THE UNIVERSITY OF CALGARY

A Graph-Based Heuristic Approach to Automated Assembly Planning

by

Xue Yan

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF MASTER OF SCIENCE
IN MECHANICAL ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

CALGARY, ALBERTA
JANUARY, 1995

© Xue Yan 1995
The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "A Graph-Based Heuristic Approach to Automated Assembly Planning" submitted by Xue Yan in partial fulfilment of the requirements for the degree of Master of Science in Mechanical Engineering.

Dr. P. Gu  
Supervisor and Committee Chairman  
Department of Mechanical Engineering

Dr. A. Dagnino  
Alberta Research Council, and  
Adjunct professor,  
Department of Mechanical Engineering

Dr. R. Li  
Department of Geomatics Engineering

January 31, 1995

Date
ABSTRACT

CAD-directed assembly sequence planning is an essential component of automated robotic assembly task planning. Many research attempts have been made in the past to integrate computer-aided design (CAD) and robot task planning to increase efficiency and effectiveness of robotic assembly automation. This thesis presents a graph-based heuristic approach for automatic generation of assembly sequences from a feature-based data base. A feature-based representation is used to model product assembly. The automatic assembly sequence planning system utilizes four major stages to generate assembly sequences without any user intervention: 1. create connective graphs based on the product feature representation; 2. decompose an assembly into sub-groups using the connective graphs; 3. generate the disassembly sequence for each sub-group formed at the stage 2; and 4. merge the disassembly sequences of the sub-groups into a complete disassembly sequence, and convert the disassembly sequence into the final assembly sequence. The assembly planning system associated with the feature-based product model has been implemented in Smalltalk - an object-oriented programming language. Several examples are included to illustrate the approach and the heuristic algorithms. The results show that the approach can be used to automatically generate assembly sequences for robot assembly task planning directly from the feature-based database.
ACKNOWLEDGEMENTS

I wish to express my very special thanks to my supervisor, Dr. P. Gu, for his professional guidance and support throughout this work.

Special thanks are extended to Dr. A. Dagnino and Dr. R. Li, the examination committee members, for devoting their time and energy towards this thesis.

Financial support provided by the Natural Science and Engineering Council (NSERC) of Canada through Research Grant OGP 010 5754, CRD Grant 143281, Industry Canada through Intelligent Manufacturing Systems Consortium Grant, and the Department of Mechanical Engineering through Teaching Assistanship is gratefully acknowledged.
Dedicated to my Parents, my Husband -- Hongwen and my Daughter -- Justine
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval Sheet</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iv</td>
</tr>
<tr>
<td>Dedication</td>
<td>v</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Assembly Systems and Assembly Planning</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Research Objective</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Research Approach</td>
<td>7</td>
</tr>
<tr>
<td>1.5 Organization of the Thesis</td>
<td>8</td>
</tr>
<tr>
<td>Chapter 2 Literature Survey on Assembly Sequence Planning</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Literature Review</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Summary</td>
<td>20</td>
</tr>
</tbody>
</table>

vi
Chapter 3  Overview of the Assembly Planning System  . . .  22

Chapter 4  Representation of an Assembly  . . .  25
  4.1  The Feature-Based Product Design Model  . . .  25
  4.2  The Assembly Model  . . .  33
    4.2.1  Liaison Graph  . . .  34
    4.2.2  Contact Relation Graph  . . .  38
  4.3  Summary  . . .  45

Chapter 5  Assembly Sequence Planning  . . .  49
  5.1  Assembly Decomposition  . . .  49
  5.2  Sequencing  . . .  54
    5.2.1  The Priority Order of the Six Contact Directions  .  56
    5.2.2  The Determination of Collision-Free Paths for Parts Disassembly  . . .  57
    5.2.3  The Stability Considerations  . . .  65
  5.3  Synthesis  . . .  71
    5.3.1  Case I: Identical Removal Directions of P  . . .  72
    5.3.2  Case II: Different Removal Directions of P  . . .  76
  5.4  Constructing Sub-Assemblies  . . .  80
  5.5  Summary  . . .  81
<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>The System Structure and Implementation Algorithms</td>
</tr>
<tr>
<td>6.2</td>
<td>Object Oriented Class Hierarchy</td>
</tr>
<tr>
<td>6.2.1</td>
<td>GeometricEntity Class</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Part Class</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Feature Class</td>
</tr>
<tr>
<td>6.2.4</td>
<td>SubAssembly Class</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Assembly Class</td>
</tr>
<tr>
<td>6.3</td>
<td>Methods and Message Passing</td>
</tr>
<tr>
<td>6.4</td>
<td>Summary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 7</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Conclusions</td>
</tr>
<tr>
<td>7.2</td>
<td>The Contribution</td>
</tr>
<tr>
<td>7.3</td>
<td>Future Research</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Appendix A</th>
<th>The Algorithm of Assembly Decomposition</th>
</tr>
</thead>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table 4.1. Assembly Relations</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The Structure of the Assembly Planning System</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>A Vertical Machining Centre Example</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>An Example of Product Assembly I</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>A Shaft (P7) of Assembly I</td>
<td>31</td>
</tr>
<tr>
<td>4.4</td>
<td>Feature-Based Representation of the Shaft of Assembly I</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Automatic Transmission Assembly II (Defazio and Whitney)</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>The Liaison Graph of Assembly I</td>
<td>36</td>
</tr>
<tr>
<td>4.7</td>
<td>The Liaison Graph of Assembly II</td>
<td>37</td>
</tr>
<tr>
<td>4.8</td>
<td>Examples of Contact and Fitting Relations</td>
<td>40</td>
</tr>
<tr>
<td>4.9</td>
<td>Generation of Contact Relations for Cylindrical Surface Fits</td>
<td>41</td>
</tr>
<tr>
<td>4.10</td>
<td>Generation of Contact Relations for Taper Fits</td>
<td>43</td>
</tr>
<tr>
<td>4.11</td>
<td>Generation of Contact Relations for Screw-Fits</td>
<td>46</td>
</tr>
<tr>
<td>4.12</td>
<td>Contact Relations between Parts of Assembly I</td>
<td>47</td>
</tr>
<tr>
<td>4.13</td>
<td>Contact Relations between Parts of Assembly II</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>The Bi-connected Graphs for Assembly I</td>
<td>52</td>
</tr>
<tr>
<td>5.2</td>
<td>The Bi-connected Graphs for Assembly II</td>
<td>53</td>
</tr>
<tr>
<td>5.3</td>
<td>Feature Selection of the First Step for Multiple Targets</td>
<td>60</td>
</tr>
<tr>
<td>5.4</td>
<td>An Example of Multiple Targets (Case I)</td>
<td>61</td>
</tr>
<tr>
<td>5.5</td>
<td>An Example of Multiple Targets (Case II)</td>
<td>62</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.6</td>
<td>An Example of Multiple Targets (Case III)</td>
<td>63</td>
</tr>
<tr>
<td>5.7</td>
<td>Examples of Qualified Candidates with Stability</td>
<td>66</td>
</tr>
<tr>
<td>5.8</td>
<td>An Example of Multiple Removal Directions Assembly--Assembly III</td>
<td>79</td>
</tr>
<tr>
<td>6.1</td>
<td>An Object-Oriented Assembly Planning System</td>
<td>84</td>
</tr>
<tr>
<td>6.2</td>
<td>A Sample Template for Product Data Input.</td>
<td>85</td>
</tr>
<tr>
<td>6.3(a)</td>
<td>The General Algorithm for the System Implementation</td>
<td>87</td>
</tr>
<tr>
<td>6.3(b)</td>
<td>The Removable Part Search</td>
<td>88</td>
</tr>
<tr>
<td>6.3(c)</td>
<td>The Methods of Part Searching In a Defined Direction</td>
<td>89</td>
</tr>
<tr>
<td>6.3(d)</td>
<td>The Synthesis of the Sequences</td>
<td>90</td>
</tr>
<tr>
<td>6.4</td>
<td>The Hierarchical Structure of the Classes</td>
<td>92</td>
</tr>
<tr>
<td>6.5</td>
<td>An Instance of Class Part</td>
<td>96</td>
</tr>
<tr>
<td>6.6</td>
<td>An Instance of Class Feature</td>
<td>97</td>
</tr>
<tr>
<td>6.7</td>
<td>An Instance of Class SubAssembly</td>
<td>98</td>
</tr>
<tr>
<td>6.8</td>
<td>The Output of the Assembly Planning for Assembly I</td>
<td>102</td>
</tr>
<tr>
<td>6.9</td>
<td>The Output of the Assembly Planning for Assembly II</td>
<td>103</td>
</tr>
</tbody>
</table>
CHAPTER 1

1. INTRODUCTION

1.1 ASSEMBLY SYSTEMS AND ASSEMBLY PLANNING

Assembly is the act of putting together all the discrete parts into a final product. Assembly systems constitute major components of modern manufacturing systems. Up to 50% of ultimate manufactured product cost is invested in the assembly phase (Sanders 1988). Due to the importance of the assembly stage, it is desired to assemble products in the most efficient way by automation. As a result, automating assembly process has been recognized by product oriented enterprises as an important means to improve productivity, quality, and reduce labour cost (Boothroyd 1992). Assembly can be classified into three categories: electronic assembly, electric assembly, and mechanical assembly. Among them, mechanical assembly is the most complex since it involves more variables than the
other two, which typically are a matter of mounting components on a flat surface. Therefore, until recently it was quite rare for mechanical assembly operations to be highly automated.

Automated mechanical assembly can be divided into two classes based on system configurations: automatic assembly system and flexible assembly system. An automatic assembly system is made up of dedicated assembly machines for assembly operations and is designed for high-volume production. Therefore, it requires parts with predefined positions, orientations and consistent dimensions and tolerances. The system operates with fixed programs or sequences. A flexible assembly system, on the other hand, attempts to relax those restrictions of dedicated automated systems, and perform various assembly operations. Robotic assembly has been recognized as an effective approach to flexible assembly. To design an automated assembly system or a flexible assembly system for certain products, the product assembly sequences must be determined first.

Assembly processes can be defined as a sequence of operations performed by an assembly system. Assembly process planning concerns the creation of steps of assembly operations based on connectivity relationships between components from which a product is assembled. In conventional mass production using automatic assembly systems, special processes and devices for assembly operations are designed and only a small number of assembly plans are required to be prepared for a defined period of time. The current trends in industry are moving more towards growing of product variations, reduction in
production times, and increasing product complexity as well as demands for even shorter delivery time. These trends require small batch size productions and process planning systems that can automatically generate assembly sequences in real time. As flexibility and adaptability of assembly systems become prominent concerns, assembly processes constantly vary to accommodate product changes. It is concluded that assembly process planning plays a dominating role in flexible assembly systems. In addition, automation of assembly planning will expedite planning execution, reduce planning cost, and improve planning quality. In concurrent engineering environments, the automation of assembly sequence planning will help designers assess assembly process requirements of different design solutions for a given product. For some products, small changes in product design can have a large impact on assembly alternatives. Furthermore, in less structured, more dynamic manufacturing systems, when same kind of products are assembled in different shops, the knowledge of all assembly sequences is needed for selection of the sequences which are more suitable to the equipment available in each shop. In such flexible and dynamic assembly environments, robots and other flexible assembly machines are common hardware facilities for assembly operations.

Robotic assembly process planning can be divided into four planning levels in a hierarchy (Wolter 1989): assembly level planning, task-level planning, manipulator-level planning, and joint-level planning. Joint-level, the lowest level in the hierarchy, specifies robots' joint motions that can be directly executed by the robot. Manipulator-level plans the motions of robots' end effecters. Task-level describes actions purely on world models
of parts, such as "Place Object1 onto Object2". Assembly-level involves the generation of a series of assembly operations (a sequence) with selection of assembly fixtures and tools. The three lower level planning levels from the joint-level up to task-level are generally called robot task planning. Extensive work has been done in robot task planning, such as Lozano-Perez and Wesley (1977), Lieberman and Wesley (1977), Binford (1979), Wesley, et al (1980), Latombe and Mazer (1981), Mazer (1983), Dufay and Latombe (1983), Popplestone and Ambler (1983), Fu, et al (1987), Chang and Wee (1988), and Nnaji (1992). Recent literature has been focused on higher level -- assembly-level planning which generates the assembly operation sequences for robot task planning.

