CHAPTER 1. INTRODUCTION

A product designed for disassembly can be taken apart easily to support applications such as assembling, maintenance, reuse and recycling [1,2]. For these applications, the disassembly analysis involves evaluating a disassembly sequence (order of component removals) from the geometric model of an assembly (A). In general, two categories of problems exist in disassembly: (let $C_i$ denotes the $i$th component in A)

1. **Complete Disassembly** (CD) involves disassembling all the components in A to obtain a CD sequence. For example, to disassemble all the components from A in Figure 1, one possible sequence is \{C_1, C_2, C_6, C_3, C_4, C_5\}, as shown in Figure 2.

2. **Selective Disassembly** (SD) involves disassembling a subset of components (C) from A to obtain a SD sequence (S). For example, to disassemble $C = \{C_4, C_5\}$ for A in Figure 1, one possible sequence $S = \{C_6, C_4, C_5\}$, as shown in Figure 3.

![Figure 1. Test assembly to illustrate complete and selective disassembly](image1)

![Figure 2. CD sequence is {C_1, C_2, C_6, C_3, C_4, C_5} for A in Figure 1](image2)
1.1 Need for Selective Disassembly Analysis

An application for CD is assembling, since reversing a disassembly sequence can potentially yield to an assembly sequence [3, 4]. For example, in Figure 2 the reverse of disassembly sequence gives an assembly sequence \{C_5, C_4, C_3, C_6, C_2, C_1\}. However, SD is often more relevant than CD for applications such as maintenance, recycling and reuse [5]. These applications usually require removal of a subset of components of A, and not the entire assembly, hence providing a need for SD.

![Figure 4. Selective disassembly of the Engine for maintenance [5]](image)
For example, aircraft engine maintenance requires the SD of the engine, which may involve prior disassembly/removal of some parts before the engine itself may be disassembled, and it does not require the disassembly of the entire aircraft. For example, Figure 4 shows the SD of engine sub-assembly from a plane assembly for maintenance. As can be imagined, the effective use of some product assemblies requires their rapid and efficient disassembly/assembly. For instance, if an aircraft-engine cannot be rapidly disassembled and reassembled for maintenance and/or repair, the aircraft may need to be taken out of service for an extended period of time, and the effective upkeep cost of the aircraft can be greatly increased.

![Figure 5. Selective disassembly of PVC and nylon components for recycling [5]](image)

Another application of SD is reuse, wherein components of one product are reused in another product. For example, SD of an instrument panel (for its reuse) from a car assembly. From the standpoint of conservation of natural resources, it is generally desirable to reuse components rather than constructing new ones. Yet another application of SD is in recycling,
which requires separation of components of different materials from A. For example, Figure 5 shows the SD of PVC and Nylon components from a dashboard assembly for recycling. However, if components to be recycled/reused cannot be quickly and efficiently disassembled from the assembly, the cost of recycling may be such that it is not worthwhile.

Therefore, SD analysis of components in an assembly, and the determination of an efficient SD solution such as one with less number of removals, is an important area of research in assembly planning.

1.2 Virtual Selective Disassembly Analysis

![Figure 6. Motivation to virtual SD analysis](image)

Virtual SD analysis involves evaluating at the product design stage various aspects of assembly including the sequence (S) to disassemble the selected components, disassembly paths, SD cost/time, ease-of-disassembly of components and sub-assembly analysis. Performing disassembly analysis on the CAD model of a product before production allows analysis for easy
separation of components, thereby reducing the disassembling cost/time for product maintenance/recycling [6], as also reducing the burden placed on the environment from the ever-increasing number of obsolete products [7].

Moreover, based on the SD analysis the designer can assess the design for disassembly [8] and in turn may perform design changes for a better product. In general, during the product development stage an assembly is analyzed for SD and the design changes are performed iteratively, until the selected components are designed satisfying some objective (such as minimal removals). For example, designing the pipe layout in an aircraft-engine involves an objective of disassembling selected components for maintenance by removing not more than a pre-defined number of component removals. Thus, as illustrated in Figure 6, performing virtual SD analysis on the CAD model of an assembly A is an important area of research in product development.

1.3 Research Goal and Objectives

Currently there exists no design-aids or software tools to perform efficient SD analysis. Moreover, manual SD analysis takes significant amount of time and may not be even possible for assemblies with larger number of components and/or complex geometry. Therefore, the goal of the current research is to develop design-aids and automation tools to perform virtual SD analysis.

In the current research, the requirement is to identify S to disassemble C. However, apart from the objective that the SD analysis should be automatic and analyze 3D geometric models, there are two other important factors: (i) Computationally feasible algorithms and (ii) Appropriate SD solution with fewer number of removals
The objective of fewer number of removals is appropriate since it is a measure of difficulty of disassembling [9]. Moreover, the product design for manufacture suggests lesser disassembly operations and easier separation of components for maintenance/recycling [10, 11].

The factors, computation time and the removals are inter-related and one is usually achieved at the cost of the other. For example, if feasible computation time is the only factor, then any of the CD solution can be extended for SD. Although a SD sequence can be obtained from a CD solution (e.g. references [3, 4 and 12]), the solution is inappropriate for SD. For example, S can be obtained by recursively disassembling components that are disassemblable in A until all the components in C are disassembled. To illustrate this, consider A in Figure 1 with C = \{C_4, C_5\}. Let \(n_r\) = number of components in S. Following the CD algorithm, S to disassemble C is \{C_1, C_2, C_6, C_3, C_4, C_5\} with \(n_r = 6\), but a better sequence exists S = \{C_6, C_4, C_5\} with \(n_r = 3\). Therefore, a CD solution is not recommended for SD analysis.

On the other hand, if an appropriate SD solution with less component removals is the only factor, then an exhaustive enumeration can be applied for SD. However, this approach is computationally expensive (typically exponential with respect to the number of components in A) since it enumerates all possible sequences, it is not recommended.

Therefore, the current research attempts to provide methods that balance the requirements of computation time and an appropriate SD solution, i.e., automatically determining a SD solution with fewer removals computed in a feasible computation time.

1.4 Assumptions of the Current Research

The assumptions for the current research are:

1. The relative motions of the components are determined without considering the tools, fixtures or robots required to achieve these motions.
2. Assemblies are assumed to be rigid (non-deformable), frictionless and defined by nominal geometry (no tolerances).

3. Components are disassemblable along simple disassembly paths such as translations and screw motions.

4. Disassembly sequences are monotonic (components are totally removed while disassembling), and non-destructive (no component is destroyed) [13].

Assumptions 1-4 are standard assumptions followed by different researchers (e.g., references [3, 4, 12, 14 and 15]) in automated assembly/disassembly analysis.

Disassembly analysis with Assumption 1 gives meaningful results, for example the fixture elements can be modeled based on the sequence determined [14], or can be modeled as constraints to components, or can be modeled as a component with constraints [9].

Assumption 3 is reasonable, since the design for manufacturing [10, 11] recommends simple motions for disassembly of components. Several researchers (e.g. references [3 and 4]) utilize this assumption, since automatic generation of disassembly sequences allowing general disassembly motion is computationally expensive [16]. Moreover, this assumption is realistic for some real world examples, as indicated by some existing assembly planning systems (e.g. references 17, 18).

1.5 Overview and Organization of this Thesis

Based on the number of components to be disassembled, the \textbf{SD} problem of disassembling \(m\) target components (C) from an assembly (A) of \(n\) components in determining a \textbf{SD} sequence (S) is categorized in two classes:

1. \textbf{SD} of one component--defined as Single \textbf{SD} (\(m = 1\)),
2. \textbf{SD} of \(m\) components (\(1 < m < n\))--defined as Multiple \textbf{SD}
For automatically determining a **SD** solution, a new approach called Wave Propagation (WP) has been developed. The approach involves topologically arranging the components in the assembly denoting the disassembly order by analyzing the assembly geometry. At first, the problems of single and multiple **SD** [19-22] are analyzed with contact-geometry and subsequently the concept is extended to include spatial constraints of components [23-24]. The WP algorithms and the geometric-reasoning methods implemented in a prototypical software system are then presented as design synthesis tool for virtual **SD** [25-34].

The rest of this thesis is organized as follows:

**Chapter 2 Related Research**: This chapter presents the related research in assembly and disassembly sequence planning and disassembly evaluation tool development.

**Chapter 3 Single Selective Disassembly**: This chapter analyzes the problem of Single **SD**. The motivation is that the disassembly analysis can be localized with respect to **C** and the analysis of all the components may not be required. A WP algorithm called Single Wave Propagation (SWP) is proposed which orders the components for disassembly from **C** (inwards) towards the disassemblable components (the assembly boundary), by analyzing the contact-geometry of the components.

**Chapter 4 Multiple Selective Disassembly**: This chapter analyzes the problem of **Multiple SD**. The motivation is, that although a SWP algorithm can be applied for multiple **SD**, a better solution with fewer component removals may be obtained if two or more components are disassembled along a common sequence. Two WP algorithms are proposed based on the contact-geometry analysis for multiple **SD**: (I) *Multiple Wave Propagation (MWP)* algorithm and (II) Priority Intersection Event (PIE) algorithm. Both the algorithms order the components in the assembly with respect to **C** and also the assembly boundary in determining a common sequence.
between the target components. The proposed algorithms differ in the number of removals and the computation time in determining a SD solution.

**Chapter 5 Global Selective Disassembly**: This chapter analyzes the problem of determining a non-interfering (collision free) SD sequence. A WP algorithm called Global Selective Disassembly (GSD) is proposed that orders the components for SD both by analyzing the spatial geometric constraints and the user-defined constraints such as components grouping and directional constraints.

**Chapter 6 Selective Disassembly Analysis**: This chapter presents the implementation of a prototypical software system called Assembly and Disassembly in Three Dimensions (A3D) and its functional features as a design tool for SD analysis.

**Chapter 7 Results and Discussion**: This chapter describes the implemented results of the WP algorithms and the developed A3D system for SD analysis. Moreover, a discussion on the proposed WP algorithms in determining a SD solution with fewer removals and feasible computation time is also presented.

**Chapter 8 Application, Results and Discussion**: This chapter presents the application results of the SD analysis for assembling, maintenance and recycling.

**Chapter 9 Discussion and Conclusion**: This thesis concludes with this chapter, presenting the research findings, the contributions of the current research and the future work.

**Appendix** presents supplementary results of the SD algorithms.
CHAPTER 2. RELATED RESEARCH

The related research in assembly/disassembly is discussed under the following sub-headings: (i) Assembly/Disassembly Survey, (ii) Assembly/Disassembly Sequencing, and (iii) Assembly/Disassembly Planners.

2.1 Assembly/Disassembly Survey

There are several survey papers on assembly and disassembly analysis. Homem-de-Mello [9] examine the important issues in assembly modeling, systematization and computerization of mechanical assembly planning. Whitney [35] provides a definition of assembly modeling and lists important aspects of mechanical assembly modeling to improve the design and manufacturing. Boothroyd [36] review Design for Assembly methods that have been developed and discuss the research trends in Design for Disassembly. Gupta [37] and Jovane [1] provide an overview of the ongoing research in product disassembly and also present the topics and trends for future activities. Penev [38] review theories for determining disassembly sequences and analyze the methods for creation of an effective disassembly strategy. Sack [39] present a survey of the techniques for determining the translational separability of 2D polygons. Hrinyak [40] examines the existing disassembly software tools available to the designers for inclusion in their design processes. Moreover, a survey of research in assembly/disassembly sequencing and planning is listed in the assembly planning bibliography [41-43].

2.2 Assembly/Disassembly Sequencing

Several representations allow the evaluation of assembly and disassembly sequences, and are presented below.
Baldwin [44], De-Fazio [45] and Henrioud [46] propose the *Assembly Sequence Diagram* to represent the ability or inability to assemble a component with other components. The user is asked a number of questions regarding the precedence relation of components in order to enumerate all assembly sequences. However, the number of such questions grow exponential to the number of components in \( A \), and therefore the computational expense of this method limits it to smaller assemblies. Further, the method is not automated insofar as users need to answer the inquiries relating to the components, and therefore the time expense of the method is further increased.

Homem de Mello [47, 48] propose *AND/OR Graph* that establishes conditions and precedence relationships between components. The AND/OR graph represents the sequences for assembling based on local motion analysis in three dimensions. A cut-set of the mating diagram of the assembly and subassemblies is used as a basis for disassembly questions. The AND/OR graph method suffers from the same disadvantages as Assembly Sequence Diagram method in that it is time-consuming to execute except for assemblies with only a small number of components.

To reduce the number of queries to the user, several researchers [49-53] propose *Knowledge-Assisted Systems* that uses pre-existing information (stored from the earlier queries to the user) to analyze similar products. In effect, these are essentially expert systems. The part interconnections and the directional constraints of the motion that bring two parts together were analyzed based on both the existing knowledge and further user queries. These methods do not significantly decrease analysis time, particularly insofar as the information stored by expert system can only be re-used in the later products of the same type.
Several researchers have utilized the computational geometric techniques for the assembly analysis. Arkin [54] uses the concept of a monotone path between obstacles to deduce a removable subassembly and a single extended translation to remove it. Shyamsundar [55] proposes the use of a convex hull based representation of assemblies for geometric reasoning. Beasley [12] considers the generation of disassembly motion for voxelized models of objects based on disassemblablity analysis. Mattikalli [15] proposes constraint-geometry-based analysis to determine the translational and rotational disassemblability of components for 3D assemblies. Ashai [56] proposes metric-based evaluation of CD sequences via disassemblability analysis of 3D assemblies. However, these proposed techniques for CD are primarily based on the analysis of contact geometry of components, and they therefore suffer from the disadvantages of the contact geometry reasoning discussed later in this document (most particularly, failure to detect the interference of components during the linear global assembly/disassembly motions).

Several researchers [57-59] have utilized the geometric information in the CAD models for the assembling analysis. Chakrabarty [60] proposes a Constraint Graph to represent the spatial constraints for assembly sequencing. However, the geometric reasoning part is done manually. Miller [61] uses ray casting techniques to derive assembly sequences, reasoning including fasteners. Lozano [62] and Valade [63] study the physical reasoning needed to accomplish the single steps in an assembly plan using the motion planning. Toussaint [64] studies the separating motions of parts in two and three dimensions. Xu [65] studies the problem of geometric path determination required to remove a portion of the assembly contained in a cavity by generating the partial medial axis of the free space within the assembly. However, all the above approaches assume sequential disassembly of components for generating CD and assembly sequences.
Several approaches have been developed to automatically determine disassemblability of a component from assembly geometry. The disassemblable components are removed one at a time to determine a CD sequence. Woo [4] proposes an algorithm for component disassemblability by considering the faces of the component that mate with the rest of the assembly. In a later disassembly study, Dutta [66] describes identifying a sub-assembly to initiate the disassembly process based on disassemblability analysis. However, all these approaches analyze the disassemblability of the components for CD based on contact geometry, and the spatial constraints are not analyzed.

Several researchers (e.g. references [3, 34]) use a Non-Directional Blocking Graph (NDBG) that describes part interactions from the blocking nature of components. The NDBG is constructed to determine a CD sequence [14] by utilizing the concept of graph partitioning [68]. Each partition is a removable subassembly, and the above process of partitioning the graph is continued to determine a CD sequence. Some of these approaches advantageously do not use strictly contact-geometry approaches, and therefore can lead to improved analysis results. However, a study on efficient partitioning of an assembly into sub-assemblies [69] show that the problem is NP-Hard and therefore very computationally expensive. Therefore, successive study [70-72] on assembly partitioning for CD sequence generation focuses on algorithms for feasible partitioning with an objective of minimal disassembly motions.

In a later study, Schweikard [72], and Halperin [73] extend NDBG partitioning for CD with multiple translations. However, while the NDBG methods represent an advance in CD analysis, they still suffer from several of the disadvantages of prior approaches, e.g., failure to consider simultaneous component removals, impaired results where the contact geometry approaches are used, and high computational expense.
2.3 Assembly/Disassembly Planners

While some of the foregoing methods are just theoretical approaches with no implementation, others of the methods have resulted in the generation of several assembly/disassembly tools [9]. Stanford Assembly Automation Tool, STAAT [18] uses NDBG for contact-based and extended translation analysis of components. Disassembly Management System, DMS [74] generate disassembly plans based on an iterative procedure that involves an input disassembly procedure from a user, and successively checks for its correctness based on trivial collision checks. The Archimedes tool [17] determines the assembly sequences based on CD analysis and uses NDBG as the basic structure. A set of CD sequences are generated based on the extended contact-geometry analysis [75]. The user-specified constraints, while not directly considered in the disassembly analysis, are modeled as a filter so as to generate a feasible assembly plan from the generated set of complete disassembly sequences [76, 77].

It is also helpful to note that the newer “second-generation” tools (e.g. Archimedes tool) perform automated complete disassembly sequencing with little user inputs. While the earlier planners (e.g. reference [46]) perform assembly and disassembly analysis via extensive user inputs, and wherein geometric reasoning is not performed.

2.4 Summary

Most of the previous work on automated assembly and disassembly planning has focused on CD and there has been little investigation of techniques for SD. In view of the prior research, there is still a need for efficient means of determining SD sequences rather than CD sequences. Preferably, any method for determining such sequences should be computationally feasible; should allow analysis of assemblies in 3D using input geometric models rather than a continuous stream of user inputs; and should be highly automated in all the other respects as well.
CHAPTER 3. SINGLE SELECTIVE DISASSEMBLY

The Single SD problem is formulated as follows: given an assembly $A$ of $n$ components and a target component $C = \{C_x\}$, automatically determine a SD sequence $S$ to disassemble $C_x$. The motivation for single SD problem is that the disassembly analysis can be localized with respect to $C_x$ and the analysis of all the components in $A$ for disassembly may not be required. This chapter proposes a new approach called Single Wave Propagation (SWP) for single SD.

3.1 Single Wave Propagation: An Overview

A notion of disassembly wave ($\tau$) is introduced to denote disassembly ordering such that a component in one wave ($\tau_a$) is disassembled after removing its adjacent component in the next wave ($\tau_{a+1}$). In determining a SD sequence, the SWP approach defines a disassembly wave from $C_x$, with the waves propagating outwards until a removable component (defined as the boundary component $C_b$) is reached.

For example, Figure 7 illustrates the disassembly wave propagation from $C_x$. An example of the disassembly wave is illustrated in Figure 8 for $A$ with $C_x = C_{16}$. $C_{16}$ is disassemblable by removing $C_{17}$ in $\tau_1$, and $C_{17}$ is disassemblable after removing $C_{18}$ in $\tau_2$. Since $C_{18} \in \beta_1$ (set of removable components, $C_b$’s), $S = \{C_{18}, C_{17}, C_{16}\}$. Similarly, another $S = \{C_4, C_{10}, C_{16}\}$. The generation of sequences, requires 13 components to be analyzed, which is significantly less than the total number of components, $n$ ($= 42$) in $A$.

Definition 3.1: (single and multi dependent component) A $C_i$ is a $k$-dependent component, if at least $k$ ($\geq 1$) adjacent components of $C_i$ must be removed for local disassembly of $C_i$. 

