

**Sub-assembly Partitioning Choice for Complex Assemblies
Based on an Action-Count-Closure Criterion**

by

Stephen J. Rhee

B.S., California Institute of Technology (1994)

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

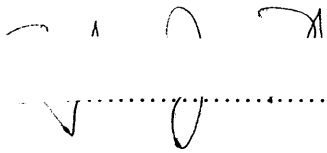
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1996

© Stephen J. Rhee, 1996. All rights reserved.

The author hereby grants to M.I.T. and The Charles Stark Draper
Laboratory, Inc. permission to reproduce and distribute copies of
this thesis document in whole or in part.

Author.....



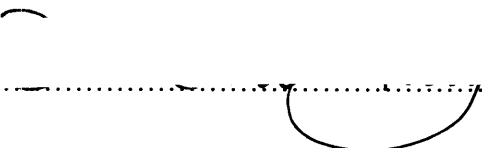
Department of Mechanical Engineering
August 7, 1996

Approved by.....



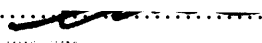
Thomas L. De Fazio
Technical Staff Member
The Charles Stark Draper Laboratory
Technical Supervisor

Certified by.....



Daniel E. Whitney
Senior Research Scientist
Center for Technology, Policy, and Industrial Development
Lecturer, Department of Mechanical Engineering
Thesis Supervisor

Accepted by.....


MASSACHUSETTS INSTITUTE
OF TECHNOLOGY
Chairman, Departmental Committee on Graduate Students

Ain A. Sonin

DEC 03 1996

Eng.

LIBRARIES

Sub-assembly Partitioning Choice for Complex Assemblies Based on an Action-Count-Closure Criterion

by

Stephen J. Rhee

Submitted to the Department of Mechanical Engineering
on August 7, 1996 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

Taking manufacturing and assembly issues into consideration during the design phase has been shown to be capable of reducing production costs of a given design. Although Design-For-Assembly (DFA) methods currently exist that aid the designer by suggesting systematic redesign, it is felt that while these methods are useful for simple designs, e.g. those having a limited number of parts, they may be inadequate for more complex assemblies. This thesis attempts to address how DFA may be used successfully for such complex assemblies through subassembly repartitioning or minor redesign.

Complex assemblies are generally characterized by having a large parts count with the assembly organized as a collection of subassemblies. In addition to having little or no design freedom, a complex assembly may contain assembly moves in which a large number of kinematic degrees of freedom (actions) must be closed, i.e. fixed, during the assembly move. This action count closure can then be taken as a quantifiable measure of difficulty of a particular assembly move. It has been determined that the design freedoms that are available to a designer for redesigning a product for easier assembly consist primarily of detail redesign, case (nonfunctional) redesign, and subassembly repartitioning. By using Assembly Sequence Analysis (ASA), an approach for suggesting how detail redesign may be applied successfully to an assembly has been studied. To address the issue of subassembly repartitioning, a tool has been developed that suggests how an assembly may be repartitioned favorably according to specified criteria. This tool searches a space of possible subassembly repartitions using a genetic algorithm. Although the developed tool is capable of handling multiple criteria, the criterion used in this thesis was based on measuring difficulty of assembly by using the action counts of the assembly. This tool is also capable of determining favorable assembly sequences in addition to subassembly repartitionings with respect to the specified criteria. It has been found that while the action count criterion is useful in locating difficult assembly moves, it may not be the driving criterion in determining good assembly sequences and subassembly partitioning schemes suggesting that other criteria, e.g. minimizing the number of reorientations or refixturings during assembly, should also be considered. Case studies of real industry assemblies subjected to these approaches are illustrated.

Thesis Supervisor: Daniel E. Whitney
Senior Research Scientist
Center for Technology, Policy, and Industrial Development
Lecturer, Department of Mechanical Engineering

Acknowledgements

First and foremost I would like to thank both my advisors, Dr. Tom De Fazio and Dr. Daniel Whitney from whom I have learned a great deal. They have not only helped me to grow as a student but as a person too. I believe that the experience they have given me will prove invaluable to me for the rest of my life.

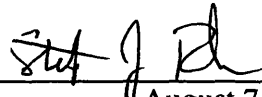
Secondly, I'd like to thank all the many friends who have given me their trust and support over the years. At the risk of leaving any one name out, I will let them all remain nameless as they should know who they are.

Finally, I would also like to thank my family for giving me their support and always having faith in me.

This material is based upon work supported by the National Science Foundation under Grant No. DMI-9414984. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Contract NSF 27000. Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein, It is published for the exchange and stimulation of ideas.

I hereby assign my copyright of this thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.


August 7, 1996

Permission is hereby granted by The Charles Stark Draper Laboratory, Inc., to the Massachusetts Institute of Technology to reproduce any or all of this thesis.

CONTENTS

1 Introduction	8
1.1 Motivation	8
1.2 Goal of Research	10
1.3 Thesis Overview	10
2 Related Works	11
2.1 Design For Analysis	11
2.1.1 Boothroyd-Dewhurst DFA Method	11
2.1.2 Hitachi Method	11
2.2 Assembly Sequence Analysis	12
2.2.1 Precedence Relation Generation	12
2.2.2 Assembly Sequence Representation	13
2.2.3 Assembly Sequence Evaluation	13
2.3 Other Assembly Planners	13
2.3.1 Subassembly Extraction	14
2.3.2 Simulated Annealing	14
2.3.3 Neural Network	14
2.3.4 Genetic Algorithm	14
2.4 Summary	15
3 Design-For-Assembly Methods for Complex Assemblies	16
3.1 Subassembly Repartitioning and Sequence Generation	16

3.1.1 Genetic Algorithms	17
3.1.2 Weighted Liaison Graph	19
3.1.3 Action Count	21
3.1.4 Assembly Sequence and Subassembly Partitioning Encoding	25
3.1.5 Crossover and Mutation Operators	28
3.1.6 Objective Function	29
3.2 DFA Through Precedence Relation Elimination	30
3.3 Summary	31
4 Case Studies of Actual Complex Assemblies	32
4.1 AFI Assembly	32
4.1.1 GA Sequence Generation	35
4.1.2 GA Subassembly Partitioning of AFI-H	36
4.1.3 GA Subassembly Partitioning of AFI-B	40
4.1.4 GA Subassembly Partitioning of AFI with No Decomposition	44
4.2 BAT Assembly	47
4.2.1 GA Sequence Generation and Subassembly Partitioning of BAT	50
4.2.2 GA Subassembly Repartitioning of BAT-F	53
4.2.3 Precedence Relation Elimination Applied to BAT	58
4.3 Summary	62
5 Conclusions and Further Work	63

List of Figures

3-1	De Fazio and Whitney's assembly from industry	20
3-2	Liaison graph for the assembly from industry	20
4-1	Detailed drawing of the AFI assembly	33
4-2	AFI with original subassembly partitioning	34
4-3	Original subassembly H; broken down into subassemblies Ha and Hb	37
4-4	Liaison diagram for AFI-H	37
4-5	Subassemblies Ha, Hb, J, and L	39
4-6	Subassembly B and broken down into subassemblies Ba, Bb, Bc, and Bd	41
4-7	Liaison diagram for AFI-B	42
4-8	Subassembly repartitioning of AFI-B	43
4-9	Suggested subassembly partitionings of AFI with no subassemblies broken down	46
4-10	Detailed drawing of the BAT assembly	48
4-11	BAT assembly with original subassembly partitioning	49
4-12	Liaison diagram for BAT assembly	49
4-13	Suggested partitioning of the BAT assembly with original subassemblies	52
4-14	Breakdown of subassembly F	54
4-15	Liaison diagram for BAT-F	54
4-16	Subassemblies A, E, F, and G before repartitioning and after	59

List of Tables

3-1	Weighted liaison graph information for assembly from industry in tabular form	21
4-1	Weighted liaison graph information for AFI-H in tabular form	38
4-2	Weighted liaison graph information for AFI-B in tabular form	42
4-3	Weighted liaison graph information for BAT in tabular form	50
4-4	Weighted liaison graph information for BAT-F in tabular form	55
4-5	Precedence relations for BAT	59
4-6	Precedence relations for BAT-RD1	60
4-7	Precedence relations for BAT-RD2	61
4-8	Precedence relations for BAT-RD3	61
4-9	Number of feasible assembly sequences	62

CHAPTER 1

Introduction

1.1 Motivation

It has been shown that it is necessary to consider manufacturing and assembly issues during the design phase of product development for numerous reasons. Doing so will allow for a better designed product in terms of ease and cost of manufacture and assembly in addition to a more robust product that best satisfies its intended functionality. Traditionally, product development began with the design process where a design engineer or design team would be responsible for producing a detailed design from which a given product would be manufactured. This design process consists primarily of product specification, preliminary design, and detail design. After the design has been finalized, product development enters the manufacturing phase where it is up to the manufacturing engineer or team to determine how best to manufacture and assemble the given design. However, since the design has already been fixed, much of the freedom in how the product can be assembled has been removed, often leading to difficult and costly assembly. It is the goal of design for assembly (DFA) methods to aid the designer in taking assembly issues into consideration during the design process to produce a design that can be better manufactured. This is done by taking advantage of the freedoms that exist during the design phase to optimize the design for manufacture.

Although DFA methods currently exist, the current methods are typically based on generic checklists and lookup tables of numerical approximations developed primarily through extensive experience [1]. It is felt by the author that these DFA methods are useful for simple designs, e.g. those having a limited number of parts, due to use of numerical tallies that can lead to significant error as more and more approximations are used with assemblies with large parts counts.

Furthermore, these DFA routines are concerned with generic rather than the design-specific issues necessary for consideration when dealing with complex assemblies.

A complex mechanical assembly is defined qualitatively as a densely packed assembly with many connections between parts and tight dimensional tolerances. This implies that a complex assembly consists of many parts usually with the assembly organized as a collection of subassemblies, with specifications given for performance, size, weight, and bulk that are considered fixed. There is little or no freedom to change material or make process substitutions, or to make parts-count reductions. In addition to having little or no design freedom, a complex assembly may contain assembly moves in which a large number of kinematic degrees of freedom (actions) must be closed during an assembly move. One can quickly see how a complex assembly differs from a simple assembly and why conventional DFA methods may no longer be applicable. Previous research in the realm of complex assemblies [2] has shown that design changes made for easier assembly occasionally ran contrary to those proposed by conventional DFA routines or practice. These included increased parts counts or other design changes that would not be suggested by current DFA practice. It has been determined in this research that the design freedoms that are readily available to a designer for redesigning a complex assembly for easier assembly consist primarily of detail redesign, case (nonfunctional) design, and subassembly repartitioning.

To develop a successful design for assembly methodology for complex assemblies, it is necessary to be able to evaluate how design changes affect assembly options for a given design. Computer-aided assembly planning has long been of interest for its ability to allow for the rapid development of assembly plans for a given design and for this reason is a useful means of evaluating design changes. Assembly sequence analysis (ASA) is one such method for computer-aided assembly planning. ASA consists of generating all feasible assembly sequences using precedence relations to maintain geometric feasibility, editing the assembly sequences using given criteria, and evaluating the remaining assembly sequences according to cost of assembly. ASA then offers a rapid means by which a design can be evaluated. Thus far, ASA has been used

primarily as a one way tool, using design information to generate an acceptable assembly plan. One can see the possibilities that ASA offers as a means of conveying information from the assembly realm back to the design process necessary in DFA to allow for an environment that promotes the idea of concurrent engineering (CE).

1.2 Goal of Research

The goal of this research is to propose a DFA methodology using ASA as a means for taking assembly issues into consideration during the design process that may be applied successfully to complex assemblies. A software tool based on a new action-count closure criterion is developed that allows for automatic subassembly repartitioning and assembly sequence generation. This software tool makes use of a genetic algorithm. In addition, an approach by which detail redesign is suggested by current ASA methods is examined. Using these tools, a design can be rapidly processed and have its assembly options presented for judgment and offer redesign or repartitioning suggestions that will make for better assembly.

1.3 Thesis Overview

Chapter 2 presents a quick review of previous work in the field on which this work is built upon or related. Chapter 3 presents an overview of genetic algorithms necessary for understanding how the developed software tool works. Chapter 4 develops the subassembly partitioning tool and examines a technique for DFA in which it can be used. Chapter 5 gives results on two real complex assemblies from industry to see how the developed tool and techniques fare. Finally, Chapter 6 presents overall conclusions and suggested directions for future work.

CHAPTER 2

Related Works

This chapter gives a brief overview of previous works related to this work. Related works have been categorized into three broad subject areas: Design For Assembly (DFA), Assembly Sequence Analysis (ASA), and advanced assembly planners.

2.1 Design For Analysis

Traditional DFA methods have relied on quantitative evaluation procedures for evaluating assemblability and general guidelines and examples to aid in design improvements. Two of the most well known methods are the Boothroyd-Dewhurst method and the Hitachi method [3].

2.1.1 Boothroyd-Dewhurst DFA Method

The Boothroyd-Dewhurst method starts with an analysis of the current assembly process. Using numerical lookup tables for parts handling and parts insertion and fastening, the assembly process is evaluated. This analysis is used to signal parts that should be considered for redesign. This method suggests that one way of best minimizing the cost of assembly, the number of parts should be minimized.

2.1.2 Hitachi Method

The Hitachi Assemblability Evaluation Method (AEM) aids in design improvements by identifying parts that need redesign by two evaluation methods. The first is the assemblability

evaluation score ratio, E, which gives an indication of the difficulty of assembly operations for parts. The second is the assembly cost ratio, K, which is used to evaluate the cost of the assembly. Using these two indicators, this method targets quality and cost in an effort for better redesign.

2.2 Assembly Sequence Analysis

De Fazio and Whitney [4] describes Assembly Sequence Analysis (ASA) as the methodology in which all mechanically feasible sequences are first generated, then they are edited based on given criteria, and finally compared on an economic basis.

2.2.1 Precedence Relation Generation

Bourjault [5] originally used a graph of contacts or liaisons between parts named a "liaison diagram" to model an assembly where a node represents a part and an arc between nodes represent a connection between two parts. He then developed an algorithm that was capable of generating all possible assembly sequences based on a series of yes-no questions based on part mates. After receiving user-input answers to these questions, the computer would then generate a series of constraints referred to as precedence relations. The form of precedence relations used in this thesis is as follows:

$$L_i \& \dots \& L_j \geq L_m \& \dots \& L_n$$

which is read as all liaisons in the set of liaisons $\{L_i, \dots, L_j\}$ must be completed previous to or simultaneously with the completion of all liaisons in the set of liaisons $\{L_m, \dots, L_n\}$.

Subsequently, De Fazio and Whitney extended this method making it more efficient by adding new rules to the automatic reasoning that resulted in less questions to be answered by the user. Independent of this work, Homem De Mello [6] approached the task of precedence constraint generation using a different set of questions. Whipple [7] developed another method

known as the "onion-skin" method for generating all valid assembly sequences on a process of questions that likens disassembly to the peeling of an onion. Baldwin [8] subsequently took these methods and incorporated them into a software tool called SPAS that generates the precedence relations by asking the necessary questions to the user.

2.2.2 Assembly Sequence Representation

After having determined the precedence relations required to generate all possible assembly sequences, it is necessary to represent the assembly sequences. In his assembly sequence generation software [9], Bourjault uses a parts tree to represent assembly sequences which is not compact but is easy to comprehend as it somewhat mimics a physical assembly line. Homem De Mello developed an And-Or Graph representation [10] to represent assembly sequences which is a compact representation but difficult to use to see how an assembly progresses. De Fazio and Whitney developed the Liaison Sequence Diagram based on a directed graph to offer a compact representation of assembly sequences that also offers more information on an assembly sequence's state by state progress.

