= C_2= r_1= r_2= 1. During the process, velocities are constrained in the range of [0, 7].

4.13 Summary

The objective of the research work aims at developing a methodology to generate a correct set of feasible and stable assembly sequences for robotic application and to optimize the generated sequences with a view to the minimize the assembly cost function and reduce the throughput using the evolutionary computational technique of artificial immune system (AIS) and particle swarm optimization (PSO) with a number of assembly constraints. The method is expected to gain more appreciation in robotic assembly sequence optimization problems. In order to make the methodology complete and effective the proposed method considers the interrelationship between the connecting parts in each possible directions of assembly. Comparison of some important techniques is presented which shows the advantages of present techniques.

Chapter 5

Results and Discussions

5.1 Overview

The results achieved by using different procedures for the products are presented in the following sections. Essentially two different types of approaches are adopted in generating the optimized assembly sequences. In the first kind of approach Immune optimization approach for robotic assembly sequence generations have been considered. The second approach uses particle swarm optimization with mutation operation to optimize the robotic assembly sequence. The following sections present the results achieved through all these methods and the related discussions and comparisons**.**

5.2 Immune Optimization Approach for Optimization of Robotic Assembly Sequence

By applying novel immune approach on the example products following result is got. Following tables represent the feasible assembly sequences and their affinity strengths generated by artificial immune system algorithm.

Table 5.1: Feasible assembly sequences and their affinity strength for example product (Gear train assembly).

S.No.	Feasible assembly sequence (antibody)	Energy level (E_{seq})	Antibody affinity strength
1	a-b-c-d-e-f-g	127.31	0.007854
2	a-b-e-c-d-f-g	98.54	0.01014
3	a-e-b-c-d-f-g	131.07	0.00762
4	b-c-d-a-e-f-g	157.20	0.00636
5	e-f-g-a-b-c-d	138.14	0.00723
6	a-b-e-c-f-d-g	211.91	0.00471
7	a-e-b-f-c-d-g	180.90	0.00552
8	a-e-b-f-c-g-d	168.31	0.00594
9	e-a-b-f-c-g-d	155.20	0.00644

The bold texts in Table.2 represent the high value of antibody affinity and the corresponding assembly sequence (antibody) is treated as the optimal assembly sequence. For the example product the best sequence, therefore, is ‘a-b-e-c-d-f-g’.

Table 5.2: Feasible assembly sequences and their affinity strength for example product (grinder assembly).

S.No.	Feasible assembly sequence (antibody)	Energy level (E_{seq})	Antibody affinity strength
1	a-b-c-d-e	78.75	**0.0127**
2	a-b-d-c-e	115.86	0.0086
3	a-b-d-e-c	96.55	0.0104
4	a-d-b-c-e	96.55	0.0104
5	a-d-b-e-c	115.86	0.0086
6	a-d-e-b-c	78.75	**0.0127**

The bold texts in Table.2 represent the high value of antibody affinity and the corresponding assembly sequence (antibody) is treated as the optimal assembly sequence. For the example product the best sequences, therefore, are ‘a-b-c-d-e’ or ‘a-d-e-b-c’.

5.2.1 Discussion

An efficient immune based methodology has been established to determine the stable, feasible and optimal robotic assembly sequence with least assembly cost. A clear explanation has been given in order to find out the feasible and stable assembly sequence from the possible number of different alternative solutions. Later, immune based algorithms namely Clonal selection and Affinity maturation have been implemented to determine the optimal assembly sequence. During the implementation, each assembly sequence and its energy value have been considered as antibody and the antibody affinity respectively. Affinity maturation has been done in two ways, first by selecting two positions randomly and secondly by considering the inverse of the selected two positions. More clones are produced on higher

affinity values and the assembly sequence having more affinity is treated as the best assembly sequence from the possible assembly sequences.

5.3 Particle Swarm Optimization Approach for Optimization of Robotic Assembly Sequence

Followings tables shows the result obtained by particle swarm optimization (PSO) with mutation operation applied on (Gear assembly).

