Improving Assembly Precedence Constraint Generation by Utilizing Motion Planning and Part Interaction Clusters

Carlos Morato\textsuperscript{a,b,*}, Krishnanand Kaipa\textsuperscript{a,b}, Satyandra K. Gupta\textsuperscript{a,b,**}

\textsuperscript{a}Department of Mechanical Engineering
\textsuperscript{b}Maryland Robotics Center
University of Maryland, College Park, MD 20742, USA.

Abstract

In this paper, we present a technique that combines motion planning and part interaction clusters to improve generation of assembly precedence constraints. In particular, this technique automatically finds, and clusters, parts that can mutually affect each other’s accessibility, and hence may impose assembly constraints. This enables generation of accurate precedence constraints without needing to examine all possible assembly sequences. Given an assembly model, our technique generates potential disassembly layers: Spatial clustering is used to generate part sets. Next, motion planning based on rapidly-exploring random trees (RRT) with multiple trees is used to evaluate the interaction between these part sets. Specifically, motion planning is used to determine which part sets can be removed from the assembly. These sets are added to the first disassembly layer and removed from the assembly. Part sets that can be removed from the simplified assembly are then added to the second layer. If the process gets stuck, parts in the parent set are regrouped, and the process continues until all disassembly layers are found. The resulting structure reveals precedence relationships among part sets, which can be used to generate feasible assembly sequences for each part set and the whole assembly. We present theoretical results related to the algorithms developed in the paper. Computational results from tests on a

*Corresponding author
**Principal corresponding author

Email addresses: cmorato@umd.edu (Carlos Morato), kkaipa@umd.edu (Krishnanand Kaipa), skgupta@umd.edu (Satyandra K. Gupta)
variety of assemblies are presented to illustrate our approach.

Keywords: Assembly sequence planning; Motion planning; Part interaction clusters.

1. INTRODUCTION

Assembling a complex product requires careful planning (De Fazio & Whitney 1987; Lee & Shin 1990; Wilson 1992; Wilson & Latombe 1994; Gupta et al. 1997; Gupta et al. 1998, 2001; Zha et al. 1998; Jimenez 2011). Shapes and sizes of parts in the assembly impose restrictions on the order in which assembly operations can be performed. In order to generate a detailed assembly plan, we need to first understand precedence constraints among assembly operations and be able to generate feasible sequences that are consistent with precedence constraints.

Intuitively, assembly precedence can be determined by analyzing accessibility (Lee & Shin 1990; Laperriere & ElMaraghy 1992; Wilson 1992; Wilson & Latombe 1994; Rajan et al. 1999). Notions of semi-infinite accessibility and infinitesimal-motion are well understood. Unfortunately, requirement of semi-infinite accessibility leads to elimination of feasible operations. On the other hand, feasibility of infinitesimal-motion alone does not guarantee that the assembly operation will be feasible. So these notions are not very useful in assembly sequence determination. However, representing accessibility in finite space is computationally challenging. So we instead would like to rely on motion planning to ensure that a proposed assembly operation is feasible.

The goal of assembly sequence planning is to generate a sequence of operations to construct a product from its individual parts. The complexity of this problem is proportional to the number of parts in the assembly (Wilson & Latombe 1994; Goldwasser et al. 1996). However, a product may be composed of a hierarchical structure, in which parts that mutually affect each other’s accessibility belong to a common cluster that form a part set; in turn multiple part sets may mutually affect each other’s accessibility constraints in the next level of assembly. Parts in such sets can be separately assembled before the final assembly of the product. Hence, the main goal of sequencing the individual parts can be divided into several subgoals that will reduce the number of assembly operations involved in each subgoal. However, the information about the interaction between part sets is not known beforehand. Therefore, we introduce a methodology to automatically detect
the part interaction clusters in a product so that the assembly sequencing problem can be applied to part sets at multiple levels of hierarchy.

We start by grouping individual parts into spatial clusters based on the proximity between parts. Each cluster identified in this way is treated as an individual part set. Now, we consider the whole assembly and use motion planning to determine which part sets can be removed from the assembly. These sets are added to the first part set removal layer and removed from the assembly. We again determine which part sets can be removed from the simplified assembly. These are then added to the second layer and removed from the assembly. If the process gets stuck, parts in the parent set are regrouped, and the process is continued until all parts have been removed from the assembly. This information is used to impose precedence constraints among the part sets. Finally, we generate feasible assembly sequences using the precedence constraints for each part set and the whole assembly.

Recent advances in rapidly-exploring random trees (RRT) based motion planning (LaValle & Kuffner 2001) enable efficient generation of motion plans in highly crowded scenes. However, parts in their final positions in the assembly have very limited amount of feasible motions. Hence, we need to ensure that moves being tried by the RRT-based motion planner are able to correctly assess motion feasibility. This requires us to create multiple trees to ensure that if a feasible path exists, then we can find it with a very high probability.

Implementing a system based on the above ideas requires tuning a large number of parameters to ensure reasonable system performance on moderately complex assemblies. This paper describes our approach for combining RRT-based motion planning and part interaction cluster detection for generating improved precedence constraints, which in turn can be used to generate feasible assembly sequences.

The input to the system is a 3D assembly comprising a set of parts that are described by their geometric models and relative positions. Our approach takes into account product characteristics such as component geometry, interrelationships between components, component material, and tolerances. We assume that the product is made of rigid parts and restrict ourselves to finding a feasible sequence of collision-free motions for each part (part set). We define each part (part set) as a free-flying object. Therefore, we do not consider grasping of the objects, the forces involved, or the stability of the part (part set) during motion. Additionally, we consider non-linearity in the assembly operations: an assembly operation may involve more than two part
sets at the same step. We assume only monotone assembly sequences: when an operation has assembled a part into a part set, that part may no longer be moved relative to that part set. Although restrictive in application, these assumptions are very common in assembly sequencing and can be applied to a majority of products.

The approach of assembly-by-disassembly relies on constructing a disassembly sequence and then reversing the entire sequence to obtain a feasible assembly sequence. In general, the steps involved in the sequences are not necessarily symmetric, for instance, when considering flexible parts which may undergo deformation during assembly. However, under our assumptions, these two operations are indeed symmetrical.

As mentioned earlier, the main challenges encountered in assembly sequence planning include sequence generation, combinatorial explosion, non-linearity, and interaction between part sets. The main contributions of this paper that address these problems include the following:

1. A spatial clustering based method to automatically detect the part interaction clusters in a product.
2. A technique that uses a variation of RRT-based motion planning in order to assess motion feasibility.
3. An algorithm to generate improved assembly precedence constraints by combining the part interaction cluster detection and the motion planning methods into an assembly-by-disassembly approach.

2. LITERATURE REVIEW

Research in automated assembly sequencing has rapidly increased over the past few decades (De Fazio & Whitney 1987; Homem De Mello & Sanderson 1991; Hoffman 1991; Krishnan & Sanderson 1991; Lee & Shin 1990; Wilson 1992; Zhang et al. 2005; Su 2007, 2009; Chen et al. 2010). The problem of finding a valid assembly sequence for general cases that allow complex combination of motions was shown to be impractical, primarily owing to the issue of combinatorial explosion (Kavraki et al. 1993; Kavraki & Kolountzakis 1995; Wilson et al. 1996). This led to an increase in the number of assumptions made in order to address restricted cases of the assembly sequence planning. Examples include monotone assembly sequences (each assembly operation leads to a final subassembly), two-handed assembly sequences (each assembly operation merges exactly two assembly parts
or components), and non-linear assembly sequences (at least two components are assembled at the same time).