2.2.3 Assembly Sequence Evaluation

Using the Liaison Sequence Diagram, Lui [11] developed a program, SED (Sequence Edit and Display) that would create the Liaison Sequence Diagram and allow for editing of the sequence diagram. Abell [12] expanded on this and Whipple's stability analysis creating a fully user interactive software tool, EDIT, for editing and evaluating assembly sequences.

2.3 Other Assembly Planners

More recently, there have been new approaches at assembly planning using assembly sequence generation. Most notable and relevant to this work include Lee and Shin's method of assembly planning using subassembly extraction [13]; Milner, Graves, and Whitney's use of

simulated annealing for generating least-cost assembly sequences [14]; Hong and Cho's assembly sequence optimization using neural networks [15]; and Bonneville, Perrard, and Henrioud's use of genetic algorithms to generate and evaluate assembly plans [16].

2.3.1 Subassembly Extraction (Lee and Shin)

Lee and Shin describe a method for assembly planning using subassembly extraction. Starting with a weighted attributed liaison graph, a hierarchical partial order graph is generated. This graph is a representation of the assembly plans that explicitly shows the subassemblies required for assembly. This graph is generated through a process referred to as subassembly extraction. Subassembly extraction consists of grouping preferred subassemblies together based on evaluation of the weighted attributed liaison graph.

2.3.2 Simulated Annealing (Milner, Graves, and Whitney)

This method uses simulated annealing as the optimization method for finding least cost assembly sequences from the feasible sequences of a mechanical product. By using De Fazio and Whitney's representation of assembly sequences and given economic and performance data for assembly tasks, optimal sequences are determined based on a cost function.

2.3.3 Neural Network (Hong and Cho)

This method uses a neural network to obtain optimal assembly sequences based upon inferred precedence constraints and the assembly costs from an expert system. Due to the large size of the search space of assembly sequences, a neural network and an expert system is used to find optimized assembly sequences.

2.3.4 Genetic Algorithm (Bonneville, Perrard, and Henrioud)

This method uses a genetic algorithm to generate feasible assembly sequences as opposed to cut-set methods that generate all feasible sequences. The genetic algorithm also evaluates the sequences, which are represented as assembly trees, according to an evaluation function and searches for optimal sequences.

2.4 Summary

Here, we have presented past and present research directly related to this thesis. This research relies on previous work in the area of ASA for generating feasible assembly sequences and subassembly partitionings. The optimization of these sequences and partitionings is similar to the work in assembly planning mentioned in this chapter but differs in the overall approach and methods used.

CHAPTER 3

Design-For-Assembly Methods for Complex Assemblies

Complex assemblies can be characterized as having a large number of parts and organized as an assembly of subassemblies among other qualities. Due to these qualities, complex assemblies generally offer little or no design freedoms. Of the design freedoms that are available to a designer for redesigning complex products for easier assembly--detail redesign, case redesign, and subassembly repartitioning--this chapter describes the methods used and the required tools for applying detail redesign and subassembly repartitioning to complex assemblies because the application of case redesign is fairly straightforward and often done in common practice. First, the issue of how a complex assembly can be favorably repartitioned is addressed. To do so requires that a measure be developed that can quantify in some manner the difficulty of assembly in order to determine what is or is not a favorable repartitioning. Then, based on this criterion, a tool can be developed that allows for the computer-aided generation of favorable subassembly repartitionings. Secondly, a DFA approach for complex assemblies using detail redesign as well as some subassembly repartitioning is examined. This technique relies on using Assembly Sequence Analysis, specifically the precedence relations derived during ASA, as a means of signaling what aspects of the design should be considered for redesign prior to the actual assembly process to allow for better assembly.

3.1 Subassembly Repartitioning and Sequence Generation

Due to the large number of parts in a complex assembly, the assembly is generally organized as an assembly of subassemblies to allow for easier final assembly. Subassemblies are then subject to constraints such as size, weight, and stability in order for proper handling during

final assembly. Generally, these subassemblies are partitioned by function which may not be optimal for ease of assembly. Thus, optimal partitionings are important in design for assembly.

In order to allow for the rapid generation of subassembly repartitionings with minimal user interaction, a computer-aided tool was developed. Given an assembly specified by a weighted liaison graph along with the precedence relations generated using current ASA tools, the developed tool is capable of suggesting subassembly partitionings and assembly sequences that best satisfy given criteria. A weighted liaison graph completely describes the parts or subassemblies and how they are connected with weights associated to the connections that give a description of the connection. The precedence relations are necessary in order to maintain geometric feasibility of the generated partitionings and sequences. Here, the only criterion used is based on action counts, which are used to weight the liaison graph, to be discussed in detail below. Since a complex assembly consists of a large number of component parts, it is necessary for the user to input an initial subassembly partitioning. This nominal partitioning usually consists of the original subassembly partitioning with one or two “suspect” subassemblies broken down into component parts such that new repartitioning schemes can be considered. A “suspect” subassembly can be identified by having a large number of internal kinematic degrees of freedom (actions). Hence, starting from a nominal subassembly partitioning, the developed tool can generate suggested partitioning schemes that compare favorably according to the criterion used. Due to the fact that the number of possible assembly sequences may grow with the factorial of the number of parts, the selection of favorable assembly sequences and subassembly partitionings becomes an increasingly more difficult problem to handle as assemblies of increasing complexity are being examined. The developed tool uses a genetic algorithm (GA) using GALib¹ to handle these difficulties.

3.1.1 Genetic Algorithms

¹ GALib is a library of C++ functions for genetic algorithms written by Matthew Wall at MIT and is available free for non-profit use at <http://lancet.mit.edu/ga/>

Genetic algorithms are search and optimization algorithms that use natural selection as a means of optimization. They get their name due to the fact that they are based on the evolutionary model of “survival of the fittest” according to natural genetics and mutation. Genetic algorithms rely on random choices to find optima in a given search space. In this respect, they are similar to simulated annealing. Also, it is because of this search technique that they differ from traditional search and optimization methods and thus can be of use in problems where the search space is too large for traditional search algorithms to operate efficiently.

As outlined by Goldberg [17], there are four major differences between genetic algorithms and traditional search and optimization algorithms.

1. GAs deal with a coding of the parameter set rather than the parameters themselves.
2. GAs explore the search space using a population as opposed to a single point.
3. GAs use an objective function to evaluate a coding instead of auxiliary information.
4. GAs use probabilistic transition rules based on randomized operators as opposed to deterministic rules.

Genetic algorithms are applied by first developing a coding for the parameter set. The possible encodings generated by the coding scheme are referred to “chromosomes” and the number of unique encodings covers the entire search space. Next, starting with an initial population of chromosomes, the randomized operators are applied. The three basic operators used in genetic algorithms are reproduction, crossover, and mutation. The members in the current population are all given fitness scores according to the objective function. Reproduction creates a new population based on probabilities for survival of members of the previous population according to their fitness scores. Pairs of chromosomes are then selected at random to mate according to the crossover operator, usually by swapping portions of their chromosomes to produce two new chromosomes. Finally, mutation occurs at random within the new population. Mutation is usually done by swapping elements within a single chromosome. Then the cycle restarts with the new generation. Using a large enough population and running the GA for a sufficient number of generations are then keys to reaching a set of optimized chromosomes. These operators will be discussed in further detail as applied to the developed genetic algorithm below.

3.1.2 Weighted Liaison Graph

An assembly is represented by a liaison graph. Each node in the liaison graph corresponds to a part or a subassembly of the assembly and each liaison in the graph corresponds to a connection between two parts or subassemblies. A liaison graph can readily be extracted given an assembly drawing. Figure 3-1 shows De Fazio and Whitney's assembly from industry (AFI) represented as a block assembly drawing. Items A through L are subassemblies that are joined during final assembly. Liaison graph information can be displayed in either graphical or tabular form. In tabular form, liaisons are enumerated and for each liaison, the two parts it connects are listed. Figure 3-2 illustrates the liaison diagram associated with the assembly from industry where the nodes represent the subassemblies.

In a weighted liaison graph, the liaisons are each assigned a weight. Here, the weighting used is the action count of the connection. Details on what action counts represent and how they are calculated are given in the next section. To represent the weighted liaison diagram in tabular form, a new field containing the weight for each liaison must be added. Additionally, some liaisons may take on different weight values depending on the current state of assembly due to the nature of action counts. This is taken care of by extending the weighted liaison graph using an additional condition field: if the liaisons listed in the condition field of a particular liaison entry have been previously completed, the weight for that particular liaison is the one listed in that entry; otherwise, it takes the default value given by the entry for that liaison in which the condition field is empty. Table 3-1 lists the weighted liaison graph information in tabular form. The weighted liaison graph gives a representation of an assembly in which information can be quickly conveyed, most importantly the connections between parts or subassemblies and their associated weights. For this reason, the weighted liaison graph is a simple and compact representation of the assembly. One of the limitations of the liaison graph is that although connections between parts are maintained, geometric dimensions and other details are not kept. Note that because of this reason, the tool developed requires that precedence relations acquired through ASA be supplied to maintain assembly constraints due to physical geometric dimensions of parts or subassemblies.

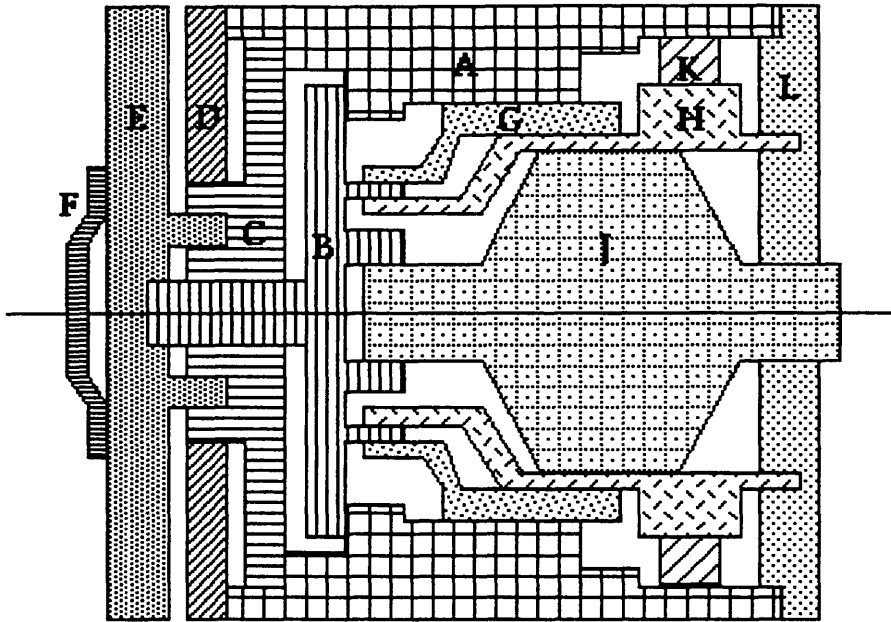


Figure 3-1: De Fazio and Whitney's assembly from industry (AFI)

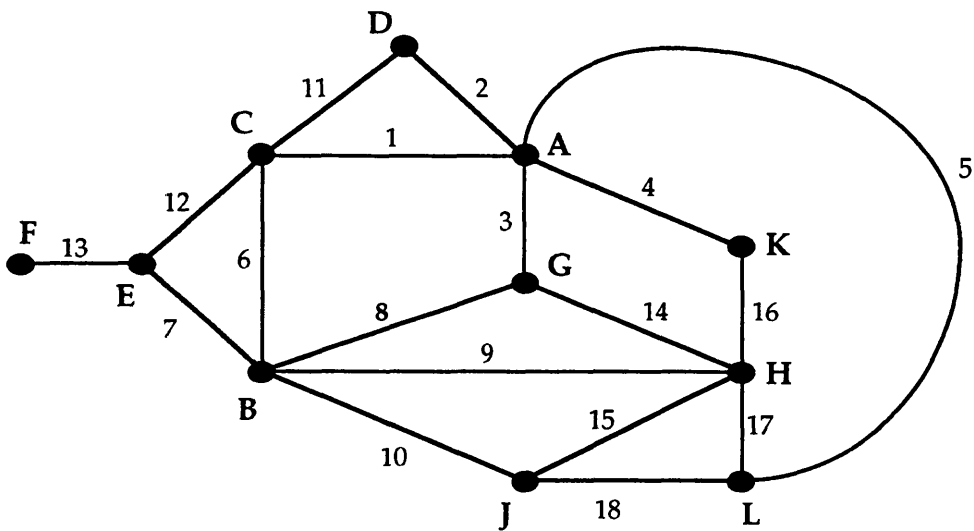


Figure 3-2: Liaison graph for the assembly from industry. Note that the numbers on the arcs merely name them and are not weights.

liaison	part	part	weight= action count	condition liaison	liaison	part	part	weight= action count	condition liaison
1	A	C	0		11	C	D	0	
2	A	D	0		12	C	E	1	
3	A	G	9		13	E	F	1	
3			6	8	14	G	H	4	
4	A	K	8		14			1	15
5	A	L	0		15	H	J	4	
6	B	C	0		15			1	14
7	B	E	0		16	H	K	7	
8	B	G	4		17	H	L	4	
8			1	3	17			1	18
9	B	H	1		18	J	L	4	
10	B	J	1		18			1	17

Table 3-1: Weighted liaison graph information for assembly from industry in tabular form

3.1.3 Action Count

De Fazio [18] defines an “action” of a subassembly as those kinematic degrees of freedom that are in addition to the six kinematic degrees of freedom associated with a solid body in space. These actions exist primarily when a subassembly contains parts that are free to move relative to the rest of the subassembly, e.g. a gear assembly with its freely rotating gear wheels. Then the action count associated with a liaison is the number of actions that must be fixed or closed in order to make that connection between the two parts or subassemblies. Thus, each liaison in the liaison graph is weighted by the number of actions that need be closed in order to establish that liaison. To illustrate, consider a planetary gear system with four planetary gears. Consider a subassembly consisting of the planetary gear carrier and the four planetary gears. In addition to the six spatial kinematic degrees of freedom associated with this subassembly, each of the four planetary gears is free to rotate. Thus, the action count associated with this subassembly is four. In order to mate the ring gear with this subassembly, the rotation of each of the four independent planetary gears must be fixed. Thus, the action count associated with this connection would be four. Typically, most part to part connections have action counts of zero. It is only when we consider subassembly

to subassembly connections where internal components of a subassembly are free to move that we see action counts greater than zero.

There are three exceptional cases that need be mentioned for calculating the action count weighting of a liaison. These are all consistent with the given definition of an “action” of a subassembly. All three cases occur due to the fact that a liaison may have different action count values depending on the current state of the assembly. These can be taken care of by using an extended weighted liaison graph with an additional condition field as mentioned in the previous section.

The first case arises because action count values are assigned to liaisons by determining the number of actions closed when that liaison is completed, taking only that liaison and the two parts it connects into consideration. As particular liaisons are completed, actions of certain subassemblies may be fixed that were taken to be free when determining the action count values of other liaisons. So, it is necessary to look at the current state of assembly when determining action count values. That is, the completion of particular liaisons causes actions of a subassembly involved to be closed so that a liaison completed subsequently involving that subassembly may have a lesser action count than its default count.

For example, consider the aforementioned planetary gear system partitioned into three subassemblies—the sun gear, the planetary gear carrier and the four planetary gears, and the ring gear. As previously stated, in order to establish the liaison between the ring gear and the four planetary gears, all four planetary gears must be aligned to mesh with the ring gear. Similarly, to establish the liaison between the sun gear and the four planetary gears, all four planetary gears must also be aligned to mesh with the sun gear. Therefore, the default action count value associated with both the ring-planet liaison and the sun-planet liaison is four. However, if the planetary gears have already been assembled to the sun gear, then they rotate in unison with the sun gear and no longer need to be individually aligned to mesh with the ring gear. Thus, the conditional action count value associated with the ring-planet liaison is one if the sun-planet liaison has already been established since only the single action of rotating the sun gear is necessary to

align the four planetary gears to mesh with the ring gear. The same situation also holds true when the sun gear is to be added to the assembly where the ring gear and the planetary gears have already been assembled. Thus, once the planetary gears have been assembled to either the ring gear or the sun gear, the resulting assembly has incurred a reduction of actions that need be closed when adding the final gear of the planetary gear system. This can be seen in the planetary gear system of the AFI assembly consisting of subassemblies G, H, and J connected by liaisons 14 and 15 (See Table 3-1 for the actual multiple action count values involved).