Table 5.3: Representation of first iterative sequence generation and updating of PSO parameters

Position X(i)	Particle (Part)	Position Shift	X_{pbest}	X_{gbest}	$X(i)_{(t+1)}$	$V(i)_{(t+1)}$	Generated Sequence	Fitness Value	Feasible (Yes/No)
1	a (fixed)	1	1		2	1			
2	f	-1	2		1	0			
3	d	0	3		3	0			
4	b	0	4	1	4	0	f-a-d-b-e-g-c	163.31	Not feasible
5	e	0	5		5	0			
6	g	0	6		6	0			
7	c	0	7		7	0			

Table 5.4: Representation of second iterative sequence generation and updating of PSO parameters

Position X(i)	Particle (Part)	Position Shift	X_{pbest}	X_{gbest}	$X(i)_{(t+1)}$	$V(i)_{(t+1)}$	Generated Sequence	Fitness Value	Feasible (Yes/No)
1	a (fixed)	1	2		3	4			
2	f	0	1		2	1			
3	d	-2	3		1	1			
4	b	0	4	1	4	0	d-f-a-b-e-g-c	216.36	Not feasible
5	e	0	5		5	0			
6	g	0	6		6	0			
7	c	0	7		7	0			

Table 5.5: Representation of third iterative sequence generation and updating of PSO parameters

Position X(i)	Particle (Part)	Position Shift	X_{pbest}	X_{gbest}	$X(i)_{(t+1)}$	$V(i)_{(t+1)}$	Generated Sequence	Fitness Value	Feasible (Yes/No)
1	a (fixed)	1	3		4	6			
2	f	0	2		2	3			
3	d	0	1		3	0			
4	b	-3	4	1	1	0	b-f-d-a-e-g-c	167.43	Not feasible
5	e	0	5		5	0			
6	g	0	6		6	0			
7	c	0	7		7	0			

Table 5.6: Representation of fourth iterative sequence generation and updating of PSO parameters

Position X(i)	Particle (Part)	Position Shift	X_{pbest}	X_{gbest}	$X(i)_{(t+1)}$	$V(i)_{(t+1)}$	Generated Sequence	Fitness Value	Feasible (Yes/No)
1	a (fixed)	1	4		5	8			
2	f	0	2		2	4			
3	d	0	3		3	3			
4	b	0	1	1	4	0	e-f-d-b-a-g-c	231.47	Not feasible
5	e	-4	5		1	0			
6	g	0	6		6	0			
7	c	0	7		7	0			

Table 5.7: Representation of fifth iterative sequence generation and updating of PSO parameters

Position X(i)	Particle (Part)	Position Shift	X_{pbest}	X_{gbest}	$X(i)_{(t+1)}$	$V(i)_{(t+1)}$	Generated Sequence	Fitness Value	Feasible (Yes/No)
1	a (fixed)	1	5		6	10			
2	f	0	2		2	5			
3	d	0	3		3	4			
4	b	0	4	1	4	3	g-f-d-b-e-a-c	180.70	Not feasible
5	e	0	1		5	0			
6	g	-5	6		1	0			
7	c	0	7		7	0			

Table 5.8: Representation of sixth iterative sequence generation and updating of PSO parameters

Position X(i)	Particle (Part)	Position Shift	X_{pbest}	X_{gbest}	$X(i)_{(t+1)}$	$V(i)_{(t+1)}$	Generated Sequence	Fitness Value	Feasible (Yes/No)
1	a (fixed)	1	6		7	12			
2	f	0	2		2	6			
3	d	0	3		3	5			
4	b	0	4	1	4	4	c-f-d-b-e-g-a	163.91	Not feasible
5	e	0	5		5	3			
6	g	-6	1		6	0			
7	c	0	7		1	0			

Since the generated sequences are all not feasible, the next iteration will be started with the sequence ‘a-f-d-b-e-g-c’ and the mutation operation is to be applied by keeping fixed part as ‘f’ with respect to all other parts.

5.3.1 Discussion

An efficient PSO based methodology has been established to determine the stable, feasible and optimal robotic assembly sequence with reduced time and minimum assembly cost. A clear explanation has been given in order to find out the feasible and stable assembly sequence from the possible number of different alternative solutions. Later, PSO based algorithm with mutation operation have been implemented to generate each possible assembly sequence. During the implementation, each part of the assembled sequence is considered as a particle. For the generated assembly sequence, after applying mutation operation in each iteration, the sequence is checked for feasibility. For all feasible sequences, fitness value is calculated and comparison of fitness values has been done between

consecutive generated sequences. Then the mutation operation is applied to the best fitness valued sequence. In similar way, mutation operation has been performed in each iteration until all possible assembly sequences have generated and finally it decides the optimal and stable robotic assembly sequence followed by the assembly constraints.