The **SWP** approach is presented for the class of single dependent components (Definition 3.1). For example, in Figure 8, \( C_{16} \) is single-dependent \( (k = 1) \), since it can be disassembled locally by removing \( C_{10} \). However, for \( A \) in Figure 9, \( C_9 \) is both single and multi dependent. \( C_9 \) is single dependent with respect to \( C_8 \) and \( C_{10} \), and multi dependent with respect to \( \{C_3, C_4\} \) and...
\{C_{15}, C_{16}\}. When SWP is applied for A in Figure 9 with \( C_x = C_9 \), the approach determines \( S = \{C_{11}, C_{10}, C_9\} \), a single dependent solution.

\[
\begin{array}{cccc}
  C_1 & C_2 & C_3 & C_4 \\
  C_6 & C_7 & C_8 & C_9 \\
  C_{12} & C_{13} & C_{14} & C_{15} \\
  C_{19} & C_{20} & C_{21} & C_{22} \\
  C_{25} & C_{26} & C_{27} & C_{28} \\
\end{array}
\]

**Figure 9.** Test assembly to illustrate single and multi dependent components

The WP approach also incorporates certain types of fasteners (such as screws, bolts, nuts, nails and rivets) as providing geometric constraints for disassembly analysis. This section illustrates the WP approach in the absence of fasteners, however, the effect of incorporating the fasteners is presented elsewhere in the document.

### 3.2 Single Wave Propagation Algorithm

This section presents the SWP algorithm for single SD in which the propagation of a wave from one component to another depends on the contact geometry of the former component with respect to the latter.

#### 3.2.1 Geometric Attributes

The SWP algorithm models the WP approach using the following definitions:

- **Mating Adjacent:** A \( C_j \) in contact with \( C_i \) (\( i \neq j \)) is defined as the *mating adjacent* [78] of \( C_i \).

  The mating adjacents of \( C_i \) form a set, \( MA_i \). For example, in Figure 10, \( MA_9 = \{C_7, C_{10}, C_{12}\} \) for \( C_9 \), and \( MA_1 = \{C_2, C_8\} \) for \( C_1 \).

- **Disassembly Directions:** The \( q^{th} \) face of \( C_i \), in contact with the \( r^{th} \) face of \( C_j \), together constitute a *mating face* [79], denoted as \( M_{ij}^{qr} \) for \( C_i \) and \( M_{ji}^{rq} \) for \( C_j \). For example, in
Figure 11, faces of \( C_2 \) mating with those of \( C_1 \) are \( M_{2,1}^{1,1} \) and \( M_{2,1}^{2,2} \). For every mating face \( M_{ij}^{q,r} \), the directions along which \( C_i \) can be locally disassembled relative to \( C_j \) are represented as a set of directions \( d_{ij}^{q,r} \) on a Gaussian Sphere [80] for 3D models and represented on a Gaussian circle for 2D models. For example, Figure 11 shows the disassembly directions \( d_{2,1}^{1,1} \) and \( d_{2,1}^{2,2} \) for the mating faces \( M_{2,1}^{1,1} \) and \( M_{2,1}^{2,2} \).

**Figure 10.** Clamp Support assembly: To illustrate single SD

**Figure 11.** Mating faces and accessibility directions
• **Accessibility**: Accessibility of $C_i$ with respect to $C_j \in MA_i$ (denoted as $AC_i^j$) is the set of directions along which $C_i$ can move relative to $C_j$ and is determined for face mating as the intersection of all $d_{i,j}^{qr}$ for $C_i$ with respect to $C_j$. For example, Figure 11 shows $AC_2^1$ computed from $d_{2,1}^{1,1}$ and $d_{2,1}^{2,2}$.

• **Disassemblability**: Disassemblability of $C_i \in A$ denoted as $\Delta_i$, is a binary value that indicates if $C_i$ is removable. $\Delta_i$ is computed as the intersection of all $AC_i^j$ of $C_i$. For example, in Figure 10, $\Delta_2 = \text{TRUE}$ for $C_2$ and similarly, $\Delta_{11} = \text{FALSE}$ for $C_{11}$. Moreover, $\Delta_b = \text{TRUE}$ for a boundary component, $C_b$.

• **Removal Influence**: If $C_i \in A$ is disassemblable in the absence of a $C_j \in MA_i$, then the removal influence denoted as $RI_i^j$ is TRUE; else $RI_i^j = \text{FALSE}$. For example, in Figure 10, $RI_{11}^{12} = \text{TRUE}$, since $\Delta_{11}$ is TRUE with removal of $C_{12}$ in $A$. Similarly, $RI_9^7 = \text{FALSE}$.

### 3.2.2 Single Wave Propagation Algorithm

Let,

• $\tau_a = \text{set of components in the } a^{\text{th}} \text{ wavefront from } C_x$.

• $T = \text{Time Step, such that when } T \text{ advances from } T = a \text{ to } T = a+1, \text{ the wavefront advances from } \tau_a \text{ to } \tau_{a+1}$.

• $\beta_1 = \text{set of boundary components or removable components in } A$.

The WP is defined in terms of the geometric factors, disassemblability and removal influence, as follows:
**Definition 3.2** (τ wave propagation) For \( a = 0 \), \( \tau_a = \{ C_x \} \). For \( a > 0 \), a τ wave propagation from \( C_i \in \tau_{a-1} \) to \( C_j \in MA_i \) exists if \( \Delta_i = \text{FALSE} \), \( C_j \notin (\tau_0 \cup \tau_1 \cdots \cup \tau_{a-1}) \) and \( \text{RI}_i^j = \text{TRUE} \), then \( C_j \in \tau_a \).

A disassembly wave topologically orders \( C_i \in A \) with respect to \( C_x \). The topological ordering denotes the disassembly ordering such that \( C_i \in \tau_{a-1} \) is disassemblable by removing its adjacent component \( C_j \), if \( C_j \in \tau_a \). A disassembly wave is represented by a Removal influence Graph (RG) whose nodes correspond to components in the disassembly wave and arcs correspond to the removal influence between the mating components. Figure 12 illustrates a WP from \( C_i \) to \( C_j \), as represented by \( C_i \rightarrow C_j \) indicating that \( C_i \) is disassemblable after removing \( C_j \) (similarly for \( C_i \rightarrow C_k \)). Therefore, \( C_i \in \tau_{a-1} \) is disassemblable after removing \( C_j \) or \( C_k \); where \( C_j \), \( C_k \in \tau_a \).

![Figure 12. RG: WP representation](image)

The WP approach analyzes \( A \) beginning from \( C_x \). If \( \Delta_x = \text{TRUE} \) then \( C_x \in \beta_1 \) and can be directly disassembled; if \( \Delta_x \neq \text{TRUE} \), the WP is recursively continued until a \( C_b \) is found in a wavefront. A sequence \( \{ C_b \sim C_x \} \) is derivable from RG: where \( C_b \sim C_x \) denotes the shortest path from \( C_b \) to \( C_x \) in the graph RG (shortest path is defined in [68]).
To illustrate this, Consider A shown in Figure 10 with \( C_x = C_{11} \). At \( T = 0 \), \( \tau_0 = \{ C_{11} \} \).

Since \( \Delta_{11} = \text{FALSE} \), the WP continues. For \( C_{11} \) in \( \tau_0 : MA_{11} = \{ C_{10}, C_{12}, C_{13} \} \), \( RI_{11}^{10} = \text{TRUE} \), \( RI_{11}^{12} = \text{TRUE} \) and \( RI_{11}^{13} = \text{TRUE} \). From the definition of WP, \( \tau_1 \) must consist of \( C_i \)'s whose removal influence value is TRUE. Therefore, at \( T = 1 \), \( \tau_1 = \{ C_{10}, C_{12}, C_{13} \} \) and edges \( C_{11} \rightarrow C_{10} \), \( C_{11} \rightarrow C_{12} \) and \( C_{11} \rightarrow C_{13} \) are inserted in RG as shown in Figure 13. The edge \( C_{11} \rightarrow C_{10} \) indicates that \( C_{11} \) is disassemblable after removing \( C_{10} \). Similarly for \( C_{11} \rightarrow C_{12} \) and \( C_{11} \rightarrow C_{13} \). Since there is no \( C_b \) in \( \tau_1 \), the WP is continued.

\[ \begin{array}{c}
\text{Figure 13. RG for } C = \{ C_{11} \} : A \text{ in Figure 10}
\end{array} \]

At \( T = 2 \), \( (\tau_0 \cup \tau_1) = \{ C_{10}, C_{11}, C_{12}, C_{13} \} \). For \( C_{10} \in \tau_1 : MA_{10} = \{ C_8, C_9, C_{11}, C_{13}, C_{14} \} \) and only \( RI_{10}^8 = \text{TRUE} \). Therefore, \( C_8 \) is inserted in \( \tau_2 \) and an edge \( C_{10} \rightarrow C_8 \) is inserted in RG. Similarly the analysis for \( C_{12} \in \tau_1 \) results in the insertion of \( C_2 \) in \( \tau_2 \) and an edge \( C_{12} \rightarrow C_2 \) in RG.

\[ \begin{array}{c}
\text{Figure 14. RG for } C = \{ C_{11} \} : A \text{ in Figure 10}
\end{array} \]
For $C_{13} \in \tau_1: MA_{13} = \{C_{10}, C_{11}, C_{12}\}$. Since $C_{10}, C_{11}, C_{12} \in (\tau_0 \cup \tau_1)$, they are not analyzed for the removal influence. Therefore, $\tau_2 = \{C_2, C_8\}$. The corresponding RG at $T = 2$ is shown in Figure 14. The edge $C_{10} \rightarrow C_8$ indicates that $C_{10}$ is disassemblable after removing $C_8$, and similarly for $C_{12} \rightarrow C_2$. In RG, the two boundary components, $C_2$ and $C_8$, both belong to $\tau_2$, so there exist two sequences, $\{C_2, C_{12}, C_{11}\}$ and $\{C_8, C_{10}, C_{11}\}$ for $C_{11}$.

3.3 Discussion

The primary attributes of the WP approach are:

- **Minimal Number of Waves**: The SWP determines a solution with minimal number of waves from $C_b \in A$ to $C_x$. Therefore, within the class of single dependent component solution, the number of components in $S$ is minimal.

- **Polynomial Computation Time**: The average computational complexity in a SD sequence determination is $O(n)$ computed as follows: Let $n =$ number of components in $A$; $M =$ number of mating faces in $A$; $M_i =$ number of mating faces of $C_i$. The geometric attributes disassemblability and a removal influence computation for $C_i$ is of order $O(M_i)$. The SWP approach involves construction of an RG for $C_x$. The number of nodes in RG is of order $O(n)$, since components cannot be repeated in the waves. On an average the number of mating adjacent’s of $C_i =$ $O(1)$ [4]. Therefore, the number of edges in RG is of order $O(n)$. Since each edge in an RG involves a removal influence computation which takes $O(M_i)$ and $O(M) =$ $O(n)$ [4], the computation time for SWP algorithm is $O(n M_i) = O(M) = O(n)$.

- **Subset of components in A are analyzed**: The number of components analyzed by the WP approach ranges from 1 to $n$ depending on the position of $C_x$ in RG. If $C_x$ is closer to the boundary (in the graph), then the number of components analyzed is significantly less than $n$. 
CHAPTER 4. MULTIPLE SELECTIVE DISASSEMBLY

The multiple SD problem is formulated as follows: Given an assembly (A) of n components and target components (C), automatically determine a SD sequence (S) for m < n components, where m = Cardinality (C). This chapter analyzes the multiple SD problem and proposes a new approach called Disassembly Wave Propagation (DWP) for multiple SD.

4.1 Motivation

![Figure 15. Test assembly to illustrate multiple selective disassembly problem](image)

One potential approach to perform the multiple SD analysis is by applying the SWP algorithm for every component in C. Although this approach may determine S for individual target components with a few removals, the resultant S for C (which is an aggregation of all sequences for \( C_x \in C \)) is not necessarily an appropriate SD sequence with fewer removals. To illustrate the multiple-component SD problem, consider A in Figure 15 with the requirement to disassemble \( C = \{C_3, C_5\} \). Let \( n_r \) denote the number of components in S. For \( C = \{C_3\} \), \( S = \{C_2, C_3\} \) with \( n_r = 2 \). For \( C = \{C_5\} \), two S’s with \( n_r = 3 \) exist: \( \{C_7, C_6, C_5\} \) and \( \{C_1, C_4, C_5\} \). Aggregating these two disjoint sequences (one with \( C = \{C_3\} \) and another with \( C = \{C_5\} \)) for C
= \{C_3, C_5\} results in \(S = \{C_2, C_3, C_7, C_6, C_5\}\) and \(\{C_2, C_3, C_1, C_4, C_5\}\) with \(n_r = 5\). However, for \(C = \{C_3, C_5\}\) a better solution exists: \(S = \{C_1, C_4, C_3, C_5\}\) with \(n_r = 4\). Therefore, a better solution may be obtained if two or more components are disassembled along a common sequence. This motivates the need for multiple SD analysis.

4.2 Disassembly Wave Propagation Approach: An Overview

For the SD of multiple components a new approach called Disassembly Wave Propagation (DWP) is proposed for the class of single dependent components (defined in Chapter 3.1). The DWP approach involves topologically ordering the components in \(A\) with respect to \(C\) (defined as \(\tau\) waves) and also with respect to the boundary of \(A\) (defined as \(\beta\) waves), and then evaluating \(S\) based on the intersection event (IE) between these waves.

A notion of disassembly wave is introduced to denote the disassembly ordering such that a component in one wave is disassembled after removing its adjacent component in the next wave. The importance of IE between the waves lies in the determination of the component at which the waves intersect. Therefore the shape of the wave in the geometry space is irrelevant; the wave merely provides the topological ordering of the components in \(A\) with respect to either \(C\) or the boundary of \(A\).

![Diagram](image)

**Figure 16.** \(C = \{C_3, C_5\}\): \(\tau\) wave at \(T=1\) and its representation
To illustrate the DWP approach, consider the test assembly shown in Figure 15 with \( C = \{C_3, C_5\} \). A \( \tau \) wave of \( C_x \in C \) topologically orders \( C_i \in A \) with respect to \( C \) and determines the disassembly ordering for \( C_x \). Let \( \tau^i \) denote the \( \tau \) wave of \( C_x \) that is \( i \) units away from \( C_x \). At every time step \( T \), the \( \tau \) wave advances from \( \tau^{T-1} \) to \( \tau^T \). For example, Figure 16 shows the \( \tau \) wave of \( \{C_3, C_5\} \) at time step \( T = 1 \) and its graphical representation. From the \( \tau \) wave for \( C_3 \): \( C_3 \in \tau_0^3 \) is disassemblable either after removing \( C_4 \in \tau_1^3 \) or \( C_2 \in \tau_1^3 \). Similarly from the \( \tau \) wave for \( C_5 \): \( C_5 \in \tau_0^5 \) is disassemblable either after removing \( C_4 \in \tau_1^5 \) or \( C_6 \in \tau_1^5 \).

![Figure 16. C = \{C_3, C_5\}: \tau \) waves and its representation](image)

A \( \beta \) wave determines the minimum number of components to be removed to disassemble \( C_i \in A \). Let \( \beta_i \) denote the \( \beta \) wave that is \( i \) units away from the boundary of \( A \). For example, Figure 17 shows the \( \beta \) wave for \( A \) in Figure 15 and its graphical representation. \( C_1, C_2, C_7 \) and \( C_9 \) are disassemblable components and form \( \beta_1 \). \( C_3, C_4, C_6 \) and \( C_8 \) form \( \beta_2 \), since they are disassemblable after removing \( C_2, C_1, C_7 \) and \( C_9 \), respectively. From the \( \beta \) wave, the minimum number of removals for \( C_4 \in \beta_2 \) is 2 and \( C_5 \in \beta_3 \) is 3.

![Figure 17. C = \{C_3, C_5\}: \beta \) waves and its representation](image)
Every IE between τ and the β waves determine a sequence for one or more target components. For example, Figure 18 shows the intersections of τ and β waves at T = 1. An IE occurs at C₄: \( \tau^1 \cap \tau^5 \cap \beta^2 \) which determines \( S = \{C_1, C_4, C_3, C_5\} \) for \( C = \{C_3, C_5\} \). Similarly, all other IE’s and the corresponding sequences are determined at every time step to evaluate an appropriate SD sequence \( S \). The above method of evaluating all IE’s for every time step is called Multiple Wave Propagation Algorithm and it is feasible for SD of \( m << n \) target components, since the maximum number of such IE’s is \( O(2^m n) \), as discussed later in this chapter.

However, for larger values of \( m \), a method called Priority Intersection Event Algorithm is proposed. The algorithm involves prioritizing IE’s based on the order of event occurrence to evaluate only the candidate events in evaluating a minimal removal \( S \). For example, IE shown in Figure 18 is a priority event, since at \( T = 1 \) the IE for \( \{C_3, C_5\} \) occurs before that of the IE’s for \( C_3 \) and \( C_5 \) that determine disjoint sequences, i.e., \( n_r = 4 \) for \( S = \{C_1, C_4, C_3, C_5\} \) determined by the IE at \( C_4 \), which is less than \( n_r = 5 \) for that of disjoint sequence \( \{C_2, C_3, C_7, C_6, C_5\} \). The following sections present the DWP approach and the algorithms in detail.

**Figure 18. IE at C₄:** \( S = \{C_1, C_4, C_3, C_5\} \) for A in Figure 15, \( C = \{C_3, C_5\} \)
4.3 Geometric Attributes for Topological Disassembly Ordering

The DWP approach defines a disassembly wave to topologically arrange $C_i \in A$, denoting the disassembly order such that a component in one wave is disassembled after removing its adjacent component in the next wave. The propagation of a wave from one component to another depends on the contact geometry of the former component with respect to the latter. The geometric attributes (accessibility, disassemblability and removal influence) used in defining the disassembly waves for DWP are defined in Chapter 3.2.1.

![Solid model of Machine Vice assembly: geometric attributes](image)

<table>
<thead>
<tr>
<th>Geometric Attributes (e.g.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_1^2 \leftarrow AC_4^2$</td>
</tr>
<tr>
<td>$AC_2^4 \leftarrow AC_7^3$</td>
</tr>
<tr>
<td>$\Delta_1 = \text{TRUE}$</td>
</tr>
<tr>
<td>$\Delta_2 = \text{FALSE}$</td>
</tr>
<tr>
<td>$RI_2^2 = \text{TRUE}$</td>
</tr>
</tbody>
</table>

**Figure 19.** Solid model of Machine Vice assembly: geometric attributes

An example, illustrating the geometric attributes is shown in Figure 19. An important assumption in the current research is that the linear direction of disassembly for components is considered for the automation analysis. For example in Figure 19, while $C_2$ having a threaded contact with $C_4$ must be disassembled in a helical motion, its net disassembly direction is linear (shown as $AC_2^4$), which is the disassembly direction assumed for thread mating. $AC_4^2 = \text{NULL}$, i.e., $C_2$ must be removed prior to disassembling $C_4$. In Figure 19, $\Delta_1 = \text{TRUE}$ for $C_1$ and $\Delta_2 = \text{FALSE}$ for $C_2$. Moreover, $RI_2^2 = \text{TRUE}$, since $\Delta_4$ is TRUE with the removal of $C_2$ in $A$. Furthermore, $RI_5^7 = \text{FALSE}$, since $\Delta_5$ is FALSE with the removal of $C_7$ in $A$. 
4.4 Disassembly Waves

The DWP approach defines two types of disassembly waves:

- $\tau$ waves from $C$, which propagate outwards.

- $\beta$ waves from the boundary of $A$ (the enclosing region of $A$ that includes zero components), which propagate inwards.