The second case is similar to the first, but instead of a reduction in an action count value due to the previous completion of a liaison, it is possible that there be an increase in the action count value of a liaison depending on the state of the current assembly, which can be specified by the liaisons that have been previously completed. That is, instead of closing out actions by the completion of a liaison between two parts or subassemblies, it is possible that by completing a liaison between two parts, new actions are introduced into the resulting assembly.

To illustrate this case, again consider the planetary gear system except each of the four planetary gears and the planetary gear carrier are no longer contained in a subassembly. The mate between a planetary gear alone and the sun gear requires no actions to be closed, but if the planetary gears are first assembled to the planetary gear carrier, then each mate between a planetary gear and the carrier would require one action to be closed. Note that this particular example would never occur in practice due to the fact that the assembly consisting of the sun gear and the four planetary gears would be unstable and is only examined for illustrative purposes.

The third special case occurs due to part or subassembly interference. This occurs when a part must first interface with a second part in order to reach its final position whereas if that second part were not present, the first part could reach its final position without obstruction. This occurred in an earlier design of the AFI assembly where the inner diameter of subassembly K was smaller. In this case, if K were already mated to A, G would first have to mate with K in order to pass through to its final mating location with A. Thus, if the liaison between A and K had been

previously established, the action count value for the liaison between A and G would take on a higher value.

A subclass due to part interference has been identified that is too difficult to model efficiently. It occurs when the placement of a first part interferes with the placement of a subsequent part not by requiring the subsequent part to establish a temporary mate but instead requiring it to take an entirely different mating approach path. Additionally, there are some cases where it is possible for a part to have several approach paths available, each with a different action count value associated with the mate. This ambivalence in the action count value depending on approach direction has been difficult to incorporate into the tool described below due to the fact that approach direction is also directly affected by the assembly orientation which has not been considered thus far. However, this third case is highly uncommon and hardly seen in practical designs.

Table 3-1 gives the extended weighted liaison graph information in tabular form. For liaisons with more than one row with different action count values listed, the entries differ by the conditions listed. These represent liaisons that fall into the aforementioned special cases where it is necessary to have multiple action count values associated with the liaisons. Thus, a listing with an empty condition field gives the default action count value, whereas those listings with liaisons noted in the condition field give the action count value if the “condition” liaisons have been previously completed.

Finally, an action completion count can also be associated with an assembly move. This is done by determining what liaisons are being completed in a particular assembly move and then totaling all the individual action counts for those liaisons. This results in the sum number of actions being completed or closed in an assembly move. It has been shown in assembly practice that completing liaisons 4 and 16 of the AFI assembly simultaneously is an assembly move that should be avoided due to the difficulty of such a move. By using actions counts, we can see that the action completion count for this move is 15. This is a high number relative to the action counts per individual liaison, thus indicating an assembly move we wish to avoid. Having to fix actions

of a subassembly during assembly requires additional effort that make such an assembly move difficult. Having to fix many actions in a single assembly move complicates the completion of the assembly move that much more so. Thus, one criterion for easier assembly can be seen as ensuring that no assembly moves have high action completion counts in an assembly sequence. Further validation of using action counts as a criterion is examined in section 4.1.1 where assembly sequences are generated according to this criterion.

3.1.4 Assembly Sequence and Subassembly Partitioning Encoding

By subassembly repartitioning and selecting particular assembly sequences, we can generate an assembly plan that avoids assembly moves with high action completion counts. For example, if subassemblies A, G, and H were partitioned together as a single subassembly, then it would be impossible to avoid an assembly move in which liaisons 4 and 16 are completed simultaneously, which has been shown previously to close a high number of actions. This would then obviously be a poor choice of subassembly partitioning. Thus, by proper subassembly partitioning, it is possible to avoid having required assembly moves with high action count closures. Due to the fact that subassembly partitioning is a combinatorial problem, a genetic algorithm is used as the search and optimization technique. In order to apply a genetic algorithm to the subassembly partitioning problem, it is necessary to devise an encoding that captures the parameters that are to be optimized, i.e. the subassembly partitioning scheme and assembly sequence.

The problem of subassembly partitioning is in essence similar to the problem of set partitioning or more specifically, graph partitioning. The use of genetic algorithms in solving set partitioning problems has been studied by Jones and Beltramo [19]. In their study, they consider several approaches for solving the set partitioning problem. The difference between subassembly partitioning and set partitioning lies in the fact that subassemblies are required to be composed of connected parts, whereas in a general set, there is no notion of connectivity between set members.

The method used for encoding subassembly partitions as a chromosome is permutation encoding using group separators considered for set partitioning problems by Jones and Beltramo and also by Bhuyan, Raghavan, and Elayavalli [20]. This method encodes a partition as a permutation of n parts and $N - 1$ group separators where n is the number of parts and N is the desired number of subassemblies. For example, given an assembly of 5 parts A, B, C, D, and E, and 3 subassemblies {A C}, {B E} and {D}, this would encode as (1 3 6 2 5 7 4) where a number that is less than or equal to the number of parts references that part (i.e. 1 references A and 2 references B, etc.) and a number such as 6 or 7 that is greater than the number of parts represents a group separator. Thus, an encoding can be read in linearly, placing parts into subassemblies and starting a new subassembly when a group separator is read. If it is desired that the number of final subassemblies be unspecified, i.e. let the algorithm decide the optimal number of final subassemblies, we can set N equal to n and ignore any empty subassemblies that are produced by an encoding, i.e. treat consecutive group separators as a single group separator. This option is referred to as using a variable number of desired subassemblies. This is also the equivalent of allowing for arborescent, or parallel, assembly sequences as opposed to being restricted to purely sequential assembly sequences.

In addition to encoding the subassembly partitioning scheme, the advantage of using the permutation encoding is that it can also encode the assembly sequence for that given subassembly partitioning. Consider the previous example encoding of (1 3 6 2 5 7 4) which decodes to (A C | B E | D) where “|” denotes that a new subassembly begins. This also encodes the associated assembly sequence. First we connect part C to part A completing the first subassembly. Then we connect part E to part B completing the second subassembly. Part D is left alone as it is the only member in its subassembly; thus, the third subassembly is automatically completed. This completes the assembling of the subassemblies. Then in the final pass, the subassemblies are assembled in the order they appear in the encoding. Thus, the subassembly containing B and E is connected to the subassembly containing A and C and then finally the subassembly containing only D is connected to the previous assembly. By using the encoding to also encode the assembly

sequence as well as the subassembly partitioning, we have avoided the problem of redundancy associated with encodings of subassembly partitions alone. For example, the encodings (1 3 6 2 5 7 4), which we considered previously, and (1 3 6 4 7 2 5) both encode the same subassembly partitions {A C}, {B E} and {D}. The difference lies in the assembly sequence they encode. For the second encoding, in the final pass of assembling the subassemblies, the subassembly containing part D alone is connected to the subassembly containing parts A and C prior to connecting the subassembly containing parts B and E as opposed to the assembly sequence encoded by the first encoding.

Due to the nature of the problem of subassembly partitioning, it is possible that invalid encodings are produced. There are two different classes of invalid encodings. The first is one in which an empty subassembly is encoded. This occurs when a group separator appears at the head or tail of the list, or when two group separators appear adjacent to each other in an encoding. If it is desired that the number of subassemblies not be fixed, i.e. variable number of subassemblies, these encodings are not invalid as by having encodings with empty subassemblies that are to be ignored, we can have a variable number of subassemblies with the encoding scheme used. Otherwise, if the desired number of subassemblies has been defined by the user, these encodings are considered invalid and need to be rejected. These can easily be rejected during crossover and mutation of the genetic algorithm to ensure that only valid encodings enter the population. However, we did the rejection in the objective function. More specifically, if invalid encodings were encountered in the population, they were evaluated and given a low objective value, or fitness score. The reason for this is though the encoding may be invalid as a whole, there may be one or several good subassembly partitions encoded in the chromosome. For this reason, we check for invalid encodings in the objective function to allow for penalization instead of rejection. In this way, we can preserve any potentially good subassembly partitions through crossover and mutation in the next generation.

The second class of invalid encodings occurs when subassembly partitions are encoded which are invalid. More specifically, this occurs when a subassembly for which all the parts are

not connected is generated. This is also checked in the objective function to allow for appropriate penalization.

3.1.5 Crossover and Mutation Operators

The permutation encoding chromosome is a list containing no repeated elements to ensure that all parts are included in the subassembly partitioning scheme. For this reason, partial match crossover is used for crossover to ensure that chromosomes generated by crossover contain no repeated elements. In reproduction using simple crossover, a portion from one parent chromosome is swapped with a portion from a second parent chromosome to create two offspring chromosomes. For example given the parent chromosomes (1 3 6 2 5 7 4) and (4 6 1 7 3 5 2), applying crossover to the second through fourth elements, the resulting offspring chromosomes would be (1 3 1 7 3 7 4) and (4 6 6 2 5 5 2). The middle three elements, the second through fourth elements, have just been swapped between the two parent chromosomes to create the offspring chromosomes. This results in invalid permutation encodings, i.e. chromosomes with repeated elements. Partial match crossover avoids this problem by using positionwise exchanges of elements. For example, in the first chromosome (1 3 6 2 5 7 4), the middle three elements, 6 2 5, are to be replaced by the middle three elements of the second chromosome, 1 7 3. Since the 6 in the first chromosome is to be replaced by a 1, the original 1 in the first chromosome becomes a 6. Likewise, the original 7 becomes a 2 and 3 becomes a 5. This results in the following two offspring chromosomes: (6 5 1 7 3 2 4) and (4 1 6 2 5 3 7). Thus, only valid permutations are created by partial match crossover.

Likewise, it is necessary to use a simple swap mutator for mutation. The simple swap mutator operates on a permutation encoding chromosome by randomly swapping two elements within the chromosome. For example, the chromosome (1 3 6 2 5 7 4) may mutate to become (1 4 6 2 5 7 3). The second and last element of the chromosome have swapped places. As mentioned previously, this may result in an invalid encoding, but since it may contain good subassembly

partitionings, it is allowed and later penalized in the objective function. The alternative would be to reject this chromosome and apply the mutation operator again until a valid encoding is produced.

3.1.6 Objective Function

Currently, the objective function relies on a simple heuristic based on the idea of action counts. Since action counts are associated with liaisons and in order to complete the assembly, all liaisons must be connected, it can be seen that the action completion count over the entire assembly is constant, i.e. the sum of the action counts for all liaisons. The goal then is to have an assembly partitioned for which the associated assembly sequence avoids assembly moves with high action completion counts. Due to the fact that the total action completion count for the entire assembly is constant and that the number of assembly moves is constant (equal to the number of parts minus one), assembly moves with high action completion counts can be minimized or avoided by spreading the total action completion count over the entire assembly sequence. This is done by use of the root-mean-square (rms) function. For each assembly move, the action count value is squared. These values are then summed over the entire assembly sequence and divided by the total number of assembly moves. Then the root of this mean is taken as the objective score. This represents the rms average action count per assembly move. By minimizing this value, the actions are being spread evenly across the entire assembly. Thus, each encoding is assigned a score according to this objective function. Additionally, invalid partitioning encodings that fall in the two aforementioned classes are then penalized by increasing their objective score. The genetic algorithm then tries to find those partitioning schemes with the best, i.e. lowest, scores.

Finally simple sharing is implemented using a simple distance function in order to generate a optimal set of subassembly partitioning schemes rather than converging on a single optimum. This function looks at the assembly sequences encoded by two different chromosomes and returns a value based on similarity of the encoded sequences. Similar chromosomes share their final objective scores by dividing the objective score over the number of similar chromosomes to assure

that the GA will not tend to a single chromosome as a solution, but favor a variety of chromosomes as an optimal solution set.

3.2 DFA Through Precedence Relation Elimination

De Fazio [21] describes another approach for applying DFA to complex assemblies. The idea is based on the belief that by eliminating assembly constraints, the number of feasible assembly sequences will increase and perhaps allow for a better assembly sequence that was previously not available with the initial set of assembly constraints. Precedence relations (PR's) derived using ASA express assembly constraints that are due to the geometric dimensions and placements of parts or subassemblies in an assembly. Thus, by eliminating PR's by either redesign or repartitioning, better assembly sequence options may become available as assembly constraints are removed.

The precedence relations derived using ASA usually take on the following form:

$$L_i \& L_{i+1} \& \dots \& L_j \geq L_m \& L_{m+1} \& \dots \& L_n$$

This is read as the set of liaisons $\{L_i, L_{i+1}, \dots, L_j\}$ must all be completed before, or concurrently with, the completion of the set of liaisons $\{L_m, L_{m+1}, \dots, L_n\}$.

By examining the liaisons involved and the parts or subassemblies they connect, most precedence relations can be traced back to geometric constraints due to part dimensions in the design. Then, by inspection, it is generally possible to either partially or completely eliminate the precedence relation using either detail redesign or subassembly repartitioning. Since precedence relations arise due to geometric constraints between parts or subassemblies, by examining the assembly and determining what geometric constraint due to the dimensions of parts or subassemblies exists causing the need for a particular precedence relation, either redesign or repartitioning can be used to remove the geometric constraint. By eliminating a PR completely or by softening a PR, i.e. eliminate elements of a PR weakening its constraint on the assembly

sequences, sequences previously eliminated by the PR will now be feasible. After having examined all the precedence relations, the designer creates a new weighted liaison graph, and ASA is again used to generate a new set of precedence relations that allow for more assembly freedom. Then by using the new liaison graph and the accompanying set of precedence relations, the GA technique described in the previous section can be used to find favorable subassembly partitions or assembly sequences.

3.3 Summary

Having developed the required tools and techniques, the next step is apply them to actual industry assemblies for validation. The results are presented in the next chapter.

CHAPTER 4

Case Studies of Actual Complex Assemblies

In this chapter, the tools and techniques described in Chapter 3 are applied to two real assemblies from industry, the AFI assembly and the BAT assembly. For each assembly we first ran some experiments on the usefulness of the GA using the action count closure criterion by generating only assembly sequences using the actual subassembly partitioning scheme. Next the effectiveness of the GA for subassembly repartitioning was examined by trying various nominal subassembly partitionings. Finally, the technique of precedence relation elimination to suggest redesign was applied to the BAT assembly and its effectiveness was examined.

4.1 AFI Assembly

The AFI assembly is a 6 speed automatic transmission for trucks and buses. This assembly exhibits all the characteristics of a complex assembly. It is an assembly of mechanically complex and densely packed parts, has a high parts count with the assembly organized as an assembly of subassemblies, has little design freedom and due to the nature of its function, and has a larger number of internal kinematic degrees of freedom, or actions. Figure 4-1 shows a detailed drawing of the entire assembly, and Figure 4-2 shows the assembly decomposed according to the original subassembly partitioning scheme. The liaison graph was given earlier in graphical form by Figure 3-2 and the weighted liaison graph was described explicitly in tabular form in Table 3-1.