Chapter 6

Conclusions and Future Works

6.1 Overview

There are several factors which influence the manufacturing cost such as material cost, machining cost, labour cost, and assembly cost. Assembly is the final step of manufacturing a product. The cost of assembly can reach up to 30% of the manufacturing cost. Assembly of product is time consuming process, so it affects both manufacturing cost as well as productivity. Modern industries are paying attention on automated robotic assembly which is very useful for saving assembly cost as well as labour cost. There may be large number of assembly sequence for a product and as no of product increases number of assembly sequence also increases. In robotic assembly sequence slight change in direction influences total assembly sequence. It is therefore, much more important to plan the assembly process and to generate the optimal sequence of commands for the assembling robots. Without the use of suitable optimization technique it is impossible to generate correct assembly sequence.

6.2 Importance and Usefulness

The result obtained in this research is very useful for correct assembly sequence generation. Some new technique is presented which may give better result than previously applied technique. Initially procedure to determine precedence constraint and connectivity constraints is described. The fitness function is formulated based on cost, precedence constraint and connectivity constraints. Assembly matrix and liaison diagram for an example product is

shown by the help of which direction and possibility of assembly can be determined. Feasible and stable assembly sequences are generated and optimal sequence is determined by novel immune approach and particle swarm optimization. The work presents a combined approach for the selection of correct method for the generation of assembly sequences in the contest of robotic assembly system, testing of the stability of the generated sequences, and finding the optimal sequence and hence is definitely a new dimension to this subject.

6.3 Future Work

Although several methods has been used for assembly sequence generation but the aim is to find out important and easy method which can be conveniently applied on any kind of products. Some of the very important works that can be done in future are:

1. Minimization of robotic travelling time can help to reduce the cost of assembly.
2. Product should be designed so that it could be picked up easily by robotic gripper.
3. Robotic gripper should be designed to work in more degree of freedom to reduce the effect of change in direction while assembling.

References:

1. Barnes, C.J., Dalgleish, G.F., Jared, G.E.M., Swift, K.G, and Tate, S.J, Assembly sequence structures in design for assembly, Proceedings of IEEE International symposium on assembly and task planning, DOI: 10.1109/ISATP.1997.615402, pp. 164 – 169, (2002).
2. Baldin, D. F., Abell, T.E., Lui, M.-C.M., De Fazio, T.L. and Whitneu, D.E, An integrated computer aid for generating and evaluating assembly sequences for mechanical products, *IEEE Trans. Rob. Auto.,* Vol. 7, NO.1, 1991, pp.78-94, (1991).
3. Cao, P.B and Xiao, R. B , Assembly planning using a novel immune approach, the international journal of advanced manufacturing technology, DOI 10.1007/s00170-005-0235-2, pp. 770–782, 2007.
4. Castro L.N. and Zuben, F.J.V. Artificial Immune Systems: Part ii – A survey of applications, Technical report, TR-DCA 01/99, pp.1-95,1999.
5. Chen,Y.M, and Lin, C.T, A particle swarm optimization approach to optimize component placement in printed circuit board assembly, International Journal on Advance Manufacturing and Technology 35: pp,610–620, DOI 10.1007/s00170-006-0777-y,2007.

6. Edmunds, R., Kobayashi, M. and Higashi, M, Generating Optimal Disassembly Process Plans from AND/OR Relationships using a Hierarchical Genetic Algorithm, Proceedings of the 21st CIRP Design Conference, pp. 16-23, (2011).
7. Galantucci, L. M., Percoco, G. and Spina, R., Assembly and disassembly by using fuzzy logic & genetic algorithms, International journal of advanced robotic systems, vol. 1, no.2, pp. 67 – 74, (2004).
8. Hong, D.S and Cho, H.S, Optimization of robotic assembly sequences using neural network, Proceedings of international conference on intelligent robots and systems, DOI: 10.1109/IROS.1993.583103, pp. 232-239, (1993).
9. Hong, D.S and Cho, H.S, A neural-network-based computational scheme for generating optimized robotic assembly sequences, Engineering applications of artificial intelligence, vol. 8, no. 2, pp.129–145, (1995).
10. Hong, D.S and Cho, H.S, A genetic-algorithm-based approach to the generation of robotic assembly sequences, Control engineering practice, vol. 7, no. 2, pp.151-159, (1999).
11. Hong Guang L. and Cong L., An assembly sequence planning approach with a discrete particle swarm optimization algorithm, The international journal of advanced manufacturing technology, vol. 50, no. 5-8, pp. 761-770, (2010).
12. Lee, S.H., assembly planning by subassembly extraction, *proceedings of 3rd ORSA/TIMS conference on flexible manufacturing systems.*1606-1611, (1989).
13. Lee, S, Backward Assembly Planning Analysis with Assembly Cost, Proceedings, IEEE International Conference on Robotics and Automation, Digital Object Identifier: 10.1109/ROBOT.1992.220107 Page(s): 2382 - 2391 vol.3 Cited by: 2, (1992).