Assembly sequence determination associated with fixture and tool selection is a basic issue in assembly-level planning. As the prerequisite of robot task planning and programming, the product assembly sequence is either given or specified by human programmers or assembly operators. The development of manufacturing automation in industry has led to the current trend of integrating computer aided design (CAD) and computer-aided manufacturing (CAM). To integrate a robotic assembly planning system with a computer-aided design system, it is essential to automatically generate assembly sequences directly from CAD data bases. These assembly sequences can then be used by a robot task planner for robot programming. The intent of this particular research is to develop a mechanical assembly sequence planner which has the ability to automatically generate assembly sequences directly from a CAD data base. The ultimate vision is to have an assembly planning system that can aid the integration of product design,
production system design, control and scheduling activities. With such integration, we can combine product design analysis and production engineering into a single system, thereby increasing the entire system efficiency.

The description of an assembly is a key element for assembly sequence generation because it provides the information for the assembly planning system to generate assembly procedures. A feature-based product design model (Gu, et al 1994) was used in this project for product representation. The product design representation provides detailed design description for those real world products, and it can support a variety of applications. The features definition is highly abstract and can describe various mechanical products. With such a design representation, the proposed assembly planning system can automatically generate assembly sequences for a variety of mechanical products, and the planning system can be used in various production environment.

1.2 PROBLEM STATEMENT

Obviously, for a flexible assembly environment, manual assembly sequence planning cannot satisfy the demand and requirement of assembly planning and integration of CAD and assembly systems. Therefore, there is a need for the development of the automated assembly planning system which can directly retrieve product information from the CAD data base and automatically generate assembly sequences for robotic assembly operations.
1.3 RESEARCH OBJECTIVE

The objective of this research is to develop an assembly planning system which can automatically generate assembly procedures from product descriptions represented in a CAD data base. Several issues are considered:

- Assembly representation. The intent is to represent an assembly in more abstract form so that the computer system can handle various mechanical products with the minimum or without human experts’ assistance.

- Grouping the components in an assembly. The mechanical assembly is complex in terms of components’ geometry and the relationships between the components. The number of components of an assembly is one of the factors which result in complexity. The intent of the component grouping is to reduce the complexity of the assembly sequence search space.

- Assembly sequence generation. The system is designed to generate preferred sequences, and provide a means of focusing the search for good solutions.
1.4 RESEARCH APPROACH

The proposed approach is a graph-based heuristic method. The information about the assembly model is retrieved from a feature-based product design model. It is assumed that an assembly sequence can be obtained by reversing the order of its corresponding disassembly sequence. The generation of assembly sequences involves four steps: 1. create connectivity relationships in graphs based on the feature-based design representation; 2. decompose an assembly into sub-groups using the connectivity graphs; 3. generate disassembly sequences for each sub-group; and 4. generate the final assembly sequence by synthesizing the sub-disassembly sequences and reversing the disassembly sequence.

The entire planning system is developed in an object oriented programming language, Smalltalk 80. The object oriented environment closely resembles the real world and facilitates the development of individual objects which are able to contain attributes, knowledge and reasoning abilities.

In order to restrict the scope of the project to a manageable level, the system was developed based on the assumptions that all the mechanical parts are rigid and the assembly relations are mechanical-oriented.
1.5 ORGANIZATION OF THE THESIS

The thesis is organized into the following chapters. Chapter 2 is a literature review of the relevant assembly sequence planning methods. Chapter 3 provides a brief overview of the proposed assembly sequence planning system. Chapter 4 presents a feature-based product design model, and the assembly is analyzed based on this representation. Chapter 5 explains assembly planning algorithms and heuristics with examples. In Chapter 6, a brief discussion of the object oriented programming system is first provided, and an in depth description of how the planning system is implemented is given. Chapter 7 contains conclusions and suggestions for the future development of the assembly process planning system.
CHAPTER 2

2. LITERATURE SURVEY ON ASSEMBLY SEQUENCE PLANNING

2.1 INTRODUCTION

Fully automatic planning of robotic assembly tasks has been extensively addressed during the last decade. The thrust of this project was to develop a new approach to generation of assembly plans for real-world products without human interventions. In order to identify the main problems and provide a basis for comparison of their related solutions, a number of existing techniques will be reviewed which include user/planner interaction, AI, heuristics, and geometric modelling and reasoning. Some useful concepts have been used in this project of automatic assembly sequence planning.

Assembly process planning mainly involves three tasks: 1) assembly modelling,
2) analysis of the assembly; 3) generation of the assembly sequences. The assembly model or the representation of product assembly is the input required by an assembly planner. Assembly analysis determines the precedence and other constraints among the components based on the product representation. Based on the results of the two aspects, the assembly planner uses built-in assembly modules, algorithms and heuristic knowledge to determine the assembly sequence. This section reviews the previous work on these aspects.

### 2.2 LITERATURE REVIEW

Several approaches have been reported regarding generation of assembly sequences including user/planner interaction, AI, heuristics, and geometrical modelling and reasoning.

Early work on interactive assembly planning was concerned about formulating a set of questions that are answered by human experts. The question-answering session leads to the complete specification of precedence relationships. To successfully answer these questions, a real understanding of the product structure is essential.

Bourjault (1984) proposed an approach to the generation of all possible assembly sequences for a product assembly. His algorithm was based on users' answers to a series of questions about assembly precedence of parts. The information contained in part list
and assembly drawings was used to characterize the assembly with a network in which nodes represented parts and the arcs between the nodes are liaisons that are essentially user-defined relations between parts.

De Fazio and Whitney (1987) found that the question and answer approach proposed by Bourjault could be a serious problem when the number of assembly relations is large, say more than seven relations. As matter of fact, real product assemblies usually have a large number of parts and relations. They proposed a technique to simplify the question & answer process. Their work was based on two points: First, a production engineer or an assembly mechanic faced with an unfamiliar product assembly is asked fewer questions, not necessarily more complicated, but more involved than Bourjault’s questions. These are a set of questions which directly evoke relationships. Second, valid liaison sequences are developed algorithmically from those equivalent relationships. As with Bourjault’s network, the nodes were the parts and liaisons represented the relations between the parts. This algorithm can significantly improve the tedious question and answer process.

Homen de Mello and Sanderson (1991) also used interactive methods to tackle the problem. It is assumed that the assembly sequence is the reverse of the disassembly sequence, and therefore, the problem of generating assembly sequences becomes the problem of generating disassembly sequences. This seems easier than directly generating assembly sequences because a disassemblable part or sub-assembly directly implies the
satisfaction of precedence relationship, whereas, in the forward planning for direct assembly, the satisfaction of precedence relations may not be known immediately until an exhaustive search is completed (Lee, 1991). The algorithm Homen de Mello and Sanderson (1991) used involves generating all the cut-sets of the assembly's graph of connections, and checking which cut-set corresponds to feasible decompositions. The checking was done by generating questions that are to be answered by a human expert. The AND/OR tree structure representations of assembly sequences were then created and the sequence was determined.

Baldwin, et. al (1991) developed a set of user-interactive computer programs for generating and evaluating assembly sequences based on a disassembly analysis. The algorithms they used were rooted in the work of Bourjault (1984, 1987), Sanderon and Homen de Mellon (1987, 1991), and de Fazio and Whitney (1987). The interactive programs which were originated from Lui (1988) and Abell (1989) provided an environment for assembly sequences selection and were useful for concurrent design.

Planning has been an important research topic in artificial intelligence. AI approach has dominated much of the research in robot task planning (Fahlman 1974, Doshi, et al, 1988). The assembly planning shares some similarities with robot task planning. Because of the complexity of mechanical assembly, there does not exist an algorithmic way of generating optimal assembly plans. AI techniques, particularly knowledge-based systems, seem to be a feasible link between computer systems and real-
world assembly.

Mazouz et al (1991) applied knowledge-based system to generation of optimal assembly sequences without users’ intervention. They defined that the part has two kinds of sides: internal and external, and that the function of the liaison between parts can be maintaining, putting on, or putting on-maintaining. A series of rules was developed based on these definitions. However, these definitions sometimes seemed ambiguous since some counter-evidence could easily be found. Nevertheless, they suggested an idea about a higher level of abstraction for representing the assembly problems, which offers an opportunity to more effectively use planning systems.

Heemskerk and van Luttervelt (1989) used heuristics for assembly sequence planning. They first grouped parts into several clusters based on some characteristics such as similar part types and similar relations between parts. The grouping reduced the combinatorial complexity of possible sequences. Then accessibility heuristics for checking collision and stability were used to reduce the number of sequences. At last, an interactive search technique was used to eliminate those trivial equivalent solutions and obtain all alternative sequences.

Huang and Lee (1989, 1990, 1991) employed a knowledge-based approach for assembly planning. Predicate calculus were used for knowledge representation of product structure, precedence constraints and resource (such as tools, fixtures) constraints. Their
system accepted the CAD description of a product as input; obtained precedence relationships between parts by interactively answering two sets of structured questions about mating tasks: pre-condition and post-condition. Then the assembly sequences, fixture configurations and tool specifications were generated through graph a search of cut-sets in the connective graph - Feature Mating Operation Graph.

Sekiguchi et al (1983) developed prototype software for automatic verification of assembly drawings and automatic generation of assembly sequences. This technique is based on the assumption that the sequence of assembly is the reverse of the sequence of disassembly. Some connective relations between two parts were defined such as fit, taper contact, etc. These relations could be expressed by a matrix comprising codes arranged in a certain order. The groups were generated, based on the connective relations. The connective relations were classified based on the degree of difficulty of assembly. The degree of difficulty was determined by the combination of the freedom of motion and the required force to change the relative position of a part in an assembly. The parts were divided into five groups: shaft-like part, parts fixed to a shaft, parts related to a shaft, housing parts and parts related to housing parts. The parts in every group were disassembled according to the defined connective relations. It seems that the method might only be suitable for some specific cases which could be defined by the concepts proposed. In addition, the merging of the sequences to determine the final assembly sequence for a product might also be a problem.
Wolter (1989) proposed a three-stage solution for assembly planning: First, guess a set of insertion trajectories for each part; Second, find blocks which are essentially the parts already in final assembly positions. For each proposed trajectory, an assembly constraints graph (ACG) was created. Third, for each part, only one trajectory is selected and the others are deleted. Then, parts are ordered based on several criteria such as directionality, manipulability and fixture complexity from the CAD database. The developed planning system uses the results of the first two stages as input (i.e. assume that the first and the second stages were already implemented) of the third stage.

The feasibility of automating geometric reasoning in assembly planning has been a motivation for development of powerful computer-based geometric modelling and reasoning tools. The emphasis has been moved to automatic reasoning of geometric interference as a means of identifying precedence relationships in assembly sequencing.

Khsola and Matlikali (1989) used a geometric modelling system to represent product assembly. All components of a product were stored in the database of a solid modeller: noodles. The geometric assembly model was interactively created from the component models by designers. Then the mating relations between the components were inferred from the geometric model and were represented in a component graph. Based on the component graph, an assembly plan which was represented in the component hierarchy—AND/OR Tree was generated by recursively splitting the assembly into groups which could be single components or subassemblies. The splitting was carried out with
a breadth first search in the space of all possible groups.

Jentsch and Kaden (1984) formulated the precedence relationships among assembly tasks into predicate calculus that could be easily implemented in a computer for reasoning feasible assembly sequences. The connective relations among parts were represented by connection graphs. Based on the precedence relations between parts, the connection graph was transformed into a blocking graph which showed the blocks of removing a part by searching all the elementary circuits of the connection graph. The disassembly sequences were generated by analyzing the blocking graph.

Ko and Lee (1987) used the mating conditions, such as against, fits, tight-fits and contact to describe the relationships between components. An assembly can be expressed by a mating graph of components, having a hierarchical structure generated by developed algorithms. A bell assembly example was used to illustrate an assembly procedure generation process showing how the technique was used. In fact, the procedure used questions and answers to place the components in the proper assembly order.

Sedas and Talukdar (1987) developed an algorithm for planning disassembly, based on the same assumption that Sekiguchi and co-workers made. They used geometric and graph methods to represent the objects. The algorithm was designed to divide a product assembly into a pair of sub-objects (sub-assemblies) and verify that the sub-assemblies can be separated. This approach is basically a simulation of interference
checking.

Lee and Gossard (1985) provided a hierarchical data structure for representing assemblies in a data base. It was divided into two parts. The first part was the data structure used to store topological and geometric information on each component in an assembly. The second part was the data structure used to store information on how all the components in an assembly were connected. A tree structure using the concept of the virtual link was created to represent the relationships between the components in an assembly.

Lee and Andrews (1985) developed a method to infer the position of objects in an assembly, which was based on the representation of assembly by partial relationships imposed on components in an assembly. Then, mating conditions were defined, which were used to reduce the computational time and generate the final solutions.