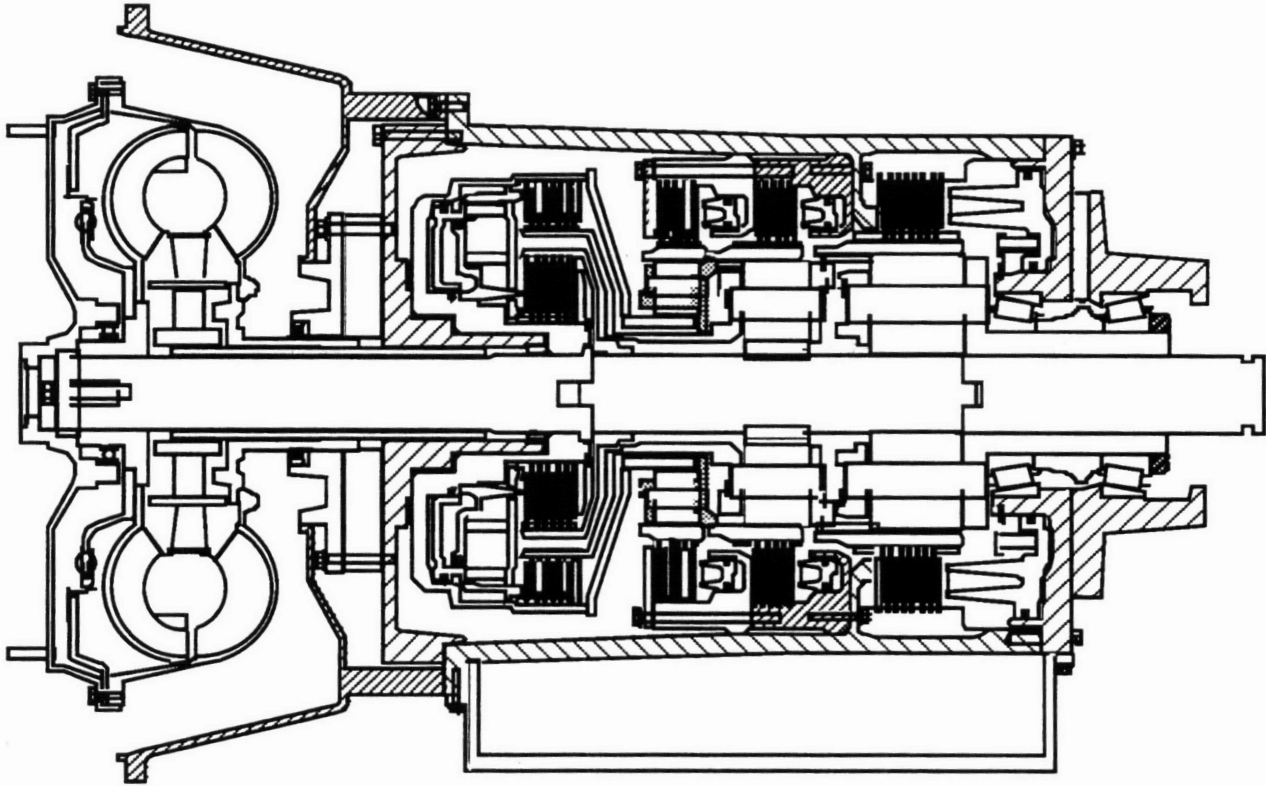


Figure 4-1: Detailed drawing of the AFI assembly

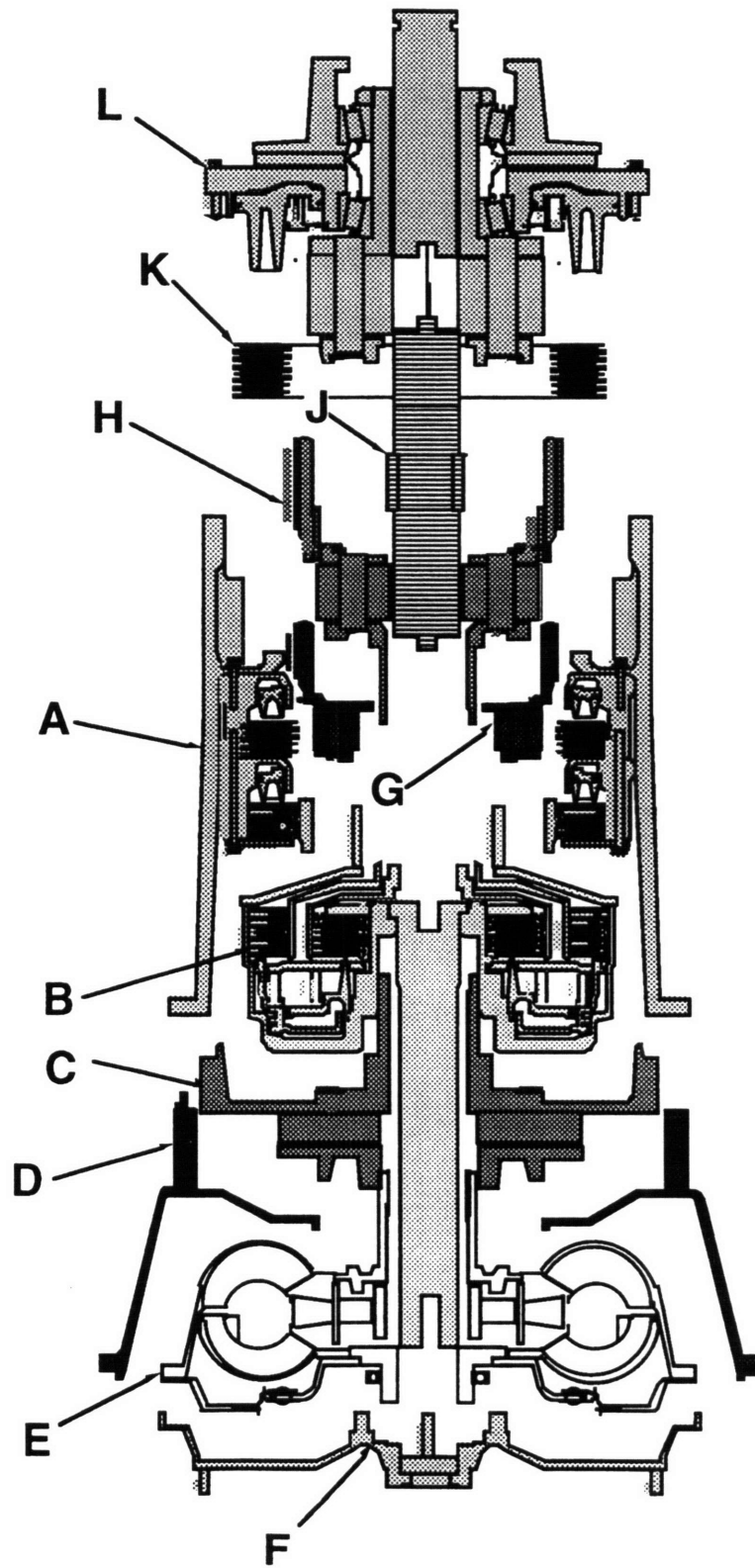


Figure 4-2: AFI with original subassembly partitioning

4.1.1 GA Sequence Generation

The first experiment run uses the GA tool developed to generate optimal assembly sequences based on the action count closure criterion. The goal is to determine the usefulness of the tool developed and the action count criterion used. Using ASA, the precedence relations are derived given the liaison graph and answering the geometric interference questions. Then, with the weighted liaison graph information and the set of precedence relations, the GA tool is used to generate only assembly sequences and not subassembly partitions using the original subassembly partitioning scheme. This is done by setting the desired number of repartitioned subassemblies to one; thus, the original partitioned subassemblies are to be assembled into one final assembly. Note that this could have also been done by setting the desired number of repartitioned subassemblies equal to the number of original subassemblies, but these are identical problems except for the fact that the latter is far more computationally expensive due to the longer encoding required. Additionally, the existence of multiple subassemblies during assembly was not allowed. The unique favorable assembly sequences generated when the GA was run for 10000 generations using a population size of 50 obtaining an objective score of 6.16443 are as follows:

```
A K G H B J C D E L F
G A K H B J C D E F L
G A K H B C J D E L F
A K G H B C J D E L F
G A K H B C J D E F L
K A G H B J C D E F L
K A G H B C J D E L F
```

Recall from chapter 3 that the objective score represents the root-mean-square average action count completion per assembly move. The sequences generally begin by building the AFI assembly starting with the casing (subassembly A), and then adding the clutch stack (subassembly K) and the planetary gear systems (subassemblies G and H). Then the rest of the assembly is completed. Note that the suggested assembly sequences are not practical as they all require multiple

reorientations due to the fact that subassembly J is generally assembled after subassembly B, followed by subassemblies D and E, and later subassembly L. This is due to the fact that only an action count based criterion was used. However, by inspecting the suggested sequences and by the fact that subassembly L does not mate with subassemblies C, D, E, or F, we may arrange one of the suggested sequences and arrive at the following sequence that will have the same objective score:

A K G H B J L C D E F

This assembly sequence works well in that starting with subassembly A, the assembly is built mostly on one end (subassemblies K, G, H, J, and L) except for subassembly B, and then on the other end (subassemblies C, D, E, and F). Hence, this experiment shows that action count does show a certain utility in determining favorable sequences, but with more criteria such as reorientation information, the genetic algorithm may be even more successful in suggesting favorable assembly sequence choices. Also, even though the GA may not always suggest all favorable sequences due to the size of the search space of the problem, the general trends in the sequences that were suggested are often useful and may lead to a practical sequence choice as shown in this experiment.

4.1.2 GA Subassembly Partitioning of AFI-H

Next, we wish to study the effectiveness of the GA tool developed based on the action count closure criterion as a means of subassembly partitioning. AFI-H represents the original AFI assembly with subassembly H broken down into two separate subassemblies, Ha and Hb, as shown in Figure 4-3. By taking this AFI-H assembly and using the genetic algorithm to repartition the 12 subassemblies into the original number of 11 subassemblies, the effectiveness of the genetic algorithm program can be studied. Here we wish to see if the broken down subassemblies of H are repartitioned together by the GA or perhaps a better repartitioning is offered. Figure 4-4 shows

the liaison diagram for the AFI-H assembly and Table 4-1 gives the weighted liaison graph information in tabular form.



Figure 4-3: (a) original subassembly H; (b) broken down into subassemblies Ha and Hb

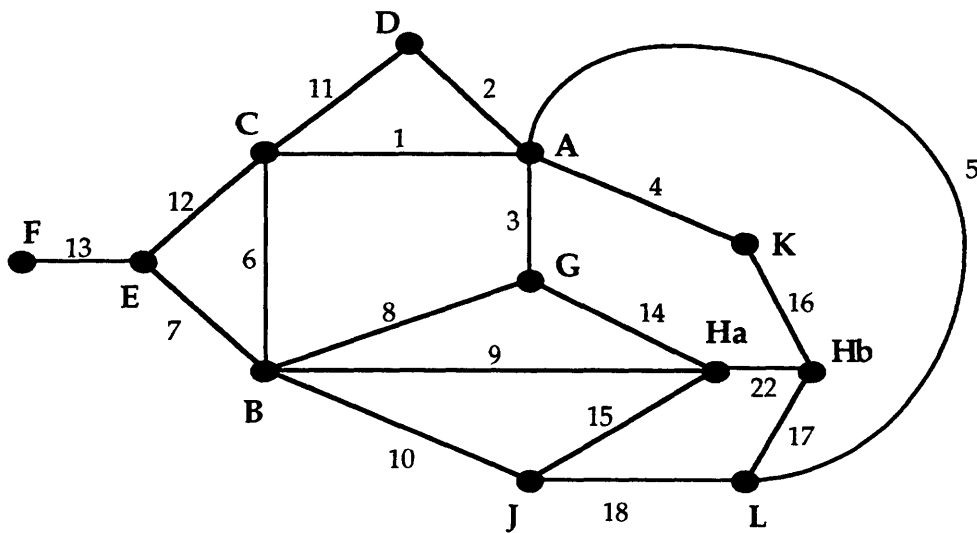


Figure 4-4: Liaison graph for AFI-H

liaison	part	part	weight= action count	condition liaison	liaison	part	part	weight= action count	condition liaison
1	A	C	0		12	C	E	1	

2	A	D	0		13	E	F	1	
3	A	G	9		14	G	Ha	4	
3			6	8	14			1	15
4	A	K	8		15	Ha	J	4	
5	A	L	0		15			1	14
6	B	C	0		16	Hb	K	7	
7	B	E	0		17	Hb	L	4	
8	B	G	4		17			1	18
8			1	3	18	J	L	4	
9	B	Ha	1		18			1	17
10	B	J	1		22	Ha	Hb	1	
11	C	D	0						

Table 4-1: Weighted liaison graph information for AFI-H in tabular form

Using the weighted liaison graph information along with the precedence relations derived with ASA, the genetic algorithm was run for 5000 generations using a population size of 100. Here, we assumed sequential assembly only in the final assembly of the 11 desired subassemblies. The resulting unique sequences with an objective score of 4.41071 suggested by the GA are as follows:

B - C - A - G - K - D - Ha - E - F - Hb - L J
 B - C - A - G - D - Ha - K - E - F - Hb - L J
 B - C - A - G - K - D - E - Hb - F - Ha - J L
 B - C - A - G - D - K - Hb - E - F - Ha - L J
 C - B - A - G - D - K - E - Ha - F - Hb - L J
 C - B - A - G - K - D - Hb - E - F - Ha - J L
 B - C - A - G - D - K - Ha - E - F - Hb - L J
 C - B - A - G - D - E - K - F - Ha - Hb - J L

An assembly sequence/subassembly partitioning is read by taking a '-' mark to indicate divisions between subassemblies, and the listed order is read to denote the assembly sequence where subassemblies within '-' marks are assembled first before final assembly. Thus, an encoding is read by taking a first pass through the encoding to assemble any subassemblies where the assembly sequence of a subassembly is given by the listed order of its components without a '-'. Next, a second pass is taken which gives the assembly sequence for the final assembly. For

example, the first listing suggests that L and J be assembled first as a single subassembly. Then, starting with subassembly B, subassemblies C, A, G, K, D, Ha, E, F, and Hb are assembled in that order. Finally, since the subassembly containing L and J is listed last, it is the final subassembly to be added to the final assembly.

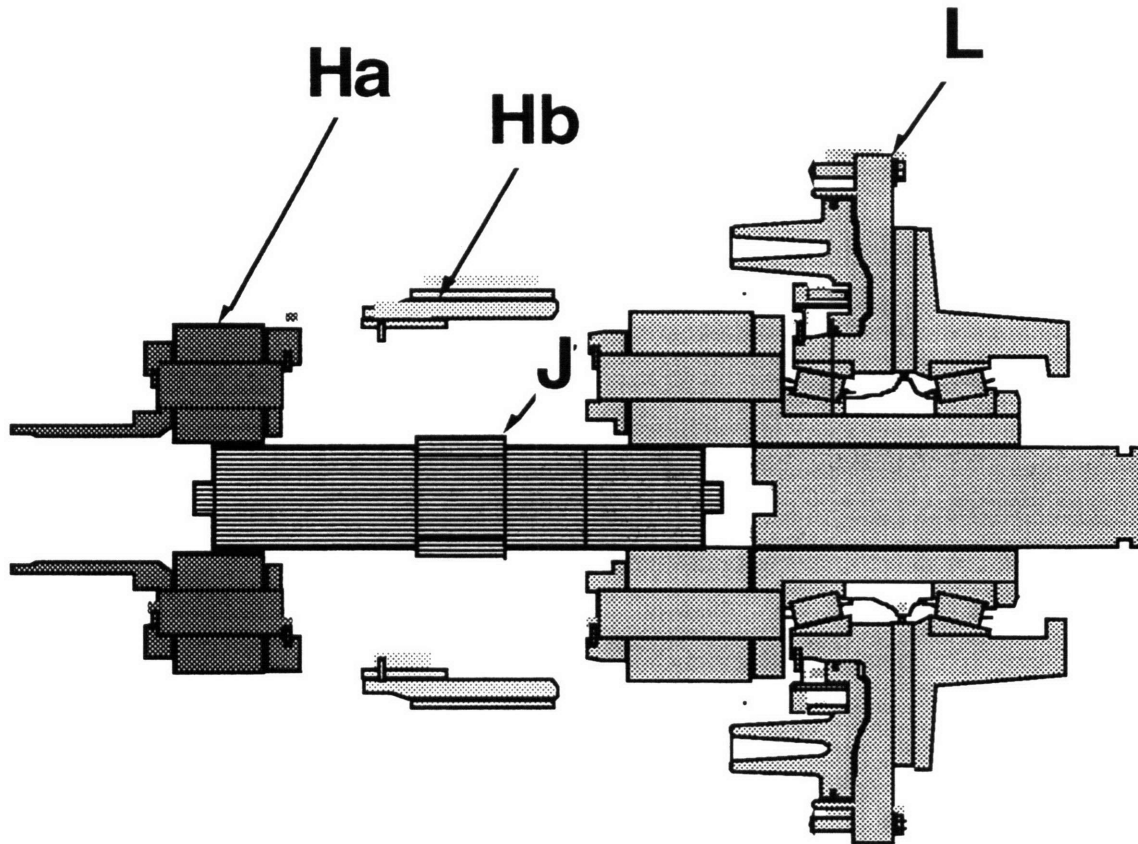


Figure 4-5: Subassemblies Ha, Hb, J, and L

Figure 4-5 shows the subassemblies Ha, Hb, J, and L in detail. What is of interest in the suggested partitionings is that Ha and Hb remain as separate subassemblies, whereas L and J have been partitioned together. This leads to the hypothesis that the action count closure criterion used gives L and J a stronger need to be combined as a single subassembly than Ha and Hb. This can be explained by the fact that subassembly L contains the planetary gears that mesh with the sun

gear contained in subassembly J and the ring gear contained in subassembly Hb. This means that if the liaison between L and J is established first, the associated action count value for the liaison is 4 and the action count value for the liaison between L and Hb is only 1 due to the fact that the planetary gears of L have been previously fixed..