Let the $\tau$ wave of $C_x \in C$ be denoted as $\tau_x$ and $\tau_x^a$ = the set of components in $\tau_x$ which are $a$ ($\geq 0$) units away from $C_x$. Then the propagation of $\tau_x$ from $\tau_x^{a-1}$ to $\tau_x^a$ is defined as follows:

**Definition 4.1** ($\tau$ wave propagation) For $a = 0$, $\tau_x^0 = \{C_x\}$. For $a > 0$, a $\tau$ wave propagation from $C_i \in \tau_x^{a-1}$ to $C_j \in MA_i$ exists if $\Delta_i = FALSE$, $C_j \notin (\tau_x^0 \cup \tau_x^1 \ldots \cup \tau_x^{a-1})$ and $RI_i^j = TRUE$, then $C_j \in \tau_x^a$.

Let $\beta_a$ = the set of components in the $\beta$ wave which are $a$ ($\geq 0$) units away from the boundary of $A$. Then the propagation of $\beta$ wave from $\beta_{a-1}$ to $\beta_a$ is defined as follows:

**Definition 4.2** ($\beta$ wave propagation) For $a = 0$, $\beta_a = \{}$. For $a = 1$, $\beta_a$ = set of all $C_b \in A$. For $a > 1$, $\beta$ wave propagation from $C_j \in \beta_{a-1}$ to $C_i \in MA_j$ exists if $\Delta_i = FALSE$, $C_i \notin (\beta_0 \cup \beta_1 \ldots \cup \beta_{a-1})$ and $RI_i^j = TRUE$, then $C_i \in \beta_a$.

A disassembly wave is represented by a Removal influence Graph (RG) whose nodes correspond to the components in the disassembly wave and the arcs correspond to the removal
influence between the components. Figure 20 illustrates a $\tau$ wave of $C_x$ from $\tau^x_{a-1}$ to $\tau^x_a$, where $C_i \in \tau^x_{a-1}, C_j \in \tau^x_a, \Delta_i = \text{FALSE}$ and $RI^j_i = \text{TRUE}$. The $\tau$ wave from $C_i$ to $C_j$, represented as $C_i \rightarrow C_j$, implies that $C_i$ is disassemblable after removing $C_j$. Similarly, Figure 20 shows a $\beta$ wave from $\beta_{a-1}$ to $\beta_a$, where $C_j \in \beta_{a-1}$ to $C_i \in \beta_a, \Delta_i = \text{FALSE}$ and $RI^j_i = \text{TRUE}$. The $\beta$ wave from $C_j \in \beta_{a-1}$ to $C_i \in \beta_a$ is represented as $C_i \rightarrow C_j$ denoting that $C_i$ is disassemblable after disassembling $C_j$ (the reason for having reverse logic of the arrow for the $\beta$ wave in $RG$ is to maintain consistency in disassembly ordering).

![Diagram of wave propagation](image)

**Figure 20. $\tau$ wave propagation and $\beta$ wave propagation**

A $\tau$ wave of $C_x \in C$ (denoted as $\tau^x$) topologically orders $C_i \in A$ with respect to $C_x$ and determines the disassembly ordering for $C_x$. For example, Figure 21 illustrates $\tau$ wave propagation of $C_5$ ($\tau^5$) for the Machine Vice assembly shown in Figure 19. $\tau^5$ propagates from $\tau^5_0$ to $\tau^5_1$ and then propagates from $\tau^5_1$ to $\tau^5_2$; where $C_5 \in \tau^5_0, C_2 \in \tau^5_1, C_1 \in \tau^5_2, \Delta_5 = \text{FALSE}, RI^5_2 = \text{TRUE}, \Delta_2 = \text{FALSE}$ and $RI^1_2 = \text{TRUE}$. The $\tau$ wave propagation from $C_5$ to $C_2$ implies that $C_5$ is disassemblable after removing $C_2$. Similarly, the $\tau$ wave propagation from $C_2$ to $C_1$ implies that $C_2$ is disassemblable after removing $C_1$. 
A $\beta$ wave determines the minimum number of components to be removed to disassemble $C_i \in A$. A $C_i \in \beta_a$ implies that the minimum number of components to be removed to disassemble $C_i$ is $a$. For example, Figure 22 shows $\beta$ wave propagation for the Machine Vice assembly shown in Figure 19. $C_1 \in \beta_1$, $C_2 \in \beta_2$, $\Delta_1 = \text{TRUE}$, $\Delta_2 = \text{FALSE}$ and $RI_2^1 = \text{TRUE}$. Minimal number of removals is 2 for $C_2 \in \beta_2$ and 1 for $C_1 \in \beta_1$.

### Figure 22. RG showing a part of $\beta_2$ wave for $A$ in Figure 19

#### 4.5 Intersection Event and Sequence Determination

The intersection of $\tau$ and $\beta$ waves (denoted as an Intersection Event, IE) determines $S$ and it is defined as follows:

**Definition 4.3 (Intersection Event):** IE is an Intersection of any $k$ ($1 \leq k \leq m$) $\tau$ wave(s) ($\tau^{x_1}$, $\tau^{x_2}$, $\ldots$, $\tau^{x_k}$; where $C_{x_1}$, $C_{x_2}$, $\ldots$, $C_{x_k} \in A$) and a $\beta$ wave at $C_w \in A$ (implies that $C_w \in \tau^{x_i}$, $\tau^{x_2}$, $\ldots$, $\tau^{x_k}$, $\beta$).
An IE is defined to determine both the disjoint (at \( k = 1 \)) and common sequences (at \( k > 1 \)) between target components; where \( k \leq m \). Every occurrence of an IE for \( k > 0 \) \( \tau \) wave(s) \((\tau^{x1}, \tau^{x2}, ..., \tau^{xk})\) determines \( S = \{C_b \leadsto C_w, C_w \leadsto C_{x1}, C_w \leadsto C_{x2}, ..., C_w \leadsto C_{xk}\}\) for \( C' = \{C_{x1}, C_{x2}, ..., C_{xk}\}\); where \( C' \subseteq C \) (where \( \subseteq \) refers to proper subset) and \( C_w \in A \). The importance of the intersection between the waves lies in the determination of the component at
which the waves intersect. Figure 23 shows a conceptual diagram of A and illustrates an IE for $C = \{C_x, C_y\}$ and the corresponding $S = \{C_b \leadsto C_w, C_w \leadsto C_x, C_w \leadsto C_y\}$.

For example, Figure 24 shows the RG for $C = \{C_4, C_5\}$ of the Machine Vice assembly in Figure 19. An IE occurs at $C_2 (\tau_2 \cap \tau_4 \cap \beta_2)$ with $k = 2$ which determines $S = \{C_1, C_2, C_4, C_5\}$ with $n_r = 4$ for $C = \{C_4, C_5\}$.

Based on the DWP approach, two algorithms are proposed for multiple SD. For $m \ll n$ target components Multiple Wave Propagation (MWP) Algorithm is presented. MWP defines time-based IE’s between disassembly waves and determines $S$ based on the order of event occurrence. However, for $m < n$ target components Priority Intersection Event (PIE) Algorithm is presented. PIE defines polynomial number of IE’s that are necessary candidate events in determining an appropriate $S$. Both algorithms determine the sequences of fewer component removals in a feasible computation time.

### 4.6 Multiple Wave Propagation Algorithm

For SD of $m$ components there are $m$ $\tau$ waves and one $\beta$ wave. Let $T$ denotes the time step. At $T = 0$, $\tau^{xk}_0 = \{C_{xk}\}$ for $C_{xk} \in C (k = 1, m)$ and the $\beta$ wave propagation is completed for all $C_i \in A$. For every time step (from $T = a$ to $T = a+1; a \geq 0$), the $\tau^{xk}$ propagates by one wave; i.e., at time step $T = a-1$, $\tau^{xk} = (\tau^{xk}_0 \cup \tau^{xk}_1 \ldots \cup \tau^{xk}_{a-1})$ and at $T = a$, $\tau^{xk} = (\tau^{xk}_0 \cup \tau^{xk}_1 \ldots \cup \tau^{xk}_a)$. For every time step $T$, IE’s are determined between $\tau$ waves and $\beta$ wave and the corresponding sequences are evaluated. At $T = 0$, disjoint sequences for every $C_{xk} \in C$ are determined. At $T > 0$, common sequences for $C' \subseteq C$ are determined. The evaluated sequences are then processed based on the order of event occurrence, i.e., by comparing every $S$ with existing sequences for $C$ based on minimal $n_r$. 
To illustrate the MWP Algorithm, consider the test assembly shown in Figure 15 with \( C = \{C_3, C_5\} \). The time-based intersection events and the wave propagation are illustrated in Table 1. The sequence in the last column of the table shows the \( S \) for \( C \) at \( T=2 \). For this example, the total number of IE’s is only 4, which can be computed in polynomial time.

<table>
<thead>
<tr>
<th>( T )</th>
<th>( \tau_T )</th>
<th>( \tau_T' )</th>
<th>IE</th>
<th>S for ( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C_3</td>
<td>C_5</td>
<td>( (\tau_0 \cap \beta_2), (\tau_0 \cap \beta_3) )</td>
<td>{C_2, C_3, C_7, C_6, C_5}</td>
</tr>
<tr>
<td>1</td>
<td>C_2, C_4</td>
<td>C_4, C_6</td>
<td>( (\tau_1 \cap \tau_1 \cap \beta_2) )</td>
<td>{C_1, C_4, C_5}</td>
</tr>
<tr>
<td>2</td>
<td>C_1</td>
<td>C_1, C_7</td>
<td>( (\tau_2 \cap \tau_2 \cap \beta_1) )</td>
<td>{C_1, C_4, C_5}</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>( S = {C_1, C_4, C_3, C_5} )</td>
</tr>
</tbody>
</table>

Table 1. Illustration of the MWP algorithm for \( A \) in Figure 15 with \( C = \{C_3, C_5\} \)

The number of IE’s for \( A \) depends on the geometric configuration of \( A \) and \( C \). However, the maximum number of IE’s for \( A \) can be exponential, i.e., \( O(2^m.n) \). In the MWP algorithm there are \( m \) \( \tau \) waves and IE occur for \( C' \subseteq C \). Therefore, the computation time to determine all the IE’s that occur at every \( C_i \in A \) is of order \( O(2^m) \). This is due to the fact that all the possible combinations of \( \tau \) waves intersecting \( C_i \in A \) must be checked for the occurrence of IE. Moreover, since \( T = O(n) \), the maximum number of IE’s is \( O(2^m.n) \).

For example, consider the Wheel Support assembly in Figure 25 with \( C = \{C_5, C_6, C_7\} \). The RG at \( T = 1 \) is shown in Figure 26. At \( T = 1 \): \( \tau_5, \tau_6 \) and \( \tau_7 \) waves from \( C_5, C_6 \), and \( C_7 \), respectively, intersect \( \beta_1 \) at \( C_4 \), i.e., \( (\tau_5 \cap \tau_6 \cap \beta_1), (\tau_5 \cap \tau_7 \cap \beta_1), (\tau_6 \cap \tau_7 \cap \beta_1) \), and \( (\tau_5 \cap \tau_6 \cap \tau_7 \cap \beta_1) \). Thus the number of checks necessary to determine all the IE’s at \( C_4 \) is \( O(2^m) \).

For every IE, \( S \) is computed in \( O(n) \) time. Therefore, the computational complexity of the MWP algorithm is \( O(2^m.n^2) \). For a smaller number of target components \( m \ll n \), the MWP
algorithm is computationally efficient as compared to the enumeration approach, which is of higher-order exponential (typically with respect to $n$).

![Figure 25. Wheel Support assembly](image)

**Figure 25. Wheel Support assembly**

![Figure 26. RG at $T = 1$ for $C = \{C_5, C_6, C_7\}$](image)

**Figure 26. RG at $T = 1$ for $C = \{C_5, C_6, C_7\}$**

### 4.7 Priority Intersection Event Algorithm

The Priority Intersection Event (PIE) algorithm is proposed for SD of $m < n$ target components. PIE modifies the MWP algorithm in the definition and determination of IE. The PIE algorithm prioritizes the IE’s and determines only the candidate events (denoted as $\phi$ events) in determining $S$ with fewer component removals.
Let at time step $T, \beta^k = \beta_1 \cup \beta_2 \ldots \cup \beta_{k \cdot \eta}; 1 \leq k \leq m$. For example, with $m = 2$, at $T = 0: \beta_0 = \{\}, \beta^1 = \beta_1 = (\beta_0)$ and at $T = 1: \beta^1 = (\beta_1), \beta^2 = (\beta_1 \cup \beta_2)$. The definition of $\phi$ events is as follows:

**Definition 4.4 ($\phi$ events):** Let

- $\Psi = \text{Set of all } \tau \text{ waves of } C' \subseteq C \text{ intersecting } C_w \in A \text{ at } T > 0$.
- $\mu = \text{Cardinality}(\Psi)$.
- $\Psi' \subseteq \Psi$, Every $\tau^x \in \Psi'$ has not been intersected by a $\beta^1$ wave or $\tau^x_T$ intersects $\beta_T$ at $T > 0$.
- $\nu = \text{Cardinality}(\Psi')$.

Then,

1. $\phi_1 \text{ event}: \text{intersection of a } \tau \text{ wave with } \beta \text{ wave, where } T = 0$.
2. $\phi_2 \text{ event}: \text{intersection of all } \nu (> 1) \tau \text{ waves of } \Psi' \text{ with a } \beta^\nu \text{ wave at } C_w, \text{ where } T > 0$.
3. $\phi_3 \text{ event}: \text{intersection of all } \mu (> 1) \tau \text{ waves of } \Psi \text{ with a } \beta^\mu \text{ wave at } C_w \text{ (where } T > 0) \text{ such that the total number of waves from } (C' \rightarrow C_w \rightarrow C_b) \text{ is less than that of } (C' \rightarrow C_b \text{'s}).$

![Diagram](image-url)

**Figure 27.** RG at $T = 0$ for $C = \{C_5, C_6, C_7\}$: A in Figure 25

A $\phi_1$ event for $\tau^{x_k}$ determines a few component removals $S = \{C_{b \rightarrow C_{x_k}}\}$ for $C_{x_k} \in C$.

To illustrate, consider RG for the Wheel Support assembly shown in Figure 27 at $T = 0$. 

![Image](image-url)
The \( \phi_1 \) events occur for \( \tau^5, \tau^6 \) and \( \tau^7: (\tau^5 \cap \beta_2) \) at \( C_5 \), \( (\tau^6 \cap \beta_2) \) at \( C_6 \) and \( (\tau^7 \cap \beta_2) \) at \( C_7 \), respectively. Therefore, \( S = \{C_4, C_5\} \) for \( C_5, S = \{C_4, C_6\} \) for \( C_6 \) and \( S = \{C_4, C_7\} \) for \( C_7 \), each with \( n_r = 2 \). Moreover, the \( \phi_1 \) event for \( \tau_x^k \) is better than other IE’s for \( C_{x_k} \in C \).

A \( \phi_2 \) event for \( v (>1) \) \( \tau \) waves: \( \tau_x^1, \tau_x^2, \ldots, \tau_x^v \in \Psi' \) determines \( S = \{C_{b_1} \sim C_{w}, C_{w} \sim C_{x_1}, C_{w} \sim C_{x_2}, \ldots, C_{w} \sim C_{x_v}\} \) for \( C' = \{C_{x_1}, C_{x_2}, \ldots, C_{x_v}\} \subseteq C \). The \( n_r \) for \( S \) from a \( \phi_2 \) event is fewer than that of the \( S \) determined for \( C_{x_1}, C_{x_2}, \ldots, C_{x_m} \) from \( \phi_1 \) events. To illustrate this, consider the conceptual assembly shown in Figure 28 with \( C = \{C_x, C_y\} \). At \( T = a \), the \( \beta_1 \) wave has not intersected \( \tau^x \) and \( \tau^y \), therefore the \( S \) for \( \{C_x, C_y\} \) will have \( n_r \geq 4a \) (2a components for \( C_{b_1} \) to \( C_x \), and 2a components for \( C_{b_2} \) to \( C_y \)). The intersection event \( \tau^x \cap \tau^y \cap \beta_2 \) at \( C_w \), determines \( S = \{C_{b_1} \sim C_{w}, C_{w} \sim C_{x}, C_{w} \sim C_{y}\} \) with \( n_r \leq 4a \). Therefore, \( \phi_2 \) event is better than the corresponding \( \phi_1 \) events. For example, from the RG shown in Figure 29, \( S = \{C_1, C_4, C_3, C_5\} \) with \( n_r = 4 \) available from a \( \phi_2 \) event for \( C = \{C_3, C_5\} \) (\( C_w = C_4 \)) is better than \( S = \{C_2, C_3, C_7, C_6, C_5\} \) with \( n_r = 5 \) available from the \( \phi_1 \) events for \( \{C_3, C_5\} \).

**Figure 28.** \( \phi_2 \) event at \( C_w \): \( S \) from \( \phi_2 \) event is better than \( S \) from \( \phi_1 \) events.

T = a
C = \{C_x, C_y\}
a = Number of components
\( \phi_2 \) at \( C_w \): \( \tau_x \cap \tau_y \cap \beta_2 \)
Figure 29. RG at $T = 1$ for $C = \{C_3, C_5\}$ for $A$ in Figure 15: $\phi_2$ event $\tau^3 \cap \tau^5 \cap \beta^2$ at $C_4$

Moreover, the $\phi_2$ event that occurs for $C' \subseteq C$ at $C_w$ is better than other IEs at $C_w$ that occur for $C'' \subset C'$ at the same time step $T$. To illustrate this, consider the RG for the Wheel Support assembly shown in Figure 26. Clearly, the $S$ available for $C' = \{C_5, C_6, C_7\}$ from a $\phi_2$ event $\tau^5 \cap \tau^6 \cap \tau^7 \cap \beta^3$ at $C_4$ is better than the other events at $C_4: \tau^5 \cap \tau^6 \cap \beta^2$ for $\{C_5, C_6\}$, $\tau^5 \cap \tau^7 \cap \beta^2$ for $\{C_5, C_7\}$ and $\tau^6 \cap \tau^7 \cap \beta^2$ for $\{C_6, C_7\}$.

