International Journal of Robotics Research 14(4), pages 335–350, 1995

Two-Handed Assembly Sequencing

Randall H. Wilson* Lydia Kavraki† Tomás Lozano-Pérez‡
Jean-Claude Latombe†

Abstract

This article considers the computational complexity of automatically determining assembly sequences for mechanical products. Specifically, we address the *partitioning problem*: given an assembly of rigid parts, identify a proper subassembly that can be removed as a rigid object without disturbing the rest of the assembly. We examine the complexity of the partitioning problem under various types of relative motions allowed for the subassemblies. We show that when arbitrary motions are allowed to separate the two subassemblies, partitioning is NP-complete.

We then describe a general framework for reasoning about assembly motions called the *interference diagram*. In its most general form the interference diagram yields an exponential-time algorithm to partition an assembly. However, two special cases of the interference diagram studied in this article yield polynomial-time sequencing algorithms. The first case occurs when assembly motions are restricted to single translations. The second case considers infinitesimal rigid motions in translation and rotation and yields a superset of all feasible partitionings. These two algorithms have important practical applications in assembly planning.

*Intelligent Systems and Robotics Center, Sandia National Laboratories, Albuquerque, NM 87185, USA. This work was performed while the author was at Stanford University.

†Computer Science Robotics Laboratory, Stanford University, Stanford, CA 94305, USA.

‡MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139, USA.

1 Introduction

Automated assembly of composite products has been a goal of robotics researchers since the beginning of the field [8; 16], and robots are used today to assemble a variety of products. However, programming these robots remains tedious and error prone, and algorithms are sought to help program them. Research in assembly planning aims to automate or partially automate the ordering of assembly operations and selection of the required tools and motions, as well as provide fast manufacturing evaluation of assembly designs. This article concerns a subproblem of assembly planning called *assembly sequence planning*, or *assembly sequencing*, in which the relative motions of the parts of the assembly are determined without considering the tools, fixtures, or robots required to achieve those motions.

Disassembly planning has been a very popular approach to assembly sequencing (see for instance [20; 33; 1; 3; 14; 15; 21; 6; 29]). The most constraints on assembly motions are present in the final assembled state of the product, so reasoning backward from the assembled state reduces the branching factor of the search considerably. The result is a sequence of relative motions for the parts called a *disassembly sequence*. Here we assume the parts are rigid, so a disassembly sequence is the reverse of an *assembly sequence*, and conversely.

In addition, we limit consideration to *monotone two-handed* assembly sequences: those that at each step merge exactly two rigid subassemblies to make a larger one. Such sequences are *two-handed* because one hand is needed to hold each rigidly moving subassembly (a table or fixture counts as a hand, if one is used) [24], and *monotone* because parts are never placed in intermediate positions relative to each other [33]. For example, the assembly in Figure 1(a) can be assembled by a monotone two-handed assembly sequence, whereas the assembly in Figure 1(b) cannot; the latter requires one part to be placed in an intermediate position and then moved again later. This limitation is well supported by typical assemblies seen in industry, and most experimental assembly planning systems today find only monotone two-handed sequences.

The success of such a disassembly planner depends on solving the *partitioning problem*: given an assembly, identify a proper subassembly that can be removed as a rigid object without moving the rest of the parts. The two subassemblies produced can be partitioned in turn, and so on, until only individual parts remain. As a result, the partitioning problem is the core of the monotone two-handed assembly sequencing problem.

In this article we consider the computational complexity of the partitioning problem. We show that when the removed subassembly may follow an arbitrary path (or an arbitrary sequence of translations), partitioning is NP-complete, even for two-dimensional assemblies of polygons. We then introduce a general framework for reasoning about the motions of subassemblies called the *interference diagram*. The interference diagram makes explicit the constraints on motion of all parts of the

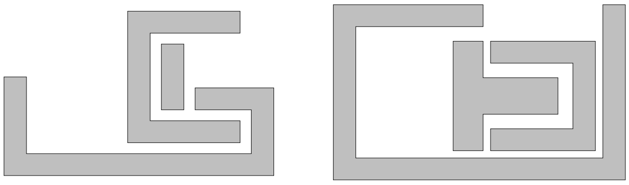


Figure 1: (a) is a monotone two-handed assembly while (b) is not.

assembly relative to all others and embodies the monotone and two-handed restrictions exactly. In this framework we analyze the conditions under which partitioning is tractable. Although the interference diagram for an assembly can be constructed in polynomial time, in its most general form it yields a partitioning algorithm that is exponential in the number of surfaces describing the assembly.