Conversely, if the liaison between L and Hb is established first, that liaison will have an action count value of 4 and the liaison between L and J will have an action count value of 1. To complicate matters, subassembly J also contains the sun gear that connects to the planetary gears contained in subassembly Ha. This results in the fact that subassembly Ha has 4 actions that need be closed when mated to subassembly J unless the ring gear in subassembly G is first assembled to Ha. By fixing J and L in a subassembly together, they are ensured to be assembled together before being assembled to any other parts in the final assembly. This will set the action count value for the liaison between J and L to 4 and the action count value for the liaison between L and Hb to 1. This is desired because otherwise there might arise a situation where J must be assembled to Ha and L simultaneously which would require that 8 actions be closed in one assembly move or L might be assembled to J and Hb simultaneously which would similarly result in the need for 8 actions to be closed in a single assembly move. Thus, by grouping J and L together in a single subassembly, this possibility can no longer occur.

This suggests that due to the nature of the action counts contained in the liaisons surrounding subassemblies H, J, and L, there may be some merit in considering partitioning the subassemblies such as to limit the possibilities of high action count closures in single assembly moves as demonstrated here. Due to the single criterion used, a new subassembly consisting of subassemblies L and J may not be practical due to the fact that the new subassembly may not necessarily be stable and that by combining L and J, the mate between J and Ha may be difficult to complete due to limited access caused by the size of L.

4.1.3 GA Subassembly Partitioning of AFI-B

AFI-B represents the original AFI assembly with subassembly B broken down into subassemblies Ba, Bb, Bc, and Bd. See Figure 4-6 for the breakdown of subassembly B. By taking this assembly and using the genetic algorithm to repartition the 14 resulting subassemblies into the original number of 11 subassemblies, the effectiveness of the genetic algorithm program can be further studied. Subassembly B has a high number of actions and is thus worth examining for repartitioning. To be more precise, we wish to see if the broken down subassemblies of subassembly B (Ba, Bb, Bc, and Bd) are repartitioned together to imitate the original subassembly partitioning, or if not, to examine how the suggested subassembly repartitioning compares with the original partitioning. Figure 4-7 shows the liaison graph and Table 4-2 gives the weighted liaison graph information in tabular form.

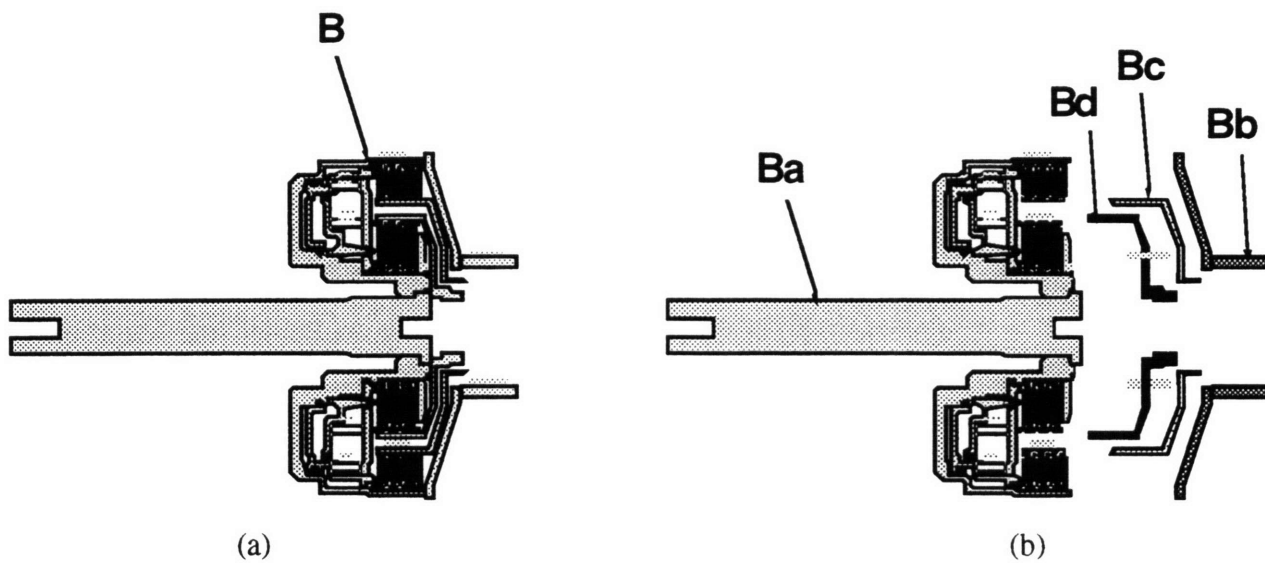


Figure 4-6: (a) Subassembly B and (b) broken down into subassemblies Ba, Bb, Bc, and Bd

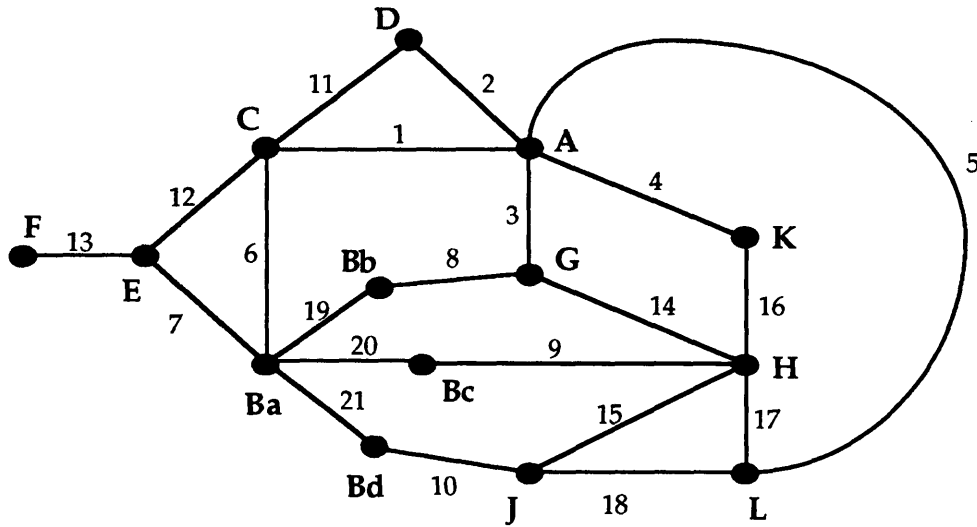


Figure 4-7: Liaison graph for AFI-B

liaison	part	part	weight= action count	condition liaison	liaison	part	part	weight= action count	condition liaison
1	A	C	0		13	E	F	1	
2	A	D	0		14	G	H	4	
3	A	G	9		14	H	J	1	15
3	A	K	6	8	15	H	L	4	
4	A	L	8		15	H	K	1	14
5	A	L	0		16	H	L	7	
6	Ba	C	0		17	H	L	4	
7	Ba	E	0		17	J	L	1	18
8	Bb	G	4		18	J	L	4	
8	Bb	G	1	3	18	J	L	1	17
9	Bc	H	1		19	Ba	Bb	0	
10	Bd	J	1		20	Ba	Bc	6	
11	C	D	0		21	Ba	Bd	5	
12	C	E	1						

Table 4-2: Weighted liaison graph information for AFI-B in tabular form

Using the weighted liaison graph information and the new precedence relations generated using ASA, the unique subassembly partitioning schemes generated with a minimum action count score of 5.18872 are as follows:

Bd - Ba - Bc - Bb - A K G - C - D - E - F - H J - L
 Bd - Ba - Bc - Bb - A G K - C - D - E - F - H J - L

Again, multiple subassemblies were not allowed during final assembly. The genetic algorithm was run for 5000 generations using a population size of 100.

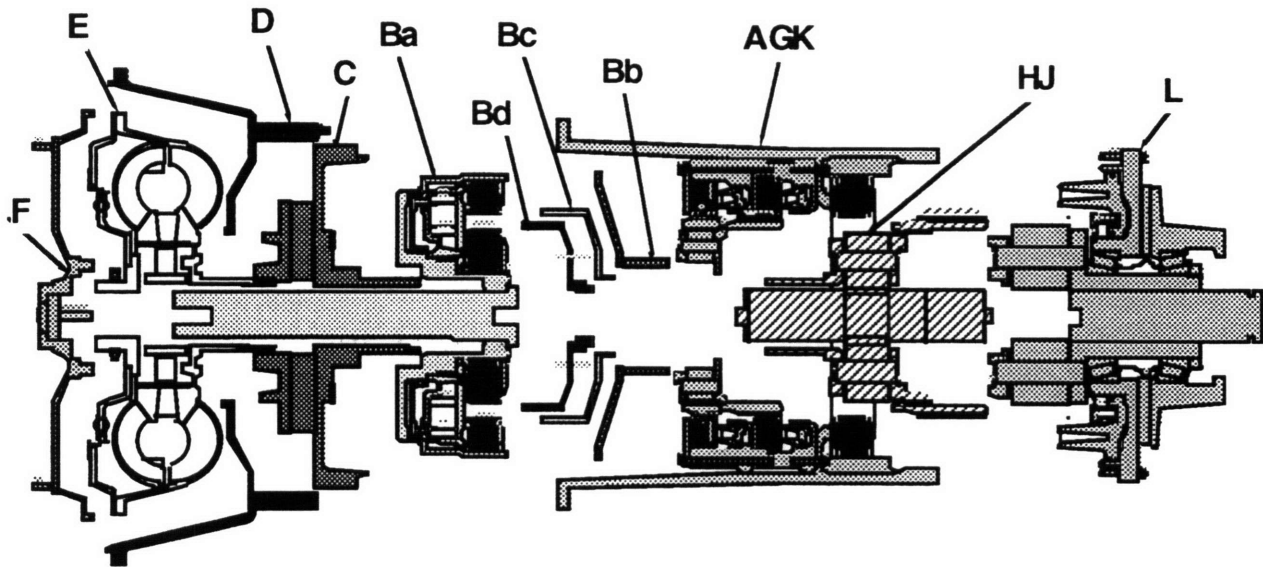


Figure 4-8: Subassembly repartitioning of AFI-B

The results indicate that A, K, and G are best partitioned into a single subassembly as are H and J as shown in Figure 4-8. By examining the physical assembly taking into account the action count criterion used, the reasons for such a repartitioning can be seen. The difficulties in the AFI assembly are due to two types of physical systems in the assembly. The first is the planetary gear system for reasons as described in detail in Chapter 3. The second type of assembly difficulty can be attributed to those subassemblies containing a set of clutch plates, such as subassembly K. In order to assemble this system of clutch plates to its mate, each of the clutch plates must be aligned in turn. Furthermore, the clutch stack consists of some plates that are to mate with an inner ring and others that are to mate with an outer ring. Thus to assemble the clutch stack to both the outer

and inner mates simultaneously, i.e. assembling K to both A and H simultaneously, would result in a horrendously high number of actions to be closed and is therefore deemed a difficult assembly move. It is for these reasons that the GA program suggests that A, K, and G be repartitioned as a single subassembly and that H and J also be repartitioned as a single subassembly. By partitioning K with A, there no longer exists the possibility that clutch stack K might be assembled to both A and H simultaneously, but instead it is required that K must be assembled to A first within a separate subassembly before being assembled as part of the final assembly. Also, by partitioning G as part of a subassembly consisting of A, K, and G, there no longer exists the possibility that the ring gear in G might need to be assembled to both the clutch stack in subassembly A and the set of planetary gears in subassembly H. Instead, G must first be mated with A to complete the suggested subassembly repartitioning before it can be mated to H. For these same reasons, it is suggested that H and J be repartitioned into a single subassembly. By doing so, there is no longer the possibility that the ring gear in H might need be mated to the clutch stack K at the same time as the set of planetary gears in H be mated to the sun gear in J and to the ring gear in G. By repartitioning H and J as a single subassembly, it is required that H be assembled with J to complete a subassembly before H can be mated with either G or K.

These results suggest that as before with AFI-H, due to the nature of the actions contained in the liaisons surrounding subassemblies A, G, K, H, and J, there may be some merit in considering repartitioning these subassemblies such as to limit the possibilities of high action count closures in single assembly moves as illustrated above. Again, since only the action count criterion was used, the suggested subassemblies may not fare well when other criteria are considered, particularly subassembly stability and access to mates between subassemblies.

4.1.4 GA Subassembly Partitioning of AFI with No Decomposition

As a final application of the genetic algorithm for subassembly repartitioning of the AFI assembly, we take the AFI assembly with its original subassembly partitions. It has been suggested that based on the action count criterion, sequential assembly would be favored over

parallel assembly. That is, given the freedom to determine the final number of subassembly partitions, the GA tool would partition the given parts or subassemblies separately, i.e. the number of final subassembly partitions would be equal to the number of input parts or subassemblies, according to the action count criterion. Thus, to test this we let the algorithm determine the final number of subassemblies to see what partitionings are suggested. This was done by setting the number of final subassembly partitions equal to the number of parts and allowing encodings with empty subassemblies by just ignoring the empty subassemblies as described in Chapter 3. Thus, if empty subassembly partitionings are generated, then the actual final number of subassemblies is less than the number of input parts or subassemblies, indicating that sequential assembly does not always rate better by the action count criterion. The resulting subassembly partitionings are:

```

D - B C A - K - - - G - - E - F - L J H
- - B C A D K - - - G - - E - F - J H L
A - B C - D - K - G - - - E - F - J H L
D - C B A K - - G - - - - E - F - H J L
E - C B A K D - G - - - - F - - H J - - L

```

Here, consecutive dashes are read as a single subassembly divider as described in detail previously. The genetic algorithm suggested 4 unique partitionings that resulted in lower final objective scores than when a sequential assembly was used (See section 4.1.1). Common to the suggested partitionings are that H and J are always partitioned together as are B and C as shown in Figure 4-9. In some cases L is also partitioned with H and J, and A is sometimes partitioned with B and C as are K and D, but less frequently. This would imply that the use of subassemblies (or branched assembly lines) can rate better by the action count criterion than the absence of subassemblies (or a sequential assembly line). A reason behind this may be that several liaisons take on multiple action count values depending on the current state of assembly.