A $\phi_3$ event determines the sequence $\{C_{b-w}, C_{w-x_1}, C_{w-x_2}, \ldots, C_{w-x_m}\}$ for $C' = \{C_{x_1}, C_{x_2}, \ldots, C_{x_m}\} \subseteq C$. The sequence evaluated from a $\phi_3$ event with minimum $n_r$ compared to $\phi_1$ events of $C'$ is a candidate event for an appropriate $S$ with fewer component removals. To illustrate this, consider the RG shown in Figure 31 for $A$ in Figure 30 with $C = \{C_3, C_4\}$. At $T = 2$, $\phi_3$ event occurs at $C_2$: the intersection of $\tau^3, \tau^4$ and $\beta^2$. Clearly, $S = \{C_1, C_2, C_3, C_4\}$ with $n_r = 4$ available from this $\phi_3$ event is better than $S = \{C_1, C_2, C_3, C_6, C_5, C_4\}$ with $n_r = 6$ available from $\phi_1$ events.
Figure 30. Solid model of Toy Assembly (exploded view) to illustrate $\phi_3$ event

Figure 31. RG at $T = 2$ for $C = \{C_3, C_4\}$: $A$ in Figure 30

<table>
<thead>
<tr>
<th>$T$</th>
<th>$C'$</th>
<th>$C_w$</th>
<th>$\phi$ events</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$C_5$</td>
<td>$C_5$</td>
<td>$\phi_1: (\tau^5_0 \cap \beta_2)$</td>
</tr>
<tr>
<td></td>
<td>$C_6$</td>
<td>$C_6$</td>
<td>$\phi_1: (\tau^6_0 \cap \beta_2)$</td>
</tr>
<tr>
<td></td>
<td>$C_7$</td>
<td>$C_7$</td>
<td>$\phi_1: (\tau^7_0 \cap \beta_2)$</td>
</tr>
<tr>
<td></td>
<td>$C_5, C_6, C_7$</td>
<td>$C_4$</td>
<td>$\phi_2: (\tau^5 \cap \tau^6 \cap \tau^7 \cap \beta_3)$</td>
</tr>
<tr>
<td>1</td>
<td>$C_5, C_6$</td>
<td>$C_5$</td>
<td>$\phi_2: (\tau^5 \cap \tau^6 \cap \beta_2)$</td>
</tr>
<tr>
<td></td>
<td>$C_5, C_6, C_7$</td>
<td>$C_6$</td>
<td>$\phi_2: (\tau^5 \cap \tau^6 \cap \tau^7 \cap \beta_3)$</td>
</tr>
<tr>
<td></td>
<td>$C_6, C_7$</td>
<td>$C_7$</td>
<td>$\phi_2: (\tau^6 \cap \tau^7 \cap \beta_3)$</td>
</tr>
</tbody>
</table>

Table 2. PIE algorithm for $A$ in Figure 25 with $C = \{C_5, C_6, C_7\}$: $\phi$ events at $T = 0, 1$. 
Based on the above argument, $\phi$ events are necessary candidate events for an appropriate SD sequence with fewer component removals and include locally best events (events that are found to yield fewer component removals at time $T$, for every $C_w$). For example, the $\phi$ events that are evaluated by the PIE algorithm for $A$ in Figure 25 with $C = \{C_5, C_6, C_7\}$ are shown in Table 2. These $\phi$ events can be determined in polynomial time.

In general, the total number of $\phi$ events for $A$ is $O(mn)$; i.e., polynomial, described as follows: From the definition of $\phi_1$ event (Definition 4.4), there can be only one $\phi_1$ event for every $C_x \in C$. Therefore, the number of $\phi_1$ events is $O(m)$. From the definitions of $\phi_2$ and $\phi_3$ events (Definition 4.4), there can be only one $\phi_2$ and one $\phi_3$ event for every $C_w \in \tau$ wave. From the definition of $\tau$ wave (Definition 4.1), a $C_w$ can appear only once in a wavefront. Therefore for $m$ $\tau$ waves the number of $\phi_2$ and $\phi_3$ events is $O(mn)$. Moreover, for every $\phi$ event, $S$ is computed in $O(n)$ time. Hence, the PIE algorithm computes the sequences for $O(mn)\phi$ events in $O(mn^2)$.

4.8 Discussion

Some of the primary attributes of MWP algorithm are:

- An IE for $k=1$ determines a sequence for $C_{x1} \in C$. However, for $k>1$, IE determines a common sequence to disassemble all the $k<=m$ target components $C_{y1}, C_{y2}, \ldots, C_{yk} \in C$.

- IE occurrence depends on the geometric configuration of $A$, i.e., if in $A$, there is no IE of $k$ ($>0$) $\tau$ wave(s) ($\tau^{y1}, \tau^{y2}, \ldots, \tau^{yk}$) with a $\beta$ wave at $C_w \in A$, then there exist no common sequence $S$ for $\{C_{y1}, C_{y2}, \ldots, C_{yk}\}$ of type $\{C_b \bowtie C_w, C_w \bowtie C_{y1}, \ldots, C_w \bowtie C_{yk}\}$.

- The computational complexity of the MWP approach is of order $O(n^2 2^m)$. For $m << n$, this algorithm is computationally feasible, and is efficient compared to the enumeration approach.
Some of the primary attributes of PIE algorithm are:

- The priority events are necessary candidate events in determining $S$ with fewer component removals and number of such events is polynomial in number.

- The order of event occurrence in $A$ depends on the geometric configuration of components in $A$ and every $IE$ for $k (> 0)$ $\tau$ wave(s) ($\tau^{y_1}, \tau^{y_2}, \ldots, \tau^{y_k}$), with a $\beta^k$ wave, determines $S$ for $C = \{C_{y_1}, C_{y_2}, \ldots, C_{y_k}\} \subseteq C$.

- The computational complexity of PIE algorithm is $O(mn^2)$. 
CHAPTER 5. GLOBAL SELECTIVE DISASSEMBLY

The global SD problem is formulated as follows: Given an assembly (A) of n components and target components (C) automatically determine a non-interfering (collision free) SD sequence (S). This chapter presents an algorithm called Global Selective Disassembly (GSD) which analyzes the geometric constraints imposed on the components for global SD.

5.1 Problem Analysis: Geometric Constraints and Automation

The geometric constraints in this research are of two types:

- **Spatial Constraints**: Constraints imposed in assembling or disassembling of a component due to the spatial position and geometry of all other components in A.
- **User-Defined Constraints**: Constraints imposed by the user on the component geometry that restricts some assembly/disassembly operations. User-defined constraints include component grouping (two or more components are grouped as a sub-assembly) and directional constraints (one or more possible assembly or disassembly directions for the components are constrained). For example, a user may wish to group a set of components based on the functionality of the sub-assembly. As another example, the user may wish to impose directional constraints to specify practical restrictions imposed in disassembling due to fixtures, the disassembly workspace, etc. To illustrate, when considering how to disassemble an automotive engine mounted within a car chassis, generally the user will want to restrict his or her analysis primarily to component removal directions that will allow them to be removed from the open hood of the car, rather than through its firewall, side panels or from the bottom of the car.
Spatial constraints on $C_i \in A$ refer not only to the constraints that result due to the mating components of $C_i$ but also due to the components that may collide with $C_i$ in different directions. If only the contact-geometry for SD analysis is considered, interference arising during the linear global assembly/disassembly motions will not be detected. To illustrate this, consider $A$ shown in Figure 32 with $C = \{C_3\}$. Let the disassembly direction be denoted as $d_j$. $C_3$ is in contact with $C_4$ (along the cylindrical surface). Performing the assembly/disassembly analysis based on the contact-geometry alone will result in a solution saying that $C_3$ is disassemblable along $d_3$ and $-d_3$. However, $C_3$ is disassemblable locally (infinitesimal translation is possible along $d_3$ and $-d_3$), not globally, i.e., $C_3$ is spatially constrained by $\{C_4, C_5\}$ along $d_3$ and by $\{C_1, C_2\}$ along $-d_3$. Therefore, SD analysis based on the spatial constraints becomes necessary to ensure the global assembly/disassembly of components.
The **GSD** algorithm models both the spatial constraints and the user-defined constraints as the geometric constraints to components, which were free to move in any disassembly directions in the absence of constraints. The algorithm analyzes the geometric constraints in determining the accessibility of components, which is followed by determining the topological disassembly ordering of the components to evaluate a **SD** sequence.

The subsequent sections present the *Global Selective Disassembly (GSD)* algorithm in detail. First the geometric attributes related to the algorithm are defined, next the **GSD** algorithm accounting for spatial constraints is presented, and then the modifications to the algorithm to incorporate user-defined constraints are discussed. The proposed algorithm is applicable for both 2D and 3D assembly geometry.

### 5.2 Geometric Attributes for Global Selective Disassembly

![Figure 33. Test assembly to illustrate the GSD algorithm](image)

The current research assumes a fixed set of disassembly directions, since determining spatial constraints is a time-consuming procedure [62]. Let **U** denote the universe of disassembly directions for all the components in **A** and **U_i** denote a set of disassembly directions of **C_i** after accounting for the directional constraints imposed on **C_i**. For example, in Figure 33, **U** = \{**d_1**, **d_2**\}. With no directional constraints applied to any other components **U_1** = **U**, i.e., **U_1** = **U_2"
= U_3 = U_4 = U_5 = \{d_1, d_2\}. However if, for example, C_1 is constrained to move and C_2 is constrained along d_1, then U_1 = \{\}, U_2 = \{d_2\}, U_3 = U_4 = U_5 = \{d_1, d_2\}.

- **Definition 5.1: (Spatial Accessibility)** The spatial accessibility of C_i, denoted as AC_i^j is the set of directions in U_4 along which C_i does not collide with C_j ∈ A (j ≠ i). For example in Figure 33: AC_5^2 = NULL and AC_2^5 = \{d_1\}.

- **Definition 5.2: (Spatial Disassemblability)** The spatial disassemblability of C_i, denoted as Δ_i, is TRUE if there exists a disassembly direction d_j ∈ U_1 along which C_i does not collide with any other C_k ∈ A (k ≠ i). A spatially disassemblable component is denoted as C_b. For example in Figure 33: Δ_2 = TRUE and Δ_5 = FALSE for C_2 and C_5, respectively.

- **Definition 5.3: (Spatial Boundary Set)** The spatial boundary set, denoted as β_r (r > 0), is defined as follows: For r = 1, β_r = Set of C_b’s in A. For r > 1, β_r = Set of C_b’s in the assembly with components A – (β_{r-1} + β_{r-2} … + β_1). For example, the spatial boundary sets in Figure 33 are β_1 = \{C_2, C_4\}, β_2 = \{C_5\}, β_3 = \{C_3\} and β_4 = \{C_1\}.

- **Definition 5.4: (Spatial Adjacent Set)** The spatial adjacent set of C_i ∈ β_r (denoted as α_i) is defined as α_i = (β_{r-1} ∪ β_{r-2} … ∪ β_1). For example in Figure 33, C_3 ∈ β_3 and α_3 = \{C_2, C_4, C_5\}. Similarly for C_5 ∈ β_2 and α_5 = \{C_2, C_4\}.

- **Definition 5.5: (Spatial Removal Influence)** Let C’ denote a sub-set of components in A. If in the absence of C’ ⊆ α_i in A, Δ_i = TRUE, then the spatial removal influence of C’ on C_i denoted as RI_i^{C’} = TRUE. For example, in Figure 33, the spatial removal influence of (C_4, C_5) on C_3 is RI_3^{4,5} = TRUE.
5.3 Algorithm

The GSD algorithm involves the following steps:

**Determination of Spatial Constraints:**

The spatial accessibility of every $C_i \in A$ with respect to every other $C_j \in A$ ($j \neq i$) are computed and the results are used to generate the spatial constraints for the components. For example, in Figure 33, $U_3 = \{d_1, d_2\}$, $AC_3^1 = \{d_1\}$, $AC_3^2 = \{d_1\}$, $AC_3^4 = \{d_2\}$ and $AC_3^5 = \text{NULL}$. Therefore, $C_3$ is spatially constrained along $d_1$ by $\{C_4, C_5\}$ and along $d_2$ by $\{C_1, C_2, C_5\}$. Similarly, the spatial constraints for other components are computed.

**Determination of Spatial Boundary Sets and Spatial Adjacent Sets:**

Based on the spatial accessibility computation, the spatial disassemblability of $C_i \in A$ and also the spatial boundary sets are computed using the Definitions 5.2 and 5.3. For example, in Figure 33, $\Delta_3 = \text{FALSE}$, since $C_3$ is constrained along $d_1$ by $\{C_4, C_5\}$ and along $d_2$ by $\{C_1, C_2, C_5\}$. Once the spatial boundary sets are computed, the spatial adjacent sets are derived from them, using the Definition 5.4.

**Construction of a Removal Influence Graph:**

Based on the spatial adjacent sets, the removal influence (Definition 5.5) computations for $C_i \in A$ are performed for every $C'$ spatially constraining $C_i$ along $d_j \in U_i$. For example, in Figure 33, $C_3$ is spatially constrained along $d_1$ by $\{C_4, C_5\}$ and along $d_2$ by $\{C_1, C_2, C_5\}$; therefore, only two $C'$, $(C_4, C_5)$ and $(C_1, C_2, C_5)$, are evaluated for spatial removal influence. For determining a SD solution with fewer removals a $C'$ with $RI_i^{C'} = \text{TRUE}$ is necessary. Therefore, a $C'$ satisfying $RI_i^{C'} = \text{TRUE}$ with fewer number of components is selected for $C_i$. In Figure 33, $(C_4, C_5) \subseteq \alpha_3$ and $(C_1, C_2, C_5) \not\subseteq \alpha_3$, where $\alpha_3 = \{C_2, C_4, C_5\}$. Therefore, $RI_3^{1,2,5} = \text{FALSE}$ and $RI_3^{4,5} = \text{TRUE}$. 
A Removal influence Graph, \( \text{RG} \), is constructed to determine the topological disassembly ordering of components. In \( \text{RG} \), the nodes correspond to the components in \( A \) and the arcs correspond to the removal influence between the components. An arc \( C_i \rightarrow C' \) in \( \text{RG} \), with an attribute \( d_j \) on the directed edge, indicates that \( C_i \) is disassemblable along \( d_j \) after removing \( C' \).

For example, consider \( A \) shown in Figure 33 with \( C = \{ C_3 \} \) and \( U = \{ d_1, d_2 \} \). The \( \text{RG} \) is shown in Figure 34. \( C_3 \) is spatially disassemblable along \( d_1 \) after removing \( C_4 \) and \( C_5 \) (\( \text{RI}_{345} = \text{TRUE} \)). \( C_5 \) is spatially disassemblable along \( d_2 \) after removing \( C_2 \) (\( \text{RI}_{52} = \text{TRUE} \)). Moreover, \( \Delta_4 = \text{TRUE} \) and \( \Delta_2 = \text{TRUE} \) for \( C_4 \) and \( C_2 \), respectively.

![Figure 34. RG for A in Figure 33: C = \{C_3\}](image)

**Determination of a Sequence:**

A sequence \( S = \{ C_b \rightarrow C_x \} \) (\( C_x \in C \)) is derivable from \( \text{RG} \) (where \( C_b \rightarrow C_x \) denotes a disassembly order from \( C_b \) to \( C_x \)) via topological sorting of \( \text{RG} \): recursively removing nodes in \( \text{RG} \) with no directed edges until \( C_x \) is removed to determine \( S \). For example, from the \( \text{RG} \) shown in Figure 34, a \( S = \{ C_2, C_4, C_5, C_3 \} \) for \( C_3 \).

The \( \text{GSD} \) algorithm is shown in Figure 35 and an illustration of the algorithm is shown in Figure 36.
Algorithm: **GSD** *(Input: A, C; Output: S)*

\[
\{
\begin{align*}
1. & \text{ Determine the spatial constraints for every } C_i \in A \text{ along every } d_j \in U_i. \\
2. & \text{ Determine the spatial boundary sets for } A. \\
3. & \text{ Determine the spatial adjacent set for every } C_i \in A. \\
4. & \text{ Construct the } RG \text{ for } C. \\
5. & \text{ Topologically sort } RG \text{ and Compute } S.
\end{align*}
\]

**Figure 35.** Global Selective Disassembly Algorithm

<table>
<thead>
<tr>
<th>Component</th>
<th>Direction</th>
<th>Spatial Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&lt;sub&gt;1&lt;/sub&gt;</td>
<td>d&lt;sub&gt;1&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;3&lt;/sub&gt;, C&lt;sub&gt;4&lt;/sub&gt;, C&lt;sub&gt;5&lt;/sub&gt;}</td>
</tr>
<tr>
<td></td>
<td>d&lt;sub&gt;2&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;2&lt;/sub&gt;, C&lt;sub&gt;3&lt;/sub&gt;}</td>
</tr>
<tr>
<td>C&lt;sub&gt;2&lt;/sub&gt;</td>
<td>d&lt;sub&gt;1&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;5&lt;/sub&gt;}</td>
</tr>
<tr>
<td></td>
<td>d&lt;sub&gt;2&lt;/sub&gt;</td>
<td>{}</td>
</tr>
<tr>
<td>C&lt;sub&gt;3&lt;/sub&gt;</td>
<td>d&lt;sub&gt;1&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;4&lt;/sub&gt;, C&lt;sub&gt;5&lt;/sub&gt;}</td>
</tr>
<tr>
<td></td>
<td>d&lt;sub&gt;2&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;1&lt;/sub&gt;, C&lt;sub&gt;2&lt;/sub&gt;, C&lt;sub&gt;5&lt;/sub&gt;}</td>
</tr>
<tr>
<td>C&lt;sub&gt;4&lt;/sub&gt;</td>
<td>d&lt;sub&gt;1&lt;/sub&gt;</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td>d&lt;sub&gt;2&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;2&lt;/sub&gt;, C&lt;sub&gt;3&lt;/sub&gt;, C&lt;sub&gt;5&lt;/sub&gt;}</td>
</tr>
<tr>
<td>C&lt;sub&gt;5&lt;/sub&gt;</td>
<td>d&lt;sub&gt;1&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;2&lt;/sub&gt;, C&lt;sub&gt;3&lt;/sub&gt;}</td>
</tr>
<tr>
<td></td>
<td>d&lt;sub&gt;2&lt;/sub&gt;</td>
<td>{C&lt;sub&gt;2&lt;/sub&gt;}</td>
</tr>
</tbody>
</table>

**Figure 36.** Illustration of the global selective disassembly algorithm for A in Fig. 33, C = \{C<sub>3</sub>\}
Another example illustrating the GSD algorithm for a 3D assembly is as follows: \( \text{C}_3 \) in Figure 37 is disassemblable along \( \text{d}_1 \) after removing \( \text{C}_4 \) and \( \text{C}_5 \), as shown by the RG. \( \text{C}_4 \) is disassemblable along disassembly directions \( (\text{d}_1, \text{d}_2) \) and \( \text{C}_5 \) is disassemblable along \( \text{d}_2 \) after removing \( \text{C}_2 \) along \( \text{d}_2 \). Therefore, from RG shown in Figure 37, \( S = \{\text{C}_2, \text{C}_4, \text{C}_5, \text{C}_3\} \) for \( \text{C}_3 \).

![Figure 37. 3D example to illustrate GSD algorithm and RG for C = \{C_3\}](image)

### 5.4 Incorporation of User-defined Constraints in the Analysis

Component grouping can be incorporated in the above GSD algorithm by treating the components grouped as a single unit (while assembling/disassembling). For example, in Figure 33 if \( \text{C}_2 \) and \( \text{C}_5 \) are grouped as a subassembly, then the group \( (\text{C}_2, \text{C}_5) \) is constrained along \( \text{d}_1 \) by \( \text{C}_3 \) since \( \text{C}_5 \) is constrained by \( \text{C}_3 \) along \( \text{d}_1 \). However, \( (\text{C}_2, \text{C}_5) \) is not constrained along \( \text{d}_2 \) since \( \text{C}_2 \) is not constrained along \( \text{d}_2 \) and \( \text{C}_5 \) is constrained only by \( \text{C}_2 \) along \( \text{d}_2 \). Figure 38 shows the RG for A in Figure 33 with \( S_1 = \{\text{C}_2, \text{C}_5\} \) and \( C = \{\text{C}_3\} \), \( S = \{\text{C}_4, S_1, \text{C}_3\} \).
Geometric constraints such as one that restricts a component to move or one that constraints along one or more disassembly direction(s) can be added to the SD analysis. This can be done through the definition of $U_i$ for $C_i$. To illustrate this, consider the engine assembly shown in Figure 32 with $C = \{C_3\}$. Figure 39 shows the SD and RG without any directional constraints where, $S = \{C_1, C_2, C_3\}$. If $C_1$ is constrained along $-d_3$ then $U_1 = \{d_1, -d_1, d_2, -d_2, d_3\}$, therefore $RI_{312} = FALSE$ along $-d_3$. Figure 40 shows the SD and RG after constraining $C_1$ along $-d_3$, where $S = \{C_5, C_4, C_3\}$.