Shpitalni et al (1989) described a system which accepted the definition of a structure (assembly) by means of a Constructive Solid Geometry (CSG) tree and used a geometrical modelling language to create supports and connectivity graphs for the structure. The system assumed that the assembly operations were straight line movements along the principle axes and that the assembly sequence for each axis was separately determined. Along a direction the sink node which did not support any object was selected to be the first candidate. In the case that multiple sink nodes were available, a
rule suggested that the one with the largest moment about the centre of the convex hull defined by the universe support base be chosen as the best candidate. The vertical support rule and the method of identification for sub-structures made it difficult to move a part along the X-axis. Although the complexity of the suitable cases for this method is limited (e.g., only +, -X and +,-Z axes direction are considered), it blazes a trail to automatically generating the assembly sequence without user intervention.

Santochi and Dini (1992) presented an assembly sequence planning system. In their work, various relationships between parts in an assembly were transferred to a single representation: contact. The contact representation was also adopted in this work since it offers much ease for computer manipulation. However, the method used to generate assembly sequences required heavy user interaction. This could be improved upon to eliminate or reduce the users' intervention.

Kroll et al. (1989) integrated geometric reasoning and knowledge based techniques to generate an assembly sequence for a structurally-stacked, uni-axial product. They first created "above graph" (some part should be above the other along the axis) for the product according to the mating conditions between parts. From this graph, an explosion graph was generated based on geometric analysis of the product. The explosion graph was an ordered directed single link of all the parts of the product. The order showed the general priority between parts, and the direction showed the geometric constraints between parts. Based on the knowledge of mechanical component assembly processes and
proceeding with the assembly process simulation, some parts which contributed to a continuous link in the explosion graph were grouped into subassemblies. The assembly sequence was generally the same as the order of parts in the explosion graph, and only the order of parts in subassemblies might have some changes.

Woo and Dutta (1991) studied automatic assembly planning from the disassembly point of view. They assumed that each component (a node) in a disassembly tree only had one parent. Two data structures were created: face adjacency graph and component mating graph, for determination of component disassemblability. The disassemblability of a component from an assembly/subassembly in a direction was determined using a ray tracing algorithm. The basic idea of the algorithm is that if rays from a component do not trace any other components, the component could be removed from the assembly. The direction was calculated from the normals of the mating faces of the components. They addressed assemblies composed of planar faceted components.

Lin and Chang (1993, 1993) presented a more complete assembly planning system based on assembly design which included geometric model - solid models and non-geometric model - frames. Connectivity relations (mating relations) and precedence constraints (collision information) for parts were generated by geometric reasoning on the solid models. Sequence constraint patterns were explicitly given in the frames for non-geometric constraints. Initially, the assembly sequences were generated solely based on the geometric constraints. The graph search methods with a three-level strategy were
used in the sequences generation. At the first level, the connectivity graph was decomposed into a tree with a root which was the base part of the assembly and several branches which were non-cycle vertices and compound cycles. At the second level, the assembly precedence diagrams were generated for both cycle-parts in the compound-cycles and non-cycle parts based on the precedence constraints. At the third level, the precedence relations were built up between non-cycle-parts and cycle-parts. Finally, the sequence constraint patterns were used for determination of final sequences.

2.3 SUMMARY

Much effort has been made for automatic and efficient assembly planning. Interactive methods require in depth understanding of product structure and assembly process. That means that the user must be an experienced process planner or a mechanical assembler. Although this approach may lose some efficiency, it is more flexible for different working environments and easily applied to real world mechanical products. AI techniques are also close to reality and efficient for assembly planning. But they require a given planning environment not to be significantly alerted. Geometric modelling and reasoning has the potential for automatic assembly planning directly from CAD data bases. However, mechanical product structures, component geometries, and assembly relations are so complex that they can not be completely represented only by geometric information. As a result, pure geometric modelling and reasoning methods need human assistance to determine the key precedence relationships. Therefore, the geometric
modelling and reasoning has to be associated with heuristics which can consider topological information about an assembly as well as functional relationships between components. This integration may be a more practical solution for real-world product assembly planning.
CHAPTER 3

3. OVERVIEW OF THE ASSEMBLY PLANNING SYSTEM

The basic function of automatic mechanical assembly planning is to perform the assembly planning task normally carried out by human assembly planners. In a manual planning, an assembly planner receives a set of engineering drawings from the design department. The planner uses the design drawings to figure out how parts can be assembled to a product. The reasoning process is mostly based on the planner’s expertise. Although there might not be any regular rules that the planner follows in deciding the assembly procedures, the primary factors that affect planning decision are parts’ geometry, connectivity relationships between parts, product functionality, and assembly feasibility. By analyzing these factors, the planner forms geometric constraints and functional constraints for the assembly process, and then generates the assembly sequences.
Based on the human reasoning process, an automated assembly planning system is proposed, shown in Figure 3.1. The system accepts a mechanical product design as input, and outputs an assembly plan that depicts a feasible assembly sequence and associated assembly process information. The mechanical product design is described in a feature-based product model. The decisions for generating feasible assembly sequences are made by a four-stage planning approach. The first stage uses the geometric and non-geometric information to generate connectivity relationships between the components of the assembly. The connectivity relationships mainly contain two types of information required for sequence generation: 1) a general liaison between components, and 2) disassembly constraints of each component for determining disassemblability. The second stage uses the general liaison to decompose an assembly into sub-groups. The components, except the common part in each sub-group, solely belong to this group. The common part provides the connections between sub-groups. This decomposition makes it possible to reduce the number of evaluations to be made in search for an optimal sequence. For each sub-group, the third stage generates sub-disassembly sequences based on removing constraints associated with the consideration of assembly stability and feasibility. In the forth stage, the sub-sequences are merged into one complete disassembly sequence according to the characteristics of the common parts and the part disassembly constraints. The integration of algorithms and heuristics are applied through the entire planning process, which will be discussed in detail in next two chapters.
DESCRIPTION OF MECHANICAL PRODUCT DESIGN

- geometric model
  - part solid models
  - assembly description
- non-geometric model
  - technological specifications
  - product functionality

REASONING OF ASSEMBLY MODELS

- geometric reasoning
- conversion mechanisms
  - connectivity relationships
    - general liaisons
    - parts moving constraints

DECOMPOSITION

- algorithm
- sub-groups

SEQUENCES GENERATION

- algorithm + heuristics
- partial disassembly sequences

SYNTHESIS

- algorithm + heuristics
- complete disassembly sequence
  - reversing assumption
  - the final assembly sequence

Figure 3.1. The Structure of the Assembly Planning System
CHAPTER 4

4. REPRESENTATION OF AN ASSEMBLY

4.1 THE FEATURE-BASED PRODUCT DESIGN MODEL

The objective of this work is to develop a method with which an assembly procedure is generated automatically based on a description of a product provided by a designer through CAD systems. The description of an assembly is one of the key elements of this effort because it determines the complexity of the method required to achieve the automation of generating assembly sequences. It is ideal that the product design model can provide sufficient information about a product and other manufacturing applications.

In the physical world, a product consists of units (sub-assemblies) and/or parts. In turn, a sub-assembly, in a similar fashion to a product, is comprised of sub-assemblies
and/or parts. Each part is described by a number of geometric entities associated with technical specifications. Figure 4.1 shows an example of a vertical machining centre (VMC) that consists of five units namely column, automatic tool changer (ATC), head stock, bed, and table. The head stock unit, for example, consists of several sub-assemblies (spindle unit, spindle motor, gear box) and a part (cover).

A product design is not only used by an assembly planner, but also needed by other systems and business functions. The proposed design representation should therefore provide detailed design descriptions for these real world entities so that the model can support a variety of applications. The information required to support these applications includes:

(1) design specifications of the product.
(2) sub-assemblies and parts.
(3) relationships between parts.
(4) bill of material.
(5) geometric shapes and dimensions of parts.
(6) material and heat-treatment requirements.
(7) geometric and dimensional tolerances.
(8) surface finish

A hierarchical product model structure developed by Gu (1992) and Gu, et al (1989, 1994) has been adopted for product design representation. The model uses five levels: product level, unit level, part level, solid primitive level, and feature level to contain the
Figure 4.1. A Vertical Machining Centre Example
above information. The product level defines the approximate size of the product, its weight, the technical specifications of the product, and components and sub-assemblies which directly form the product. The unit level provides the information about sub-assemblies including sizes of assemblies, weight, specifications, relations of sub-assemblies and components which directly form the unit. In the part level, the information includes part name, classification code if applicable, material specifications, weight, heat-treatment, relations with other parts or sub-assemblies, and primitives and Boolean operations for a chosen solid modeller. The solid primitive level concerns information required by a solid modeller. The feature level contains all the detailed information for description of the form features, such as boundary type, dimensions, location and orientation, tolerances, surface finish, and the like. The information required by the assembly planning is explained in details:

For parts:

part name: a unique identifier. All parts belonging to the same product must have different names.

part type: e.g. screw bolt, shaft, bearing, etc.

For features:

feature name: a unique identifier defined within the part. Other parts may use the same feature name for at most one of their features.

feature type: e.g. cylinder, chamfer, face, etc.

locations: three dimensional Cartesian coordinates and three orientation parameters.

relations: assembly relations with other parts such as fit and contact.
Three product assembly examples: Assemblies I, II, and III will be used for illustration of the algorithms and heuristic methods throughout the thesis. Assembly I is shown in Figure 4.2 which has sixteen parts. One of its parts, part p7 is a shaft shown in Figure 4.3. The shaft is described using the feature-based representation as shown in Figure 4.4. A more comprehensive version of the product model and its applications in process planning for machining and inspection can be found in the reference (Gu et al, 1994). Assembly II and Assembly III, shown in Figure 4.5 and Figure 5.8 respectively, will be introduced later.
Figure 4.2. An Example of Product Assembly I
Figure 4.3. A Shaft(P7) of Assembly I
Figure 4.4. Feature-Based Representation of the Shaft of Assembly I
4.2 THE ASSEMBLY MODEL

The assembly planning system integrates algorithms and heuristic procedures to generate assembly sequences. As mentioned in Chapter 3, the planning strategy consists of four major steps: 1. create connectivity graphs; 2. decompose the assembly into sub-groups using the connectivity graphs; 3. generate disassembly sequences for every sub-group; and 4. synthesize the disassembly sequences of sub-groups, and determine the assembly plan by reversing the disassembly sequence.

In this section, an assembly model aiming to minimize the user's role is sought which is the first step in the planning strategy. An assembly can be described by specifying its components and their relationships in the assembly. The components themselves are directly described using the part level and feature level information of the product model discussed in the section 4.1. The relationships between the components in the assembly are specified by the connectivity graphs. These graphs embody all the connective relations between the components and physical constraints. Two types of connectivity graphs are generated for a product: liaison graphs and contact relation graphs. A liaison graph represents the connective relations between all parts within a product. This graph will be used to decompose an assembly into sub-assemblies at the second stage of the planning strategy. The contact relation graphs represent the directional relationships between the contacting parts including both geometric and non-geometric constraints. These graphs will be used, at the third stage of the planning strategy, to verify
whether a part is removable for disassembly.

4.2.1. Liaison Graph

The generation of a liaison graph is based on the information retrieved from the feature-based representation of a product. Although this representation does not directly provide the connective relationships between parts, it links a part to other parts through the features of the part. At the feature level, one of the feature's parameters is assembly relation. The assembly relation supplies two kinds of information: the part name (ID) that the feature connects, and the type of relationship between the feature and the connected part such as fit. For every part, all the related parts can be identified. Then, the planning system uses a collection {} to collect all the related parts. In the collection, the first position is filled with the part, and the remaining positions are filled with its related parts. Linking the collections of every part in a product forms the liaison graph of the product.

Figures 4.3 and 4.4 show the part p7 design and its feature representation. Through one of its features, f3, the related parts p2 and p3 are identified; through the feature f6, the related part p1 is found. Similarly, for all the remaining features of the part shown in Figure 4.3, the related parts are collected. The part p7 and its related parts p1, p2, p3, p8, p9, p13 and p14 are collected in a collection {p7 p1 p2 p3 p8 p9 p13 p14}. In this way, an assembly is represented by a liaison graph. The liaison graphs for Assembly I shown in Figure 4.2 and Assembly II given in Figure 4.5 were generated and are shown in
Figure 4.5 Automatic Transmission Assembly II (de Fazio and Whitney)
Figure 4.6. The Liaison Graph of Assembly I
Figure 4.7. The Liaison Graph of Assembly II
Figure 4.6 and Figure 4.7, respectively. The liaison graph only provides general assembly relation information without any details on how the parts are related. To automatically generate a feasible assembly sequence for a product, the detailed descriptions of assembly relationships are needed.