14. Liao, C., Tseng, C. and Luarn, P., A discrete version of particle swarm optimization for flow shop scheduling problems, Computers & Operations Research, 34(10), pp.99–111, (2007).
15. Lowe, G., and Shirinzadeh, B, Dynamic assembly sequence selection using reinforcement learning, Robotics and Automation, Proceedings, ICRA '04, IEEE International conference, Vol: 3, pp. 2633 – 2638, (2004).
16. Luiz, S. H. M. and Arthur, C. S, Representations of mechanical assembly sequences, IEEE transactions on robotics and automation, vol. 7, no. 2, pp. 211-227, (1991).
17. Mascle, C. and Figour, J, Methodological approach of sequences determination using the disassembly method, Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing, DOI: 10.1109/CIM.1990.128149, pp. 483-490, (1990).
18. Röhrdanz F, Mosemann H, Wahl F M "Constraint Evaluation for Assembly Sequence Planning". Proceedings of 1997 IEEE International Symposium on Assembly and Task Planning, Marina del Rey, California. 263-268, 1997.
19. Shiang, F.C and Yong, J.L., The application of multi-level genetic algorithms in assembly planning, Journal of Industrial Technology, vol.17, no. 4, pp. 1-9, (2001).
20. Schutte, J et al., Evaluation of a particle swarm algorithm for biomechanical optimization, Journal of Biomechanical Engineering; 127(3), Volume 4099, pp.465–74, (2006).
21. Shin, C.K., Hong, D.S and Cho, H.S, Disassemblability analysis for generating robotic assembly sequences, Proceedings of IEEE International Conference on Robotics and Automation, DOI: 10.1109/ROBOT.1995.525457, pp. 1284-1289, (1995)

22. Shen, B., Yao, M and Yi, W.S., Heuristic information based improved fuzzy discrete PSO method for solving TSP, Proceedings of 9th Pacific Rim International Conference on Artificial Intelligence, Guilin, China, pp 859–863, DOI: 10.1007/11801603_95, (2006).
23. Sugato, C. and Jan, W., A structure-oriented approach to assembly sequence planning, IEEE transactions on robotics and automation, vol. 13, no. 1,pp. 14-29, (1997).
24. Surajit, S., Biswal, B.B., Dash, P. and Choudhury, B.B, Generation of optimized robotic assembly sequence using ant colony optimization, Proceedings of 4th IEEE conference on automation science and engineering, pp. 894-899, (2008).
25. Tiam, H. E., Zhi, K.L., Walter, O. and Chuck, M., Feature-based assembly modeling and sequence generation, Computers and industrial engineering, vol. 36, no. 1, pp.17-33, (1999).
26. Wang, C., Li-shing, H. and David J. C, Heuristics for assembly sequencing and relative magazine assignment for robotic assembly, Computers and industrial engineering, vol. 34, no. 2, pp. 423-431, (1998).
27. Wang, J.F., Liu, J.H. and Zhong, Y.F, A novel ant colony algorithm for assembly sequence planning, The international journal of advanced manufacturing technology, vol. 25, no.11-12, pp.1137-1143, (2004).
28. Zhou, X., Du, pingan., Zhou, Y, A model based approach to assembly sequence planning, International Conference on Mechatronics and Automation, ICMA, DOI: 10.1109/ICMA.2007.4303611, (2007).