Figure 39. SD of the engine assembly and the corresponding RG for $C = \{C_3\}$
5.5 Discussion

The GSD algorithm determines a non-interfering SD sequence and can be applied for both single and multiple SD. For SD of \( m \) (\( \geq 1 \)) components, the algorithm is applied to every \( C_x \in C \) to obtain a combined RG. The topological sorting of the RG obtains an appropriate S with fewer numbers of simultaneous component removals to disassemble the \( m \) target components. However, for \( m = n \) target components from A, a non-interfering CD sequence with fewer number of simultaneous removals is determined.

5.5.1 Number of Removals

The sequence determined by the GSD algorithm has fewer numbers of simultaneous removals, and is discussed as follows: Let...
• \( n_r \) = number of component removals in \( S \)

• \( n_s \) = number of simultaneous removals to disassemble \( C \)

• \( n_w \) = number of waves from the assembly boundary \( \beta_1 \) to \( C \).

The attribute \( A_x \) for \( C_x \) determines whether \( C_x \) is globally disassemblable by evaluating the spatial constraints of the components. If \( C_x \in \beta_q \) then \( n_s = q \) and \( C_x \) could have not been disassemblable at \( \beta_p \) \((p < q)\) since, the disassemblability of \( C_i \in A \) is evaluated for all of the disassembly directions in \( U_i \). Moreover, in evaluating the spatial removal influence of components for \( C_i \in A \), only \( C' \subseteq \alpha_i \) are disassembled. Therefore for \( C_x \in \beta_q \), \( n_r = n_w = q \) for \( S \) determined by the algorithm and the sequence evaluated has minimum number of simultaneous removals.

In addition, the spatial removal influence computation further refines the solution set for \( C_x \) by ensuring that all \( C_i \in \alpha_x \) does not have to be disassembled for \( C_x \) and \( n_r \) for \( S \) is locally minimum among the sequences that can be determined for \( C_x \in \beta_r \). For the example shown in Figure 34 with \( C = \{C_3\}, \ S = \{C_2, C_4, C_5, C_3\}, \ n_r = 4 \) and \( n_s = 3 \) since \( C_2 \) and \( C_4 \) can be disassembled simultaneously. Similarly for \( A \) in Figure 32, with \( U = \{d_1, -d_1, d_2, -d_2, d_3, -d_3\} \) and \( C = \{C_3\}, \) the algorithm determines \( S_1 = \{C_1, C_2, C_3\} \) and \( S_2 = \{C_5, C_4, C_3\} \) with \( n_r = n_s = 3 \).

### 5.5.2 Computation Time

The computational complexity of the \textbf{GSD} algorithm for \( C_x \in C \) is polynomial, and is discussed as follows: Let \( n = \) Number of components in \( A \), \( d = \text{Cardinality} \ (U) \), \( F_1 = \) Number of faces of \( C_i \in A \), and \( F = \) Total number of faces of all the components in \( A \).
1. **Step 1:** The spatial constraint of $C_i$ along $d_j$ can be computed in $O(F_i F)$. Therefore for $d$ directions and $n$ components, the computational complexity is $O(d F^2)$.

2. **Step 2:** Determination of every spatial boundary set involves traversal of $O(d n^2)$ entities of spatial constraints. The number of boundary sets is $O(n)$. Therefore, the computational complexity to determine all the boundary sets is $O(d n^3)$.

3. **Step 3:** Determination of the spatial adjacent set involves traversal of boundary set entities and since, there are $O(n)$ spatial boundary sets and $O(n)$ components, the computational complexity is $O(n^2)$.

4. **Step 4:** Construction of an $RG$ involves searching the spatial constraint set based on spatial adjacent set, which takes $O(d n)$.

5. **Step 5:** A sequence can be identified by topological sorting of $RG$ in $O(n^2)$, since $RG$ has $O(n)$ nodes corresponding to the number of components and $O(n^2)$ edges representing the spatial constraints.

Therefore the computational complexity of the GSD algorithm is $O(d F^2 + d n^3)$, i.e., polynomial.

One observation to note is that during the construction of $RG$, to determine a $SD$ sequence only one $C'$ satisfying $RI_i^{C'} = TRUE$ with fewer number of components is selected for $C_i$. This results in a polynomial computation time for a sequence determination. However, if all $C'$ satisfying $RI_i^{C'} = TRUE$ condition are taken for analysis then all the $SD$ sequences with fewer number of simultaneous removals are determined. While this may be useful, the number of $SD$ solutions may grow in a power series for the worst case. Therefore it is appropriate to evaluate one $SD$ sequence for $C$ and if an alternative solution is required then it can also be evaluated in a similar fashion.
CHAPTER 6. SELECTIVE DISASSEMBLY ANALYSIS

Performing disassembly analysis manually takes significant amount of time and becomes difficult for assemblies with larger number of components, and for assemblies with complex geometry. Moreover small changes in the design such as that of component shape or constraints require the designer to perform the disassembly analysis afresh since the geometric constraints of the assembly are changed. However, the product design phase is an iterative procedure, which involves performing changes to design and evaluation of SD results. While the CD techniques are highly inappropriate for SD, there is a need for developing a SD analysis tool.

A regular engineering practice observed by engineering companies is to use an automation solution as an initial result and subsequently modifying the solution to derive at a result satisfactory to their engineering requirements. Therefore, a semi-automated means of SD analysis is necessary allowing the designer to perform the engineering reasoning and the software system to perform the geometric reasoning, for the SD analysis. The requirements of such a design-tool for SD analysis were derived in the current research from discussions with several automotive and aerospace companies.

A new prototypical software system (Assembly and Disassembly in Three Dimensions, A3D) has been developed in the current research, to assist the designer in performing an automated and interactive SD analysis. The SD algorithms proposed in the current research have been implemented in the A3D system to perform the automation analysis. Several techniques allowing the designer to reason the geometry for SD analysis have also been implemented. This chapter presents the A3D system implementation and its functional features as a design-aid in performing the virtual SD analysis.
6.1 A3D System: A Software tool for Selective Disassembly Analysis

A snapshot of the interface of the A3D system for virtual SD analysis is shown in Figure 41. The A3D system uses C++ as a programming language, OpenGL™ as a graphics library, WorldToolKit™ as a development library, ACIS™ and PARASOLID™ for geometric computations and runs both on Unix™ and Windows-NT™ operating systems.

The A3D system focuses on SD analysis, which involves generating, evaluating, editing, validating and animating assembly/disassembly sequences and the paths for 3D geometric models. A3D maintains a hierarchical assembly structure and allows the user to add constraints,
edit the design, and compute the resultant sequence, paths and cost/time. **A3D** allows the user to generate complex paths/motions of the components and validates the resultant assembling/disassembling operation. In addition the user can perform several virtual manufacturing analysis such as interception checking, clearance checking, accessibility analysis of components, and design rule checking. The functional features of the **A3D** system are presented below.

### 6.1.1 Functional Features

- Reads in assembly CAD models created from different modelers. File Formats: PARASOLID, SAT, IGES, STL, DXF, RENDER, WAVEFRONT and VRML.
- CAD Plugins to automatically load assembly models created in CAD systems like Unigraphics™ and Pro-Engineering™.
- Selective (sub-set of components or all components) assembly/disassembly.
- Design-rule checking: e.g. checks assembly for intersections (interlocking components), clearances and accessibility.
- Computes geometric attributes such as accessibility and disassemblability of components.
- Automatically generates disassembly sequences and paths using the **SD** algorithms.
- Digitizes/animates assembling and disassembling.
- Reads/saves sequence/path from/to a file.
- Allows the user to edit assembly/disassembly sequences & paths.
- Validates assembly/disassembly sequences & paths, edited by the user.
- Allows the user to add/remove directional constraints & computes the resultant assembly/disassembly sequence & paths.
• Allows the user to group/ungroup components & computes the resultant sequence and paths.

• Allows the user to edit the overall component shape (Stretching/scaling/transforming components) & computes the resultant sequence and paths.

• Provides visualization features such as Zoom/Pan/Rotate of viewpoints and transformation of models.

• Provides performance-boosting features: Level-of-Detail, Culling, Rendering, etc.

• Operating System: Unix and Windows.

6.1.2 CAD Plugin

Designers use different CAD systems such as ProEngineer™ and UniGraphics™ to model the assembly. Therefore the software-plugins that can automatically load the assembly models generated in different CAD systems into the A3D system are developed. The principle notion here, is to keep the virtual disassembly analysis independent of the CAD systems, so that models can be created in different CAD systems, but can be analyzed more efficiently and rapidly for assembly/disassembly.
Figure 42 shows an assembly model generated in UniGraphics™ (UG) CAD system that is being transferred to A3D system via an UG-Plugin. The UG-Plugin software code extracts the assembly information including the assembly configuration (sub-assemblies) and generates an A3D file, a file format that can be loaded in the A3D system.

6.1.3 Disassemblability and Accessibility Computation

The A3D system uses the WP algorithms for automated SD sequence and path generation. However the designer may desire to modify the sequence or path generated or may explore alternative designs. Therefore a user-interface has been added in the A3D system using which the designer can query on any component for its disassemblability and accessibility values. Using these geometric attributes the designer can reason an assembly for SD.

6.1.4 Sequence Generation, Editing and Validation

Sequence editing and validation allows the user to modify the sequence generated for a particular application due to extraneous constraints. The A3D system allows the user to edit an automatically generated S and validates the resultant S (feasibility of S is checked) through the disassembly waves and accessibility computation of components.

For example, Figure 43 shows the SD of the crankshaft sub-assembly with $C = \{4\}$ and Figure 44 shows the result of a sequence editing procedure performed. The component 4 is edited to disassemble along the direction Y-; however, as Y- for component 4 results in a collision, the A3D system reports this error as a feedback text in the A3D system.
Figure 43. SD of the Crankshaft sub-assembly for $C = \{4\}$

Figure 44. Sequence editing and validation in the A3D system
6.1.5 Path Generation, Editing and Validation

Based on the application requirements, the user should be able to modify the paths generated. Therefore editing features have been added in the A3D system allowing the user to generate complex paths/motions of components, and validate the resultant assembling/disassembling operation. Figure 45 shows the result of path editing performed on \( C_{11} \), where path1 is the generated path and path2 is the edited path.

![Figure 45. Path editing in the A3D system](image)

Two methods have been implemented to perform the path editing: (I) Defining a path of a component using path elements (cube shown in the Figure 45) and allowing the user to modify the path elements. These path elements represent the 3D position and 3D orientation of the component along the path. (II) Allowing the user to attach the component to an input device (e.g. 2D mouse or 3D mouse or 3D pointer or 3D Glove) in defining/editing a path.
In both the cases the validity (non-interference) of the path is checked and reported. Figure 46 shows the result of defining an invalid path for \( C_1 \). The parts interfering with \( C_1 \) are shown in red color. In this example, \( C_1 \) was attached to a 3D mouse (which is not shown in the figure) in defining a path and the validity of the path is checked via dynamic collision checking of components and the result of interference is shown during the path editing process itself.

**Figure 46.** Path editing and validation in the A3D system

### 6.1.6 Overall Shape Editing and Validation

![A3D Geometry Editing](image1)

**Figure 47.** Shape editing and validation in the A3D system
The A3D system allows the user to edit the overall shape of the components (X, Y and Z dimensions) in the assembly and performs the SD analysis for the new design. Using the A3D interface, the validity of the previously computed sequence and paths for the new design can be checked. Figure 47 shows the result of editing the shape of C2 and its effect on a previously computed sequence. C2 is interfering with C7 and the error is reported via graphical display (the interfering components are shown in red color).

6.1.7 Component-Grouping and Directional Constraints

The GSD algorithm determines S automatically by computing the geometric accessibility of components. However, the user inputs may be required sometimes for complex systems. For example, to remove the air-conditioner flow-channels from a car assembly (dashboard sub-assembly is attached to the car), the SD algorithms (with non-destructive, translational disassembly assumptions) cannot determine S automatically. This is due to the fact that the air-conditioner flow-channels and the fasteners are not accessible and are blocked by the entire dashboard sub-assembly and the metallic frame of the car.

However in practice one possible disassembly procedure is to remove the entire dashboard sub-assembly from the car as a single unit followed by the removal of the air-conditioner flow-channels from the dashboard sub-assembly. The information about removing the entire dashboard unit first as a target component followed by disassembling the air-conditioner flow-channels as the target component may be provided as input by the user or obtained from a knowledge-base.

Therefore the functional features allowing the user to specify user-defined constraints such as component grouping and directional constraints and performing the SD analysis have been developed in the A3D system (the procedure to incorporate user-defined constraints is
presented in Section 5.4). However the current implementation expects the user to specify the sub-assembly and the A3D system is not integrated with any other knowledge-based one.

6.1.8 Selective Disassembly Cost and Time Evaluation

Once a SD sequence and disassembly paths for the components are computed (in either automated or semi-automated fashion), the designer may desire to evaluate the SD solution for parameters such as disassembly time or cost. Evaluation of these parameters may facilitate the designer to assess the design based on the SD results.

Therefore to quickly assess the design, a simple cost/time evaluation procedure has been developed. The standard disassembly time to remove different fasteners and components of different weights and sizes are derived from a standard database [82]. For a generated SD sequence, the SD cost is then expressed as a product of labor rate (LR) and SD time to remove the components and fasteners in S. For the Augmentor (I) assembly (Figure 64) with LR = 20$/hr, the SD cost for S = \{OL33, OL55, OL44\} is $.76 which includes removal of 3 components (45 seconds) and 8 screws (92 seconds).

6.1.9 Design Rule Checking

One of the main requirements of the SD analysis is to assess the design for various engineering rules related to product assembly and SD results. Therefore a functional feature allowing the user to perform several design rule checking related to the number of components in the SD sequence, disassembly cost for the target component, disassembly paths of certain component(s) in the sequence, clearance and interference between the components, has been added. For example, Figure 48 shows the interference-check performed for the interlocking-components.
6.1.10 Design Comparison

The SD analysis performed in the A3D system may be used to compare alternate product designs in order to determine a better design for SD. In general, comparison of alternate product design considers a disassembly-rating index, such as the number of disassembly operations or removals in S. For example Figure 49 shows two alternate designs of a Screw Jack assembly (exploded view is shown). Design-1 differs from Design-2 only in C3 (washer with a cut in
Design B), such that $C_3$ can be directly disassembled. With the number of removals as a criterion, clearly Design B is better than that of Design-1. The above analysis suggests that with minor modifications in design, components to be maintained or recycled may be easily disassembled resulting in efficient product maintenance and recycling.

6.2 A3D System Implementation: Variations

Apart from a regular UNIX/Windows based implementation of the A3D system two other implementations of the A3D system, one with a Virtual Reality (VR) Interface and another with a web-based interface are developed.

6.2.1 Virtual Reality Interface for A3D

For feasibility and ergonomic analysis of assembling and maintenance applications a virtual human in SD analysis is important; therefore a VR interface for A3D has been developed. The VR-interface provides an intuitive and an easy-to-use environment for designers by facilitating 3D-hand tracking, voice command, and stereoscopic visual display for SD analysis. For testing of the A3D-VR software system, the VR hardware set-up of the Virtual Design Studio [83, 84] developed for shape creation, has been utilized.

Figure 50 shows a designer performing virtual disassembly analysis in the A3D-VR system and Table 3 shows the hardware/software details. In the A3D-VR system, immersion is provided via a large vertical screen upon which stereoscopic images are projected. The multi-sensory VR interface allows the user to combine the voice and hand inputs to manipulate and analyze the assemblies in the design space. Generating and editing of complex paths is simpler in a VR environment. Moreover, the designer in the VR environment following the generated sequence and paths can grasp objects with hands and moves them around, detach parts from the
assemblies for feasibility study in product assembling and maintenance. Another application for VR-based A3D system is Visualization of artifacts for its assembly and disassembly.

![A3D implementation system with a virtual reality interface](image)

**Figure 50.** A3D implementation system with a virtual reality interface

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrohome Projector</td>
<td>Analysis (Automated &amp; Human-assisted)</td>
</tr>
<tr>
<td>Custom Frame (with mirror, screen etc.)</td>
<td>Geometric Kernel: PARASOLiD™</td>
</tr>
<tr>
<td>SGI™ Octane/Intergraph™ TDZ</td>
<td>Graphics: WorldToolKit™ and OpenGL™</td>
</tr>
<tr>
<td>Ascension™ trackers and emitter</td>
<td>Language: ANSI C++</td>
</tr>
<tr>
<td>Custom 3D Pointer</td>
<td>Voice Recognition: IBM VoiceType™</td>
</tr>
<tr>
<td>Pair of 5th™ Gloves</td>
<td>Interaction (Device Drivers, Hand input library, Voice library and 3D menus)</td>
</tr>
<tr>
<td>Crystal Eyes™ LCD glasses and emitter</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3.** Details about the A3D virtual reality system
6.2.2 Web-Interface for A3D

To enable distributed computing, visualization and to support eManufacturing (internet-based manufacturing analysis, which involves maintenance, assembling and recycling analysis), a Java-based web-interface was developed for the A3D system. Figure 51 shows a client-server model for the A3D web-interface. The A3D-Client is a Java/Java3D based code for user-interface and graphics and the A3D-Server is written in C++ and Parasolid, for SD computations. The A3D-Client passes the user inputs to the A3D-Server, which performs the SD analysis and sends the result to the A3D-Client. The motivation for a web-based A3D system implementation, is mainly due to the fact that the SD analysis involves multiple-operations (accessibility, disassemblability, sequence, path, cost, time, editing, constraints, etc.) which may be distributed and computed as multiple-packets of queries and results suitable for web-based implementation. However proving the effectiveness of the web-based implementation and its performance is not within the scope of the current research.
CHAPTER 7. RESULTS AND DISCUSSION

This chapter presents the results and discussion on the proposed SD methods and the A3D system as a design tool for SD analysis.

7.1 Contact-Geometric Constraints and Selective Disassembly

The contact-based SD algorithms were tested for several assemblies in the A3D system and some of the results for single and multiple SD are presented below.

The Wheel Support Assembly: Figure 54 shows the SD of the Wheel Support assembly (Figure 52) generated by the SWP algorithm for $C = \{C_3, C_6\}$. $S = \{C_9, C_2, C_3, C_8, C_7, C_6\}$ with $n_r = 6$. Similarly, Figure 55 shows the corresponding SD result generated by the MWP algorithm for $C = \{C_3, C_6\}$. The MWP algorithm determines two sequences at $T = 1$ with $n_r = 5$: $S_3 = \{C_8, C_7, C_6, C_4, C_3\}$ and $S_2 = \{C_9, C_2, C_3, C_4, C_6\}$. Figure 55 shows the SD result $S_2$ to disassemble $C$ and the RG at $T = 1$ for $C = \{C_3, C_6\}$ is shown in Figure 53.

![Figure 52. Exploded view of the Wheel Support assembly for SD analysis](image)
Figure 53. RG at $T = 1$ by MWP for the Wheel Support assembly with $C = \{C_3, C_6\}$

Figure 54. SD of the Wheel Support assembly for $C = \{C_3, C_6\}$ from SWP

Figure 55. SD of the Wheel Support assembly for $C = \{C_3, C_6\}$ from MWP
**The Screw Jack Assembly:** Figure 56 shows the SD of the Screw Jack assembly for \( C = \{C_4, C_6\} \) generated by the MWP algorithm. The RG for \( C \) at \( T = 2 \) is also shown. In this example, \( C_3 \) is threaded to \( C_2 \) hence, \( R_4^3 = \text{FALSE}, R_6^5 = \text{FALSE} \) and \( \tau_2^4 = \{\} \). Therefore, \( C \) can be disassembled only along \( S = \{C_7, C_6, C_5, C_4\} \) with \( n_r = 4 \), identified by \( (\tau^4_0 \cap \beta_4), (\tau^6_0 \cap \beta_2) \) and \( (\tau^4_2 \cap \tau^6_0 \cap \beta_2) \).