The assembly process starts from an initial configuration in which all the parts are in a completely disassembled state and finishes with a goal configuration corresponding to the final assembly. An assembly sequence plan specifies the order in which each part and/or subassembly must be inserted into an incrementally expanding subassembly that eventually leads to the final assembly. The components to be assembled may be quite different from each other in terms of component geometry, precedence, accessibility, and other types of constraints. Therefore, assembly sequence planning for a 3D assembly, with large number of parts and complex assembly relationships between its individual parts, is a large-scale combinatorial problem (Goldwasser et al. 1996; Baldwin et al. 1991). Although the associated constraints play an important role in limiting the number of sequences (Kavraki et al. 1993), there are numerous combinations of valid feasible assembly sequences (Homem De Mello & Sanderson 1991). Among these, some sequences optimize the assembly process with respect to one or more criteria (Lambert 1997, 1999, 2002, 2006).

Our literature survey presents several key approaches in the field and describes methods that are closely related to the techniques presented in this paper. Interested readers may refer to some of the survey papers (Gupta et al. 1997; Zha et al. 1998; O’Shea et al. 1998; Lambert 2003; Jimenez 2011) for a more comprehensive coverage of the field.

2.1. Assembly-by-disassembly and Graph Theory based Approaches

Woo & Dutta (1991) used the “onion peeling” approach and introduced connection graphs for disassembly analysis. Later, this approach was used by Srinivasan (1998) to derive optimal disassembly sequences. Chen et al. (1997) used the onion peeling approach for parallel disassembly of components. Most existing methods are based on graph theory and involve additional information such as contact analysis, fastener matrix, disassembly precedence matrix, etc. (Gupta et al. 1997; Zha et al. 1998; Lambert 2003); standard graph search algorithms can be applied to graph representations of assembly structures in order to determine feasible or optimal assembly sequences (Selva et al. 2010). Wilson & Latombe (1994) introduced non-directional blocking graph, a compact data structure to represent a combinatorial set of part intersections for an assembly product, which implicitly contains the geometric constraints (Chen et al. 2002). Romney et al. (1995)
extended this approach into their Stanford assembly analysis tool (STAAT). Romney’s work mainly focused on developing geometric assembly planning models rather than on optimization aspects. Khosla and Roy developed an assembly sequence method from a 3D CAD model and exact geometry representation based face adjacency graph. The method used data related to the instances of parts and subassemblies to generate assembly sequences (Srinivasan 1998). A large number of parts in this representation leads to very expensive geometric tests with the costs of nearest neighbor function and the collision checking procedure representing the major bottle necks in performance (Chen 1999). A method to reduce spatial representation and computational time to perform assembly simulation was introduced by Iacob et al. (2011).

2.2. Artificial Intelligence based Approaches

De Mello and Sanderson formulated the assembly sequence problem as a discrete search and optimization problem. They proposed AND/OR graphs to represent precedence relations between parts (De Fazio & Whitney 1987; Homem De Mello & Sanderson 1991). Following this approach, a variety of artificial intelligence methods were proposed using genetic algorithms (Chen & Liu 2000; DeLit et al. 2001; Guan et al. 2002; Hui et al. 2008), neural networks (Sinanoglu 2005; Chen et al. 2008), data mining (Duflou et al. 2008).

2.3. Motion Planning based Approaches

Preliminary motion planning approaches were developed for integrated assembly simulation systems (Gupta et al. 1998, 2001; Thomas & Wahl 2010); the goal was to perform assembly operation planning that includes assembly sequence planning, tool planning, and task planning. Motion planning based on randomized methods have became popular in path planning. Algorithms such as probabilistic roadmaps (PRM) (Kavraki et al. 1996) and RRT (LaValle & Kuffner 2001) have been used to solve high DOF motion planning problems. However, current motion planners may not work well on part disassembly due to highly constrained environments that require the generation of a finer motion of the parts. Sundaram et al. (2001) described the disassembly planning problem as a repeated occurrence of “narrow” passages in the Configuration space that makes it impractical to use PRM for these problems. However, Le et al. (2009) performed assembly sequence analysis based on RRT by using collision models. They also showed the
complexity of the domain (Xiao & Ji 2001) and the possibility of improvements to the representation. In the same direction, Jaillet et al. (2008); Cortes et al. (2008) proposed some strategies to improve the performance of motion planning methods to solve the assembly sequencing problem.

2.4. Subassembly Detection Based Approaches

Dini & Santochi (1992) formulated one of the first approaches to detect subassemblies based on a mathematical model of the product. The model was a function of the product’s interference, contact, and connection matrices. Zhang et al. (2002) described a procedure to automatically derive the feasible assembly sequence and detect the subassemblies for automobile body assembly. The procedure is based on the definition of a connection matrix and a contracted matrix. These two matrices represent the precedence constraint knowledge among components and sub-assemblies. Wang & Chen (2005) constructed a hierarchical assembly system and used a matrix operation to generate assembly sequences. Later, Xing et al. (2007) extended this assembly modeling to completed types of subassemblies, and thus generated feasible assembly sequences automatically. However, in all these cases, a complete description of the assembly was needed.

3. ASSEMBLY PRECEDENCE CONSTRAINTS FRAMEWORK

On the one hand, assembly sequence planning is a large-scale combinatorial problem; however, on the other hand, the assembly precedence relationships between parts make it a highly constrained problem. The number of potential assembly sequences is given by $n!$, where $n$ is the number of parts in the assembly. This leads to a combinatorial explosion in the number of sequences. In addition, the linearity assumption of placing one part at a time may not be valid in complex assemblies where some parts cannot be singly added/removed (i.e., in isolation with respect to others) and the numbers of parts to be added/removed simultaneously is not always the same. These kind of non-linearities in assembly sequences increases the size of the solution space to $(2n−2)!/(n−1)!$. However, absolute constraints such as geometrical, precedence, and accessibility severely reduce the number of potential assembly sequences. Nevertheless, determining these constraints for an assembly problem dramatically increases the problem complexity. Assembly sequence planning was shown to be NP-complete (Kavraki et al. 1993; Wilson et al.
In our framework, we combine motion planning and part interaction clusters in order to derive precedence relationships that can be used to generate assembly sequences for complex assemblies. We consider a mechanical product as a hierarchical structure of part sets. Usually, parts in each set can be separately assembled before the final assembly of the product. Hence, the original goal of sequencing the individual parts can be divided into several smaller subgoals that will reduce the number of assembly operations involved in each subgoal. This, thereby, reduces the combinatorial explosion of the number of sequences that span the solution space. Since the structure of interaction between the part sets is not known beforehand, we introduce a methodology to automatically extract this information so that the assembly sequencing problem can be applied to part sets at multiple levels of hierarchy.

3.1. Problem Formulation

The input to the system is a 3D assembly model of a mechanical product, which is a geometrical representation of a set of individual parts (that constitute the product) in their assembled configuration. The output of the system is a set of precedence constraints that can be used to generate a feasible assembly sequence in which components are assembled to give rise to the desired product.

The CAD models used in this paper were obtained from the META team at Vanderbilt University. These models, originally created in SolidWorks,
Figure 2: Example of an error in the input CAD model of a complex chassis assembly caused due to intersection between two of its parts.

were converted into a stereolithography (.STL) format and used as inputs to the system. Given the assembly model in .STL format, the system automatically extracts the total number of individual parts $n$ and all the implicit geometrical information based on polygonal triangulation. An example of the polygonal triangulation generation of a simplified chassis assembly is shown in Fig. 1(a). The assembly model must satisfy some consistency requirements in order to have a feasible 3D workspace. For this purpose, all the assembly parts must respect their shape and volume as a rigid body in every pose; no intersections of part models in the assembled configuration are allowed. An example of error in the CAD model of a complex chassis assembly, which is caused due to intersection between two of its parts is shown in Fig. 2. These issues were resolved by manually adjusting the dimensions and positions of the part causing the intersection before using the assembled model as input to the system. For example, in Fig. 2, the CAD model of the spacer part (shown in red) was modified by changing its length and position in the assembly by trial-and-error until the part intersection was resolved. In addition, the position and orientation of each part in relative and absolute reference frames are automatically queried (refer to Fig. 1(b)) and used to compute the transformation between the two frames. All the
input data analysis, except resolving part intersection errors, was carried out automatically.