4.2.2. Contact Relation Graphs

There are a variety of assembly relationships; several of which have been defined in this research and are summarized in Table 4.1. Depending upon the feature geometry, an assembly of any two connected parts of a product may be defined with one of these relations. The variety of assembly relations combined with an unlimited possible part geometries makes it extremely difficult to develop a unique mathematical model to solve

<table>
<thead>
<tr>
<th>Relation</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>pressure fit</td>
<td>prfit</td>
</tr>
<tr>
<td>push fit</td>
<td>pufit</td>
</tr>
<tr>
<td>movable fit</td>
<td>mfit</td>
</tr>
<tr>
<td>loose fit</td>
<td>lfit</td>
</tr>
<tr>
<td>taper fit</td>
<td>tfit</td>
</tr>
<tr>
<td>spline fit</td>
<td>sfit</td>
</tr>
<tr>
<td>ring fit</td>
<td>rfit</td>
</tr>
<tr>
<td>key fit</td>
<td>kfit</td>
</tr>
<tr>
<td>screw fit</td>
<td>scfit</td>
</tr>
<tr>
<td>contact</td>
<td>contact</td>
</tr>
<tr>
<td>connect</td>
<td>connect</td>
</tr>
</tbody>
</table>

the assembly problem. However, the product assembly relations are grouped into two
classes: *fit* and *contact* in the feature level as shown in Figure 4.8. With the help of the features' information, these two classes of assembly relations between parts are converted into a single representation, *contact relation*, in some directions. Through the assembly *relation* between a feature and a related part, contact relations in different *directions* can be defined. For example, in Figure 4.8(a), part B fits into part A along the +X direction (assume that the fitting axis is along the X axis) through its feature f_B; in Figure 4.8(b), part B contacts part A through its feature f'_B along the +X direction, and part A contacts part B through its feature f'_A along the -X direction. There exists a contact between features f_A and f_B in any direction perpendicular to the fitting axis (x) as shown in Figure 4.8(a). If only the directions parallel to the three coordinate axes are considered, a fit relation can be replaced by contact relations in the six directions in a coordinate system, ±X, ±Y, ±Z, as shown in Figure 4.9. A new representation of the relationships between parts A and B in contact relations in the defined directions is summarized as follows:

<table>
<thead>
<tr>
<th>Contact</th>
<th>+X</th>
<th>-X</th>
<th>+Y</th>
<th>-Y</th>
<th>+Z</th>
<th>-Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part A</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Part B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

There are several fitting relations such as pressure-fit, push-fit, position-fit, movable-fit, screw-fit, taper-fit, and ring-fit. All the fitting relations can be substituted by contact relations in the directions perpendicular and/or parallel to the fitting axis along
Figure 4.8. Examples of Contact and Fitting Relations

(a) fit

- $f_A$ (orientation $+x$) fits in part B
- $f_B$ (orientation $+x$) fits in part A

(b) contact

- $f_A'$ (orientation $-x$) contacts part B
- $f_B'$ (orientation $+x$) contacts part A
Figure 4.9. Generation of Contact Relations for Cylindrical Surface Fits
the three coordinate axes. The first three fitting relations given above have a common characteristic; namely, the contact between two cylindrical surfaces. Therefore, the above examples shown in Figures 4.8 and 4.9 can be directly used to generate contact relations. Upon examining the contact characteristics of screw-fit and taper fit, it is clear that the contact relations not only exist along the directions perpendicular to the fitting axes, but also appear in other directions. For the taper-fit as shown in Figure 4.10, the contact directions are \( N_A, N_B, N'_A, N'_B \) between parts A and B. All these directions are composed of two of the six directions:

\[
\begin{align*}
N_A &= \{-Z, +X\} \\
N_B &= \{+Z, -X\} \\
N'_A &= \{-Y, +X\} \\
N'_B &= \{+Y, -X\}
\end{align*}
\]

Where \(+X\) and \(-X\) are the directions along the fitting axis.

The contact relationships between parts A and B exist in the directions perpendicular to the fitting axis and along the fitting axis. The representation of the contact relationships between parts A and B is summarized as follows:

<table>
<thead>
<tr>
<th>Contact</th>
<th>+X</th>
<th>-X</th>
<th>+Y</th>
<th>-Y</th>
<th>+Z</th>
<th>-Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part A</td>
<td>B</td>
<td></td>
<td>B</td>
<td></td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>Part B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.10. Generation of Contact Relations for Taper Fits
Whether there exist contact relations along the fitting axis of a screw fit depends upon the type of the female parts. If the female part is other than a screw-nut, then the contact relations may exist not only in the directions perpendicular to the fitting axis, but also in the direction along the fitting axis. For example, the female part, part A, in Figure 4.11(a) is a structure and cannot be removed (because of its size, weight, etc), it is equivalent to having a constraint along the fitting axis. In this case, the contact relation also exists along the assembly fitting direction. The contact relations defined here provide two types of information: the connective relationship between parts and the removal constraints in the contact directions. In Figure 4.11(a) part A cannot move in the -X direction when part B is fixed. This means that part A has a moving constraint in the -X direction. So, for this case, an extra contact relation is added between part A and part B in the -X direction along the fitting axis. That is, part A contacts part B in the -X direction. If the female part is simply a screw-nut as in Figure 4.11(b), both the screw bolt and the nut can be removed to disassemble the pair, which is similar to the contact relationship of two cylindrical surfaces. For a unified representation in our system, a contact constraint along the fitting axis for the female part is defined. Therefore, the screw-nut in Figure 4.11(b) is defined to have a contact to the screw-bolt in the -X direction. This extra contact relation between a screw-bolt and a screw-nut associated with the heuristic algorithms described in next section ensures that they are disassembled as a pair in the disassembly sequence. The extra contact relation is defined based on the degree of difficulty of disassembling one part from the other. The contact relation graphs represent both geometric and non-geometric constraints for parts in a product. Using the
above method, the contact relations of the parts in Assembly I and Assembly II were generated and are shown in Figure 4.12 and Figure 4.13, respectively.

4.3 SUMMARY

The representation of a product is important for creating an assembly sequence planner in integration of CAD and robot task planning. The feature-based product design model is chosen for the representation of the product because it can provide more complete product information. Based on the product information, the assembly model is generated for the assembly sequence planning application. In this model, the connectivity relationships between parts are represented by liaison graphs and contact relation graphs. This particular assembly relations description minimized the user’s intervention for assembly planning.
When part B is fixed, part A cannot be removed along the fitting axis; that is, A cannot move away from B in \(-X\) direction.

So, we define that:
A contacts to B in \(-X\) direction which is along the fitting axis.

From \(f_A\) and \(f_B\) we get:
A contacts to B in \(\pm Z\), \(\pm Y\) directions
B contacts to A in \(\pm Z\), \(\pm Y\) directions

* A can move along the fitting axis when B is stand still.

* For unified representation, A is set to contact B in \(-X\) direction.

Figure 4.11. Generation of Contact Relations for Screw-Fits
<table>
<thead>
<tr>
<th></th>
<th>contacts +X</th>
<th></th>
<th>contacts −X</th>
<th></th>
<th>contacts +Z</th>
<th></th>
<th>contacts +Y, −Y, −Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>p8</td>
<td>p1</td>
<td>p7</td>
<td>p1</td>
<td>p7</td>
<td>p1</td>
<td>p7</td>
</tr>
<tr>
<td>p2</td>
<td>p7</td>
<td>p2</td>
<td>p3</td>
<td>p2</td>
<td>p7</td>
<td>p2</td>
<td>p7</td>
</tr>
<tr>
<td>p3</td>
<td>p2</td>
<td>p3</td>
<td>p4</td>
<td>p3</td>
<td>p7 p5</td>
<td>p3</td>
<td>p7</td>
</tr>
<tr>
<td>p4</td>
<td>p3</td>
<td>p4</td>
<td>p5</td>
<td>p4</td>
<td>p5</td>
<td>p4</td>
<td>p5</td>
</tr>
<tr>
<td>p6</td>
<td>p5</td>
<td>p6</td>
<td>p5</td>
<td>p6</td>
<td>p5</td>
<td>p6</td>
<td>p5</td>
</tr>
<tr>
<td>p8</td>
<td>p9 p10</td>
<td>p8</td>
<td>p10</td>
<td>p8</td>
<td>p7 p5</td>
<td>p8</td>
<td>p7</td>
</tr>
<tr>
<td>p10</td>
<td>p11 p12</td>
<td>p10</td>
<td>p11</td>
<td>p10</td>
<td>p11</td>
<td>p10</td>
<td>p11</td>
</tr>
<tr>
<td>p14</td>
<td>p15 p16</td>
<td>p14</td>
<td>p15</td>
<td>p14</td>
<td>p15</td>
<td>p14</td>
<td>p15</td>
</tr>
<tr>
<td>p16</td>
<td>p13 p17</td>
<td>p16</td>
<td>p17</td>
<td>p16</td>
<td>p17</td>
<td>p16</td>
<td>p17</td>
</tr>
</tbody>
</table>

Figure 4.12. Contact Relations between Parts of Assembly I
Figure 4.13. Contact Relations between Parts of Assembly II
CHAPTER 5

5. ASSEMBLY SEQUENCE PLANNING

5.1 ASSEMBLY DECOMPOSITION

The structure of a real mechanical product assembly is usually complex and contains a large number of parts. To reduce this complexity, an assembly is decomposed into several sub-assemblies. The sub-assemblies can be further decomposed into simpler sub-assemblies. Figure 4.1 showed an example where the head stock is a sub-assembly that can also be decomposed into spindle sub-assembly, motor sub-assembly, etc. The connectivity graphs including liaison graphs and contact relation graphs can be used to describe the sub-assembly formation.
As shown in Figures 4.6 and 4.7, an assembly is represented in a liaison graph which is defined by

\[ CG = (V, E) \] (1)

where \( CG \) represents a connectivity graph, \( V \) is a set of vertices, and \( E \) is a set of edges connecting the vertices. Every edge \( e \in E \) is associated with a pair of vertices \((u, v)\), where \( u \) and \( v \in V \). It is also assumed that both sets \( V \) and \( E \) of a graph are finite (Tremblay and Sorenson, 1984).

The vertices depict parts in an assembly and the edges represent the assembly relationships between parts. Since a complete \( CG \) represents a product or unit assembly, the decomposition of an assembly into several sub-groups is equivalent to breaking a complete \( CG \) into pieces. Before describing the decomposition methodology, a few concepts are first defined:

- **liaison graph**: a graph that for every pair of vertices \( u \) and \( v \) on the graph, there exists a path connecting these vertices.
- **cutvertex**: a node whose removal causes the graph to become disconnected.
- **bi-connected graph**: a liaison graph without any cutvertex.

The decomposition of an assembly starts with finding cutvertices in a \( CG \).
defined above, the cutvertex is a vertex in a CG which, if removed, will break the graph down into two or more pieces. For example, a cutvertex in Figure 4.6 is part p7. If p7 is removed, the graph is split into two pieces [p12 p13 p14 p15 p16] and [p1 p2 p3 p4 p5 p6 p8 p9 p10 p11]. Thus, Assembly I (its liaison graph is shown in Figure 4.6) can be decomposed into two sub-groups: {p7 p12 p13 p14 p15} and {p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11}. The decomposition process is repeated for every sub-graph. If a cutvertex is found in a sub-graph, this sub-graph is then split again into two or more pieces. The decomposition process for an assembly continues until every sub-graph is a bi-connected graph. From both {p7 p12 p13 p14 p15 p16} and {p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11}, the cutvertices, p5 and p13, are found respectively. So, the two sub-graphs are split again into a total of four sub-graphs (sub-groups): {p5 p6}, {p1 p2 p3 p4 p5 p7 p8 p9 p10 p11}, {p7 p13 p14} and {p12 p13 p15 p16} as shown in Figure 5.1. The sub-graphs of Assembly I in Figure 5.1 are bi-connected without any cutvertex. Removing any single part from the sub-graphs, for example, the sub-group {p7 p13 p14}, will not divide the graph into two or more disconnected pieces. Therefore, they are bi-connected graphs. In the same manner, Assembly II shown in Figure 4.7 is also decomposed into two bi-connected sub-groups: {Q E F} and {A B C D G H J K L M N O P E} which are shown in Figure 5.2. The splitting algorithm is provided in Appendix A.
Figure 5.1. The Bi-connected Graphs for Assembly I
Figure 5.2. The Bi-connected Graphs for Assembly II
One should note that the term sub-group is used above instead of sub-assembly. The distinction between these two can be made as follows:

A sub-group is defined as a sub-liaison graph whose components in the final assembly sequence may not be assembled one after the other.