![Figure 56. SD of the Tail Stock assembly for \( C = \{C_4, C_6\} \) from MWP and the RG at \( T = 2 \)](image)

**The Gear Reducer Assembly:** For the Gear Reducer Assembly shown in Figure 57, the RG at \( T = 2 \) for \( C = \{C_2, C_3, C_4, C_{12}, C_{13}, C_{16}, C_{20}, C_{22}\} \) is shown in Figure 58. The PIE algorithm determines a SD sequence \( S = \{C_1, C_2, C_3, C_4, C_{14}, C_{13}, C_{12}, C_{15}, C_{16}, C_{23}, C_{22}, C_{21}, C_{20}\} \) with \( n_r = 13 \), identified by: \( (\tau^2 \cap \tau^3 \cap \tau^4 \cap \beta^3), (\tau^{12} \cap \tau^{13} \cap \beta^3), (\tau^{16}_0 \cap \beta_2) \) and \( (\tau^{20} \cap \tau^{22} \cap \beta^2) \).
Figure 57. Exploded view of the Gear Reducer assembly

Figure 58. RG at $T = 2$ by PIE for Gear Reducer with $C = \{C_2, C_3, C_4, C_{12}, C_{13}, C_{16}, C_{20}, C_{22}\}$
The SD algorithms incorporate fasteners in the analysis as follows: Determining $S$ for $C$ by ignoring the existence of fasteners, and subsequently determining the fasteners that need to be removed to disassemble all $C_i \in S$, and modifying $S$ appropriately [19]. For example, in Figure 57 (with fasteners), $J_1, J_2, J_3, J_4 \in A$ should be removed to disassemble $C_1, C_{14}, C_{15}, C_{23} \in S$, respectively. Therefore, the resultant $S = \{J_1, C_1, C_2, C_3, C_4, J_2, C_{14}, C_{13}, C_{12}, J_3, C_{15}, C_{16}, J_4, C_{23}, C_{22}, C_{21}, C_{20}\}$ for $C = \{C_2, C_3, C_{12}, C_{13}, C_{16}, C_{20}, C_{22}\}$.

<table>
<thead>
<tr>
<th>A</th>
<th>Figure</th>
<th>C</th>
<th>S</th>
<th>$n_f$</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clamp Holder</td>
<td>Figure 10</td>
<td>${C_{11}}$</td>
<td>${C_2, C_{12}, C_{11}}$</td>
<td>3</td>
<td>SWP</td>
</tr>
<tr>
<td>Machine Vise</td>
<td>Figure 19</td>
<td>${C_4, C_5}$</td>
<td>${C_1, C_2, C_4, C_5}$</td>
<td>4</td>
<td>MWP</td>
</tr>
<tr>
<td>Wheel Support I</td>
<td>Figure 52</td>
<td>${C_3, C_6}$</td>
<td>${C_{9}, C_2, C_3, C_4, C_6}$</td>
<td>5</td>
<td>MWP</td>
</tr>
<tr>
<td>Screw Jack</td>
<td>Figure 56</td>
<td>${C_4, C_6}$</td>
<td>${C_7, C_6, C_5, C_4}$</td>
<td>4</td>
<td>MWP</td>
</tr>
<tr>
<td>Wheel Support II</td>
<td>Figure 25</td>
<td>${C_5, C_6, C_7}$</td>
<td>${C_4, C_5, C_6, C_7}$</td>
<td>4</td>
<td>PIE</td>
</tr>
<tr>
<td>Lego Toys</td>
<td>Figure 30</td>
<td>${C_3, C_4}$</td>
<td>${C_1, C_2, C_3, C_4}$</td>
<td>4</td>
<td>PIE</td>
</tr>
<tr>
<td>Test I</td>
<td>Figure 8</td>
<td>${C_{16}}$</td>
<td>${C_4, C_{10}, C_{16}}$</td>
<td>3</td>
<td>SWP</td>
</tr>
<tr>
<td>Test II</td>
<td>Figure 9</td>
<td>${C_9}$</td>
<td>${C_{11}, C_{10}, C_9}$</td>
<td>3</td>
<td>SWP</td>
</tr>
<tr>
<td>Test III</td>
<td>Figure 15</td>
<td>${C_3, C_5}$</td>
<td>${C_1, C_4, C_3, C_5}$</td>
<td>4</td>
<td>MWP</td>
</tr>
<tr>
<td>Gear Reducer</td>
<td>Figure 57</td>
<td>${C_2, C_3, C_4, C_{12}, C_{13}, C_{16}, C_{20}, C_{22}}$</td>
<td>${C_1, C_2, C_3, C_4, C_{14}, C_{13}, C_{12}, C_{15}, C_{16}, C_{23}, C_{22}, C_{21}, C_{20}}$</td>
<td>13</td>
<td>PIE</td>
</tr>
</tbody>
</table>

Table 4. Summary of results for contact-based SD algorithms
7.2 Contact-Based Selective Disassembly Algorithms: Analysis

Table 4 summarizes the SD results for the 10 test assemblies analyzed, based on the contact-geometry algorithms. A comparison of \( n \) (number of components in the assembly) and \( n_r \) (number of component removals) for the corresponding \( C \) (target components) is shown in the bar charts (Figures 59 and 60). Based on the results, it can be observed that the contact-based SD algorithms determine \( S \) for \( C \) such that \( n_r \leq n \), i.e., unlike a CD solution which involves disassembling all the components in \( A \), the SD algorithms determine a SD solution (a solution with a few removals/disassembly operations).

Another key observation about SWP, MWP and PIE approaches is that these algorithms ensure a SD solution with fewer total number of disassembly waves between the target component(s) and the assembly boundary, for the class of single dependent components.

For example, in the Wheel Support assembly (Figure 52) when SWP algorithm is applied for \( C = \{C_3\} \) it determines a solution with \( S = \{C_9, C_2, C_3\} \). The total number of disassembly waves between \( C_3 \) (target) and \( C_9 \) (a boundary component) for this \( S \) is 3, which is better compared to other SD sequences such as \( S = \{C_8, C_7, C_6, C_4, C_3\} \) with the total number of disassembly waves being 5.

Similarly, if MWP/PIE algorithms are applied with \( C = \{C_3, C_6\} \) (Figure 52), they determine a SD solution \( S = \{C_9, C_2, C_3, C_4, C_6\} \). The total number of disassembly waves between \( \{C_3, C_6\} \) (target components) and the assembly boundary for this \( S \) is 5, which is better compared to other SD sequences such as \( S = \{C_9, C_2, C_3, C_8, C_7, C_6\} \) with the total number of disassembly waves being 6.
Figure 59. Bar chart-I comparing $n$ and $n_r$ for the contact-based SD results.

Figure 60. Bar chart-II comparing $n$ and $n_r$ for the contact-based SD results.
7.2.1 Comparison of SWP and MWP/PIE Algorithms

Both the SWP and the MWP/PIE algorithms can be applied to single dependent class of SD problems; however there exists a trade-off between the number of removals ($n_r$) and the number of (disassemblability + removal-influence) computations denoted as $O_p$. Table 5 compares the SWP and MWP/PIE algorithms applied to the same test assemblies based on $n_r$ and $O_p$. For the MWP/PIE algorithms the minimum number of $O_p$ required in determining a SD solution is tabulated.

Based on the presented results it can be observed that: (I) $n_r$ of SWP $\geq$ $n_r$ of MWP/PIE and (II) $O_p$ of SWP $\leq$ $O_p$ of MWP/PIE. Moreover, the SWP algorithm is efficient (fewer $O_p$) than the MWP/PIE algorithms for single SD and the MWP/PIE algorithms are appropriate (fewer $n_r$) than the SWP algorithm for multiple SD.

<table>
<thead>
<tr>
<th>A</th>
<th>Figure</th>
<th>C</th>
<th>$n_r$</th>
<th>$O_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>SWP</td>
<td>MWP/PIE</td>
</tr>
<tr>
<td>Wheel Support</td>
<td>Figure 52</td>
<td>{C₃, C₆}</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Screw Jack</td>
<td>Figure 56</td>
<td>{C₄, C₆}</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Clamp Holder</td>
<td>Figure 10</td>
<td>{C₁₁}</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Test III</td>
<td>Figure 15</td>
<td>{C₃, C₅}</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Lego Toys</td>
<td>Figure 30</td>
<td>{C₃, C₄}</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table 5.** Comparison of SD results from SWP and MWP/PIE algorithms
7.2.2 Comparison of MWP and PIE Algorithms

Both MWP and PIE algorithms order IE’s based on the time step T; however the events identified at each T and the number of such events differ. The MWP algorithm starts at T = 0 with disjoint sequences for C and at every time step increment (T > 0) tries to identify S that is better (fewer \( n_r \)) than previously computed S. For T > 0, all new IE’s are identified. Therefore, MWP algorithm can be processed until some user-defined limiting time-step \( T' \), determining a locally optimum S (a solution with lesser removals) at T = T'. However, the PIE algorithm defines \( \phi \) events with respect to C and T, thereby determining an appropriate S with fewer removals only after all the \( \phi \) events are processed.

For example, the RG of MWP and PIE algorithms for A in Figure 30 is shown in Figures 61 and 62. The MWP algorithm at T = 1: \( (\tau^3_0 \cap \tau^4_1 \cap \beta_3) \) at C3 determines S = \{C1, C2, C3, C4\} and \( (\tau^1_1 \cap \tau^4_0 \cap \beta_3) \) at C4 determines S = \{C6, C5, C4, C3\}. However, for the PIE algorithm, there are no \( \phi \) events at T = 1. For this example, S with fewer removals is reached at T = 1 by the MWP and not by the PIE algorithm. That is, at T = 1: S = \{C1, C2, C3, C4\} with \( n_r = 4 \) by the MWP algorithm and S = \{C6, C5, C4, C1, C2, C3\} with \( n_r = 6 \) by the PIE algorithm.

![Figure 61. RG at T = 1, C = \{C3, C4\} for the test assembly in Figure 30: from MWP algorithm](image)
Figure 62. RG at $T = 1$, $C = \{C_3, C_4\}$ for the test assembly in Figure 30: from PIE algorithm

<table>
<thead>
<tr>
<th>A</th>
<th>Figure</th>
<th>C</th>
<th>m</th>
<th>S</th>
<th>n</th>
<th># of events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test III</td>
<td>Figure 15</td>
<td>${C_3, C_5}$</td>
<td>2</td>
<td>${C_1, C_4, C_3, C_5}$</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Machine Vise</td>
<td>Figure 19</td>
<td>${C_4, C_5}$</td>
<td>2</td>
<td>${C_1, C_2, C_4, C_5}$</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Wheel Support II</td>
<td>Figure 25</td>
<td>${C_5, C_6, C_7}$</td>
<td>3</td>
<td>${C_4, C_5, C_6, C_7}$</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>Lego</td>
<td>Figure 30</td>
<td>${C_3, C_4}$</td>
<td>2</td>
<td>${C_1, C_2, C_3, C_4}$</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Screw Jack</td>
<td>Figure 56</td>
<td>${C_4, C_6}$</td>
<td>2</td>
<td>${C_7, C_6, C_5, C_4}$</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Gear Reducer</td>
<td>Figure 57</td>
<td>${C_2, C_3, C_4, C_{12}, C_{13}, C_{16}, C_{20}, C_{22}}$</td>
<td>8</td>
<td>${C_1, C_2, C_3, C_4, C_{14}, C_{13}, C_{12}, C_{15}, C_{16}, C_{23}, C_{22}, C_{21}, C_{20}}$</td>
<td>23</td>
<td>293</td>
</tr>
</tbody>
</table>

Table 6. Comparison of SD results from MWP and PIE algorithms

In general, the MWP computes all the IE’s, whereas the PIE algorithm computes only the $\phi$ events (which are necessary but need not be sufficient to ensure absolute minimum removals).

Moreover, the MWP and PIE algorithms determine a SD solution in a feasible computation time: lower-order exponential and polynomial respectively. Therefore, MWP is feasible for m
<< n target components and PIE for m < n. Table 6 compares the performance of MWP and PIE algorithms for various test assemblies. One observation is that the number of IE’s of PIE is lesser than that of the MWP algorithm.

7.3 Spatial-Geometric Constraints and Selective Disassembly

The global SD algorithm was tested for 20 test assemblies in the A3D system and some of the results for single and multiple SD are presented below.

*The Augmentor Assembly (I):* Figure 63 shows an Augmentor assembly in the A3D system and the selection of the target component \( C = \{\text{OL44}\} \) by the user using a virtual pointer. The SD of the Augmentor assembly for \( C = \{\text{OL44}\} \) generated by the GSD algorithm is shown in Figure 64. The corresponding RG for \( C \) is also shown in the Figure 64, where \( S = \{\text{OL33, OL55, OL44}\} \) with \( n_r = 3 \) and \( n_s = n_w = 2 \).

![Figure 63. Selection of the target component C = \{OL44\} for the Augmentor assembly I](image-url)
The Augmentor Assembly (II): The GSD algorithm analyzes the spatial constraints imposed by both the contact and non-contact geometry as illustrated by the following result. Figure 65 shows the SD of an Augmentor sub-assembly with $S = \{\text{ol}11, \text{ol}1, \text{ol}33, \text{ol}22, \text{s}1, \text{ol}3, \text{ol}2, \text{s}21\}$ generated for $C = \{\text{s}21\}$ and Figure 66 shows the corresponding RG generated by the GSD algorithm. In this example $\text{ol}33$ is constrained by $\text{ol}3$ (both by contact and spatially). Therefore, to ensure global disassembly, even between two mating components, analyzing the contact-geometry alone is not sufficient but spatial analysis is necessary. The components in $\beta_1$ are disassembled simultaneously followed by the components in $\beta_2$ and $\beta_3$. Therefore, the simultaneous disassembly groups are $(\text{ol}33, \text{ol}22, \text{ol}11, \text{ol}1)$, $(\text{ol}3, \text{ol}2, \text{s}1)$ and $(\text{s}21)$ with $n_r = 8$ and $n_k = n_w = 3$. 
Figure 65. SD of the Augmentor assembly (II) from GSD for $C = \{s21\}$

Figure 66. RG for the Augmentor assembly (II) from GSD: $C = \{s21\}$
The Tailstock Assembly: Figure 68 shows the SD of the Tail Stock assembly for $C = \{C_9\}$ generated by the GSD algorithm. The Tail Stock assembly and the corresponding RG for $C$ is shown in Figure 67, where $S = \{C_{12}, C_{10}, C_{11}, C_9\}$ with $n_c = 4$ and $n_s = n_w = 3$.

Figure 67. Tail Stock Assembly and the RG generated by the GSD algorithm for $C = \{C_9\}$

Figure 68. SD of the Tail Stock assembly for $C = \{C_9\}$ from GSD
The Motor Assembly: Figure 69 shows the SD of a Motor assembly generated for $C = \{C_3, C_4\}$ by the GSD algorithm. For this example, the GSD algorithm is applied for $C_3$ and $C_4$ and the corresponding RG for $C$ is shown in Figure 70, where $S = \{C_1, C_2, C_3, C_4\}$ with $n_r = 4$ and $n_s = n_w = 3$. The simultaneous disassembly groups are $(C_1)$, $(C_2)$ and $(C_3, C_4)$. In this example, $C_1$ is a threaded component such that $\Delta_1 = \text{TRUE}$ and $R_{I2}^1 = \text{TRUE}$. While animating the sequence, $C_1$ is disassembled along a helical path defined by the pitch and lead of the thread (represented in the assembly model).

**Figure 69.** SD of the Motor assembly for $C = \{C_3, C_4\}$ from GSD
**The Cab Assembly:** Figure 72 shows the SD of the Cab assembly for $C = \{C_3, C_4\}$ generated by the GSD algorithm. The Cab assembly and the corresponding RG for $C$ is shown in Figure 71, where $S = \{C_1, C_4, C_2, C_3\}$ with $n_r = 4$ and $n_s = n_w = 2$. The simultaneous disassembly groups are $(C_1, C_4)$ and $(C_2, C_3)$, as illustrated by Figure 72.

**Figure 70. RG of the Motor assembly for $C = \{C_3, C_4\}$**

**Figure 71. Cab assembly and the RG generated by the GSD algorithm for $C = \{C_2, C_3\}$**
The Cell Phone Assembly: Figure 74 shows the SD of the Cell Phone assembly for $C = \{C_2, C_3\}$ generated by the GSD algorithm. The Phone assembly and the corresponding RG for $C$ is shown in Figure 73, where $S = \{C_3, C_4, C_1, C_2\}$ with $n_r = 4$ and $n_s = n_w = 3$. The simultaneous disassembly groups $(C_3, C_4)$, $(C_1)$ and $(C_2)$. In this example, the disassembly of $C_3$ can be clubbed with any simultaneous disassembly groups, since there are no dependents to $C_3$ in RG.
Other possible simultaneous disassembly groups are \((C_4), (C_3, C_1)\) and \((C_2); (C_4), (C_1)\) and \((C_3, C_2)\), both with \(n_s = n_w = 3\). Moreover in this example the snap-fits (which results in interlocking between components) were not modeled therefore a \textbf{SD} sequence is determined with rigid component assumption.

\[\quad\]

\textbf{Figure 74. SD} of the Cell Phone assembly for \(C = \{C_2, C_3\}\) from GSD

\[\quad\]