Next, we formalize the problem by laying out the assumptions underlying the framework, some definitions, and the problem statement as follows:

**Assumption 1.** All parts are rigid, all connections between parts are rigid, and once a connection or liaison is made, it remains in this way.

**Assumption 2.** Screws and nuts are included as members of the assembly; therefore, they belong to the assembly as parts. Any fastening method other than screw/nut must be removed from the 3D assembly model before our approach can be applied.

**Assumption 3.** The geometric disassembly and assembly precedence constraints are only based on the information present in the assembly model.

**Assumption 4.** The disassembly sequence is completely reversible to turn it into an assembly sequence.

We make the assumption 2 as we address only assemblies that use screws and nuts for fastening. Therefore, some modifications have to be made to our framework presented in the paper before it can be applied to assemblies that use other fastening methods. For example, if glue is used for fastening two parts, then this must be specified the input data. Then, this additional information can be used in the precedence constraints analysis accordingly.

**Definition 1.** Posture: The posture of a part $\omega$ is defined as $q^\omega = (p^\omega, \theta^\omega)$, where $p^\omega \in \mathbb{R}^3$ is the position $(x^\omega, y^\omega, z^\omega)$ in the Euclidian space and $\theta^\omega \in \mathbb{R}^3$ is the orientation $(\alpha^\omega, \beta^\omega, \gamma^\omega)$ in the Euclidian space.

**Definition 2.** Assembly: We define the assembly as a set of parts $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n \mid \forall \omega_i : q^{\omega_i} \in \mathbb{R}^6, c^{\omega_i} \in \mathbb{R}^3\}$, where $q^{\omega_i}$ and $c^{\omega_i}$ represent the posture and the center of mass of the part $\omega_i$, respectively. The set $\Omega$ includes all the parts in the input assembly.

**Definition 3.** Obstacles: We define a set of obstacles $O = \{o_1, o_2, \ldots, o_n \mid \forall o_i : q^{o_i} \in \mathbb{R}^6\}$, where $q^{o_i}$ represents the posture of an obstacle $o_i$. When we select a part to evaluate its motion feasibility, all the remaining parts of the assembly are considered as potential obstacles. Therefore, initially, the set $O$ includes all the parts in the input assembly.
Definition 4. Part set: A part set is defined as a set of parts and/or part subsets. A part set is considered to be a parent $C_p$ for a set of child sets $\{C^c_i : i = 1, \ldots, k\}$; in turn, each child set is a parent for subsequent child sets, recursively, until each child is a single part. Accordingly, it is clear that the root part set $C^r$ refers to the whole assembly $\Omega$.

Definition 5. Disassembly structure: The disassembly structure for a given $n$-part assembly $\Omega$ is defined as a set of hierarchical layers $H = \{h_1, h_2, \ldots, h_{|H|}\}$, where $h_i$ represents the $i^{th}$ disassembly layer and $|H|$ is the number of layers. Every layer is composed of its part sets $h_i = \{C_1, C_2, \ldots, C_{|h_i|}\}$, where every $C_i$ is defined according to Definition 4 and $2 \leq \sum_{i=1}^{\|H\|} |h_i| = n$.

Definition 6. Disassembly part set precedence $\prec$: Let $C^c_i, C^c_j \in C^p$. If $C^c_j$ can be removed only after removing $C^c_i$, then we say $C^c_i \prec C^c_j$.

Definition 7. Disassembly part set equivalence $\sim$: Let $C^c_i, C^c_j \in C^p$. If $C^c_i$ can be removed before $C^c_j$ and $C^c_j$ can be removed before $C^c_i$, then we say $C^c_i \sim C^c_j$.

Definition 8. Disassembly layer precedence $\prec\prec$: Let $h_i, h_j \in H$ such that $i < j$ (i.e., layer $h_i$ is generated before layer $h_j$), then we say $h_i \prec\prec h_j$. Moreover, if $C_i \in h_i$ and $C_j \in h_j$ then $C_i \prec\prec C_j$.

Definition 9. Disassembly layer equivalence $\approx$: Let $h \in H$. If $C_i, C_j \in h$, then we say $C_i \approx C_j$.

Definition 10. Assembly precedence $\rightarrow$: Let $C^c_i, C^c_j \in C^p$. If $C^c_j$ can be assembled only after assembling $C^c_i$, then we say $C^c_i \rightarrow C^c_j$.

Definition 11. Assembly equivalence $\leftrightarrow$: Let $C^c_i, C^c_j \in C^p$. If $C^c_j$ can be assembled either before or after assembling $C^c_i$, then we say $C^c_i \leftrightarrow C^c_j$.

Problem Statement: Given a 3D assembly model of a $n$-part mechanical assembly $\Omega$ (Definition 2), find a disassembly structure $H = \{h_1, h_2, \ldots, h_{|H|}\}$ (Definition 5) along with a set precedence relationships $P$ that can be used to generate feasible assembly sequences.

Next, we describe our overall approach to find a solution to the above problem.
3.2. Overview of Approach

Initially, with the assumption that the largest part of the assembly is the driving part that guides the assembly process, we extract this part from the given CAD model and keep it aside. If there are multiple largest parts, then heuristic information (e.g., which of these parts has maximum surface contact with the ground) is used to pick one of them. When there is lack of such information, one of them is picked randomly.

Next, we group the remaining \( n - 1 \) parts into \( k \) \((=2)\) part sets using spatial \( k \)-means clustering. Under this new arrangement, the assembly is composed of \( k + 1 \) part sets—the largest part, part set 1, part set 2, \ldots, part set \( k \)—in the first step. Now, we verify the assembleability of this new assembly. For this purpose, we use motion planning to determine which part sets can be removed from the assembly (explained in Section 4). These part sets are added to the first disassembly layer and removed from the assembly. We again determine which part sets can be removed from the simplified assembly. These part sets are then added to the second layer and removed from the assembly. If the process gets stuck before all part sets have been removed, we go to the first step, in which parts are rearranged into a different grouping by increasing the number of clusters by one. This results in \( k+1 \) new clusters. The cycle is continued until all part sets have been removed from the assembly and all part set removal layers are found. The techniques and algorithms used to extract part interaction clusters are described in detail, using an illustrative example, in Section 5.

Note that until now, the disassembly layers have been generated for the whole assembly while considering the identified clusters as individual part sets. Therefore, the above procedure is applied recursively to generate disassembly layers for each cluster identified in the previous step. The information extracted during the above procedure is used to impose precedence constraints among the part sets. Finally, we generate feasible assembly sequences using the precedence constraints for each part set and the whole assembly. Some theoretical results related to the properties of the proposed algorithms are presented in Section 6. Computational results from tests on a variety of assemblies are reported in Section 7.

Based on the complexity of the assembly, we cannot reject some part sets that do not pass the assembleability test. This occurs when at least two parts must be moved into their final assembly locations simultaneously. Therefore, whenever there is a failure, we recheck the assembleability for the largest assembly part and a single set obtained by merging the part sets. If
the assembly structure passes the test then a temporary assembly location is needed to generate a feasible assembly sequence. This process gives us the information about the existence of nonlinearity in the assembly. An example of this assembly scenario is explained in Section 7.5.

Next, we describe the principal techniques used to implement the above approach—motion planning, generation of disassembly layers, and spatial partitioning based part interaction cluster extraction.