However, restrictions on the motions allowed to separate two subassemblies give rise to special cases of the interference diagram that can be analyzed in polynomial time. We consider two cases here. In the first case, assembly motions are restricted to single translations to infinity, and the resulting algorithm identifies a removable subassembly in $O(n^2v^4)$ time, for an assembly of n polyhedra having a total of v vertices. In the second case, infinitesimal rigid motions (including translation and rotation) are considered; the set of movable subassemblies thus identified is a superset of those that can be removed by arbitrary paths. A subassembly that can translate or rotate a small distance from the rest of the assembly is identified in $O(mc^5)$ time, for a polyhedral assembly with m pairs of parts in contact, whose contacts are described by $c \in O(v^2)$ point–plane contact constraints.

2 Related Work

Although we consider only monotone two-handed assembly sequences in this article, other authors have examined more general cases. In nonmonotone assembly sequences, parts may assume intermediate positions in subassemblies. Natarajan [24] and Wolter [33] show that assembly sequencing is PSPACE-hard when nonmonotone sequences are allowed. Hoffman [13] describes one of the few nonmonotone assembly planning systems.

In assembly sequences that are not two-handed, more than one subassembly may move independently at the same time. Palmer [26] considers the infinitesimal partitioning problem in this case: determining whether a feasible set of simultaneous, infinitesimal motions exists for the parts of a polygonal assembly. He shows that this problem is NP-complete by a reduction from 3-Satisfiability. However, few industrial assemblies require the simultaneous independent motion of many parts.

In this article a subassembly always refers to a set of parts in fixed relative position.

Most experimental assembly planning systems find only monotone two-handed assembly sequences. These systems address the partitioning problem in a variety of ways. Several systems [3; 15; 21] enumerate all possible subassemblies and test each for removal, an approach that is practical for small assemblies but exponential-time in the worst case. Another approach is to consider only sequences in which each operation mates a single part with a subassembly [20; 32]. This approach reduces the partitioning problem to checking for the motion of each part but risks losing some, maybe all, of the assembly sequences for a product. Finally, many systems predefine a small set of assembly trajectories or heuristically choose trajectories based on features of the parts or their contacts [33; 2; 13; 6]. However, this approach requires that extra planning information be present in the assembly description and risks missing some assembly sequences.

Arkin, Connelly, and Mitchell [1] use the concept of a monotone path among obstacles to derive a polynomial-time algorithm for partitioning an assembly of polygons in the plane with a single infinite translation. This corresponds to the first special case of the interference diagram considered below (Section 5). However, their approach is significantly different from ours and does not directly extend to the 3D case.

In the rest of this article, we analyze the partitioning problem from a purely geometric and computational point of view. In Section 3 we show that when an arbitrary translational path is allowed to separate two subassemblies, the partitioning problem for assemblies of simple polygons in the plane is NP-complete. Section 4 introduces the interference diagram, a novel configuration-space formulation of the partitioning problem. Although the interference diagram for an assembly can be constructed in polynomial time, its analysis requires exponential time. However, in Sections 5 and 6 we show that when assembly motions are restricted to single translations and infinitesimal rigid motions, respectively, the resulting special cases of the interference diagram yield polynomial-time partitioning algorithms that are quite useful in practice. Finally, Section 7 examines the boundary between tractable and intractable cases of the partitioning problem and identifies open problems.