\textit{The Shaft Support Assembly:} Figure 75 shows a sequential disassembly of all the components in the Shaft Support assembly, \(S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}\}\), generated by the \textbf{GSD} algorithm. The solution determined by the algorithm can be executed either in sequential or parallel fashion. In case of parallel disassembly, the simultaneous disassembly groups are \((C_1, C_2), (C_3, C_5), (C_4, C_6), (C_7), (C_8), (C_9)\) and \((C_{10})\) with \(n_r = 10\) and \(n_s = 7\).
Table 7 summarizes the results of the 20 test assemblies analyzed with the global SD algorithm. Based on the test results as also the definition of the GSD algorithm it can be observed that a non-interfering SD solution is determined.
<table>
<thead>
<tr>
<th>A</th>
<th>Figure</th>
<th>C</th>
<th>S</th>
<th>$n_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aero-Engine I</td>
<td>Fig. 85</td>
<td>{rjf21}</td>
<td>{rjf18, rjf16, rjf4, rjf5, rjf9, rjf8, rjf7, rjf6, rjf21}</td>
<td>4</td>
</tr>
<tr>
<td>Aero-Engine II</td>
<td>Fig. 87</td>
<td>{S1}</td>
<td>{rjf18, rjf16, S1}</td>
<td>2</td>
</tr>
<tr>
<td>Aero-Engine III</td>
<td>Fig. 89</td>
<td>{rjf20}</td>
<td>{rjf11, rjf10, cy, rjf20}</td>
<td>2</td>
</tr>
<tr>
<td>Aero-Engine IV</td>
<td>Fig. 91</td>
<td>{rjf20}</td>
<td>{rjf13, rjf12, rjf11, rjf10, rjf22, rjf20}</td>
<td>3</td>
</tr>
<tr>
<td>Tail stock</td>
<td>Fig. 67</td>
<td>{C9}</td>
<td>{C12, C10, C11, C9}</td>
<td>3</td>
</tr>
<tr>
<td>Augmentor I</td>
<td>Fig. 64</td>
<td>{OL44}</td>
<td>{OL33, OL55, OL44}</td>
<td>2</td>
</tr>
<tr>
<td>Augmentor II</td>
<td>Fig. 65</td>
<td>{s21}</td>
<td>{ol11, ol1, ol33, ol22, s1, ol3, ol2, s21}</td>
<td>3</td>
</tr>
<tr>
<td>Dashboard I</td>
<td>Fig. 96</td>
<td>{C16}</td>
<td>{C4, C10, C11, C8, C12, C16}</td>
<td>5</td>
</tr>
<tr>
<td>Dashboard II</td>
<td>Fig. 101</td>
<td>{C5, C6, C9, C12, C18, C19}</td>
<td>{C5, C6, C18, C19, C4, C10, C11, C8, C9, C12}</td>
<td>4</td>
</tr>
<tr>
<td>Plane I</td>
<td>Fig. 93</td>
<td>{C4}</td>
<td>{C1, C2, C4}</td>
<td>2</td>
</tr>
<tr>
<td>Plane II</td>
<td>Fig. 95</td>
<td>{S1}</td>
<td>{C1, C2, C3, S1}</td>
<td>3</td>
</tr>
<tr>
<td>Engine I</td>
<td>Fig. 39</td>
<td>{C3}</td>
<td>{C1, C2, C3}</td>
<td>3</td>
</tr>
<tr>
<td>Engine II</td>
<td>Fig. 40</td>
<td>{C3}</td>
<td>{C5, C4, C3}</td>
<td>3</td>
</tr>
<tr>
<td>Crankshaft</td>
<td>Fig. 43</td>
<td>{4}</td>
<td>{3f, 2, 4}</td>
<td>3</td>
</tr>
<tr>
<td>Motor</td>
<td>Fig. 69</td>
<td>{C3, C4}</td>
<td>{C1, C2, C3, C4}</td>
<td>3</td>
</tr>
<tr>
<td>Test I</td>
<td>Fig. 33</td>
<td>{C3}</td>
<td>{C2, C4, C5, C3}</td>
<td>3</td>
</tr>
<tr>
<td>Test II</td>
<td>Fig. 37</td>
<td>{C3}</td>
<td>{C2, C4, C5, C3}</td>
<td>3</td>
</tr>
<tr>
<td>Shaft Support</td>
<td>Fig. 75</td>
<td>All</td>
<td>{C1, C2, C3, C5, C4, C6, C7, C8, C9, C10}</td>
<td>7</td>
</tr>
<tr>
<td>Cab</td>
<td>Fig. 71</td>
<td>{C2, C3}</td>
<td>{C1, C4, C2, C3}</td>
<td>2</td>
</tr>
<tr>
<td>Cell Phone</td>
<td>Fig. 73</td>
<td>{C2, C3}</td>
<td>{C4, C3, C1, C2}</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 7.** Summary of results for the global SD algorithm: SD Sequence
The **GSD** algorithm ensures the global (collision free) disassembly by analyzing both the spatial constraints (contact/non-contact geometric constraints) and also the user-defined constraints imposed on the geometric models of an assembly. The hyper-edge from every \( C_i \) in the \( \text{RG} \), connects to all the components that need to be removed for the global disassembly of \( C_i \). To illustrate, consider the Aero-Engine assembly (Figure 85) with \( C = \{ \text{rjf21} \} \). The prior components for removals to ensure global disassembly of \( \text{rjf21} \) are \( (\text{rjf6, rjf7, rjf8, rjf9, rjf16, rjf18}) \), which are connected by a hyper-edge from \( \text{rjf21} \). By recursively checking every dependent component for its global disassembly, the **GSD** algorithm ensures a non-interfering \( \text{SD} \) sequence.

### 7.4.1 Number of Removals

Let,

- \( n = \) number of components in \( A \)
- \( n_r = \) number of component removals in \( S \)
- \( n_s = \) number of simultaneous removals to disassemble \( C \) (from observation of \( A \))
- \( n_w = \) number of waves from \( \beta_1 \) to \( C \).

From the 20 test assembly results, charts comparing \( n, n_r, n_w \) and \( n_s \) are shown in Figures 76 to 80. One observation is that \( n_r \leq n \), i.e., the **GSD** algorithm determines an appropriate \( \text{SD} \) solution with fewer removals. Another key finding based on the results, is that the \( n_s = n_w \). Since the **GSD** algorithm minimizes the number of waves from \( C \) to \( \beta_1 \) and the components in a wave are disassembled independent of each other, the **GSD** algorithm ensures a \( \text{SD} \) solution for \( C \) with minimal \( n_s \).
Figure 76. Bar chart-I comparing $n$, $n_r$, $n_w$ and $n_s$ for the global SD results

Figure 77. Bar chart-II comparing $n$, $n_r$, $n_w$ and $n_s$ for the global SD results
Figure 78. Bar chart-III comparing $n$, $n_r$, $n_w$ and $n_s$ for the global SD results.

Figure 79. Bar chart-IV comparing $n$, $n_r$, $n_w$ and $n_s$ for the global SD results.
Figure 80. Bar chart-V comparing $n$, $n_r$, $n_w$ and $n_s$ for the global SD results

<table>
<thead>
<tr>
<th>Assembly</th>
<th>Figure</th>
<th>$n$</th>
<th>$F$</th>
<th>$d$</th>
<th>$C$</th>
<th>Computation Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Step (1)</td>
</tr>
<tr>
<td>Dashboard</td>
<td>Fig. 96</td>
<td>52</td>
<td>5120</td>
<td>6</td>
<td>${C_{16}}$</td>
<td>23.210</td>
</tr>
<tr>
<td>Engine</td>
<td>Fig. 32</td>
<td>7</td>
<td>8322</td>
<td>6</td>
<td>${C_3}$</td>
<td>10.942</td>
</tr>
<tr>
<td>Motor</td>
<td>Fig. 69</td>
<td>35</td>
<td>15552</td>
<td>6</td>
<td>${C_3, C_4}$</td>
<td>126.962</td>
</tr>
<tr>
<td>Plane</td>
<td>Fig. 93</td>
<td>45</td>
<td>18070</td>
<td>6</td>
<td>${C_4}$</td>
<td>171.204</td>
</tr>
<tr>
<td>Augmentor</td>
<td>Fig. 65</td>
<td>15</td>
<td>32159</td>
<td>6</td>
<td>${s21}$</td>
<td>329.86</td>
</tr>
<tr>
<td>Aero-Engine</td>
<td>Fig. 85</td>
<td>17</td>
<td>184622</td>
<td>6</td>
<td>${rjf21}$</td>
<td>2182.625</td>
</tr>
<tr>
<td>Crank Shaft</td>
<td>Fig. 43</td>
<td>4</td>
<td>1245</td>
<td>6</td>
<td>${4}$</td>
<td>3.756</td>
</tr>
<tr>
<td>Shaft Support</td>
<td>Fig. 75</td>
<td>10</td>
<td>527</td>
<td>6</td>
<td>all</td>
<td>8.567</td>
</tr>
<tr>
<td>Cell Phone</td>
<td>Fig. 73</td>
<td>12</td>
<td>872</td>
<td>6</td>
<td>${C_2, C_3}$</td>
<td>12.451</td>
</tr>
<tr>
<td>Cab</td>
<td>Fig. 71</td>
<td>40</td>
<td>24617</td>
<td>6</td>
<td>${C_3, C_4}$</td>
<td>276.742</td>
</tr>
</tbody>
</table>

Table 8. Summary of results for global SD algorithm: Computation Time
7.4.2 Computation Time

Table 8 lists the computation time of the GSD algorithm (steps 1-5 shown in Figure 35) for various test assemblies. The testing was performed on an Intergraph-TDZ-2000, Dual 450 MHz Pentium-II machine.

For the test assemblies in Table 8: Figure 81 shows a graph-plot between the number of faces \((F)\) in \(A\) and the accessibility computation-time (step 1 in the GSD algorithm). Figure 82 shows a graph-plot between the number of components \((n)\) in \(A\) and the sequence determination time (Steps 2-5 in the GSD algorithm). While the results show that the accessibility computation time and the sequence determination time increase with \(F\) and \(n\) respectively, it is difficult to ascertain the behavior given the oscillations in the data. For this purpose, log-log plots (Figures 83 and 84) are constructed and the data is observed to have a linear behavior for both the plots. The equations of the best fits are determined from the plots and are shown in the Figures 83 and 84. By observing the coefficients of these best fits (trend lines) the accessibility computation time and the sequence determination time are both polynomial.

The lack of uniformity in the distribution of the data points along the x-axis for the accessibility computation time (especially point 5 in Figure 83) is attributed to the fact that the assembly models tested were chosen at random. The accessibility computation-time depends on the number of components that are spatially constraining every other component in \(A\) and also the number of exact collision or intersection checking to be performed between the components in \(A\) for SD analysis. Since these values are different for the tested assemblies, the points seem fluctuating. Moreover, the accessibility-computations just involve either positive or negative answers. Therefore during accessibility computation the exact intersection region need not have to be computed and even a positive answer from simple collision checks based on extended
bounding-box checks between the components, can avoid hefty computation time for collision checking. Another point to note, is that the relationship between the number of faces and the computation time is not one-to-one, due to the same reason listed above; i.e. assembly models with same $F$ can have different accessibility computation time and vice-versa. However, the accessibility computation time is bounded by the worst case computation time $O(dF^2)$.

Similarly the lack of uniformity in the distribution of the data points along the x-axis for the sequence computation time (especially point 3 in Figure 84) is attributed to the fact that the assembly models tested were chosen at random. The $RG$ construction depends on the number of components that are spatially constraining every other component and also the number of components that can be simultaneously disassembled. Since these values are different for the tested assemblies, the resultant points seem fluctuating. Another point to note, is that the relationship between the number of components and the computation time is not one-to-one, due to the same reason listed above; i.e. assembly models with same $n$ can have different sequence computation time and vice-versa. However the sequence computation time is bounded by the worst case computation time $O(dn^3)$.

Based on these results, it is evident that the computation time for determining a non-interfering $SD$ sequence by the $GSD$ algorithm is polynomial. The Appendix section presents supplementary results of the $GSD$ algorithm for a pattern assembly in which the variations of the data-points will be less.
**Figure 81.** Accessibility computation time plot ($F$ vs Time) for GSD results

**Figure 82.** Sequence computation time plot ($n$ vs Time) for GSD results
Figure 83. Log-Log plot for the accessibility computation time with trend line

Figure 84. Log-Log plot for sequence computation time with trend line
7.5 User-Defined Constraints and Selective Disassembly

Using the interface provided in the A3D system the user can add user-defined constraints such as component grouping and directional constraints and can generate a new SD sequence for the modified inputs.

7.5.1 Grouping Components and Sequence Determination

A sub-assembly is treated as a single unit (component) by the GSD algorithm. Grouping of components into sub-assemblies may change the SD sequence and paths for the target component(s), which is presented in this section with the Aero-Engine SD results.

Figure 85. SD of the Aero-engine assembly for $C = \{rjf21\}$ from GSD
Figure 86. RG for the Aero-Engine assembly from GSD for C = \{rjf21\}

Figure 85 shows the SD of the Aero-engine assembly generated by the GSD algorithm for C = \{rjf21\} with no components grouped. The corresponding RG for C is shown in Figure 86, where S = \{rjf18, rjf16, rjf4, rjf5, rjf9, rjf8, rjf7, rjf6, rjf21\} with \(n_r = 9\) and \(n_s = n_w = 4\). The simultaneous disassembly groups are (rjf18, rjf16, rjf4), (rjf5), (rjf9, rjf8, rjf7, rjf6) and (rjf21).

The grouping of components into subassemblies change the assembly configuration and hence the geometric constraints. The user-interface for defining sub-assemblies using the A3D system is shown in Figure 87. With components grouped as a subassembly S1 = \{rjf5, rjf6, rjf7, rjf8, rjf9, rjf21\}, Figure 87 shows the SD result for C = \{S1\}. The corresponding RG for C is shown in Figure 88, where S = \{rjf18, rjf16, S1\}. The simultaneous component removal groups are (rjf18, rjf16) and (S1) with \(n_r = 3\) and \(n_s = n_w = 2\).
Figure 87. SD of the Aero-engine assembly from GSD for $C = \{S1\}$

Figure 88. RG for the Aero-engine assembly from GSD for $C = \{S1\}$
7.5.2 Directional Constraints and Sequence Determination

Adding/removing directional constraints to a component change the disassembly directions for that component and may affect the geometric constraints imposed on the other components in the assembly, which is presented in this section with the Aero-Engine SD results.

![Diagram of Aero-engine assembly](image)

**Figure 89.** SD of the Aero-engine assembly from GSD for $C = \{ \text{rjf20} \}$

![Diagram of RG](image)

**Figure 90.** RG for the Aero-engine assembly from GSD for $C = \{ \text{rjf20} \}$
Figure 89 shows the SD of the Aero-engine assembly generated by the GSD algorithm for $C = \{\text{rfj20}\}$ without any directional constraints applied to any components. The corresponding RG for $C$ is shown in Figure 90, where $S = \{\text{rfj11, rfj10, cy, rfj20}\}$ with $n_r = 4$ and $n_s = n_w = 2$. The simultaneous disassembly groups are $(\text{rfj11, rfj10, cy})$ and $(\text{rfj20})$.

Figure 91. SD of the Aero-engine assembly from GSD for $C = \{\text{rfj20}\}$ with cy constrained

However if the user constrains cy from moving, then the resultant $S = \{\text{rfj13, rfj12, rfj11, rfj10, rfj22, rfj20}\}$ generated for $C = \{\text{rfj20}\}$, as shown in Figure 91. Figure 91 also shows the user-interface for defining directional constraints. The corresponding RG for $C$ is shown in Figure 92, where the simultaneous disassembly groups are $(\text{rfj13, rfj12, rfj11, rfj10})$, $(\text{rfj22})$ and $(\text{rfj20})$ with $n_r = 6$ and $n_s = 3$. 
Discussion: Performance Boosting

In the process of generating a feasible SD solution, the designer may be performing several iterations and variations in the product design. Especially when performing SD analysis in the A3D system with a VR interface or Web interface, the time taken for SD analysis and animation of SD results becomes an important issue. The techniques implemented in the A3D system to improve the computation and simulation performances are discussed below.

7.6.1 Computation

The SD sequence generation from the GSD algorithm involves three steps: (I) computing the accessibility of the components, (II) construction of a Removal Influence (RG) graph, and (III) determination of a SD sequence from RG. The computation time for these steps are $O(dF^2)$, $O(dn^3)$ and $O(n^2)$ respectively. Therefore an incremental approach has been utilized for SD analysis.

- While performing a SD analysis, the accessibility of the components is computed and stored along with the A3D file, thereby reusing the accessibility results for further analysis.

Figure 92. RG for the Aero-engine assembly from GSD for $C = \{rjf20\}$ with cy constrained
• When the target component(s) are changed, then only Step III is recomputed.

• When the user-defined constraints or the shape of the components are modified, then only steps II and III are recomputed after incrementally updating the accessibilities of the corresponding components.

For example, Table 9 shows the computation time in determining a SD solution of an Aero-engine assembly (Figure 85) for various changes in the design inputs. For this example, Steps I, II and III together takes 2182.654 seconds. By following the above incremental approach, the results show a drastic reduction in the computation time, as shown in the Table.

<table>
<thead>
<tr>
<th>Assembly</th>
<th>Figure</th>
<th>Operation before SD</th>
<th>Computation Time</th>
<th>% Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aero-Engine</td>
<td>Fig. 85</td>
<td>Changing the target component from $C = {rjf21}$ to $C = {rjf20}$</td>
<td>$&lt; 10^{-3}$ seconds</td>
<td>99.9999</td>
</tr>
<tr>
<td></td>
<td>Fig. 89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fig. 85</td>
<td>Grouping components such that $C = {S1}$</td>
<td>0.021 seconds</td>
<td>99.9990</td>
</tr>
<tr>
<td></td>
<td>Fig. 87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fig. 89</td>
<td>Adding directional constraints to cy; $C = {rjf20}$</td>
<td>0.029 seconds</td>
<td>99.9986</td>
</tr>
<tr>
<td></td>
<td>Fig. 91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fig. 85</td>
<td>Changing the shape of $rjf16$; $C = {rjf21}$</td>
<td>148.34 seconds</td>
<td>93.2036</td>
</tr>
</tbody>
</table>

Table 9. Incremental approach and the computation time for sequence generation

7.6.2 Graphics

Minimizing the time lag between the user-input and the VR system response can achieve the objective of real-time simulation. From the software perspective, the following techniques have been implemented in the A3D system.
• **Level of Detail**: Multiple representation of shapes with different resolution (or triangulation) are stored and swapped for display based on the eye position of the user and the distance with respect to the geometrical shape in the virtual environment.

• **Culling**: Selective display of components of interest to the user and either not showing the rest of the components or showing only an abstract shape, during simulation.

• **Wireframe/Shading**: Displaying some components in wireframe and some shaded, based on the choice of the user.

Testing on the **A3D** system showed an increase in animation speed of about 10 fold with Level-of-detail technique, 3 fold with culling technique and 5 fold with wireframe/shading.
CHAPTER 8.  APPLICATION, RESULTS AND DISCUSSION

This section presents the application results of the SD technology for product maintenance, assembling and recycling.

8.1 Application: Maintenance and Service

Figure 93. Aircraft sub-assembly: SD of the engine for in-place maintenance

Product maintenance requires removal of certain components in A for in-place or for replacement maintenance. Therefore performing SD analysis allows determination of a SD
sequence with fewer numbers of removals to reach the component(s) to be serviced, i.e. with less time. Moreover the SD analysis provides the user with information regarding the feasibility of disassembling the components for maintenance, accessibility of components and the disassembly time involved.

Application of the SD analysis for in-place and replacement maintenance is illustrated, with a test result for plane engine maintenance. Figure 93 shows the SD of one of the outer shell (C4) of the engine generated by an automated SD method followed by the engine component disassembly for in-place maintenance. For example, the disassembly of rotor (C5) is shown in Figure 94, generated by sequence and path editing in the A3D system.

![Figure 93. SD of one of the outer shell of the engine](image1)

**Figure 94. SD of the rotor (C5) from the Aircraft sub-assembly for in-place maintenance**

For the same plane assembly, Figure 95 illustrates the test result for the replacement maintenance. Firstly, the aircraft engine (S1) is disassembled as a single unit from the aircraft
subassembly with $S = \{C_1, C_2, C_3, S1\}$, with $n_r = 4$ and $n_s = 3$, where the component grouping and automated SD is performed. Secondly, the rotor unit ($C_6$) is disassembled from $S1$ (defined as a new assembly), where $S = \{C_4, C_5, C_6\}$, with $n_r = 3$ and $n_s = 3$.

One key observation is that the components that need to be disassembled for maintenance depend on whether the requirement is in-place or replacement maintenance. Accordingly, based on whether it is in-place or replacement maintenance, the SD sequence and the time will be different. Therefore, if both options are evaluated then the selection of one over the other can be made, based on the SD evaluation results.

![Aircraft sub-assembly](image1)

Figure 95. Aircraft sub-assembly: SD of the engine for replacement maintenance

Another result showing the sequence and time evaluation for the SD of the instrument panel from the Dashboard assembly of an automobile for service is presented below. Figure 96
shows the Dashboard assembly and the selection of the target component $C_{16}$ (Instrument Panel) for SD. The SD is shown in Figure 97 with $S = \{C_4, C_{10}, C_{11}, C_8, C_9, C_{12}, C_{16}\}$.