4. MOTION PLANNING TO ASSESS FEASIBILITY OF PART SET DISASSEMBLY

The capability of sampling-based motion planners to perform assembly analysis of complex product models can be mainly attributed to their computational efficiency. For example, the Manhattan-like RRT based planner presented in Cortes et al. (2008) can handle models with hundreds of degrees of freedom. Therefore, as mentioned earlier, we rely on motion planning to evaluate the feasibility of an assembly operation. In particular, we developed a multiple RRT based motion planning algorithm to compute a collision-free escape path to move a part set from its assembled configuration to a given location that lies outside of the assembly.

Considering that the parts are moved and disassembled one by one, an assembly admits a disassembly sequence if an escape path for disassembling each part $\omega_i \in \Omega$ can be found. Given the initial assembled state configuration $\{q_{\omega_i}^{\text{initial}} : i = 1, 2, \ldots, n\}$, the problem consists of computing a collision-free escape path, from $q_{\omega_i}^{\text{initial}}$ to a disassembled configuration $q_{\omega_i}^{\text{goal}}$ for all $\omega_i \in \Omega$.

Rapidly-exploring random trees (RRT) based motion planning provides feasible solutions for part navigation in crowded scenes, a problem representative of searching non-convex, high-dimensional spaces. A RRT is constructed incrementally by randomly sampling valid configuration states and quickly computing the nearest neighbor to states that already belong to the tree. This process quickly reduces the expected distance of a randomly-chosen point to the tree. However, RRT cannot be directly applied to complex assemblies that are composed of large number of parts, which often lead to scenes with high obstacle densities. The large number of obstacles generates a highly constrained environment with very narrow passages, resulting in the RRT’s failure to generate a valid state. This, thereby, increases the false negatives dramatically.
In order to assess assembly feasibility, we used the RRT with multiple random trees based motion planning approach described in our earlier work (Morato et al. 2012). The approach performs a robust analysis of part motion feasibility. It uses multiple RRTs that dynamically modify the environment description in order to generate a valid escape path. It dynamically modifies the number of trees for each assembly part based on the environment constraints associated with the part. A highly constrained environment may require a large number of trees to find the escape path for an assembly part. Multiple RRTs provide greater robustness across narrow passages and crowded environments.

Strandberg (2004) proposed a multiple RRT algorithm, in which the number of trees is kept fixed. However, in our approach, the number of trees is changed on-the-fly; that is, a new tree is added only if it is required by merging the information about the “old” tree with the “new” tree instantaneously, thereby, increasing the probability of finding a valid state in the next step by using the knowledge of the previous trees.

5. GENERATION OF PART INTERACTION CLUSTERS

5.1. Basic Idea Behind Identification of Disassembly Layers

The motion planning technique uses the geometric information extracted from the assembly model to find an escape path that allows a part to be completely separated from the assembly. This enables the system to determine which part movements are forbidden, which movements are feasible, and thereby, determine which part(s) can be removed at each stage of disassembly. Using Definition 5, we consider a hierarchical disassembly structure $H = \{h_1, h_2, \ldots, h_{|H|}\}$, such that $h_i \ll h_{i+1}, \forall i \neq |H|$. Therefore, each layer $h_i$ represents a precedence constraint for the layer $h_{i+1}$ for disassembly.

For simplicity, we first describe the generation of disassembly layers without grouping part into clusters and by applying motion planning only to individual parts. Specifically, the system considers each part and uses the motion planning algorithm to check if it is physically blocked by another part before it can be fully removed out of the assembly. Parts that can be removed in this manner during the first attempt fill the first layer $h_1$. This process is repeated to fill the second layer $h_2$ and so on, until all the parts are disassembled. As a result, the process is bounded by a maximum of $n^2$ iterations. The resulting hierarchically layered structure comprising part groups that can be removed at each layer gives rise to disassembly precedence relations.
Algorithm 1 Assembly sequencing using DMRT based motion planning.

**Input:** $\Omega, O$

**Output:**
- Assembleability $\in \{\text{TRUE, FALSE}\}$,
- AssemblySequence

**Procedure** $\text{DMRT}(\Omega) \rightarrow (\text{DisassemblyLayer}, \mathcal{O})$

1. $O \leftarrow \mathcal{O};$
2. while $(|O| \neq 0)$ do
   1. $\tau$.
   2. $O \leftarrow \text{RemovePart}(O, \omega);$
   3. while $(\text{EscapePathFound} == \text{FALSE})$ do
      1. $q_{\text{rand}} \leftarrow \text{RandomStage}();$
      2. for each TREE do
         1. $q_{\text{near}} \leftarrow \text{NearestNeighbor}(q_{\text{rand}}, \tau);$
         2. $u \leftarrow \text{SelectInput}(q_{\text{rand}}, q_{\text{near}});$
         3. $\text{CheckCollision}();$
         4. if $(\text{Collision} == \text{FALSE})$ then
            1. $q_{\text{new}} \leftarrow \text{Newstate}(q_{\text{near}}, u, \Delta t);$
            2. $\tau$.
            3. $\tau$.
         else
            1. $\text{ReinitializeNewTree}();$
      end if
   end for
3. end while
4. if $(\text{EscapePathFound} == \text{TRUE})$ then
   1. $\text{DisassemblyLayer} \leftarrow \text{AddPart}((\text{DisassemblyLayer}, \omega);$
   2. $O \leftarrow \text{AddPart}(O, \omega);$
   3. $\mathcal{O} \leftarrow \text{RemovePart}(\mathcal{O}, \omega);$
   4. $|\text{Paths}| + +;$
5. end if
6. end while
7. $\mathcal{O} \leftarrow \Omega; i = 1; |\text{Paths}| = 0;$
8. while $(|\mathcal{O}| \neq 0)$ do
   1. $(h_i, \mathcal{O}) \leftarrow \text{DMRT}((\mathcal{O});$
   2. $H \leftarrow \text{AddLayer}(H, h_i);$
9. end while
10. if $(|\Omega| == |\text{Paths}|)$ then
    1. $\text{AssemblySequence} \leftarrow \text{ReverseSequence}();$
    2. $\text{Return} (\text{TRUE}, \text{AssemblySequence})$
else
    1. $\text{Return} \text{Assembleability} = \text{FALSE}$
end if
Figure 3: (a) 23-part assembly. (b) Exploded view. (c) Generation of disassembly layers for 23-part assembly.

Generation of such disassembly layers for a chassis assembly of 23 parts is shown in Fig. 3.

The disassembly layers generated by the above process can be reversed and turned into a linear assembly sequence as shown in Algorithm 1. However, the above process does not fully represent the precedence relations between parts across different layers. For instance, $H$ provides precedence relation between layers $h_i \prec≺ h_j$ for disassembly. Therefore, if $\omega_i \in h_i$ and $\omega_j \in h_j$, then $\omega_i \prec≺ \omega_j$. That is $\omega_i$ can be removed before $\omega_j$. However, this doesn’t necessarily imply whether or not $\omega_j$ can be removed before $\omega_i$. Consequently, we do not know for sure if $\omega_i \Rightarrow \omega_j$ for assembly. Also, the precedence relations between the parts in the same layer are not evaluated by this method.

In the next subsection, we show how the assembly structure identified by
Figure 4: Definition of the centers of mass \( L = \{c_1^\omega, c_2^\omega, \ldots, c_n^\omega\} \) based on the 3D assembly model. \( L = \{(0, 26.85, -232.28), (0.22, 107.425, -360.295), \ldots, (0, 55.35, -323.785)\} \).

Our new approach reveals improved precedence relationships by reorganizing the parts in different layers into different part sets.