A sub-assembly is a sub-group whose components in the final assembly sequence are assembled one after the other.

To explain the distinction further, for example, two sub-groups are formed \{A, B, C, D\} and \{E, F, G\}. The final assembly sequence is \{A, B, E, F, G, C, D\}. In this example, the sub-group \{E, F, G\} is a sub-assembly because its components are adjacent in the final assembly sequence. The other group is not a sub-assembly because its components in the final assembly sequence are not adjacent and the sequence is interrupted by other components. More discussions about the sub-assemblies are given in section 5.4.

5.2 SEQUENCING

The process of assembly and disassembly is reversible assuming that the components are rigid and there is no internal energy stored (Woo, 1987). In this project, automatic assembly is studied from the point of view of disassembly under this assumption.

To generate the assembly sequence for a product, we first generate the disassembly
sequence, and then reverse the order of the disassembly sequence. The key criterion for determining whether a part can be removed from a product assembly is: there is a collision-free path for the part removal in a defined working environment. If a part satisfies this condition, the part can be disassembled from the product assembly. There are many possible solutions for a planning problem. To generate a good assembly plan near optimal, some of constraints which need to be considered are listed and explained below:

1. Manipulability. To perform the more difficult operations with the easier handling parts. For example, to attach a bolt to a gearbox, it is preferred to fix the gearbox while screwing the bolt rather than fixing the bolt while screwing the gearbox. This has been considered in section 4.2 in constraints representation with contact relation graphs.

2. Directionality. To assemble/disassemble the parts as much as possible from a single direction. This simplifies the fixtures, requires a less dextrous robots, and avoids extra operations to reorient the work piece. This requirement will be discussed in section 5.2.1.

3. Stability. It is preferred to sequence the operations that the partially built assembly can hold itself as much as possible. This reduce the fixture requirements and simplifies the assembly operations. This consideration will be discussed in section 5.3.
Based on the above considerations, the system generates a feasible disassembly sequence. In the planning system, the basic constraints for each part in an assembly are represented in contact relation graphs in which six contact directions (±X, ±Y, ±Z) for each part are defined. Therefore, collision-free paths are searched for a part in these six directions. If a part does not have any constraint for its removing, it may be disassembled. After the part passes the structure stability checking, its disassemblability is approved, and the liaison graph and contact relation graphs for the sub-group are updated. As there are six contact directions, the disassembly priority order in the six directions is first established.

5.2.1. The Priority Order of the Six Contact Directions

The priority order of the six directions (±X, ±Y, ±Z) is determined by considering the stability of the structure in the disassembly process and the ease of automated assembly operations. Since the +Z assembly direction is considered the most stable orientation of the product assembly, removing a part from top to bottom is known as the most preferable disassembly operation, and it has the highest priority. The direction -Z has the lowest priority, since it usually requires that either the product be turned over or a special fixture be used. The priority order among the ±X and ±Y, which are horizontal directions, can not be determined at this early stage based on above two considerations. So, we temporarily set the priority order among them: +X, -X, +Y, -Y. When the sequence is generated, the order of these four directions is then finally determined.
according to the number of parts disassembled in a direction based on the ease of automated assembly operation. The larger the number of the parts disassembled in a direction is, the higher the priority of the direction is. The priority order of the six directions is therefore +Z, +X, -X, +Y, -Y, and -Z.

The precedence relations among parts are first determined by the priority of the direction that a part can be removed. For example, if one part (say part A) can be removed in the +Z direction and another part (say part B) can be removed in the +X direction, then part A is removed prior to part B; if a part (say part C) can be removed either in the +Z direction or in the +X direction, then part C is removed in the +Z direction.

5.2.2. The Determination of Collision-Free Paths for Parts Disassembly

The degree of difficulty in disassembling a part depends upon the number of parts that block the way in which the part is removed. The more parts a part is blocked by, the more difficult it is to move the part. Some of these parts have contact relations with the part; the others do not (they are non-touch constraints for the part to be removed). Therefore, the solution of searching for a collision-free path in a given direction is proceeded with two stages. The first is to search for a part that does not contact any other parts in the given direction (the part is called contact-free part). The second is to check if the part found at the first stage has non-touch constraints. For ease of our discussion,
we call the part being searched for at the first stage a target part and the target part that passes the second stage checking (having a collision-free path) a candidate part.

At the first stage, a target part which is contact-free in a direction can be found directly from contact relation graphs. The contact relation graphs are defined in six directions (±X, ±Y, ±Z). The search of the target part follows the priority order of the six directions from high to low. Once the target part is found on a certain order of the direction, the system will collect all the contact-free parts (target parts) in this direction for next stage checking and will not search for other target parts in lower order of directions.

It is possible that there are simultaneously several target parts available in one direction. In this case, the following heuristic algorithms are used to find the best target. These heuristic algorithms are based on the feature locations of the target parts and their contact relations. To illustrate the algorithm, it is assumed that \( N \) is the direction in which the target part has a contact-free path, \( \bar{N} \) is the opposite direction of \( N \).

**Heuristic Rules for Multiple Targets Elimination:**

**STEP 1** Among the features of each target part, search for the feature that contacts another part in the \( \bar{N} \) direction and is the farthest along the \( \bar{N} \) direction. An example from Assembly II is shown in Figure 5.3.
STEP 2  Among the features selected in the first step, search for the farthest one along the N direction. If a feature is found, then the part that has the feature is the best target. An example from Assembly II is shown in Figure 5.4.

STEP 3  If no feature is found in STEP 2, it means that the features found in STEP 1 are all on the same plane. Therefore, among all the other features of those target parts, search for the feature that is the farthest along the N direction. If there exists such a feature, the part having this feature is the best target. An example from Assembly II is shown in Figure 5.5.

STEP 4  If no feature is found in STEP 3, it means that all the targets have the same priority, and any one of them can be removed first. Therefore, the best target is selected arbitrarily. An example is provided in Figure 5.6.
In $+X$ direction, part L is one of target parts.
In $-X$ direction, part L contacts parts A, H and J through its features $f_{L1}$, $f_{L2}$, $f_{L3}$ and $f_{L4}$.

Among these features, $f_{L4}$ is the farthest in $-X$ direction.

Figure 5.3. Feature Selection of the First Step for Multiple Targets
In +X direction, parts L and K are the target parts. $f_{L4}$ and $f_{K1}$ are the farthest features contacting other parts (A, H, J) in -X direction for parts L and K, respectively. In +X direction, $f_{L4}$ is farther than $f_{K1}$. So, part L is removed prior to part K.

Figure 5.4. An Example of Multiple Targets (Case I)
Parts F and O are the target parts in -X direction. The features $f'_F$ of part F and $f'_O$ of part O which contact part E in +X direction are on the same plane. So, in -X direction, the farthest feature of the two parts is searched. It is $f'_F$ which belongs to part F. As a result, part F is removed prior to part O.

Figure 5.5. An Example of Multiple Targets (Case II)
A, B, C are candidates.
$f_A$, $f_B$, and $f_C$ are on the same plane.

Also, $f_A'$, $f_B'$, $f_C'$ are on the same position along N direction:
So, A, B, C have the same priority.

Figure 5.6. An Example of Multiple Targets (Case III)
For the second stage, when a part is contact-free in a direction, it does not necessarily mean that the part has a collision-free path in the direction. There may exist a non-touch block on the path of the part's removal. Therefore, for the best target just determined with the above algorithm, we use the following heuristic algorithms to check if it has non-touch constraints in the contact-free direction N:

Again assume the opposite direction of N is $\bar{N}$.

**HEURISTIC RULES FOR NON-TOUCH CONSTRAINTS IDENTIFICATION:**

**STEP 1**

From the features of the parts that the target part does not contact in $\bar{N}$, search for the feature (say $F$) which is the farthest in N with a normal in $\bar{N}$. If the feature is found which belongs to the target part, then the target part is approved to be a candidate part which has a collision-free path in N. If the feature found does not belong to the target part, then go to **STEP 2**.

**STEP 2**

Search for the feature (say $f$) of the target which has its normal in N and is the farthest in N. In the N direction, if $f$ is located farther than or equal to $F$, then the target part is approved to be a candidate. If $F$ is located farther than $f$, then the target part is rejected. The system searches for another target part from other directions.
The examples from Assembly I are shown in Figure 5.7. In Figure 5.7(a), part p4 is the target in the -X direction. Feature $f_{p4}$ of part p4, whose normal is the +X direction, is the farthest in the -X direction (STEP 1). So, p4 is the candidate part having a collision-free path in the -X direction. In Figure 5.7(b), part p7 is the target in the -X direction. Feature $f_{p5}$ of part p5 is the feature found in STEP 1. Feature $f_{p7}$ of part p7 is the feature found by STEP 2. In the -X direction, since $f_{p7}$ is farther than $f_{p5}$, part p7 is approved to be a candidate part. In Figure 5.7(c), the part p8 is the target in the -X direction. $f_{p5}$ of part p5 is the feature found by STEP 1, and $f_{p8}$ is the one found in STEP 2. Since $f_{p5}$ is farther than $f_{p8}$ in the -X direction, part p8 is rejected.

After a part is identified as a candidate, the determination of its removal is made by considering the stability of the assembly structure.

5.2.3. The Stability Considerations

Instability of an assembly structure in the disassembly process may occur if a part removed supports other parts. When the collision-free path of a part (the candidate determined above) is along the +Z direction, which usually means that the candidate part does not support any other parts in the +Z direction (except a screw-bolt which fits with a screw-nut), the removal of the candidate will not affect the stability of an assembly structure. The candidate is then directly approved to be removed. In the +Z direction, if the candidate part is a screw-bolt, then a screw-nut is searched from the related parts of
Figure 5.7. Examples of Qualified Candidates with Stability
the screw-bolt. If the search is failed, then the screw-bolt is treated as other types of parts having collision-free paths in +Z. The procedures discussed above will apply. If the search is successful, then a temporary support is added to the screw-nut, and the screw-bolt is removed in the +Z direction. The screw-nut is removed right after the screw-bolt is disassembled in the -Z direction.

When a collision-free path of a part (the candidate) is in a horizontal direction (e.g., +X, -X, +Y, or -Y), an instability may occur if the candidate is removed. In the example shown in Figure 5.7(a), when part P11 is first removed along the +X direction, part P10 will lose its support and become unstable. Therefore, when a part has a collision-free path in a horizontal direction, the decision on whether it can be removed will be made through a three-step heuristic check.

**Heuristic Rules for Stability Check in Horizontal Directions:**

**STEP 1**
Check whether the candidate part contacts any other parts in the +Z direction. If the candidate part does not contact any part in the +Z direction, then the candidate part is approved to be removed. If the candidate part does contact some part(s) in the +Z direction (these parts are called virtual unstable parts), proceed to **STEP 2**.

**STEP 2**
For each of these virtual unstable parts, search for its contacting parts
(these parts are called supporting parts) in the \(-Z\) direction. If the number of supporting parts is more than one, it means that the virtual unstable part has other support in addition to the candidate’s support. Then the candidate part is approved and can be removed. If for some or all of the virtual unstable parts, the number is one, then the candidate is not approved and suspended in the candidate position (this kind of candidate is called suspended candidate) because the removal of the candidate part may cause other parts to become unstable and those virtual unstable parts are real unstable parts. The system then searches for another candidate. If another one is found, the suspended candidate is taken out from the candidate position, the system proceeds with the new one. If no such a candidate is found, the suspended candidate is back to being a candidate and proceed to STEP 3.

**STEP 3** Add a temporary support to the unstable part(s). Then the candidate is approved and removed. The unstable part is removed right after the candidate. The removal direction of the unstable part is either the same as that of the candidate or the \(-Z\) direction. If the former is tenable, then the unstable part and candidate part are grouped into a sub-assembly in the final assembly sequence.