**Figure 96.** Dashboard sub-assembly: selection of the Instrument panel ($C_{16}$) for service

**Figure 97.** Dashboard sub-assembly: SD of the Instrument panel ($C_{16}$) for service

Figure 98 shows the cumulative SD time plot points generated by the A3D system. The SD time = 177.5 seconds for sequential disassembly; which includes removal of 10 screws (140
seconds), 2 pins (8 seconds), 1 nut (4 seconds), 2 clips (8 seconds) and 7 components (17.5 seconds) in S.

![Components in the Selective Disassembly Sequence](image)

**Figure 98.** Dashboard assembly: disassembly time graph for maintenance

In this example, S is practically feasible and also efficient if components are assumed to be rigid. However, the solution determined is not a minimum time sequence. In practical situations, C₁₂ does not have to be removed, since C₁₂ is of a flexible material, it can be bent and C₁₆ may be removed directly after removing C₉. Therefore the minimum disassembly time sequence for sequential disassembly is \{C₄, C₁₀, C₁₁, C₈, C₉, C₁₆\} with disassembly time = 171 seconds; which excludes removal of a clip (4 seconds) and C₁₂ (2.5 seconds).

### 8.2 Application: Assembling

As previously noted, a selective (or complete) disassembly sequence can be reversed and used for assembling components. Evaluation of an assembly in the A3D system for SD analysis provides the user with information regarding the feasibility of assembling the components, the accessibility of components as also the sequence and paths to assemble/disassemble the
components in an assembly. For example, Fig. 100 shows an exemplary A3D result for the Augmentor sub-assembly illustrating the simultaneously assembling of components.

Figure 99 shows a Java applet generated by the A3D system for visualization. Since the A3D system allows rapid and full visualization of assembly and disassembly, it may be fully integrated and considered in engineering analyses of product assemblies.

Figure 99. A3D applet for visualization of assembly/disassembly
Figure 100. Augmentor: simultaneous assembling of components
### 8.3 Application: Recycling and Reuse

Performing SD analysis for recycling allows determination of a SD sequence for separating components of different materials (target components). Moreover easier and quicker separation of materials for recycling yields profit, resulting in greater impetuses for the companies to recycle a product. In addition, the SD analysis may facilitate the companies to evaluate the disassembly cost, if and when the product is disassembled for recycling/reuse.

![Dashboard assembly: SD of PVC and nylon components for recycling](image)

**Figure 101.** Dashboard assembly: SD of PVC and nylon components for recycling

**Example:** An exemplary result is shown in Fig. 101 for the SD of Nylon and PVC components for recycling from the Dashboard assembly (generated by the A3D system). Figure 101 shows the generated $S = \{C_5, C_6, C_{18}, C_{19}, C_4, C_{10}, C_{11}, C_8, C_9, C_{12}\}$ for SD of Nylon ($C_5, C_6, C_9, C_{18}, C_{19}$) and PVC ($C_{12}$) material components.
Figure 102. Dashboard assembly: value-sequence graph for recycling

Figure 102 shows the cumulative cost and return-value plot points generated by the A3D system. For this example, the disassembly cost-computation procedure listed in Section 7.1.7 is used with LR = 20$/hr. The total disassembly cost = $1.84; which includes removal of 10 screws (140 seconds), 4 clips (16 seconds), 9 pins (36 seconds), 1 nut (4 seconds), and 10 components (25 seconds) in S.

The recycling return values and the disposal cost values are incorporated as follows. Nylon and PVC have a high net (accounting for the processing cost) material recycling value of $.516/lb and $.20/lb respectively. The cumulative return value plot (Figure 102) shows the recycling value obtained by disassembling and recycling of Nylon and PVC components. For example, a return value of $2.78 at C_{12} indicates the net value in recycling C_5 to C_{12}, except C_4, C_{10}, C_{11} and C_8 (which is neither PVC nor Nylon).
The cumulative disassembly and disposal cost plot shows the cost of disassembling the components and disposing off the rest of the assembly. For example, the American State of Wisconsin is used as a location for disposal, where the standard disposal cost for non-hazardous material is $.015/lb. A cumulative cost of $2.65 at $C_{12}$ indicates the total disassembly cost of disassembling components from $C_5$ to $C_{12}$ as well as the disposal of all the components in the assembly except for the PVC and Nylon material components.

The profit in $SD$ and recycling/disposing off the components are available from the value-sequence plot (Figure 102). For example, $SD$ of Nylon and PVC material components and the disposal of the rest of the assembly results in a total profit of $.13 (= $2.21/hr) with a net recycling value of $2.78 and a total disassembly/disposal cost of $2.65.

In this example, $S$ is practically feasible and also economical. However, from the value-sequence graph, a sub-set of $C$ for a profitable recycling can also be obtained. For example, in Figure 102, at $C_{19}$ the total profit is $.20 (= $9.23/hr) with the net recycling value = $1.69 and the total disassembly/disposal cost = $1.49. To achieve this locally optimum profit, components $C_5$ to $C_{19}$ are disassembled, the Nylon material components \{$C_5$, $C_6$, $C_{18}$, $C_{19}$\} are recycled, and the rest of the assembly is disposed off.
CHAPTER 9. DISCUSSION AND CONCLUSION

This chapter summarizes the research findings and contributions of the current research and discusses the future work.

9.1 Contributions

The first contribution of this research is the new concept of *disassembly waves* (represented as a removal influence graph) for selective disassembly analysis. The disassembly wave embeds the assembly geometry and topologically arranges components denoting the disassembly order, thereby enabling determination of a selective disassembly sequence, as showed by the test results from the developed prototypical software system.

The development of the contact-based and global *Selective Disassembly Algorithms* for efficient selective disassembly constitutes another contribution of this research. As the test results showed, within the class of problems analyzed and the assumptions for the current research, the proposed algorithms enable determining a selective disassembly solution with fewer removals computed in a feasible computation time. The *GSD* algorithm determines a non-interfering SD sequence with minimal simultaneous component removals in a polynomial computation time. For the contact-based analysis, the *SWP* and *MWP/PIE* algorithms determine fewer removal sequences for single and multiple component selective disassembly, respectively.

Another contribution of this research is the development of a new design-tool (*Assembly and Disassembly in Three Dimensions, A3D*) which enables performing automated and interactive virtual selective disassembly analysis to support maintenance, recycling and assembling applications. The proposed design tool geometrically reasons about the assembly model and allows the designer to generate, edit, validate, evaluate and animate selective
disassembly sequence and paths. While there exist design-aids to perform disassembly analysis, some are not automated and the rest perform complete disassembly of components, and not the selective disassembly analysis. Moreover, unlike modeling the user-defined constraints as filters (as done by some of the existing research) which involves generation of all the solutions, the A3D system incorporates the user-defined constraints such as component grouping and directional constraints directly in the algorithm, that enables modeling application knowledge in the selective disassembly analysis.

9.2 Future Work

A discussion on the limitations of the current research and the future topics of research in virtual SD analysis is presented in this sub-section.

9.2.1 Limitations Due to the Assumptions and Relaxation

One limitation of the current approach (due to the Assumption 1, Section 1.4) is that the components are assumed to be free-floating and the tools/fixtures required to achieve disassembly are not analyzed. With the incorporation of directional-constraints in the A3D system, the fixture elements may be either modeled as a component with directional constraints or may be represented as directional constraints to the components in the assembly. However it has been a regular practice [9] to perform disassembly sequencing (a Macro assembly analysis) first followed by tools/fixtures planning (a Micro assembly analysis). Several techniques available for tool/fixtures planning are listed in the assembly planning bibliographies [41-43].

A second limitation (due to the Assumption 2, Section 1.4) is that the assemblies are assumed to be rigid, frictionless and defined by nominal geometry. However, relaxation of this assumption in the SD algorithms may require a new procedure for computing the geometric-
attributes by representing flexible assemblies [85], friction [86], gravity [87] and the variational geometry due to tolerance [88].

A third limitation (due to the Assumption 3, Section 1.4) is the assumption of simpler disassembly paths for automated disassembly. While this assumption may be appropriate [10, 11], some assemblies may require complex paths in disassembling the target components, as shown with the plane sub-assembly example (in Section 8.1). Therefore two potential solutions, using the A3D system interface, are proposed: (I) The user pre-defining the required path in the universe of disassembly directions and (II) The user editing the sequence and paths in generating a complex-path for the component. The future work includes a modified procedure for the accessibility and disassemblability of components based on a global path planning technique (such as [65]).

A fourth limitation (due to the Assumption 4, Section 1.4) is that the disassembly sequences are monotonic and non-destructive. Analysis of intermediate states in disassembly operation [89] and inclusion of destructive means [13] for SD, represent future research.

9.2.2 Contact-Based Analysis and Global Disassembly

An inherent assumption of contact based disassembly, results in non-detection of interference during linear global disassembly motions. While the GSD algorithm ensures a non-interfering SD sequence, the contact-based algorithms result in a local disassembly of components. Therefore the methods extending SWP, MWP and PIE algorithm concepts for global-disassembly have to be researched.

Several of the CD methods [9] are all contact-based. However some of the extension procedures followed by other researchers to incorporate global-disassembly for CD is as follows: (i) part sweeping and collision detection [73], (ii) virtual contacts to express global constraints
between parts [9], and (iii) creating virtual mating based on convex hull and relative convex hull of parts [55]. Incorporation of such techniques for SD is a topic for future research.

Figure 103. Test assemblies to illustrate path-containment and component-containment

One potential approach proposed by the current research, to ensure a non-interfering SD sequence for the contact-based SD analysis is a modified procedure for disassemblability and removal-influence of components; hence it is illustrated with the following example. Consider Figure 103(a) with \( C = \{ C_1 \} \). From the contact-based analysis: \( \Delta_1 = \text{FALSE}, \Delta_2 = \text{TRUE}, RI_1^2 = \text{TRUE} \). In this example, \( C_2 \) is disassemblable globally along \( d_1 \); however \( C_1 \) is locally disassemblable (an infinitesimal translation is possible) and globally constrained by \( C_3 \). The current research introduces a notion of ‘component-containment’; e.g. \( C_1 \) is disassemblable globally since \( C_1 \) can be fully contained within \( C_2 \) by moving \( C_1 \) along \( d_2 \) direction. Since the component-containment for \( C_1 \) with respect to \( C_2 \) is TRUE, the problem of removing \( C_1 \) is reduced to the problem of removing \( C_2 \). Thereby following the above procedure, \( C_2 \) is disassemblable globally along \( d_1 \) and \( C_1 \) along \( d_2 \) followed by \( d_1 \).
Now consider Figure 103(b) where $C_1$ cannot be contained within $C_2$. For this case the current research introduces another concept called ‘path-containment’: Since the component-containment of $C_1$ with respect to $C_2$ is FALSE, $C_1$ is globally disassemblable along $d_2$, if $C_2$ is also disassembled along $d_2$. The direction $d_2$ is derived from the attribute $\text{RI}_1^2$ which is TRUE along $d_2$. Thereby, following both the concepts, the disassembly sequence is \{ $C_3$, $C_2$, $C_1$ \}.

Therefore, by combining the global-disassemblablility, component-containment and the path-containment concepts, the global-disassembly of components for contact-based analysis within the class of single dependent components may be ensured. However the above concepts require further analysis and represent future research.

### 9.2.3 Knowledge-Assisted Selective Disassembly

The application knowledge base (or rules) may dictate certain disassembly constraints, based on the knowledge gained about a particular domain [90]. The current research provides component-grouping and directional constraints as user-defined constraints, which may model some of the application knowledge in the SD analysis. However the extension of the above concept of knowledge-assisted SD has to be further researched.

![Figure 104. Disassembly constraints (precedence relation)](image)

For example in an Aero-Engine disassembly, the knowledge base may dictate a constraint “disassemble the fan blade before the hub”--a precedence relation. The constraints imposed on
disassembly, can be a hard constraint (must be followed) or a soft constraint (suggestions). To illustrate this, consider the concept diagram of $A$ in Figure 104 with $C_x$ to be disassembled. Examples for hard constraints are “remove $C_y$ before $C_x$” and “remove $C_z$ first”. In this example, due to the hard constraints, three components must be removed; $C_z$ followed by $C_y$, which is followed by $C_x$. One potential approach is as follows: Applying the GSD algorithm first with $C_z$ as the target and $\{C_x, C_y\}$ constrained; removing all the components in the sequence from $A$; then applying the GSD algorithm with $C_y$ as target and $C_x$ constrained; removing all the components in the sequence from $A$; and then applying the GSD algorithm with $C_x$ as the target. However, if there are soft constraints such as “prefer removing $C_y$ before $C_x$” then the possibility of removing $C_y$ before $C_x$ has to be analyzed. Moreover, if along with the constraints there are other objectives such as minimal component removals, then the solution to this problem is not straightforward. Performing knowledge-assisted SD (knowledge available as rules or disassembly procedure) is a topic for future research.

### 9.2.4 Other Topics for Future Research

An important enhancement for SWP, MWP and PIE algorithms will be the incorporation of multi-dependent components in the analysis and determining a minimal removal sequence in a feasible computation time.

While the current research performs SD analysis with an objective of minimal removals, other objectives such as minimal cost has to be researched. One potential approach to determine $S$ with an objective such as minimal cost or weight etc. is first to determine a set of sequences, $\{S\}$, from disassembly waves and then determining $S$ that satisfies the required objective from $\{S\}$[25]. Although this approach may not guarantee an absolute optimum solution, it allows an initial pruning of the solution space based on the removals and is also computationally efficient
than enumerating all the possible sequences. However generalization for optimal SD sequencing requires further analysis and is a topic for future research.

Another area for future work is the determination of sub-assemblies for disassembly using the geometric-reasoning techniques. One other area for future research is the development of computationally efficient procedures for collision checking to realize real-time simulation.

9.3 Conclusion

In conclusion, the principal motivation for this research was to develop a design-aid for selective disassembly analysis and it has been shown that the developed selective disassembly methods and the geometry reasoning techniques implemented in the prototypical software system (Assembly and Disassembly in Three Dimensions) allow realization of such a design analysis tool.

The test results with the prototypical software developed in the current research, clearly demonstrate the determination of a non-interfering (collision-free) selective disassembly sequence by the Global Disassembly Algorithm in a polynomial (feasible) computation time. Moreover the selective disassembly sequence determined by this algorithm has minimal number of simultaneous component removals in disassembling the target component.

The test results also showed that the Wave Propagation algorithms determine an efficient selective disassembly solution compared to the enumeration approach (based on the computation time) and the complete disassembly techniques (based on the number of removals). Moreover, for the class of single dependent disassembly of components, the single and multiple wave propagation algorithms ensure a minimal component removal sequences for single and multiple disassembly respectively. While the contact geometric constraints are determined in a less
computation time, the contact-based selective disassembly analysis may not ensure global disassembly of components and therefore the analysis of spatial constraints has to be explored.

The ability allowing the designer to generate, edit, validate, evaluate and animate the assembly and disassembly sequences of components by the developed prototypical software *(Assembly and Disassembly in Three Dimensions)* demonstrate its applicability for virtual selective disassembly analysis. The test results also demonstrate the applicability of the selective disassembly techniques for maintenance, assembling and recycling. Moreover the observation of the incremental computation of the geometric attributes in sequence determination shows the usefulness of the selective disassembly methods for distributed and real-time evaluation systems, as observed from the developed prototypical software’s with virtual reality interface and web-interface.

A key observation from the user-defined constraints is that the application knowledge may be modeled in the selective disassembly analysis. While the incorporation of grouping and directional constraints in the automation analysis is demonstrated in this research, other means to achieve knowledge-assisted selective disassembly have to be explored. Another observation to note, which is not captured in the presented results is that the developed prototypical software does not simply provide static and separate views of the assembled and disassembled CAD models. Rather it allowed the user to obtain an animated view of the assembled and disassembled models, a feature that is regarded to be particularly valuable.

While several topics of future research remain, the results of this thesis establish the feasibility of the proposed wave propagation algorithms and the developed prototypical software system as a potential automation tool for virtual selective disassembly analysis alongside its usefulness for the engineering applications.
REFERENCES


APPENDIX: SUPPLEMENTARY RESULTS

Figure 105. A Cube-Matrix assembly of 3x3 (= 9) components

Further to understand the behavior of the GSD, the algorithm was applied to an assembly of cubes arranged in a matrix fashion. For example, Figure 105 shows a cube-matrix assembly of 3x3 components. Different cube-matrix assemblies of various matrix sizes were analyzed. For each assembly the center component in the matrix was selected as C (for example, the center component for the 3x3 cube matrix assembly is shown in Figure 105 with a mark *) and the universe of disassembly directions as \{+X, -X, +Y, -Y\}. Figure 106 shows the effect of changing \(n_w\) on \(n_s\) for a cube-matrix assembly with \(d = 4\). The results show that the \(n_s = n_w\) for GSD.

Two variations of the above analysis were also performed.

- Figure 107 shows the effect of changing \(n\) on \(n_s\) for a cube-matrix assembly with \(n_w = 2\) and \(d = 4\). The result shows that the \(n_s\) is independent of \(n\) and is always equal to \(n_w\), which is 2 in this case.

- Figure 108 shows the effect of changing the position of C on \(n_s\) for a cube-matrix assembly with \(n = 35 \times 35\) and \(d = 4\). The target component is changed along the 18\(^{th}\) row from 1 to 35.
When the position of $C$ is changed the $n_w$ of $C$ is also changed and hence $n_s$, such that $n_s = n_w$. For example, when $C$ is at the 18th column in the 18th row, the $n_s = n_w = 18$. Similarly, when $C$ is at the 19th column in the 18th row, the $n_s = n_w = 17$.

The results show that the GSD algorithm determines a minimal $n_s$ solution for $C$. Moreover, $n_r \leq n$ for the cube-matrix assembly (Figure 109) and it is of order $O(n^{0.5})$, i.e, a SD solution.

Figure 106. Effect of $n_w$ on $n_s$ for a cube-matrix assembly by varying $n$ with $d = 4$

Figure 107. Effect of changing $n$ on $n_s$ for a Cube-matrix assembly with $n_w = 2$ and $d = 4
**Figure 108.** Effect of changing the position of $C$ on $n$, for a cube-matrix assembly along the 18th column with $n = 35 \times 35$ and $d = 4$

**Figure 109.** Effect of changing $n$ on $n_r$ for a cube-matrix assembly with $d = 4$
In a matrix assembly, the number of components spatially constraining every other component is the same for all the components. Moreover, since the shape of the components are the same, the accessibility computation time between any two spatially constraining components will also be the same in A. The test results for the cube-matrix assembly with C as the center component in A and \( d = 4 \), are as follows:

- The accessibility computation-time plot (Figure 110) for a cube-matrix assembly shows a polynomial behavior (order 2) with respect to F.
- The effect of changing the number of disassembly directions on the accessibility computation time (Figure 111) for a cube-matrix assembly of size 33x33, shows a linear behavior.
- The computation time plot to construct RG (Figure 112) from accessibility for a cube-matrix assembly, shows a polynomial behavior (order 2) with respect to n.
- The sequence computation time plot (Figure 113) from RG for a cube-matrix assembly, shows a linear behavior with respect to n.

Based on the results, we can observe a polynomial computation time for the GSD algorithm.

![Graph](image)

**Figure 110.** Accessibility computation time plot (F vs Time)
**Figure 111.** Accessibility computation time plot ($d$ vs Time)

**Figure 112.** RG computation time plot ($n$ vs Time)
Figure 113. Sequence computation time from RG (Plot $n$ vs Time)