5.2. Spatial Partitioning based Part Interaction Cluster Formation

We developed a spatial partitioning algorithm based on \( k - \) means clustering technique in order to generate part interaction clusters. K-means clustering (MacQueen, 1967) is a method commonly used to automatically partition a data set into \( k \) groups. It proceeds by selecting \( k \) initial cluster centers and then iteratively refining them as follows: (a) Each instance \( c_i^{\omega} \) is assigned to its closest cluster. (b) Each cluster center \( C_j \) is updated to be the mean of its constituent instances. The algorithm converges when there is no further change in assignment of instances to clusters or alternatively, when there is no change in the location of the cluster centers. The pseudocode for spatial clustering is given in Algorithm 2.

We illustrate the part interaction cluster extraction technique using the example shown in Fig. 4. We initialize the number of clusters \( K = 2 \) and initialize the cluster centers using instances chosen at random from the assembly set \( \Omega \). The data set is composed of 3D positions that represent the
**Algorithm 2** K-Means based spatial partitioning.

**Input:**
- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$
- $L^\Omega = \{c^{\omega_1}, c^{\omega_2}, \ldots, c^{\omega_n}\}$ (Centers of mass for parts to be clustered)
- $K$ (number of clusters)

**Output:**
- $C = \{p_1, p_2, \ldots, p_K\}$ (Set of partitions)  
- $L^C = \{c^{p_1}, c^{p_2}, \ldots, c^{p_K}\}$; (cluster centroids)
- $m : \Omega \rightarrow C$ : (cluster membership)

**Procedure** SpacePartitioning($\Omega, L^\Omega, K$) $\rightarrow$ ($C, L^C, m$)

\[ L^C \leftarrow \text{InitialValue}(\text{Random selection of } K \text{ values from } L^\Omega); \]

for $\omega_j \in \Omega$ do

\[ m(\omega_j) = \arg \min_{k \in \{1, 2, \ldots, K\}} \text{Distance}(c^{\omega_j}, c^p_k); \]

end for

while ( there is change in $m$) do

for $j \in \{1, \ldots, k\}$ do

Recompute $c^{p_j}$ as the centroid of $\{\omega \mid m(\omega) = p_j\}$;

end for

for $\omega_j \in \Omega$ do

\[ m(\omega_j) = \arg \min_{k \in \{1, 2, \ldots, K\}} \text{DISTANCE}(c^{\omega_j}, c^p_k); \]

end for

end while

Return $C, m$
Figure 5: First cycle of partitioning for the chassis assembly. Two valid part sets $A$ and $B$ are obtained with $k = 2$.

Figure 6: Second cycle of partitioning for the chassis assembly. $B$ is partitioned into two valid part sets $C$ and $D$. 
center of mass $c_i^\omega$ of each part $\omega_i$. We use the Euclidean distance metric to compute closeness of a data point to cluster centers. The cluster centers of the product affect the assembly feasibility in the sense that the resulting parts in one cluster may or may not be physically separated from parts in other clusters. Therefore, whenever a part partitioning corresponding to a set of cluster centers is not assembleable, then $K$ is incremented by 1 and new cluster centers are identified. This process is repeated until cluster centers that lead to a successful partitioning are found.

Every time an assembly set is repartitioned into a set of $K$ clusters, we must test whether the new organization of the parts is feasible to assemble or not. For this purpose, we treat these clusters as individual part sets by merging all elements that belong to one cluster into one part set. Therefore, the reorganized assembly structure is composed of the set $\Omega_{\text{reorg}} = \{\omega_{\text{rest}}, C_1, C_2, \ldots, C_K\}$, where $\omega_{\text{rest}}$ is either the largest part in the first iteration or the merged part set composed of all the assembly elements excluded from the current partitioning analysis. The pseudocode used to implement assembleability testing is shown in Algorithm 3. We introduce a parameter $c_{\text{min}}$, which represents the minimum number of elements contained in a valid cluster. That is, whenever $|C_i| \leq c_{\text{min}}$, $C_i$ is not partitioned further. In all our experiments $c_{\text{min}} = 6^1$.

The first cycle of partitioning ($K = 2$) is shown in Fig. 5. In this case, the reorganized assembly $\Omega_{\text{reorg}}$ is composed of three part sets: Part set $A$, part set $B$, and $\omega_{\text{rest}}$, which is equal to the largest part of the assembly $\omega_{\text{largest}}$ detected at the beginning of the algorithm. Symbolically, $\Omega_{\text{reorg}} = \{\omega_{\text{largest}}, A, B\}$. The motion planning module finds that the resulting assembly structure is assembleable. Therefore, the algorithm proceeds to the second cycle of partitioning. Note that $|A| = 4 (< c_{\text{min}})$. Therefore, $A$ doesn’t undergo further partitioning. However, note that $|B| = 18 (> c_{\text{min}})$. Therefore, $B$ is further partitioned into $C$ and $D$ as shown in Fig. 6. The new assembly is composed of three part sets $C$, $D$, and $\omega_{\text{rest}}$, where $\omega_{\text{rest}}$ is a merged part set composed of $\omega_{\text{largest}}$ and $A$. Therefore, $\Omega_{\text{reorg}} = \{\omega_{\text{largest}}, A, (B \rightarrow C + D)\}$ The updated assembly structure is verified to be assembleable. In the third cycle, $C$ doesn’t undergo partitioning

\footnote{A low value of cmin (= 6) for the cluster results in obtaining manageable part clusters at the leaf node of the assembly tree structure whenever possible. Similar results will be obtained if this value is changed to 5 or 7.}
Figure 7: Third cycle of partitioning for the chassis assembly. $D$ is partitioned into two valid part sets $E$ and $F$.

$|C| = 6$ and $D$ is partitioned into $E$ and $F$ ($|D| = 12$) (Fig. 7). Similarly, $F$ is partitioned into $G$ and $H$ (Fig. 8). However, the motion planning module finds that the resulting partitioning is not feasible for assembly. We continue repartitioning of a invalid cluster only when at least one of its subclusters contains $c_{min}$ elements. Since $|G| < 6$ and $|H| < 6$, the algorithm halts resulting in $F$ as a single part set. Therefore, We obtain the final assembly structure $\Omega_{reorg} = \{\omega_{largest}, A, (B \rightarrow (C \rightarrow (D \rightarrow E + F)))\}$. The pseudocode to implement the spatial partitioning and motion planning based automated generation of assembly precedence constraints is given in Algorithm 4.

As mentioned earlier, a linear assembly sequence is one in which each operation places a single part into the assembly. Although not all products can be assembled linearly, such sequences are used in manufacturing owing to their simplicity. Here, we showed that our approach can be used to achieve more complex organizational levels in which assembly of part sets can be assigned to different stations and later, these part sets can be put together to form the final assembly.
Figure 8: Fourth cycle of partitioning for the chassis assembly. $F$ is partitioned into two part sets $G$ and $H$. However, this partitioning fails the assembleability test making the part sets invalid.

Algorithm 3 Assembleability testing.

**Input:**
- $\omega^\text{assembly}_{\text{rest}}$ (Part excluded from partitioning analysis. Initially it is the largest part).
- $K$ (number of clusters)

**Output:**
- $\text{Assembleability} \in \{\text{TRUE, FALSE}\}$,
- $\text{AssemblySequence}$

**Procedure** VerifyAssembly: $(\Omega, \omega^\text{assembly}_{\text{rest}}, K) \rightarrow (\text{Assembleability, AssemblySequence})$

Assembleability = FALSE;

while $(k \leq \Omega \lor \text{AssemblySequence} == \text{FALSE})$ do

$(C, L^C, m) \leftarrow \text{SpacePartitioning}(\Omega, L^1, K)$;

for $k \in \{1, 2, \ldots K\}$ do

$(\hat{\Omega}) \leftarrow \text{MergeElements}(C, m, \Omega)$;

end for

$(\text{Assembleability, AssemblySequence}) \leftarrow \text{DMRT}(\hat{\Omega})$;

end while

Return $(\text{Assembleability, AssemblySequence})$;
Algorithm 4 Spatial partitioning and motion planning based automated generation of assembly precedence constraints.