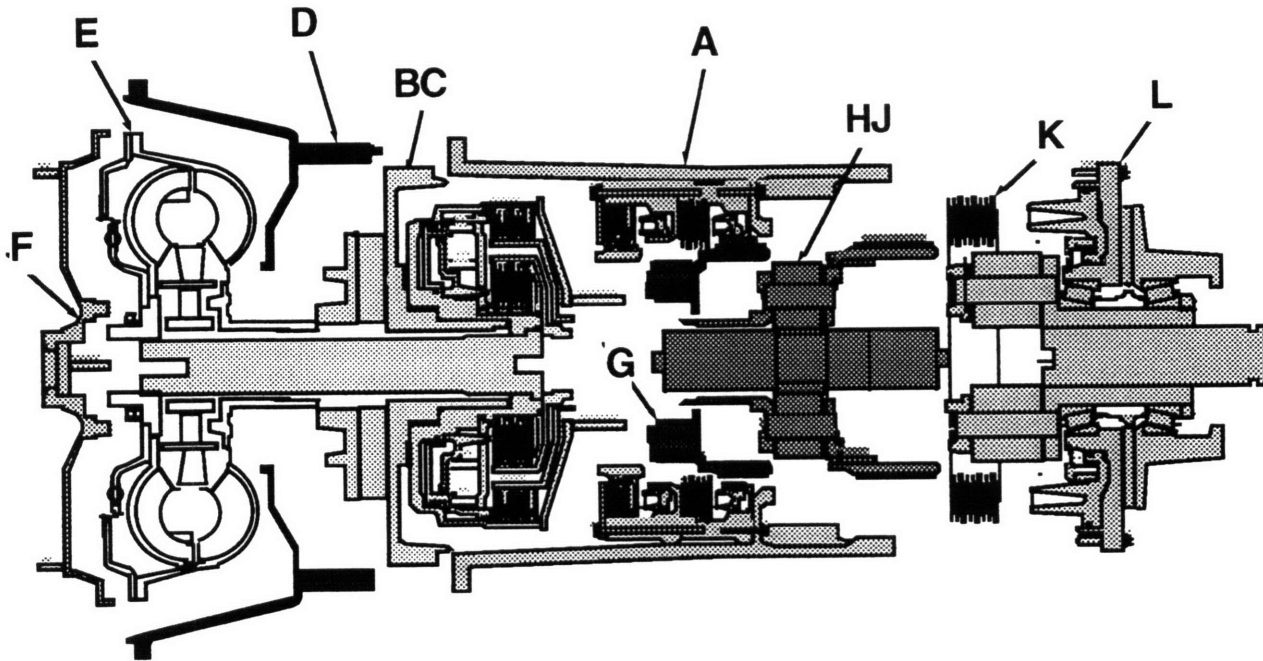


Figure 4-9: Suggested subassembly partitionings of AFI with no subassemblies broken down

Recall that in breaking down subassembly H into Ha and Hb (section 4.1.2), the suggested subassembly repartitionings included L and J partitioned as a single subassembly. Also, in breaking down subassembly B into Ba, Bb, Bc, and Bd, suggested subassembly repartitionings included H and J partitioned together as a single subassembly and A, K, and G partitioned together as a single subassembly. Note that in these two cases, the final number of subassemblies was set to 11 to match the original number of subassemblies. Although the previous results show some correspondence with results in this section, there is some variation which can easily be explained by the limits defined for each experiment run. Due to the fact that the previous two required that there be exactly 11 final subassemblies, it was not possible to have a repartitioning that included H and J together in AFI-H and one where B and C were together in AFI-B because in both cases, this would result in less than 11 final subassemblies. It is possible then to conclude that were the final number of subassemblies into which to repartition not fixed in the previous two examples, AFI-H and AFI-B, then subassembly repartitionings including H, J, and L together and B and C together

would most likely be suggested by the GA program. Due to current hardware and software limitations, such runs with a large assembly having B or H broken down and given the freedom to determine the final number of subassemblies were unable to offer any real results considering the resulting size of the search space.

Again, some of the suggested subassemblies may be impractical when other criteria are taken into consideration. For example, a subassembly consisting of H, J, and L may be too large to handle and when mating with the rest of the assembly, limited access to the mating areas may lead to difficult assembly. Also, some of the assembly sequences generated are impractical due to the number of reorientations required. However, by examining the suggested subassembly partitionings and assembly sequences, it is possible to generate the following sequence/partitioning by observation based on the results of the GA tool with minor modifications for simple consideration of other necessary criteria:

D - B C - A - K - G - J H - L - E - F

This sequence requires only one reorientation at the end to install subassemblies E and F. The partitioning are based on the suggested partitionings generated by the GA tool using the action count criterion. Thus, results from the GA tool can be useful in subassembly partitioning and assembly sequence generation although other criteria may need to be considered.

4.2 BAT Assembly

The BAT assembly is a car transaxle. In many ways, the BAT assembly is similar to the AFI assembly. They both display all the characteristics of complex assemblies. The BAT assembly differs in that it is not completely axially symmetric as is the AFI assembly and therefore may serve as a good contrast to the AFI assembly. Figure 4-10 shows a detailed drawing of the BAT assembly, and Figure 4-11 shows the BAT assembly with its original subassembly

partitioning. The liaison graph is given in graphical form in Figure 4-12 and in tabular form in Table 4-3.

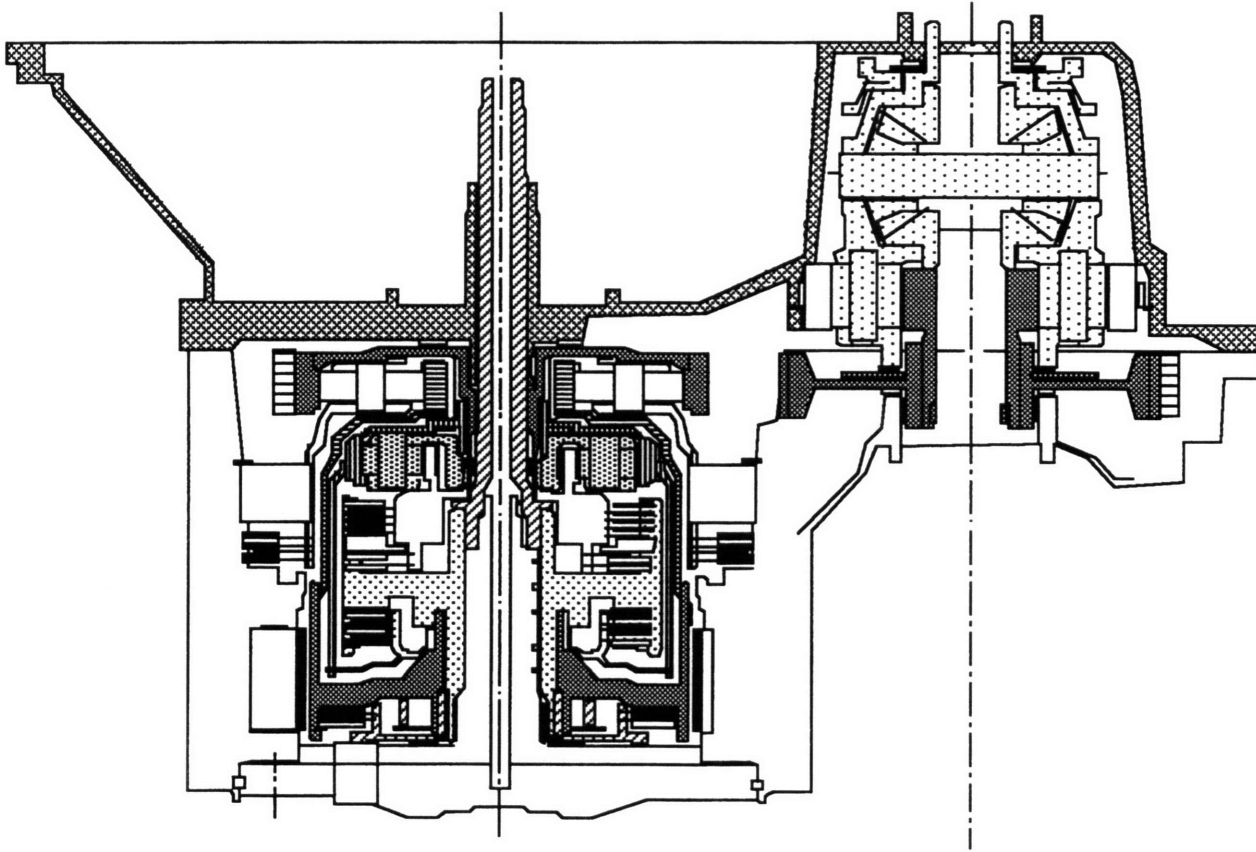


Figure 4-10: Detailed drawing of the BAT assembly

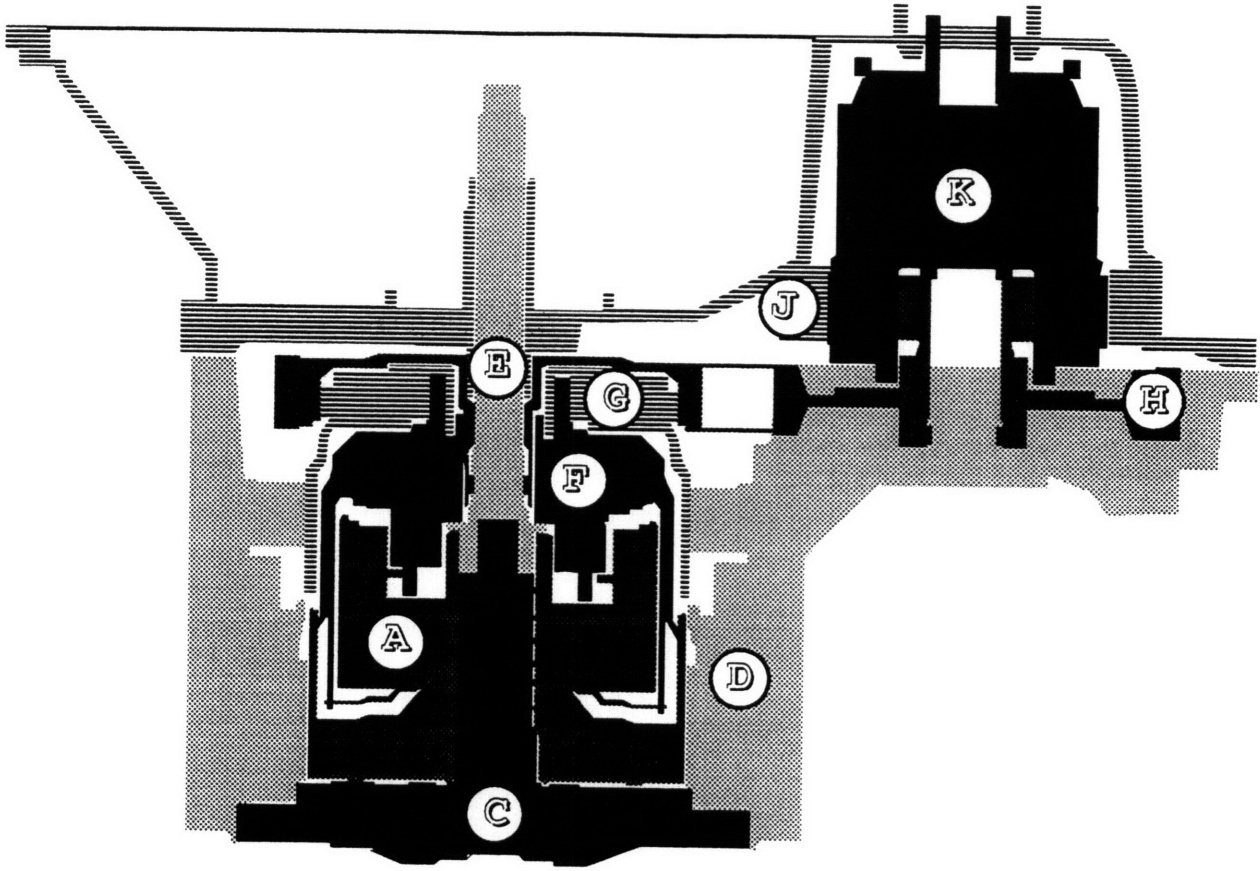


Figure 4-11: BAT assembly with original subassembly partitioning

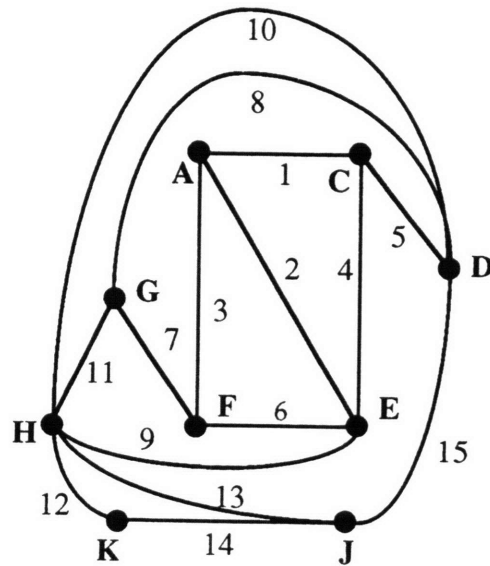


Figure 4-12: Liaison graph for BAT assembly

liaison	part	part	weight= action count	condition liaison	liaison	part	part	weight= action count	condition liaison
1	A	C	0		10	D	H	0	
2	A	E	0		11	G	H	3	
3	A	F	7		11			0	7
4	C	E	0		11			4	8
5	C	D	0		11			1	7, 8
6	E	F	0		12	H	K	3	
7	F	G	4		12			1	14
7			1	11	13	H	J	0	
8	D	G	4		14	J	K	3	
8			5	11	14			1	12
9	E	H	0		15	D	J	0	

Table 4-3: Weighted liaison graph information for BAT in tabular form

4.2.1 GA Sequence Generation and Subassembly Partitioning of BAT

First, we ran the GA only generating assembly sequences for the BAT assembly using the original partitioning as we did for the AFI assembly. The goal here is to see how well the action count closure criterion works for the BAT assembly. Using the liaison graph information and the precedence relations generated using ASA, the genetic algorithm was run for 1000 generations with a population size of 100 with only sequential assembly allowed. The following assembly sequences with an objective function score of 3.39117 were generated:

DGFHEAKCJ	DGFHEAKJC
GFDHEAKCJ	GFEADHCKJ
EAFGDHCKJ	GDFHEACKJ
GDFEAKCJ	GFDHEACKJ
FEAGDHCKJ	GFDEAHCKJ
A EFGDHCKJ	GFEADHCKJ
DGHFEAKJC	EAFGDCHKJ
A EFGDHCKJ	FEAGDCHKJ
FGDHEACKJ	A EFGDCHKJ
GFEADHCKJ	EAFGDHCKJ
FEAGDHCKJ	EFGADHCKJ
EAFGDHCKJ	FGDHEACKJ

As with the AFI assembly, most of the sequences generated are impractical due to the obvious need for multiple reorientations to accommodate most of the suggested sequences. This can be seen by the fact that both subassemblies C and J fall close to the end of most of the sequences yet they are physically located at opposite ends of the assembly requiring that the assembly most likely be reoriented several times if the suggested assembly sequences were to be used. This is not to say that the action count closure criterion is not a good criterion but may be inadequate by itself.

To avoid multiple reorientations, the GA was run again to find favorable sequences but with subassembly C fixed as the first subassembly to go into the final assembly. Two runs were done, one for 1000 generations and one for 500 generations both using a population size of 50. The results are as follows with an objective function score of 3.93701:

```
CADEFGHKJ
CDAEFGHKJ
CAEFDGHKJ
```

The resulting sequences are more practical than the results initially obtained, but due to the constraint that the assembly sequences must start with subassembly C, the final objective score obtained was not as low as for the first results. This suggests that choosing optimal sequences using only the action count criterion is not sufficient, but by taking other criteria into consideration such as requiring that the assembly start with subassembly C, we may then use the action count criterion for selecting optimal sequences as was done here.