The examples from Assembly I are shown in Figure 5.7 (a), (b) and (c). In Figure
5.7(a), part p11 has a collision-free path in the +X direction and therefore is a candidate part. Part p10 gets its only support from p11. It is an unstable part. Therefore, part p11 is suspended, and the system searches for another candidate. In the -X direction, part p4 is found as an approved candidate to be removed since part p5 it contacts in the +Z direction contacts other parts p3, p8, and p9 etc. in the -Z direction. In the meantime, part p11 is not a candidate any more, and the system proceeds from p4, then p3, then p2. In Figure 5.7(b), part p7 is a suspended candidate in the -X direction since part p1 has its only support from p7. So, the system searches for other possible candidates in other directions. The search fails (note: in +X, part p11 is still not an approved candidate). As a result, unstable part p1 gets a temporary support, part p7 is back to be an approved candidate and is removed in the -X direction. Then p1 is removed in the -Z direction. In Figure 20(c), again, part p11 is a suspended candidate in the +X direction and part p10 is an unstable part. Other candidates can not be found in Figure 5.7(c). So, p10 gets a temporary support, p11 becomes an approved candidate and is removed in the +X direction. Since part p10 can be removed right after p11 in the same direction, +X, parts p11 and p10 are grouped into a sub-assembly. That is, in the final assembly sequence, parts p11 and p10 can be treated as an unit to be assembled/disassembled to/from the assembly.

It seldom happens that a part has to be removed in the -Z direction because the -Z direction is set to have the lowest priority to be considered. It is recommended that a product be set up in such a manner so that most of the parts can be removed without
having to consider the -Z direction. If there is a part (the candidate part) having a collision-free path only in the -Z direction, this part can be removed without the stability examination except if the candidate is a screw-bolt. If a candidate part other than a screw-bolt does not contact any other part in the -Z direction, it means that the candidate part does not have any support from the -Z direction and is not a supporting part for the structure, therefore, the candidate part can be removed. Otherwise, the structure itself may not be stable. When the candidate part is a screw-bolt, its function is usually to fasten another part up to some structure, then the removal of the screw-bolt will cause the part it contacts in the +Z direction unstable. Therefore, when the candidate part removed in the -Z direction is a screw-bolt, a temporary support is added to the unstable part. Then the screw-bolt is removed in -Z direction. The unstable part is removed right after the screw-bolt in the same direction.

According to the above heuristic algorithms, the disassembly sequence of every sub-group can be generated. For the example Assembly I shown in Figure 4.2, the disassembly sequences of its four sub-groups and the removal directions of every part are:

1. \{p4(-x)\rightarrow p3(-x)\rightarrow p2(-x)\rightarrow p7(-x)\rightarrow p1(-z)\rightarrow p11(x)\rightarrow p10(x)\rightarrow p9(x)\rightarrow p8(x)\rightarrow p5(x)\}
2. \{p6(z)\rightarrow p5(z)\}
3. \{p14(z)\rightarrow p13(x)\rightarrow p7(x)\}
4. \{p15(x)\rightarrow p16(-z)\rightarrow p12(x)\rightarrow p13(x)\}
For the Assembly II shown in Figure 4.5, the disassembly sequences and the removal directions of its two sub-groups are:

1. \{M(x)\rightarrow L(x)\rightarrow K(x)\rightarrow J(x)\rightarrow H(x)\rightarrow G(x)\rightarrow O(-x)\rightarrow E(-x)\rightarrow P(-x)\rightarrow D(-x)\rightarrow N(-x)\rightarrow C(-x)\rightarrow B(-x)\rightarrow A(-x)\}

2. \{Q(-x)\rightarrow F(-x)\rightarrow E(-x)\}.

5.3 SYNTHESIS

For any two sub-groups that have a common part (cutvertex), their disassembly sequences are merged into one sequence. For the sake of this discussion, the following notations are introduced:

[A_i] and [A_j]: two sub-groups of a product.

{S_i} and {S_j}: disassembly sequences of [A_i] and [A_j], respectively.

P: the common part (cutvertex) of {S_i}/[A_i] and {S_j}/[A_j].

{S}: the resulting sequence after merging {S_i} and {S_j}.

Assume that both the sub-groups {S_i} and {S_j} have some parts that must be disassembled before and after common part P, this being the more general and difficult case. Then, {S_i}, {S_j} and {S} can be written in the following form:

for \{S_i\}: \{\{S'_i\}\rightarrow P\rightarrow \{S''_i\}\}

for \{S_j\}: \{\{S'_j\}\rightarrow P\rightarrow \{S''_j\}\}

for \{S\}: \{\{S'\}\rightarrow P\rightarrow \{S''\}\}
Where \( \{S_i'\}, \{S_j'\} \) and \( \{S'\} \) are the sequences of the parts that are disassembled before the common part \( P \) in \( \{S_i\}, \{S_j\} \) and \( \{S\} \), respectively.

\( \{S_i''\}, \{S_j''\} \) and \( \{S''\} \) are the sequences of the parts that are disassembled after the common part \( P \) in \( \{S_i\}, \{S_j\} \) and \( \{S\} \) respectively.

Therefore,

\[
\{S'\} = \{S_i'\} \rightarrow \{S_j'\} \text{ or } \{S_j'\} \rightarrow \{S_i'\}
\]

\[
\{S''\} = \{S_i''\} \rightarrow \{S_j''\} \text{ or } \{S_j''\} \rightarrow \{S_i''\}
\]

The merging of any two sub-groups is done by generating the correct order sequence between \( \{S_i'\} \) and \( \{S_j'\} \) for \( \{S'\} \) and between \( \{S_i''\} \) and \( \{S_j''\} \) for \( \{S''\} \). The methods to do this are developed for the two different situations where \( \{S_i'\} \) and \( \{S_j'\} \) are listed before common part \( P \) in the sequences \( \{S_i\} \) and \( \{S_j\} \), and where \( \{S_i''\} \) and \( \{S_j''\} \) are listed behind \( P \). Around common part \( P \), there are two cases to be considered:

1. when the removal directions of \( P \) in the two sub-groups are the same;
2. when the removal directions of \( P \) in the two sub-groups are different.

### 5.3.1. Case I: Identical Removal Directions of \( P \)

For the first case, when the removal directions of \( P \) in the two sub-groups are the same, we use a heuristic algorithm to order the two sequences of the two sub-groups.
To generate the correct order of \(\{S'_1\}\) and \(\{S'_j\}\), the following heuristic algorithm is applied:

**HEURISTIC RULES FOR THE DETERMINATION OF THE ORDER \(\{S'_1\}\) AND \(\{S'_j\}\):**

**STEP 1** Search backward for parts \(P'_i\) and \(P'_j\) from \(\{S'_i\}\) and \(\{S'_j\}\) respectively which have the same removal direction as common part \(P\).

**STEP 2** If both \(P'_i\) and \(P'_j\) are found, then the order of \(P'_i\) and \(P'_j\) determines the order of \(\{S'_1\}\) and \(\{S'_j\}\). The order of \(P'_i\) and \(P'_j\) is determined according to the rules of multiple target parts elimination discussed early. The order of \(\{S'_1\}\) and \(\{S'_j\}\) is the same as the order of \(P'_i\) and \(P'_j\).

**STEP 3** If only one of \(P'_i\) and \(P'_j\) is found, the sequence which has the part is set after the other. For example, if \(P'_i\) from \(\{S'_i\}\) has the same removal direction as \(P\) while no \(P'_j\) is found, then the merged sequence of \(\{S'_i\}\) and \(\{S'_j\}\) is \(\{\{S'_j\}\rightarrow\{S'_i\}\}\).

**STEP 4** If none of \(P'_i\) and \(P'_j\) is found, then the sequences \(\{S'_1\}\) and \(\{S'_j\}\) can be rewritten in the following form:

\[
\{S'_1\} = \{\{S_{i1}'\} \rightarrow P_{i1}'\}
\]

\[
\{S'_j\} = \{\{S_{j1}'\} \rightarrow P_{j1}'\}
\]
where $P_{i1}'$ is the last part of sequence $\{S_1'\}$. $\{S_{i1}'\}$ contains the parts before part $P_{i1}'$ in $\{S_1'\}$. $P_{j1}'$ is the last part of sequence $\{S_j'\}$. $\{S_{j1}'\}$ contains the parts before part $P_{j1}'$ in $\{S_j'\}$. The order of $\{S_1'\}$ and $\{S_j'\}$ is the same as the order of $P_{i1}'$ and $P_{j1}'$. When the removal directions of $P_{i1}'$ and $P_{j1}'$ are different, the order of $P_{i1}'$ and $P_{j1}'$ is determined by the priority of the directions along which they are removed. The priority of the six defined directions ($\pm X$, $\pm Y$, $\pm Z$) are discussed previously. When the removal directions of $P_{i1}'$ and $P_{j1}'$ are the same, the order of $P_{i1}'$ and $P_{j1}'$ is determined according to the rules of multiple target parts elimination discussed early.

The merger of the sequences for the two sub-groups of Assembly II shown in Figure 5.4 is given below.

For Assembly II,

\[
\{S_1\} = \{M(x) \rightarrow L(x) \rightarrow K(x) \rightarrow J(x) \rightarrow H(x) \rightarrow G(x) \rightarrow O(-x) \rightarrow E(-x) \rightarrow P(-x) \rightarrow D(-x) \rightarrow N(-x) \rightarrow C(-x) \rightarrow B(-x) \rightarrow A(-x)\}
\]

\[
\{S_j\} = \{Q(-x) \rightarrow F(-x) \rightarrow E(-x)\}.
\]

where $E$ is the common part and removed in -X direction.

\[
\{S_1'\} = \{M(x) \rightarrow L(x) \rightarrow K(x) \rightarrow J(x) \rightarrow H(x) \rightarrow G(x) \rightarrow O(-x)\}
\]

\[
\{S_j'\} = \{Q(-x) \rightarrow F(-x)\}
\]

Parts $O$ and $F$ are the first parts found having the same removal direction by backward searching in $\{S_1'\}$ and $\{S_j'\}$ respectively. By applying the rules regarding multiple target
parts discussed in the previous section, part F must be removed prior to part O as shown in Figure 5.5. The merged sequence is

\[
\begin{align*}
\{(Q\rightarrow F) &\rightarrow (M\rightarrow L\rightarrow K\rightarrow J\rightarrow H\rightarrow G\rightarrow O)\rightarrow E\rightarrow P\rightarrow D\rightarrow N\rightarrow C\rightarrow B\rightarrow A \} \\
= \{Q\rightarrow F &\rightarrow M\rightarrow L\rightarrow K\rightarrow J\rightarrow H\rightarrow G\rightarrow O\rightarrow E\rightarrow P\rightarrow D\rightarrow N\rightarrow C\rightarrow B\rightarrow A \}.
\end{align*}
\]

For the two sub-sequences \{p14(z)\rightarrow p13(x)\rightarrow p7(x)\} and \{p15(x)\rightarrow p16(-z)\rightarrow p12(x)\rightarrow p13(x)\} of Assembly I, the common part, p13, is removed in the +X direction. The partial sequences removed before p13 are:

\[
\begin{align*}
\{S'_i\} &= \{p14(z)\} \quad \text{and} \\
\{S'_j\} &= \{p15(x)\rightarrow p16(-z)\rightarrow p12(x)\}.
\end{align*}
\]

In the above two partial sequences, part p12 from \{p15(x)\rightarrow p16(-z)\rightarrow p12(x)\} has the same removal direction as p13. It is set by STEP 3: \{p14(z)\rightarrow p15(x)\rightarrow p16(-z)\rightarrow p12(x)\}. The merged sequence of the two sub-groups is \{p14(z)\rightarrow p15(x)\rightarrow p16(-z)\rightarrow p12(x)\rightarrow p13(x)\rightarrow p7(x)\} = \{S_1\}.

To generate the correct order of \{S'_i\} and \{S'_j\}, the following heuristic algorithms are applied:

**Heuristic Rules for the Determination of the Order of \{S'_i\} and \{S'_j\}:**

Search forward for part P" from both \{S'_i\} and \{S'_j\} which are farthest along the removal direction of common part P based on the heuristic procedures about
multiple target elimination discussed in section 5.2.2. The sequence of disassembly sub-group having part \( P'' \) is set prior to the other. For example, if \( P'' \) is found from \( \{S_{i''}\} \), then \( \{S_{i''}\} \) is prior to \( \{S_{j''}\} \). That is \( \{S_{i''}\} \rightarrow \{S_{j''}\} \). Otherwise, the sequence will be \( \{S_{j''}\} \rightarrow \{S_{i''}\} \).

5.3.2. Case II: Different Removal Directions of \( P \)

For the second case, which the removal directions of the common part are different in the two sub-groups, one of following situations may correspond to it:

1) common part \( P \) is the last part in the both sequences of the two sub-groups.
2) common part \( P \) is the last part of one of the two sequences.
3) common part \( P \) is not the last part of either sequence.