**Input:** $\Omega$

**Output:** Assembleability $\in \{\text{TRUE, FALSE}\}$; PartSet; TotalPartSetSequence

1. $\omega_{\text{largest}} \leftarrow \text{IdentifyLargestPart};$
2. $\Omega \leftarrow \text{RemovePart}(\omega_{\text{largest}});$
3. $\omega_{\text{rest}} \leftarrow \omega_{\text{largest}};$
4. $L^\Omega \leftarrow \text{PartCenterOfMass}(\Omega);$
5. $c_{\text{min}} = 6;$
6. $\text{Cluster} = \{C_1, C_2, \ldots, C_u \forall u \in \mathbb{R}^+\}$ (List of all clusters in the assembly);
7. $\text{ClusterSet} = \{C_1 : C_1 = \Omega\}$
8. $\text{NumberOfNewClusters} = 1;$
9. $i = 0;$
10. **while** $(\text{NumberOfNewClusters} > 0)$ do
11.     $\text{NumberOfPreviousClusters} = \text{NumberOfNewClusters};$
12.     $i + +; k = 1;$
13.     $\omega_{\text{assembly}}^{\text{rest}} \leftarrow \omega_{\text{largest}} + \text{ClusterSetElements}(C_1, \ldots, C_{i-1})$
14.     **repeat**
15.         $k + +;$
16.         Assembleability $\leftarrow \text{VerifyAssembly}(C_i, \omega_{\text{assembly}}^{\text{rest}}, k);$
17.     **until** (Assembleability $== \text{FALSE}$)
18.     **if** (Assembleability $== \text{TRUE}$) then
19.         **for** $C_j \in \{1, \ldots, K\}$ do
20.             **if** $|C_j| \leq c_{\text{min}}$ then
21.                 $\Omega \leftarrow C_j;$
22.                 PartSetSequence$\text{,}_j \leftarrow \text{DMRT}(\Omega);$
23.             **else**
24.                 ClusterSet $\leftarrow \text{AddCluster}(C_j);$
25.             **end if**
26.         **end for**
27.     **else**
28.         $(\Omega) \leftarrow \text{MergeElementsInCluster}(C, m, \Omega);$  
29.         (Assembleability, AssemblySequence) $\leftarrow \text{VerifyAssembly}(C, \omega_{\text{assembly}}^{\text{rest}}, k);$  
30.         **if** (Assembleability $== \text{TRUE}$) then
31.             AssembleSametime($C_i \in C$);
32.             ClusterSet $\leftarrow \text{AddCluster}(C)$
33.         **end if**
34.     **end if**
35.     $\text{NumberOfNewClusters} = |\text{ClusterSet}| - \text{NumberOfPreviousClusters};$
36. **end while**
37. **if** $(\sum_{i=1}^{\text{ClusterSet}} |C_i|) = |\Omega|)$ then
38.     Return TotalClusterSequence $\leftarrow \text{TotalAssembly}(\text{ClusterSequence}_j);$  
39. **else**
40.     Return Assembleability $= \text{FALSE};$
41. **end if**
6. PROPERTIES OF PROPOSED ALGORITHMS

In this section, we present some theoretical results related to the techniques and the associated algorithms developed in this paper. The notations and definitions used in the lemmas and theorems were described in Section 3.1.

**Lemma 1.** Transitive precedence: If \( C_i \prec C_j \) and \( C_j \prec C_k \), then \( C_i \prec C_k \).

*Proof.*

Let \( C_i, C_j, C_k \in H \).

From Definition 6, we have \( C_j \prec C_k \)

\[ \Rightarrow C_k \text{ can be removed only after removing } C_j. \quad (1) \]

However, \( C_i \prec C_j \)

\[ \Rightarrow C_j \text{ can be removed only after removing } C_i. \quad (2) \]

Therefore, from (1) and (2), we have \( C_i \prec C_k \).

\[ \blacksquare \]

For example, in the 23-part assembly (Fig. 3), Part-23 \( \prec \) Part-4 and Part-4 \( \prec \) Part-5. Therefore, according to Lemma 1, Part-23 \( \prec \) Part-5. Also, from visual inspection of the exploded view of the part assembly (Fig. 3(b)), it can be verified that Part-23 must be removed before removing Part-5, which is consistent with the above lemma.

**Lemma 2.** Let \( C_i, C_j \in H \). Now, if \( C_i \approx C_j \) then \( C_i \sim C_j \).

*Proof.* \( C_i \approx C_j \) \( \Rightarrow \) \( C_i, C_j \in h \) for some \( h \in H \). That is, the part sets belong to the same layer. Therefore, from Definition 5 and Definition 6, we have \( C_i \sim C_j \).

\[ \blacksquare \]

**Remark 1.** Note that the converse may or may not be true.

For example, in the 23-part assembly (Fig. 3), Part-16 \( \approx \) Part-17 (the two parts belong to the same layer). Therefore, Part-16 \( \sim \) Part-17 (Part-16 can be removed before Part-17 and vice-versa). Note that Part-6 \( \sim \) Part-4. However, the converse, Part-6 \( \approx \) Part-4, is not true.

**Lemma 3.** Let \( C_i \in h_i \) and \( C_j \in h_j \) for some \( h_i, h_j \in H \), where \( i \neq j \). Now, if \( C_i \prec C_j \) then \( C_i \prec\prec C_j \).

[24]
Proof. Suppose \( \neg(C_i \prec\prec C_j) \). This implies \( C_j \prec\prec C_i \) or \( C_i \approx C_j \).

If \( C_j \prec\prec C_i \), from Definition 8 \( h_j \prec\prec h_i \). This implies \( \neg(C_i \prec C_j) \), which is a contradiction.

If \( C_i \approx C_j \), then from lemma 2, we have \( C_i \sim C_j \), which is also a contradiction. Therefore, we have the result. \( \square \)

**Remark 2.** Note that the converse may or may not be true. In particular, the converse is true if, additionally, there is no \( C_k, k \neq i \) such that \( C_k \prec C_j \).

For example (Fig. 3), Part-4 \( \prec \) Part-5. From Lemma 3, Part-4 \( \prec\prec \) Part-5. Note that the converse is also true as Part-3 \( \prec \) Part-5 is not true. Now, Part-4 \( \prec\prec \) Part-6. However, the converse, Part-4 \( \prec \) Part-6, is not true.

**Theorem 1.** For any two adjacent layers \( h_i \) and \( h_{i+1} \), a layer precedence relationship \( h_i \prec \prec h_{i+1} \) is established if and only if, for every part set \( C_{i+1} \in h_{i+1} \), \( \exists \) at least one part set \( C_i \in h_i \) such that \( C_i \prec C_{i+1} \).

Proof. First, we prove the “if” part: Let \( \emptyset \subseteq h_i \) such that \( |\emptyset| \geq 1 \) and \( C_i \prec C_{i+1}, \forall C_i \in \emptyset \). Using lemma 3, we have \( C_i \prec\prec C_{i+1}, \forall C_i \in \emptyset \). This implies \( h_i \prec \prec h_{i+1} \). Hence, the “if part” is proved.