Appendices: **A** (Coding for generation of feasible assembly sequence)

```
#include<stdio.h>
#include<conio.h>
static char nei[5][5];
int stack[5],top=-1,size=5;
int adj[5][5]={{0,1,0,1,0},{1,0,1,0,0},{0,1,0,0,0},{1,0,0,0,1},{0,0,0,1,0}};
static int flag[5];
void adjacent(int);
void push(int);
int pop();
char valueSwitch(int );
void stackSeq();
void emptyTheStackNow();
void findNeighbour(int k[5][5]);

main()
{
  findNeighbour(adj);
  stackSeq();
  return 0;
}
void stackSeq()
{
   int i,j,sum,k;
   for(i=0;i<5;i++)
   {
   push(i);
      for(j=0;j<5;j++)
      {
         if(nei[i][j]<5)
         {
          push(nei[i][j]);
          adjacent(nei[i][j]);
          while(top!=-1)
          {
          emptyTheStackNow();
          }
          printf("\n");
          for(k=0;k<5;k++)
          {
          flag[k]=0;
          }
          sum=j+1;
          if(nei[i][sum]<5)
          push(i);
         }
      }
   }
}
void emptyTheStackNow()
{
   int k=pop();
   adjacent(k);
}
void adjacent(int z)
{
int j;
for(j=0;j<5;j++)
{
   if(adj[z][j]==1)
   {
   if(flag[j]!=1)
   {
   push(j);
   adjacent(j);
   }
   }
}
}
```

```
int pop()
{
   int val;
   if(top==-1)
   printf("stack underflow\n");
   val=stack[top];
   top=top-1;
   return val;

}
void push(int x)
{
   char c;
   flag[x]=1;
    c=valueSwitch(x);
   printf("%c",c);
   //getch();
   if(top==size)
   printf("stack overflow \n");
   top=top+1;
   stack[top]=x;
}
void findNeighbour(int k[5][5])
{
int i,j,temp;
for(i=0;i<5;i++)
  {
  for(j=0;j<5;j++)
  {
  nei[i][j]=98;
  }
  }
   for(i=0;i<5;i++)
   {
   temp=0;
      for(j=0;j<5;j++)
      {
         if(k[i][j]==1)
         {
         nei[i][temp++]=j;
         }
      }
   }
   for(i=0;i<5;i++)
  {
  for(j=0;j<5;j++)
  {
  printf("%d  ",nei[i][j]);
  }
  printf("\n");
  }
}
char valueSwitch(int l)
{
   switch(l)
   {
      case 0:return 'a';

      case 1:return 'b';

      case 2:return 'c';

      case 3:return 'd';

      case 4:return 'e';

   }
}
```

B (Coding or determination of E sequence)

```
#include<iostream.h>
#include<conio.h>
#define cp 35
#define cc 45
#define rs 0.5
#define rt 0.5
#define cj 45
    int i;
   float x,mi,li,p=0,m,df,tdf=0,NT;
   float J,Ep,Ec,Ej,Eseq,NTj,Ca=0.0,Cas,Cnf,Cn=0.0;;
void main()
   {
   x=0.0;
   for(i=1;i<=2;i++)
   {
   cout<<"Enter the value for mi for "<<i<<" th value\n";
   cin>>mi;
   x=x+mi;
   }
    Ep=cp*x;
    cout<<"EP="<<Ep<<endl;
   for(i=1;i<=2;i++)
      {
       cout<<"Enter the value for lamda Li for "<<i<<" th value\n";
       cin>>li;
       p=p+li;
       }
     Ec=cc*p;
     cout<<"Ec="<<Ec<<endl;
   cout<<"Enter the value for m \n";
   cin>>m;
      for(int j=1;j<=m;j++)
         {
          for(int a=1;a<=2;a++)
            {
             cout<<"Enter the degree of freedom for "<<a<<" th component\n";
             cin>>df;
             tdf=tdf+df;
            }
           Ca=Ca+(i/(12*j))+tdf;
          }
            Cas=(Ca/12);
            cout<<"Cas="<<Cas<<endl;
      for(int b=1;b<=m;b++)
         {
          for(int c=1;c<=m;c++)
          {
           cout<<"Enter the value for NTj for "<<b<<" th value \n";
           cin>>NT;
           NTj=(NTj+NT)*b;
           }
          // cout<<"NTj"<<NTj<<endl;
          NTj=NTj/b;
          cout<<"NTj"<<NTj<<endl;
          Cn=Cn+NTj;
         // cout<<"Cn"<<Cn<<endl;
         }
         Cnf=Cnf/m;
         cout<<"Cnf"<<Cnf<<endl;
         J=rs*Cas+rt*Cnf;
         cout<<"J="<<J<<endl;
       Ej=cj*J;
       cout<<"Ej="<<Ej<<endl;
       Eseq=Ep+Ec+Ej;
       cout<<"The value of Eseq is "<<Eseq;
     }
```

PUBLICATIONS

Journals

1. B. B. Biswal, B.B. Deepak, and Y. Rao, “Optimization of Robotic Assembly Sequence using Immune based Technique”, Accepted in journal of Manufacturing Technology Management.

2. B. B. Biswal, B.B. Deepak, and Y. Rao, “Optimization of Robotic Assembly Sequence using Prticle Swarm Optimization with Mutation Operation”, Communicated to journal of Inderscience Management(under review).