Finally, one last GA run was done where the number of subassemblies was allowed to vary but the sequences were still constrained to start with C. This resulted in only one optimal sequence/partitioning:

```
--CAEF----DGH-KJ-
```

This sequence resulted in a much lower score of 3.10242 than the previous two examples, indicating that according to the action count criterion, there is some merit to partitioning C, A, E, and F together as a single subassembly, D, G, and H together, and finally K and J together as opposed to the original partitioning of 9 separate subassemblies as shown in Figure 4-13.

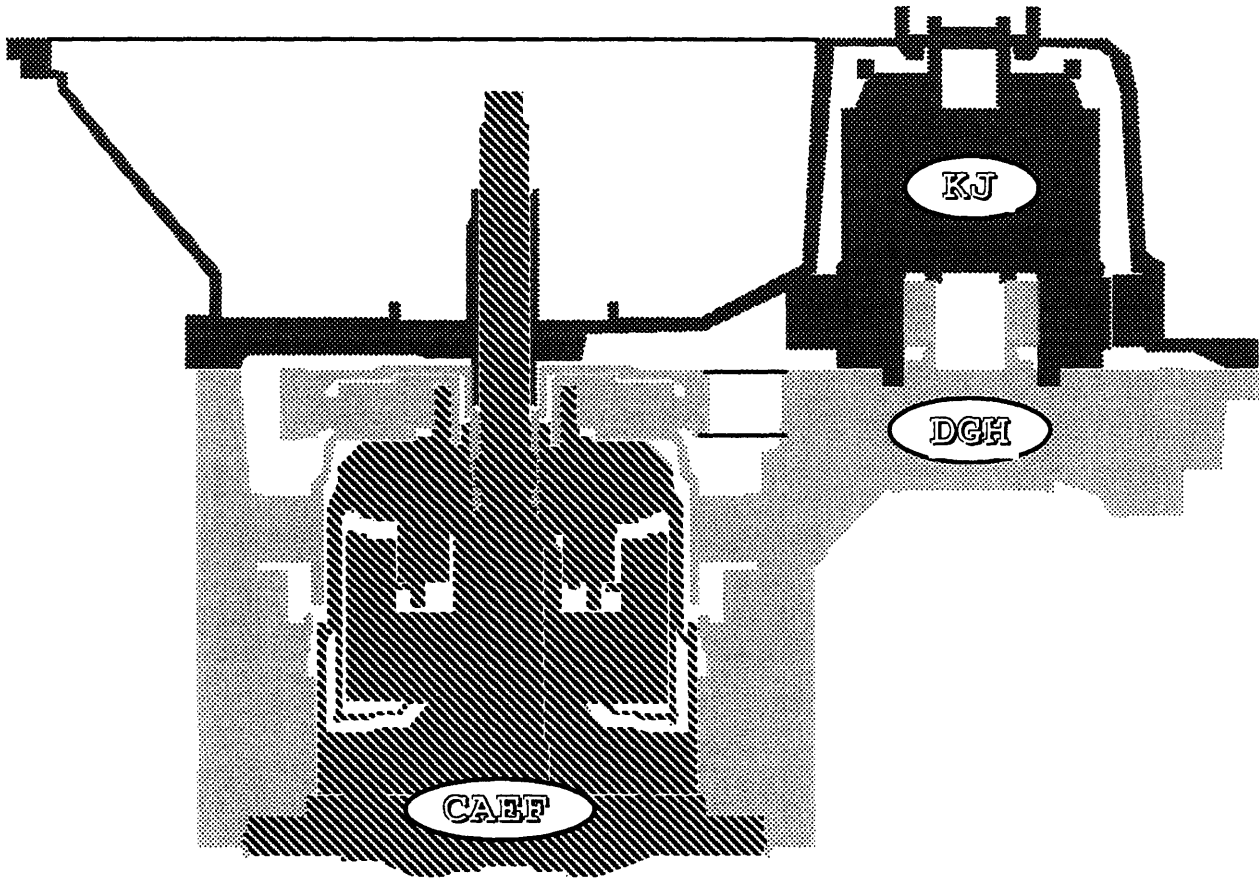


Figure 4-13: Suggested partitioning of the BAT assembly with original subassemblies

The results of this experiment suggest that the action count closure criterion alone may not be sufficient but by taking other criteria into consideration such as avoid multiple reorientations and refixturings, favorable partitionings can be generated as was done here.

4.2.2 GA Subassembly Repartitioning of BAT-F

Next we wish to test the GA tool for suggesting repartitionings beginning with a nominal subassembly partitioning scheme. BAT-F represents the original BAT assembly with subassembly F broken down into smaller subassemblies and component parts F1, F2, F3, and F4 as shown in Figure 4-14. Subassembly F was chosen to be broken down due to the large number of actions contained within the subassembly. The liaison graph is given in Figure 4-15 with the weighted liaison graph information given in tabular form in Table 4-4. In determining the action count values for the weighted liaison graph, we first encountered an example of where multiple action count values were necessary due to the introduction of new kinematic degrees of freedoms as subassemblies or parts come together. This can be seen in the breakdown of subassembly F. Take parts F3 and F4 for instance. F3's mate to subassembly A has a default action count value of 0 because no actions are closed by the mate. The same is true for the liaison between F4 and A. But consider a mate between subassembly A and a subassembly consisting of parts F3 and F4. Once mated to form a separate subassembly, F3 and F4 are free to rotate with respect to each other due to the roller bearing involved in their mate. Thus, the action count value for the mate between A and a subassembly consisting of F3 and F4 is now 1 because of the new internal kinematic degree of freedom.

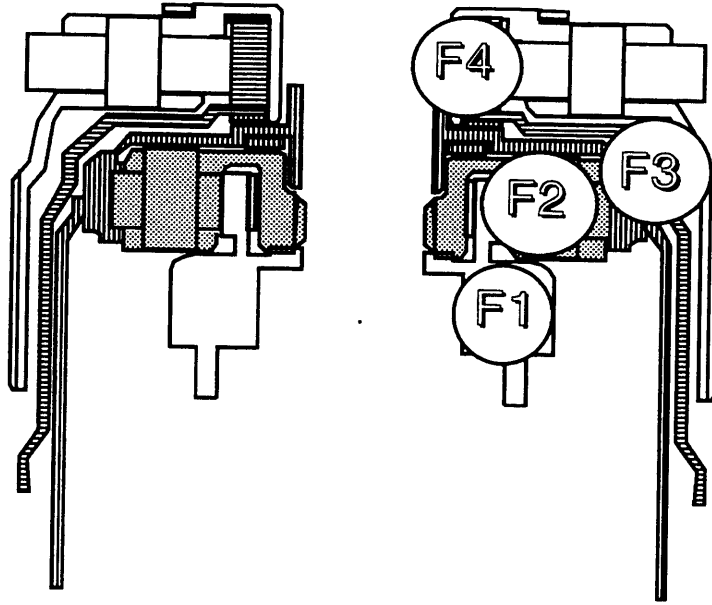


Figure 4-14: Breakdown of subassembly F

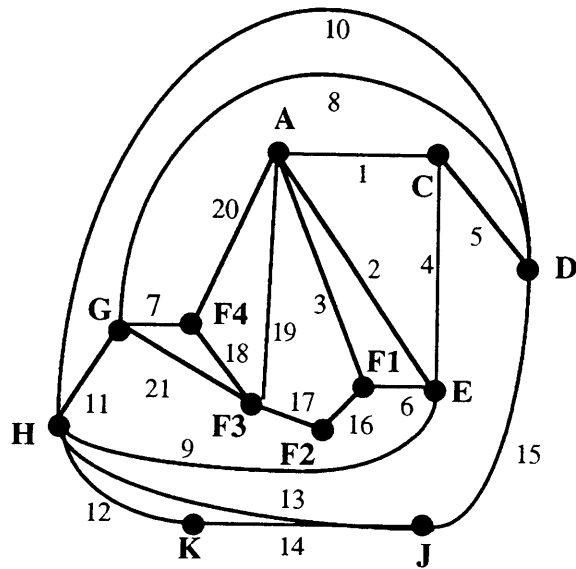


Figure 4-15: Liaison graph for BAT-F

condition liaison	part	part	weight= action count	condition liaison	liaison	part	part	weight= action count	condition liaison
1	A	C	0		12	H	K	3	
2	A	E	0		12			1	14
3	A	F1	5		13	H	J	0	
3			6	16, 17	14	J	K	3	
4	C	E	0		14			1	12
5	C	D	0		15	D	J	0	
6	E	F1	0		16	F1	F2	4	
7	F4	G	4		16			1	17
7			1	11	17	F2	F3	4	
8	D	G	4		17			1	16
8			5	11	18	F3	F4	0	
9	E	H	0		19	A	F3	1	
10	D	H	0		19			1	18
11	G	H	3		20	A	F4	0	
11			0	7	20			1	1, 5, 7
11			4	8	21	G	F3	0	
11		E	1	7, 8					

Table 4-4: Weighted liaison graph information for BAT-F in tabular form

Using the weighted liaison graph information and the precedence relations from ASA, the GA is used to repartition BAT-F into 9 subassemblies. The purpose of this experiment is to see if the suggested partitionings will match the original 9 subassemblies or if not, what improvements may be suggested by the action count closure criterion. A sample set of subassembly repartitionings generated by the GA run for 5000 generations using a population size of 250 achieving an objective function score of 2.69678 is given as follows:

G - F4 - D - F3 - H - F1 E A F2 - K - C - J
D - G - F4 - H - F3 - E F1 F2 A - K - C - J
F4 - G - D - F3 - H - E F1 F2 A - K - C - J
G - F4 - D - F3 - H - A E F1 F2 - K - J - C
D - G - F4 - H - F3 F2 - E F1 A - K - J - C
D - G - F4 - H - F3 - A E F1 F2 - K - C - J
G - F4 - D - H - F3 F2 - E A F1 - K - C - J
F4 - G - D - H - F3 F2 - E A F1 - K - C - J
G - F4 - F3 - D - H - E F1 F2 A - K - J - C
F4 - G - D - F3 - H - A E F1 F2 - K - C - J

The common trend in all the generated partitionings is to have E, A, and F1 together as a separate subassembly and F2 and F3 as another separate subassembly or to have E, A, F1, and F2 repartitioned as a separate subassembly. This indicates that having some of the components of F partitioned together fares well with the action count closure criterion, but subassemblies F1 and F2 are better off partitioned with subassemblies E and A than with the rest of F as in the original partitioning scheme.

Again, these sequences themselves are not practical for the same reasons as mentioned in section 4.2.1 so again the constraint that all sequences must start with subassembly C was imposed. This proved to be a rather computationally intensive problem for the GA, so we started the GA with an initial population set to the following sequence/partitioning which has an objective score of 2.96954:

C - A - D - F1 E - F3 F2 - F4 - G H - K - J

Using a population size of 200, after 5000 generations, the GA arrived at the following optimal sequence/partitionings with an objective score of 2.92328:

C - D - A - F1 E - F2 F3 - G F4 - H - K - J
C - D - A E - F1 - F2 F3 - G F4 - H - K - J
C - D - A - F1 E F2 - F3 - G F4 - H - K - J
C - D - A E F1 - F2 - F3 - G F4 - H - K - J
C - D - A - F1 F2 E - F3 - G F4 - H - K - J

We arrived at pretty much the same subassembly partitionings as before, at the cost of a higher objective score based on the action count closure criterion, but with the benefit of assembly sequences that are practical in terms of other criteria such as reducing the number of assembly reorientations and refixturings. Thus, the action count closure criterion found optimal subassembly partitionings but without consideration of other criteria beforehand, the sequences

were impractical. This indicates that the action count closure criterion is a useful criterion, but should not be the only criterion considered when looking for optimal assembly sequences.

Recall now that in the previous runs, the desired number of subassemblies was fixed to match the original number of subassemblies. This could account for the fact that all four components of F could not be partitioned into a single subassembly along with E and A due to the fact that by doing so, the resulting partitionings would result in fewer than the original number of subassemblies. Therefore, this constraint was removed for the next run and the algorithm was allowed to vary the number of subassemblies to determine the number that worked best. The GA was again run for 5000 generations using a population size of 250. A sample of the resulting partitionings with an objective function score of 2.43085 are as follows:

F1 E A - F3 F2 F4 - D G H - K - J - C
F3 F4 - F1 F2 E A - G D H - K - J - C
F3 F2 F4 - F1 E A - G D H - K - J - C
A E F1 F2 - F3 F4 - D G H - K - J - C
F1 E A - F2 F3 F4 - D G H - K - J - C
F4 F3 - E F1 F2 A - D G H - K - C - J

The results all have D, G, and H repartitioned as a single subassembly. A, E, and F1 are also repartitioned together as are F3 and F4 with F2 repartitioned with either one of these two subassemblies. These results correspond well with those in section 4.2.1 for the BAT assembly with the original subassembly partitioning scheme when the number of final subassemblies was also allowed to vary. Recall that for the BAT assembly with the original subassembly partitioning scheme, D, G, and H were partitioned together as were A, E, C, and F. Now, having broken down F into smaller subassemblies, we can see that it is favorable for F1 alone to be partitioned with A and E and for F3 and F4 to be partitioned as a separate subassembly. Thus, by breaking down a complicated subassembly such as F, the GA can more finely repartition the original subassemblies according to the criterion used.

Again, the sequences that were generated were impractical so another GA run was done with the constraint that assembly sequences must begin with subassembly C. Using an initial population of 250, the GA was run for 2000 generations and achieved the resulting partitionings/sequences with an objective score of 2.57611:

```

C - F1 F2 E A -- F3 F4 --- G D H --- K -- J
- C - F1 E A --- F3 F4 F2 - D G H K ---- J -
- C - F1 E A F2 --- F4 F3 - D G H -- K --- J
C -- F1 E A -- F2 F3 F4 - G D H ---- K J -
- C - F1 E A F2 - F3 F4 --- G D H K ---- J -
C - F1 E F2 A --- F3 F4 -- G D H K ---- J
C - F1 F2 E A - F3 F4 ---- G D H ---- K J -
- C - F1 E A F2 -- F3 F4 -- G D H ---- K J
- C - F1 F2 E A - F3 F4 --- G D H K ---- J

```

The subassembly partitionings remain pretty much the same with the exception that K is often partitioned with the subassembly containing G, D, and H, and also with subassembly J as it was with the original BAT assembly in section 4.2.1. Again, after breaking down a complicated subassembly, the GA is useful in repartitioning the original subassemblies according to the action count criterion used, but with consideration given to other criteria, sequences that are more practical can also be generated with the GA tool.

4.2.3 Precedence Relation Elimination Applied to BAT

The precedence relations for the BAT assembly are first generated using ASA. The precedence relations are listed in Table 4-5. Next, by examining the precedence relations, the geometric interferences that are the basis for individual precedence relations can be determined. For the BAT assembly, there are three classes of precedence relations. The first class can be eliminated or reduced by repartitioning the roller bearing of E with F as shown in Figure 4-16 so that E can mate with A and pass completely through G. Precedence relations that fall in this class are PR#1, PR#2, and PR#9. The second class can be eliminated or reduced by detail redesign to allow G to pass completely through D. This includes PR#4 and PR#5. The final class, which

includes the remaining precedence relations, can not be eliminated by detail redesign or subassembly repartitioning.