Next, we prove the “only if” part:

Given \( h_i \prec \prec h_{i+1} \) (3)

Suppose the result is untrue. That is, for some part set \( C_{i+1} \in h_{i+1} \), \( \neg(C_i \prec C_{i+1}) \) for every \( C_i \in h_i \).

\[ \Rightarrow C_i \sim C_{i+1}, \forall C_i \in h_i \] (4)

or \( C_{i+1} \prec C_i, \forall C_i \in h_i \) (5)

(4) \( \Rightarrow \) \( C_{i+1} \in h_i \). However, this is a contradiction, since \( h_i \cap h_{i+1} = 0 \).

(5) \( \Rightarrow \) \( C_{i+1} \prec\prec C_i \) (From lemma 3). Equivalently, \( h_{i+1} \prec\prec h_i \), which is also a contradiction. Hence, we have the result. \( \square \)

From Fig. 3(c), \( h_2 \prec\prec h_3 \), where \( h_2 = \{ \text{Part-2, Part-11, Part-14, Part-20} \} \) and \( h_3 = \{ \text{Part-21, Part-22} \} \). Note that Part-2 \( \prec \) Part-21 and Part-2 \( \prec \) Part-22. That is, for every Part-i \( \in h_3 \), there exists at least one Part-j \( \in h_2 \) such that Part-j \( \prec \) Part-i. This is consistent with Theorem 1. Note
that Part-21 and Part-22 have no individual precedence relationships with
the remaining parts in h_2 even though they fall in different layers.
From theorem 1, we can state the following corollary.

**Corollary:** For any two layers h_i and h_j, a layer precedence relationship
h_i ≺≺ h_j is established if and only if, for every part set C_j ∈ h_j, ∃ at least
one part set C_i ∈ h_i such that C_i ≺ C_j.

From Fig. 3(c), h_1 ≺≺ h_3, where h_1 = { Part-8, Part-9, Part-10, Part-12,
Part-13, Part-15, Part-16, Part-17, Part-18}. Note that Part-12 ≺ Part-21
and Part-13 ≺ Part-22, which is consistent with the above corollary.

**Theorem 2.** Given the n-part assembly Ω, the part interaction clusters
extracted by Algorithm 4 admit a tree structure T with Ω as its root part set
and a monotonically decreasing bound on its branching factor. Further, the
branching converges in a finite number of partition levels that is bounded by
n − 2.

Proof. Let Ω = {ω_1, ω_2, . . . , ω_n}. Applying Algorithm 4, we get Ω = C^r =
{C_1, C_2, . . . , C_k, ω_{largest}}, where each C_i is a child part set of the parent part
set C^r according to the Definition 4 and ω_{largest} is the largest part. Algorithm
4 is applied recursively on each child part set until each part set is a single
part. By considering each part set as a node and connecting each child part
set to its parent by an edge, the resulting part sets can be represented by a
tree structure T, where Ω is the root node.

Let C^p = {C^p_1, C^p_2, . . . , C^p_r} for some node C^p ∈ T. Note that the branching
factor at C^p = r. The corresponding bound on the branching factor is
|C^p|. This implies 2 ≤ r ≤ |C^p|. Similarly, the bound on branching factor
at each C^p_i ∈ C^p = |C^p_i|. However, |C^p_i| < |C^p|, which implies a monotonically
decreasing bound on the branching factor of the tree. Note that the
maximum bound on the branching factor is n − 1, which occurs at the first
partition level.

Consider a path from the root node to the leaf node. We note that the
maximum number of partition levels is obtained by choosing a k = 2 partition
at each level i, with |n − 1 − i| number of part sets in one partition and a
single part in the other partition. This results in the following branching
structure:

\[ C^1 = \{C^2, \omega : |C^2| = n - 2, \omega \in C^1 - C^2\} \]
\[ C^2 = \{C^3, \omega : |C^3| = n - 3, \omega \in C^2 - C^3\} \]
\[ C^i = \{ C^{i+1}, \omega : |C^{i+1}| = n - i - 1, \omega \in C^i - C^{i+1} \} \]

\[ C^{n-2} = \{ C^{n-1}, \omega : |C^{n-1}| = 1, \omega \in C^{n-2} - C^{n-1} \} \] (6)

From (6), \(|C^{n-1}| = 1\). This implies that the branching terminates since the minimum number of parts in a part set for it to be partitioned is two. Therefore, Algorithm 4 partitions \(\Omega\) into part sets in a finite number of partition levels, which is bounded by \(n - 2\).

The tree structures obtained for the 23-part chassis assembly, the 73-part chassis assembly, the crank shaft assembly, and the radial engine assembly are shown in Figs. 11, 14, 17, and 20, respectively. In all these examples, the branching converges in a finite number of partition levels.

**Theorem 3.** The number of possible assembly sequences \(AS_n\) of an \(n\)-part assembly based on part interaction cluster structure extracted by Algorithm 4 is \(\sum_{i=1}^{n-1} SA_i \cdot SA_{n-i}\), where \(SA_i\) is the number of possible assembly sequences for the parent part set with \(i\) child part sets.

**Proof.** Let Algorithm 4 partition the parts into two part sets \(C_1\) and \(C_2\). Let \(|C_1| = u\) and \(|C_2| = v\) such that \(u + v = n\), for some \(u\) and \(v\) \(\in\) \(\mathbb{R}^+\). Accordingly, let us assume assembling \(u\) parts and \(v\) parts separately, and then combining the \(u\)-part and \(v\)-part sets to form the \(n\)-part assembly. In this particular case, the total number of possible assembly sequences of an \(n\)-part assembly is \(SA_u \cdot SA_v\). In the general case, \(SA_n\) is the sum of all such cases:

\[
AS_n = SA_1 \cdot SA_{n-1} + SA_2 \cdot SA_{n-2} + ... + SA_{n-1} \cdot SA_1
\]

\[= \sum_{i=1}^{n-1} SA_i \cdot SA_{n-i} \]

\[\square\]
7. COMPUTATIONAL RESULTS

We report the results from application of our technique to four different assemblies: (a) Chassis assembly (b) Crankshaft assembly (c) Radial crankshaft assembly, and (d) The five part puzzle.

7.1. Simple Chassis Assembly

In the previous section, we used the 23-part chassis assembly (Fig. 4(a)) in order to illustrate our approach of combining part interaction cluster extraction and motion planning to generate assembly sequences. Figure 9(a) shows the part sets identified by the system and Fig. 9(b) shows the corresponding assembly structure representing the precedence relations between different part sets and individual parts within each part set. Note that the algorithm identifies four part sets, in which $F$ precedes $E$ ($F \prec E$) and the rest of the part sets do not have any additional precedence constraints. This means that $A$, $C$, and $F$ can be fit into the assembly (part 1) in any order, but $E$ must be assembled after $F$. Also, note that the individual parts in each set can be assembled in parallel at different locations without any precedence between parts across different part sets. However, note that individual parts admit precedence relations with each other within each part set. The disassembly layers generated by the approach used in our earlier work (Morato et al. 2012) can be reversed and turned into a linear assembly sequence. However, they do not fully represent the precedence relations between parts across different layers. In contrast, the algorithm developed in this paper combines part cluster detection and disassembly layer extraction to generate precedence constraints for assembly. This distinction is illustrated in Fig. 10. For instance, part 16 belongs to the first disassembly layer and part 7 belongs to the seventh disassembly layer (Fig. 10(a)). This implies that part 7 can be assembled before part 16. However, it is not clear if the converse is true. The assembly structure identified by the new approach (Fig. 10(b)) clarifies this precedence relation: Part 7 and part 16 have no precedence constraint with respect to each other. That is, the set of parts ($\{2, 3, 4, 5, 6, 12, 13, 21, 22, 23\}$) that has to be removed before part 7 can be removed does not include part 16. The new assembly structure reveals these precedence relationships by reorganizing the parts in individual layers into interaction clusters, while maintaining the same hierarchy of layers found in Fig. 10(a). Figures 11(a) and (b) show the directed acyclic graph representation of the assembly structures obtained by the two techniques, respectively.
7.2. Complex Chassis Assembly

Now, we report results on a more complex assembly comprising 74 parts, which is obtained by augmenting the previous assembly with 51 new parts. Two views of this 74-part chassis assembly are shown in Fig. 12. The disassembly layer generation and the directed acyclic graph representation of the assembly structure are shown in Figures 13 and 14, respectively.