For the first situation, the removal direction of \( P \) is not important since it is the last part to be removed. The HEURISTIC RULE of STEP 4 for the first case discussed in section 5.3.1 can be applied here to determine the order of the two sequences. The removal direction of \( P \) is set to be the same as the one having higher priority. For example, in the two sequences \( \{p6(z)\rightarrow p5(z)\} \) and \( \{p4(-x)\rightarrow p3(-x)\rightarrow p2(-x)\rightarrow p7(-x)\rightarrow p11(x)\rightarrow p10(x)\rightarrow p9(x)\rightarrow p8(x)\rightarrow p5(x)\} \) of Assembly I, common part \( p5 \) is removed in different directions. The parts linked to common part \( p5 \) are \( p6 \) and \( p8 \) respectively. Since the removal direction of the part \( p6 \) which is \( +Z \) is prior to the removal direction of the
part p8 which is +X, the merged sequence is \(\{p6(z)\rightarrow p4(-x)\rightarrow p3(-x)\rightarrow p2(-x)\rightarrow p7(-x)\rightarrow p11(x)\rightarrow p10(x)\rightarrow p9(x)\rightarrow p8(x)\rightarrow p5(z)\}\) = \(\{S_2\}\).

For the second situation, the removal direction of common part P which is the last part in one of the two sequences is not important, and therefore, it is set to be the same as the removal direction of P in the other sequence. Then the order of the two sequences is determined based on the heuristic algorithm discussed in the first case. For example, in the two sequences \(\{S_1\} (\{p14(z)\rightarrow p15(x)\rightarrow p16(-z)\rightarrow p12(x)\rightarrow p13(x)\rightarrow p7(x)\})\) and \(\{S_2\} (\{p6(z)\rightarrow p4(-x)\rightarrow p3(-x)\rightarrow p2(-x)\rightarrow p7(-x)\rightarrow p1(-z)\rightarrow p11(x)\rightarrow p10(x)\rightarrow p9(x)\rightarrow p8(x)\rightarrow p5(z)\})\) of Assembly I, common part p7 is removed in different directions: +X and -X. In the former sequence \(\{S_1\}\), since part p7 removed in the -X direction is the last part, its removal direction is set to be +X as in the latter sequence \(\{S_2\}\). Then, the two sequences are merged according to the rule of STEP 3 in the first case. Part p2 from the latter sequence \(\{S_2\}\) has the same removal direction as common part p7 does. So, the merged sequence is \(\{p14(z)\rightarrow p15(x)\rightarrow p16(-z)\rightarrow p12(x)\rightarrow p13(x)\rightarrow p6(z)\rightarrow p4(-x)\rightarrow p3(-x)\rightarrow p2(-x)\rightarrow p7(-x)\rightarrow p1(-z)\rightarrow p11(x)\rightarrow p10(x)\rightarrow p9(x)\rightarrow p8(x)\rightarrow p5(z)\}\) which is the resulting sequence for Assembly I.

For the third situation, the sequences \(\{S_i\}\) and \(\{S_j\}\) have global defects though they are locally correct. The different removal directions of common part P in \(\{S_i\}\) and \(\{S_j\}\) show that part P cannot actually be removed before both the parts in \(\{S_i''\}\) and the parts in \(\{S_j''\}\) are removed. Therefore, one of the originally generated sequences \(P \rightarrow\)
\{S_1''\} or \{P->\{S_j''\}\} needs an adjustment. To determine which one of the two sequences (\{S_1''\} and \{S_j''\}) is to be adjusted, the rule is that the one which has less number of parts is adjusted. The adjustment is to reverse the sequence and the removal directions of the parts in the sequence so that common part \(P\) is the last part in the new sequence. Assuming that \{P->\{S_i''\}\} has fewer parts, it is reversed to become \{\{S_i''\}->P\}, and the disassembly sequence (\{S_i\} = \{\{S_i'\}->P->\{S_i''\}\}) of \([A_i]\) becomes \{\{S_i'\}->\{S_i''\}->P\}, where \{S_i''\} is the reverse of \{S_i''\}. The disassembly sequence of \([A_i]\) is still \{\{S_j'\}->P->\{S_j''\}\}. The rules used to sort the sequences before common part \(P\) can be applied to merge the two sequences together. The final point to make is that the disassembly direction of common part \(P\) in the final disassembly should be the one where \(P\)'s group is not adjusted.

Figure 5.8(a) shows an assembly structure with seven parts, A, B, C, D, E, F, G. Its liaison graph and contact relation graph were generated and are shown in Figure 5.8(b). The next task is the decomposition of the assembly into the sub-groups. Part A was identified as a cutvertex, and the assembly was decomposed into two sub-groups as shown in Figure 5.8(c). For the two sub-groups, the disassembly sequences were created as follows and are also shown in Figure 5.8(d):

\[
\{S_1\} = \{C(z)->B(z)->A(z)->D(z)\} \\
\{S_2\} = \{F(x)->A(x)->E(x)->G(x)\}
\]
Figure 5.8. An Example of Multiple Removal Directions Assembly--Assembly III
Obviously, common part A has different removal directions in the two sub-groups. Therefore, the above rules were applied. Since, in \( S_1 \), the parts behind common part A is fewer than that in \( S_2 \), the adjustment was carried out on the sequence \( \{ A(z) \rightarrow D(z) \} \) in \( S_1 \). The new sequence of this portion in \( S_1 \) became \( \{ D(-z) \rightarrow A(x) \} \). The resulting sequence for the first sub-group \( S_1 \) is therefore

\[
S_1 = \{ C(z) \rightarrow B(z) \rightarrow D(-z) \rightarrow A(x) \}
\]

After merging, the final sequence of disassembly of the structure is (Figure 5.8(e))

\[
\{ C(z) \rightarrow B(z) \rightarrow D(-z) \rightarrow F(x) \rightarrow A(x) \rightarrow E(x) \rightarrow G(x) \}
\]

After all the sub-group sequences were merged into one sequence, the disassembly sequence was reversed to become a complete assembly sequence.

\[
\{ G(x) \rightarrow E(x) \rightarrow A(x) \rightarrow F(x) \rightarrow D(-z) \rightarrow B(z) \rightarrow c(z) \}
\]

5.4 CONSTRUCTING SUB-ASSEMBLIES

A sub-assembly is defined here as a group of two or more individual parts, such that the parts can be assembled together separately from other parts of the product and that the sub-assembly itself can be treated as a individual part in the assembly sequence to be assembled to the product structure.

Based on this definition, it is required that suitable sub-assemblies should have minimal connections with other parts. Therefore, the group of parts which construct a bi-connected graph for the connectivity relationships discussed in section 4.2 is the
candidate. In addition, the sub-assembly requires that all its members must be successive components in the generated complete assembly sequence (in section 5.3). As the result of the requirements for a sub-assembly, a group of parts can construct a sub-assembly if they satisfy the following two conditions:

1. The connective graph of the parts is bi-connected.
2. The parts are successive components in the generated complete assembly sequence.

Take Assembly I as an example, it is split into four bi-connected groups with corresponding sub-sequences: \( \{p4->p3->p2->p7->p1->p11->p10->p9->p8->p5\} \) of group [A], \( \{p6->p5\} \) of group [B], \( \{p14->p13->p7\} \) of group [C], and \( \{p15->p16->p12->p13\} \) of group [D]. The generated final complete sequence \( \{S\} \) is \( \{p14-> (p15->p16->p12->p13) ->p6-> (p4->p3->p2->p7->p1->p11->p10->p9->p8->p5) \} \). The parts of group [A] and [D] are successive in the final assembly sequence \( \{S\} \). Therefore, groups [A] and [D] are sub-assemblies while groups [B] and [C] are not. Then the sequence can be: \( \{p14->(A)->p6->(D)\} \), where \( \{A\} \) and \( \{D\} \) are the corresponding sequences of group [A] and [D] respectively.

5.5 SUMMARY

This chapter presents algorithms and heuristics for generating assembly sequences. The system decomposes an assembly into sub-groups by identifying cutvertices from the
liaison graph. For each sub-group, it uses built-in assembly logic and heuristics to generate disassembly sequences based on contact relation graphs. The merger of disassembly sequences for a pair of sub-groups is carried out by identifying the disassembly sequence of the common part in the two sub-group disassembly sequences. The final assembly sequence is obtained by reversing the disassembly sequence.
CHAPTER 6

6. IMPLEMENTATION

6.1 THE SYSTEM STRUCTURE AND IMPLEMENTATION ALGORITHMS

The assembly planning system was developed in Smalltalk 80 as shown in Figure 6.1. It consists of three main parts: 1. a data access, 2. a product data base -- the assembly model, and 3. an assembly sequence generator. The results are stored in an output file. The data access is essentially a user interface with which the user inputs product information based on the feature-based representation. To assist the user in the data entry process, a template is provided to the user to input the information describing the product. The template adopts the exact format of the product representation from Feature-Based product Design Model (as shown in Figure 4.4). Figure 6.2 shows the sample template.
Figure 6.1 An Object-Oriented Assembly Planning System
Assembly 1
16
p1 Gear
14
f1 face
73 60 60 180 -90 90
1
contact p7
f2 cylo
76 60 60 180 -90 90
0
f3 face
78 60 60 0 90 -90
.
.
p2 washer
4
f1 face
66 60 60 180 -90 90
l
contact p3
.
.
Figure 6.2. A Sample Template for Product Data Input
As the user inputs the product information, the interface transforms the input to a format that is directly used by the assembly planner to generate the assembly sequence. The product data base -- the assembly model -- stores the product information in two levels, part level and feature level, corresponding to the product design representation. The syntax is predefined and will be discussed in the following section. The assembly sequence generator retrieves the information from the product data base, and uses a four-stage procedure to complete the planning task. Figure 6.3 shows the implementation algorithms. Figure 6.3(a) is the general planning process; Figure 6.3(b) shows the search procedures for removable parts; Figure 6.3(c) is the details of part searching in a given direction; Figure 6.3(d) is the synthesis of the final assembly sequence. During planning, the intermediate results of each stage are stored in the results output for the next stage planning. The final planning results are stored in an output file.
Figure 6.3(a). The General Algorithm for the System Implementation
Figure 6.3(b). The Removable Part Search
The Methods of Part Searching in a Defined Direction

- **Search in a defined direction**
- Searching for contact-free part(s) -- target part(s)
- The target(s) is/are found
- False
- True
- n-number of target parts
- True
- False
- n>1
- Eliminating the unsuitable target part(s)
- False
- True
- The best target part
- Searching for non-touch constraints
- False
- True
- Found constraints
- Candidate<-- target part
- Checking other part(s)' stability if rmoving the candidate
- False
- True
- The parts are stable
- False
- True
- A part that can be disassembled is found

Figure 6.3(c)
Figure 6.3(d). The Synthesis of the Sequences
6.2 OBJECT ORIENTED CLASS HIERARCHY

The assembly planning system has been implemented in Smalltalk: an object oriented programming language. In the object oriented programming environment, every real world entity is modelled as an object. An object has a state and behaviour. The state of an object is a set of attributes called instance variables of the object. The behaviour of an object is a set of methods, each of which describes how an operation on the object is to be computed. A class groups all the objects that share the same state and behaviour. Objects that belong to a class are called instances of the class. This research applies the object oriented concept to construct the product data model. The overall object oriented class structure of the assembly planning system is shown in Figure 6.4. Five new classes: GeometricEntity, Part, Feature, SubAssembly, and Assembly have been developed. Class GeometricEntity is directly placed under the Smalltalk 80 ultimate superclass Object. Classes Part, Feature, and SubAssembly are the immediate sub-classes of the GeometricEntity class, These classes are defined for describing the product design and partial assembly plans respectively which will be discussed individually later in sections 6.2.1 to 6.2.4. The Assembly class is placed under the Smalltalk 80 OrderedCollection class. The Assembly class is responsible for all the sequences planning activities. The class hierarchy has been developed in such a manner that the concept of inheritance is used as much as possible.
Figure 6.4. The Hierarchical Structure of the Classes
6.2.1 GeometricEntity Class

The GeometricEntity class abstracts the common characteristics from the Part class, the Feature class, and the SubAssembly class. The objects of part, feature, and sub-assembly are grouped into the GeometricEntity class since they share same state and behaviour, such as the type of the object and the name of the object. Therefore, each instance of the GeometricEntity class has the following instance variables or attributes: name and type. The value of instance variable name is the unique identifier of an object. The value of instance variable type represents the sort of the object. These instance variables are inherited by all the subclass of the class hierarchy though the actual values contained in the instance variables may differ. For example, if the object is aPart (an instance in Part class), its type could be bolt, shaft, or gear; if the object is aFeature (an instance in Feature class), its type could be face, cylinder, or spline.

The methods in GeometricEntity class are used for creating new instances. These methods will be inherited by its sub-classes.