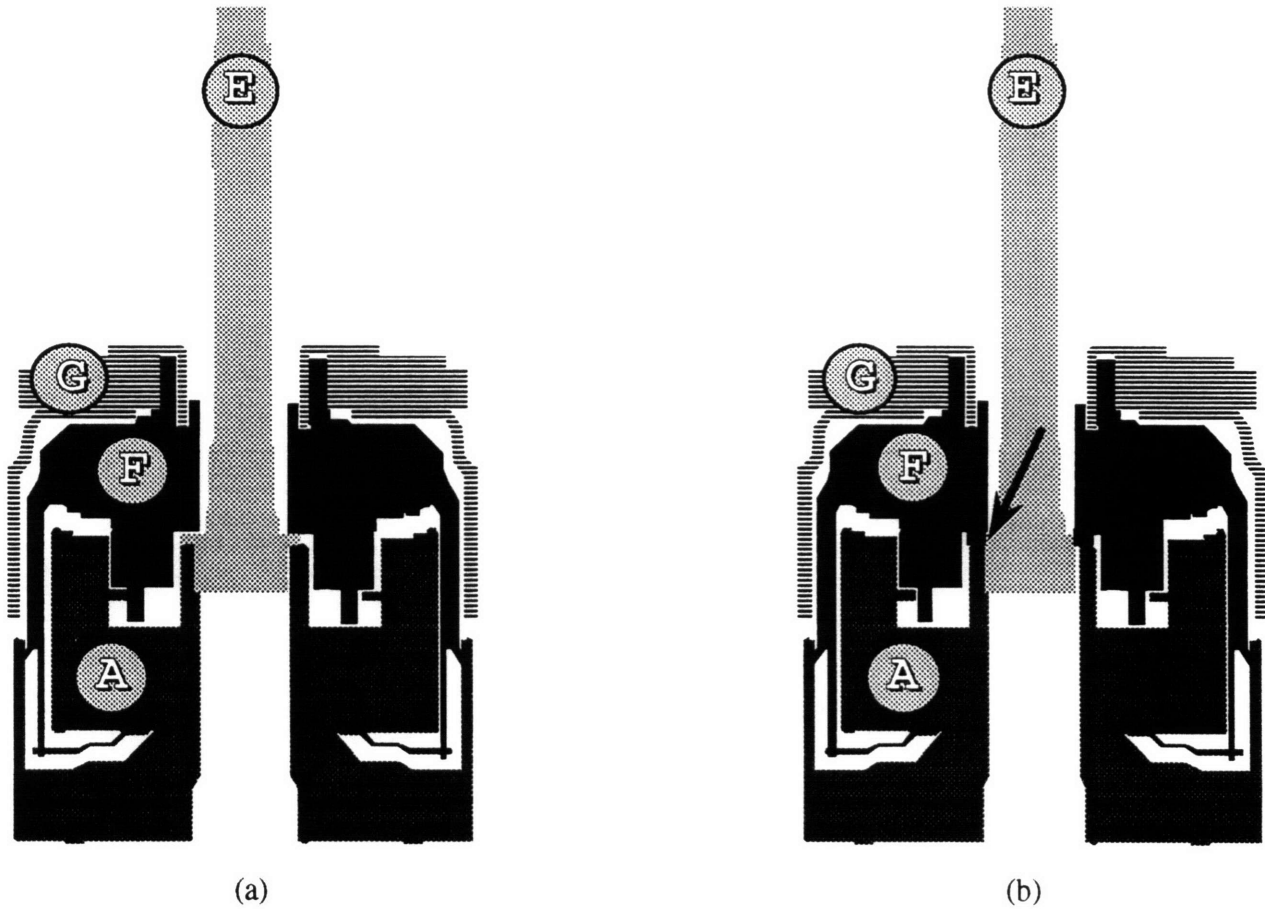


Figure 4-16: (a) Subassemblies A, E, F, and G before repartitioning and (b) after. Arrow points to area where repartitioning occurred.

	<i>Precedence Relation</i>
1	1 & 2 \geq 4
2	2 & 6 \geq 3
3	6 \geq 9
4	5 & 8 \geq 1 & 2 & 3 & 4 & 6 & 7
5	8 & 11 \geq 10
6	10 & 11 & 13 \geq 8 & 15
7	14 \geq 15
8	1 \geq 5 & 8
9	2 & 4 \geq 1 & 5 & 8
10	7 & 11 \geq 6 & 9
11	8 & 10 & 12 & 13 \geq 11 & 14 & 15
12	12 & 14 \geq 13

13	$3 \& 6 \geq 1 \& 2 \& 4 \& 5 \& 14 \& 15$
14	$2 \& 3 \& 4 \geq 1 \& 5 \& 6 \& 14 \& 15$
15	$1 \geq 5 \& 14 \& 15$
16	$3 \& 6 \& 7 \geq 1 \& 2 \& 4 \& 5 \& 8$

Table 4-5: Precedence relations for BAT

By incorporating these suggested redesigns or repartitionings, we arrive at three new designs for BAT: BAT-RD1, BAT-RD2, BAT-RD3. All three have the roller bearing of E repartitioned with F. In addition, BAT-RD2 contains a redesign so that G can pass through D, but A can no longer pass through D due to the detail redesign. Finally, BAT-RD3 allows for both G and A to pass through D by further redesign. The precedence relations for BAT-RD1, BAT-RD2, and BAT-RD3 are given in Tables 4-6, 4-7, and 4-8, respectively.

	<i>Precedence Relation</i>
1	$1 \& 2 \& 3 \geq 4 \& 6$
2	$2 \& 4 \& 6 \geq 1 \& 3$
3	$6 \geq 9$
4	$5 \& 8 \geq 1 \& 2 \& 3 \& 4 \& 6 \& 7$
5	$8 \& 11 \geq 10$
6	$10 \& 11 \& 13 \geq 8 \& 15$
7	$14 \geq 15$
8	$1 \geq 5 \& 8$
9	$4 \geq 5 \& 14 \& 15$
10	$7 \& 11 \geq 6 \& 9$
11	$8 \& 10 \& 12 \& 13 \geq 11 \& 14 \& 15$
12	$2 \& 3 \& 4 \& 7 \geq 1 \& 5 \& 6 \& 8$
13	$12 \& 14 \geq 13$
14	$3 \& 6 \geq 1 \& 2 \& 4 \& 5 \& 14 \& 15$
15	$1 \& 2 \& 6 \geq 3 \& 4 \& 5 \& 14 \& 15$
16	$3 \& 6 \& 7 \geq 1 \& 2 \& 4 \& 5 \& 8$

Table 4-6: Precedence relations for BAT-RD1

	<i>Precedence Relation</i>
1	$1 \geq 5$
2	$2 \& 4 \& 6 \geq 1 \& 3$
3	$6 \geq 9$
4	$10 \geq 2 \& 3 \& 6 \& 9$
5	$7 \& 11 \geq 6 \& 9$

6	10 & 13 >= 15
7	12 & 14 >= 13
8	2 & 3 & 4 & 7 >= 1 & 5 & 6 & 8
9	8 & 11 >= 1 & 5 & 10
10	8 & 10 >= 3 & 7 & 11
11	3 & 6 & 7 >= 1 & 2 & 4 & 5 & 8
12	1 & 2 & 3 >= 4 & 6

Table 4-7: Precedence relations for BAT-RD2

	<i>Precedence Relation</i>
1	1 & 2 & 3 >= 4 & 6
2	2 & 4 & 6 >= 1 & 3
3	6 >= 9
4	5 & 10 >= 1 & 2 & 3 & 4 & 6 & 9
5	8 & 11 >= 5 & 10
6	10 & 13 >= 15
7	12 & 14 >= 13
8	1 >= 5 & 8
9	2 & 3 & 4 & 7 >= 1 & 5 & 6 & 8
10	7 & 11 >= 6 & 9
11	3 & 6 & 7 >= 1 & 2 & 4 & 5 & 8

Table 4-8: Precedence relations for BAT-RD3

The precedence relation sets get relatively weaker, i.e. less constraining on the set of feasible assembly sequences, in the progression: BAT, BAT-RD2, BAT-RD1, and BAT-RD3. This can be seen by the number of feasible sequences generating using ASA (specifically, EDIT) as shown in Table 4-9. As shown in the table, BAT-RD1 offers more assembly sequences than does BAT due to the fact that precedence relations were removed or softened by repartitioning; thus, removing assembly constraints to allow for more assembly freedom. Furthermore, the numbers suggest that the redesign in BAT-RD2 may add more constraints that it relieves, yet it should allow for a new branch of sequences that were previously not feasible and is worth exploring. Finally, by redesigning such that both G and A can pass through D in BAT-RD3 results in the largest number of possible assembly sequences; thus, assembly constraints have effectively been removed through redesign in addition to repartitioning to allow for more feasible assembly sequences.

	<i>BAT</i>	<i>BAT-RD1</i>	<i>BAT-RD2</i>	<i>BAT-RD3</i>
# undeleted sequences	324	635	469	706

Table 4-9: Number of feasible assembly sequences

First, assembly sequences were generated for BAT-RD1, BAT-RD2, and BAT-RD3 using the GA and compared with those generated for BAT. The sequences were all relatively similar with the same objective function score. Yet when we constrained that the sequences must start with subassembly C as was done previously for BAT, both BAT-RD2 and BAT-RD3 had sequences with better objective function scores than did those for BAT and BAT-RD1. These sequences are the same for both BAT-RD2 and BAT-RD3 and have an objective function score of 3.39117 as opposed to 3.93701 for those suggested for BAT and BAT-RD1:

C E A F G D H K J
C A E F G D H K J

It can be seen that these sequences became possible due to the redesign in both BAT-RD2 and BAT-RD3 that allows G to pass through D. Thus, it can be seen that by examining the precedence relations, one can determine design changes, either detail redesign or subassembly repartitioning, that will loosen the precedence relations and therefore the assembly constraints to allow for more feasible assembly sequences. By doing so, it is possible to find assembly sequences that rate better by the action count closure criterion.

4.3 Summary

In this chapter, the tools and methods developed in the previous chapter have been applied to actual complex assemblies to explore their usefulness in practice. In some cases, the results were as expected and in others, offered new insight previously not thought of.

CHAPTER 5

Conclusions and Further Work

The goal of this research was to develop the techniques and tools required to apply Design-For-Assembly to complex assemblies. Noting how current DFA methods are based on experience and approximations and are difficult and often inadequate on more complicated assemblies, one can see that this is not an easy task. As the applications of computer-aided tools become more widespread, it is only reasonable to expect that assembly planning techniques become more and more capable in aiding a designer in keeping assembly and manufacture in mind when developing new products. This thesis presents one such computer-aided tool to aid in DFA of complex assemblies by allowing for rapid evaluation of assembly options by design changes and in assembly planning itself by suggesting subassembly partitioning schemes along with assembly sequences. The results presented in the previous chapter show how the genetic algorithm based tool is capable of handling the difficult problem of assembly sequence and subassembly partitioning choice for complex assemblies. This thesis also presents a methodology for applying DFA that relies on ASA wherein the GA tool proves to play a useful part. The developed tool is powerful in that not only are subassembly partitionings evaluated, but so are associated assembly sequences at the same time. Also, by using a genetic algorithm, the tool is capable of keeping all possible assembly sequences in the search space, and by using precedence relations derived through ASA, geometric reasoning is fully automated ensuring that resulting subassembly partitionings and assembly sequences are geometrically feasible.

The action count closure criterion used by the GA tool has also proven to be useful in locating and avoiding difficult assembly moves, but as discovered, it may not be the driving criterion in determining assembly sequences and subassembly partitioning schemes. The GA

based tool developed is fully capable of handling multiple criteria due to the way the genetic algorithm uses an objective function for scoring partitions/sequences it evaluates, but would require additional initial user input and a scale on which competing criteria can be measured. There are many criteria that could be considered such as minimizing the number of reorientations or refixturings during assembly or minimizing the tolerance buildup during assembly to mention a few. Future work would include the incorporation of multiple criteria which would require that the criteria be based on quantifiable measures of assembly difficulty and that the issue of how to weigh competing criteria in the objective function be resolved.

Bibliography

- [1] G. Boothroyd & P. Dewhurst, "Product Design for Assembly," Boothroyd-Dewhurst, Inc., Wakefield, RI, 1987.
- [2] S. Miyakawa and T. Shigemura, The Hitachi Assemblability Evaluation Method (AEM), *Proc. Int. Conf. on Mfg. Systems and Environment—Looking Toward the 21st Century (The Japan Society of Mech. Engrs.)*, pp. 277-282, 1990.
- [3] J. L. Nevins & D. E. Whitney, Editors, *Concurrent Design of Products and Processes*, McGraw-Hill Publishing Co., New York, 1989.
- [4] T. L. De Fazio and D. E. Whitney, "Simplified generation of all mechanical assembly sequences," *IEEE J. Robotics and Automation*, vol. 3, no. 6, pp. 640-658, Dec. 1987 (Corrections *ibid*, vol. 4, no. 6, pp. 705-708, Dec. 1988).
- [5] A. Bourjalt, "*Contribution à une approche méthodologique de l'assemblage automatisé: élaboration automatique des séquences opératoires*," Ph.D. dissertation, Faculté des Sciences et des Techniques de l'Université de Franche-Comté, Nov. 1984.
- [6] Luis S. Homem de Mello and A. C. Sanderson, "Automatic generation of mechanical assembly sequences," Technical report, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Dec. 1988.
- [7] Russell W. Whipple, "Assembly Sequence Generation Using the 'Onion-Skin' Method and Sequence Editing Using Stability Analysis," Master's thesis, MIT, June 1990.
- [8] Daniel Baldwin, "Algorithmic Methods and Software Tools for the Generation of Mechanical Assembly Sequences," Master's thesis, MIT, May 1990.
- [9] A. Bourjault, "*Outils méthodologiques pour la définition de systèmes d'assemblage automatisés*," Technical report, Centre de Recherche Microsystems et Robotique, Université de Franche-Compte, Feb. 1987. Also computer implementation of these tools: "SAGA - Systeme d'Elaboration Automatique de Gammes d'Assemblage, Version 1.2," Feb. 1988.
- [10] Luis S. Homem de Mello and A. C. Sanderson, "AND/OR graph representation of assembly plans," *IEEE Trans. Robotics Automat.*, vol. 6, no. 2, pp. 188-199, Apr. 1990.
- [11] M-C. Lui, "Generation and Evaluation of Mechanical Assembly Sequences Using the Liaison Sequence Method," Master's thesis, MIT, May 1988.

- [12] T. E. Abell, "AN Interactive Software Tool for Editing and Evaluating Mechanical Assembly Sequences Based on Fixturing and Orientation Requirements," Master's thesis, MIT, Aug. 1989.
- [13] S. Lee and Y. G. Shin, "Assembly planning based on subassembly extraction," in *Proc. 1990 IEEE Conf. on Robotics and Automation* (Cincinnati, OH), pp. 1606-1611, May 1990.
- [14] J. M. Milner, S. C. Graves, D. E. Whitney, "Using Simulated Annealing to Select Least-Cost Assembly Sequences," in *Proc. 1994 IEEE International Conference on Robotics and Automation* (San Diego, CA), vol. 3, pp. 2058-2063, May 1994.
- [15] D. S. Hong and H. S. Cho, "Optimization of Robotic Assembly Sequences Using Neural Network," in *Proc. of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Yokohama, Japan), vol. 1, pp. 232-239, Jul. 1993.
- [16] F. Bonneville, C. Perrard, J. M. Henrioud, "A Genetic Algorithm to Generate and Evaluate Assembly Plans," in *Proc. 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation* (Paris, France), vol. 2, pp. 231-239, Oct. 1995.
- [17] D. E. Goldberg, "Genetic Algorithms: In Search, Optimization and Machine Learning," Addison-Wesley, Reading, MA, 1989.
- [18] T. L. De Fazio, Personal Communication, Mar. 1995.
- [19] D. R. Jones and M. A. Beltramo, "Solving partitioning problems with genetic algorithms," in *Proc. Forth Intl. Conf. on Genetic Algorithms*, pp. 442-449, 1993.
- [20] J. Bhuyan, V. Raghavan, and V. Elayavalli, *Genetic-Based Clustering*, Technical Report 90-4-1, The Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA, March 1990.
- [21] T. L. De Fazio, Personal Communication, Oct. 1995.