7.3. Crankshaft Assembly

The crankshaft assembly, shown in Fig. 15, consists of 41 assembly parts: One crank shaft, 4 shafts, 4 lower shafts, 4 inner bearings, 4 bearings, 8 pins, 12 rings, and 4 piston heads. The disassembly generation of a past-set identified by the algorithm is shown in Fig. 16. Figure 17 shows the directed acyclic graph representation of the assembly structure extracted by the algorithm. From this figure, it is clear that the algorithm correctly identifies a feasible nested configuration of part set.

7.4. Radial Crankshaft Assembly

The radial crankshaft assembly, shown in Fig. 18, consists of 50 assembly parts: One master shaft, one radial bearing, 4 shafts, 4 bolts, 5 inner bearings, 5 bearings, 10 pins, 15 rings, and 5 piston heads. The disassembly generation of a part set identified by the algorithm is shown in Fig. 19. Figure 20 shows the directed acyclic graph representation of the assembly
Figure 10: Precedence relations between layers in the hierarchical exploration structure for the chassis assembly: (a) Simple disassembly generation. (b) Disassembly generation coupled with cluster detection.

structure extracted by the algorithm. From this figure, it is clear that the algorithm correctly identifies a feasible nested configuration of part set.

7.5. The Five Parts Puzzle

The five-part puzzle (Fig. 21) is used to illustrate the ability of the algorithm to deal with the issue of nonlinearity. In particular, this assembly example is representative of a scenario in which some parts cannot be assembled one by one in a linear order; rather, they must be simultaneously moved to their final assembly locations. The algorithm implementation can be explained in the following steps:

1. Find part sets to test feasibility with $C_{\text{min}} = 1$. No feasible partition is found with $k = 1, 2, 3$ and 4.
2. Test feasibility with the largest part + a single part containing the four remaining parts. The assembly structure passes the test. The result of this step is that the set of four parts must be assembled before placing in the final position.
3. Recursively partition the four-part set and test feasibility. This results in two part sets $A$ and $B$. 

Figure 11: Directed acyclic graph representation of the assembly structure for the chassis assembly: (a) Simple disassembly generation. (b) Disassembly generation coupled with cluster detection.

Figure 12: The 73 part chassis assembly: (a) View 1. (b) View 2.
Figure 13: Disassembly layer generation for the 73-part chassis assembly by combining motion planning and part interaction cluster detection methods.
Figure 14: Directed acyclic graph representation of the assembly structure for the 73-part chassis assembly.

Figure 15: Crank shaft assembly

Figure 16: The disassembly generation of one part interaction cluster identified by the algorithm for the crank shaft assembly.
Figure 17: Directed acyclic graph representation of the precedence relations between assembly parts and part sets for the crank shaft assembly.

Figure 18: Radial engine assembly

Figure 19: The disassembly generation of one part set identified by the algorithm for a Radial engine assembly.
The resulting assembly sequence is given below:

1. Assemble the two parts in part set \( A \).
2. Assemble the two parts in part set \( B \).
3. Assemble the previous two part sets into one single part set.
4. Assemble the resulting part set into its final position inside the largest part.

8. CONCLUSIONS

We presented a framework that combined motion planning techniques and part interaction cluster extraction to guide the generation of feasible assembly sequences for complex mechanical assemblies. We showed that our approach can be used to achieve complex organizational levels of an assembly in which assembly of part sets can be assigned to different stations and later, these part sets can be put together to form the final assembly. We also showed how our approach addresses the issue of nonlinearity in assemblies. The system generated the assembly precedence constraints directly from 3D assembly...
models as it used only information about geometric and physical constraints. In situations where part assembly involves a single translation and a single rotation, methods without using path planning are computationally more efficient than our approach. However, our approach performs better when applied to more complex assemblies that involve multiple translations and rotations. Our approach explores many different feasible assembly directions. Our approach checks for feasibility in a given assembly state. So it can handle situations where different directions become feasible based on the state of the assembly. For example, if the left direction is blocked in a given assembly state, and the right direction is checked for feasibility and selected (if appropriate).

Our approach presented in the paper considered each part as a free-flying object. However, in a realistic scenario, as tools and human hand operations will be used during assembly, some of the precedence constraints generated by our approach may become infeasible. In order to address this issue, our approach must be augmented by incorporating motion planning for human-hand and tool models into the formulation. This will result in the generation of modified precedence constraints that cater to the spatial constraints imposed by the tools and the hands. Our technique was proposed for mechanical assemblies that are composed of finite numbers of non deformable parts. Hence, assemblies that contain flexible parts, where the flexible property of a part is exploited to fit it into the assembly, were out of the current scope of our current approach. In addition, an analysis beyond pure geometric reasoning is required for assemblies that contain parts requiring force for placement.
9. ACKNOWLEDGMENTS

This work was partially supported by DARPA AVM iFab Project (Contract # FA8650-11-C-7128). This project was a large collaborative effort involving Carnegie Mellon University, Lockheed Martin, Pratt and Miller, University of Maryland, and University of Michigan. We acknowledge and thank all members of our team for their valuable input. We especially would like to thank David Bourne from Carnegie Mellon University for sharing his valuable insights and feedback. We are also thankful to META team at Vanderbilt University for providing the chassis assembly used in this paper.

REFERENCES


Kavraki, L. and Kolountzakis, M., 1995. ”Partitioning a planar assembly into two connected parts is NP-complete”. Information Processing Letters, 55(3) 159–165.

BIOGRAPHIES

Carlos W. Morato received his BS in Systems Engineering from the Technical University of Oruro, Bolivia in 1998. He holds two Masters degrees, a MSc (Cum Laude) in Computer Science from San Andres University, La Paz, Bolivia and a MSc in Aerospace Engineering from University of Maryland, College Park in 2001 and 2011 respectively. He also received a graduate specialization degree in Robotics and computer vision from Alicante University in 2008. He was a RETECA-IDSIA research Fellow from 2005 to 2006. Currently, he is a PhD student at the Maryland Robotics Center, University of Maryland, College Park, USA. He is a member of IEEE.

Krishnanand Kaipa received his BE (Hons.) in Electrical Engineering from the Birla Institute of Technology and Science, Pilani, India, in 1998. He received his MSc (Engg.) and PhD from the Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India, in 2004 and 2007, respectively. He received best master thesis and best PhD thesis awards. He was a JRD Tata Research Fellow for two years (2007 -2008). He worked as a postdoctoral researcher at the Department of Computer Science, University of Vermont, Burlington, USA, from 2008 to 2010. Currently, he is a postdoctoral researcher at the Maryland Robotics Center, University of Maryland, College Park, USA. Dr. Kaipa is member of the IEEE and the CLAWAR Association.

Satyandra K. Gupta received his Master of Technology (M.Tech.) degree in production engineering from the Indian Institute of Technology, Delhi,
in 1989; his Bachelor of Engineering (B.E.) degree from the University of Roorkee (currently known as the Indian Institute of Technology, Roorkee) in 1988; and his Ph.D. degree in mechanical engineering from the University of Maryland (UMD), College Park, in 1994. He is currently a Professor in the Department of Mechanical Engineering and the Institute for Systems Research at UMD. Dr. Gupta is a Fellow of ASME and a Senior Member of IEEE.