6.2.2 Part Class

The Part class is created to represent the complete information about every part of the product assembly model. The information is obtained by reasoning the product design model that has been discussed in detail in section 4.2. The Part class inherits all
the instance variables of its super class. In addition, each instance of class Part has the following instance variables: component, neighbour, relatingComponent, n_relation (n = ±X, ±Y, or ±Z), and featurePosition. The instance variable component provides all the features that the current instance of the Part class, aPart, holds. The features are the instances defined in class Feature. The instance variable Neighbour contains all the related parts' names for aPart. RelatingComponent records the features' names of aPart that has relations with other parts. N_relation uses a dictionary to collect related parts of aPart and the position values of the aPart's features connecting those parts in a particular direction n. FeaturePosition uses an array to hold a list of features' position values (n_min and n_max) of aPart in six defined directions (±X, ±Y, ±Z). The normal of the features is n (n = ±X, ±Y, ±Z), and the position values are n_min and n_max. An example of an instance of Part is shown in Figure 6.5.

The Part class contains a number of methods. It also inherits all the methods of its super class. All the methods deal with creating new instances and modification in the data access process. There are no class variables for this class.

6.2.3 Feature Class

Class Feature is used to depict the location of a feature of a part and relations between the feature of the part and other parts in the assembly model. This information is directly retrieved from the product design model. Feature has two instance variables:
location and relation. Location specifies the position (x, y, z) and orientation (nX, nY, nZ) of the current instance in class Feature, aFeature. Relation collects all part names that aFeature has connection with. An example of an instance of Feature is shown in Figure 6.6.

Class Feature inherits all the methods of its super class. It also has some methods for creating new instances for itself. There are no class variables for this class.

6.2.4 SubAssembly Class

The SubAssembly class is used to store generated temporary sub-assemblies and permanent sub-assemblies in assembly planning. So-called permanent sub-assemblies are those defined in section 5.4. So-called temporary sub-assemblies are the groups of parts which are preferred to be assembled together as sub-assemblies, but are not consistent with the sub-assembly definition in section 5.4, like the case discussed in section 5.2.3. The SubAssembly class inherits the instance variables from its superclass GeometricEntity: type and name. In addition, it has an instance variable: sequence, which records the disassembly sequence of the parts contained in the current instance, aSubAssembly. An example of an instance of class SubAssembly is shown in Figure 6.7.

The SubAssembly class inherits all the methods of its super class. It also has its own methods. All the methods are used to created new instances.
Figure 6.5. An Instance of Class Part
Figure 6.6. An Instance of Class Feature
Figure 6.7. An Instance of Class SubAssembly
6.2.5 Assembly Class

The Assembly class is created for all the operations of assembly sequences generation. It has no instance variables. The Assembly class has two class methods and a lot of instance methods. The two class methods form the user interface. The instance methods are used for reasoning the product information stored in the Part class and Feature class and generating appropriate assembly sequences.

6.3 METHODS AND MESSAGE PASSING

In the object-oriented environment, the assembly plan is achieved through a set of objects communicating with each other. When an object receives a message it determines whether it has an appropriate operation or method to allow it to respond to the message. The definition of the method describes how the object will react upon receiving the message.

All the methods created for assembly sequences generation are resided in the Assembly class. The methods include two class methods and a number of instance methods. One class method, dataAccess, reads in the product information from the template file based on the feature-based product design model, then converts the information into the assembly model which is a format of product representation to be more easily used by the assembly planner. The other class method, example, outputs the
intermediate and final result, into a data file. This file can be read and printed out by the user.

There are many instance methods created in the Assembly class to implement the algorithms described in Chapter 5. These methods are grouped into several groups for different planning stages: liaison, splitting, sequencing, and synthesising. The liaison group clusters the methods for building connectivity relation graphs -- liaison graph and contact relation graphs. The splitting group contains the methods for decomposing an assembly into sub-groups. The sequencing group collects the methods for generating disassembly sequences for sub-groups. The synthesising group gathers the methods for merging the sub-sequences into a complete assembly sequence. The methods in the sequencing group are split into three sub-groups: targetSearch, candidateSearch and stabilityCheck. The methods in the targetSearch group determine the best target part. The methods in the candidateSearch group identify the non-touch constraints for the target part. The methods in the stabilityCheck group make sure that the assembly structure is stable when a part is being removed. In the synthesising group, the methods are divided into two groups: sameRemovalDirection and difRemovalDirection group. The sameRemovalDirection group is used for merging two sub-sequences when their common part P is removed in the same directions. The difRemovalDirection group is designed for dealing with the case that common part P is removed in different directions.

Through the communication between the objects in the Assembly class and objects
in the Part class as well as the Feature class, the product information stored in the Part class and the Feature class is analyzed based on the heuristic algorithms described in Chapter 5 to accomplish the planning task.

To test the algorithms, several examples were used. They are Assembly I shown in Figure 4.2, Assembly II shown in Figure 4.5, Assembly III shown in Figure 5.8 for this thesis, and other examples in references (Yan and Gu, 1994, Gu and Yan, 1994). The Assembly II example was taken from the literature in order to allow comparison of this method with other methods. Figures 6.8 and 6.9 show the records generated as output for the first two examples by the planning system during the assembly planning using the above approach. The planning system was run on Sun 3/60. The running time for Assembly I, Assembly II, and Assembly III is 5'32", 2'40", and 42" respectively.

6.4 SUMMARY

This chapter describes the implementation of the system in an object oriented environment: Smalltalk 80. The detailed implementation algorithms are given. Several new classes are created to store the product information and carry out the planning operations. The major methods generated in these new classes are explained to show the inner mechanisms of the system and the establishment of the user interface.
### Mechanical Product Assembly Sequence Generation

#### Step 1. Establishing the liaison of the product

| p1 | p7 | p8 |
|    |    |    |
| p2 | p3 | p7 |
| p3 | p2 | p4 |
| p4 | p3 | p5 |
| p5 | p3 | p4 |
| p6 | p5 |
| p7 | p1 | p2 |
| p8 | p1 | p5 |
| p9 | p5 | p7 |
| p10| p9 | p11|
| p11| p5 | p9 |
| p12| p13| p16|
| p13| p12| p15|
| p14| p7 | p13|
| p15| p13| p16|
| p16| p12| p15|

#### Step 2. Grouping the parts

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
<th>p7</th>
<th>p8</th>
<th>p9</th>
<th>p10</th>
<th>p11</th>
</tr>
</thead>
<tbody>
<tr>
<td>p5</td>
<td>p6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p7</td>
<td>p13</td>
<td>p14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p12</td>
<td>p13</td>
<td>p15</td>
<td>p16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Step 3. Ordering every group

<table>
<thead>
<tr>
<th>p4</th>
<th>p3</th>
<th>p1</th>
<th>p7</th>
<th>p1</th>
<th>p11</th>
<th>p10</th>
<th>p9</th>
<th>p8</th>
<th>p5</th>
</tr>
</thead>
<tbody>
<tr>
<td>p6</td>
<td>p5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p14</td>
<td>p13</td>
<td>p7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p15</td>
<td>p16</td>
<td>p12</td>
<td>p13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Step 4. Merging every ordered group into one disassembly sequence

|----|----|----|----|----|----|----|----|----|----|----|----|----|

#### Step 5. Generating assembly sequence


End

---

Figure 6.8. The Output of the Assembly Planning for Assembly I
Mechanical Product Assembly Sequence Generation

Step 1. Establishing the liaison of the product

A N P D L M K Q B C
B A G H J C E O
C E J N A D
D P C A
E Q B C O F
F Q E
G H A B
H B L K J
L J A K H M
J L H B
K L H A
M L A
P A D
Q B E
Q E F
N C A

Step 2. Grouping the parts

G H A B L K J N P D M C O E
F Q E

Step 3. Ordering every group

M L K J H G O E P D N C B A
Q F E

Step 4. Merging every ordered group into one final disassembly sequence

Q F M L K J H G O E P D N C B A

Step 5. Generating assembly sequence

A B C N D P E O G H J K L M F Q

End

Figure 6.9. The Output of the Assembly Planning for Assembly II
CHAPTER 7

7. CONCLUSIONS

7.1 CONCLUSIONS

Automatic generation of assembly sequences from a CAD database is a complex issue because of the combination of a variety of assembly relations and the unlimited possible part geometries. A graph-based heuristic approach in association with a feature-based product design representation is implemented and a number of test cases have been presented to automatically generate assembly sequences. The assembly sequence of a product is defined as the reverse of the disassembly sequence. To generate a feasible disassembly sequence, the planner uses the feature information to create two types of connective graphs: the liaison and contact relation graphs. The liaison graphs are used to form sub-groups by identifying cutvertices in the graphs. The decomposition of an
assembly into sub-groups reduces the degree of the complexity in implementation of the planning system. The contact relation graphs are defined by transferring all the assembly relations into "contact" relations which are then used to determine the precedence relationships between the components of a product. This unified representation has provided a means for achieving the automation of the assembly sequence planning. The contact relation graphs are used to determine the disassembly sequences for each of the sub-groups. The merging of disassembly sequences for a pair of sub-groups is carried out by identifying the disassembly sequence of the common part in the two sub-group disassembly sequences. The algorithms using disassembly direction and the feature location and orientation were also discussed with examples. Several examples were used to test the effectiveness of the approach. The results show that the approach can be used to automatically generate assembly sequence directly from CAD data bases.

7.2 THE CONTRIBUTION

The graph-based heuristic approach proposed in this study for generating assembly plans has been proven to be very efficient. In summary, the assembly planning system developed in this research possesses the following features:

1. The assembly precedence are determined through geometrical analysis as well as functional relation analysis of the product. Various precedence constraints are incorporated in the contact relation graphs which make it easy to implement the
automatic assembly planning.

2. The decomposition of the assembly into sub-groups reduces the complexity of the relationships among the components in the assembly without any artificial simplification.

3. The heuristic rules are generated by analyzing assembly structure and mechanical part characteristics using features. They are sufficiently general for assembly sequences planning.

4. It is an automated system, no human intervention is required. The system can be used to integrated CAD database with robot task planning and other assembly application systems.

5. The assembly sequences generated are feasible and applicable to practical mechanical products.

7.3 FUTURE RESEARCH

There are many directions that the future research can take with respect to this project. One important direction is to integrate the assembly planning system with the STEP Model since STEP provides an international standard data format for the product information. This integration would build a bridge between research efforts and industry practice.

For an product assembly, there may exist several feasible and applicable assembly
sequences. The optimal sequence highly depends on the working environment such as available tools and fixtures. In the current system, only one feasible assembly sequence is generated for a product. There is a potential for the system to be able to generate all the feasible assembly sequences.

In addition, the system could be further improved to specify the tool requirements according to the generated different assembly sequences. Thus, the optimal sequence could be determined based on a particular working environment.

Furthermore, with the concurrent engineering or life-cycle design gradually dominates the mechanical design area, most product life cycle factors such as assemblability/disassemblability, serviceability, and recyclability are considered in early design stage. This system could be further developed and extended to assist the designer in balancing various life-cycle factors during the product design process.
REFERENCES


Bourjault, A., 1987, "Outils methodologiques pour la definition de systems d’assemblage automatises" Universite de Franche-Comte Centre de Recherche Microsystemes et Robotique, Feb.


Cincinnati, Ohio, May 13-18, pp. 1594-1599.


Lozano-Perez, T. and Wesley, M. A., 1977, "LAMA: A Language for Automatic Mechanical Assembly", *Proc. of 5th Int'l Joint Conf. on AI*, pp. 188-191.


APPENDIX A: THE ALGORITHM OF ASSEMBLY DECOMPOSITION

*Assume that the initial assembly is ASM.
*Array Primitive-ASM[maximum] is used to hold all the resulting Primitive Assemblies.

Assembly.Split{

    Temp-Asm[100]; /**<temporal assembly array*/
    CutVertexFound = False;
    Current-Part = the first part in This assembly;
    While(Current-Part != Null){
        i = 0;
        for(every neighbour of current part, sayB){
            Temp_Asm[i]<--Current-Part;
            Temp-Asm[i]<--connect-on-part(B);
            i = i+1;
            remove Current-Part from this assembly;
        }
        if(i=1){/*not a cutVertex*/
            put everything in Temp_Asm[0] back to the original assembly(this assembly);
            Primitive-Asm<-- this assembly;
        }
        if(i>1){/*is a cutVertex*/
            for(j=0; j<i; j++){
                Temp-Asm[j].Split();
            }
            CutVertexFound = True;
        }
        Current-Part = Current-Part.next;
    } /*end while loop*/
    if(CutVertexFound == False){
        Primitive-Asm<-- This Assembly;
    }
}