3 Complexity of Partitioning

The assembly partitioning problem consists of identifying a subassembly of a given assembly that can be removed as a rigid object without disturbing the other parts of the assembly. In this section we show that deciding whether such a subassembly exists is an NP-complete problem. We outline the proof for an assembly of polygons that can translate and rotate in the plane. A more detailed version of the proof can be found in [18]. The NP-completeness result also extends to some restricted variants of the partitioning problem that are presented at the end of this section.

Let a *rigid motion* of a subassembly S be a set of simultaneous motions (translations and rotations) of the parts of S that preserve the relative positions of these parts throughout the motion. The subassembly S is then called a *rigid* subassembly.

Planar Partitioning (PP) *Given a set A of nonoverlapping polygons in the plane, decide if there is proper subset S of A that can be separated from $A \setminus S$ by a collision-free rigid motion of S.*

We will show that PP is NP-complete. It is clearly in NP, since a nondeterministic algorithm can guess S and then invoke a path planner to find the path of S out of the assembly. Schwartz and Sharir [28] have shown that path planning can be done in polynomial time for the case considered here.

We show that PP is NP-hard by a reduction from 3-Satisfiability (3-SAT), a well known NP-complete problem [9]. An instance of 3-SAT is a set of clauses $C = \{c_1, c_2, \ldots, c_m\}$ on a set of boolean variables $U = \{u_1, u_2, \ldots, u_n\}$, where each clause is a disjunction of 3 terms. A term is either a variable u_i or the negation of a variable $\overline{u_i}$. The problem is to determine whether there exists a truth assignment of the variables that satisfies the conjunction of the clauses.

For any instance of 3-SAT, we construct in polynomial time an assembly of nonoverlapping polygons that can be partitioned if and only if a satisfying truth assignment exists. Figure 2 shows the *outside box* and the *key* of the constructed assembly. Other parts are contained in the *assignment mechanism* and the *AND/OR mechanism*, detailed in Figures 3 and 4, respectively. Our construction is summarized in the following:

- A rigid subassembly must be removed to partition the assembly A. Call this subassembly S.
- The part labeled "*key*" in Figure 2 blocks the only exit gate of the assembly. Hence, it must be a member of S.
- The key can be removed only through the assignment mechanism. For this to happen, some other parts of the assignment mechanism must be in S. These parts represent a truth assignment for the variables of the 3-SAT instance and move rigidly with the key.
- S can be removed only through the AND/OR mechanism. This mechanism enforces the clauses of the 3-SAT instance.
- Finally S is moved to the cavity on the upper left, rotated, and removed through the gap left by the key.

The assignment mechanism (Fig. 3) consists of (1) the walls of the assignment, which are drawn in gray; (2) the key, which is a 3×1 rectangle drawn in black; and (3) two 3×1 white rectangles for each of the variables of the 3-SAT instance.

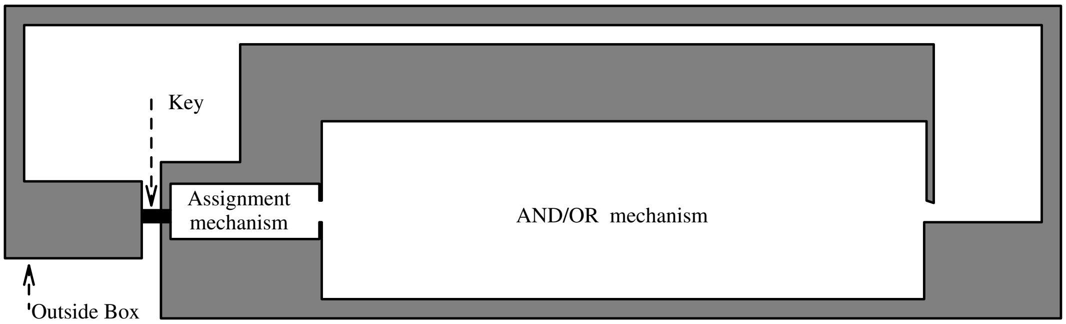


Figure 2: A sketch of the final assembly.

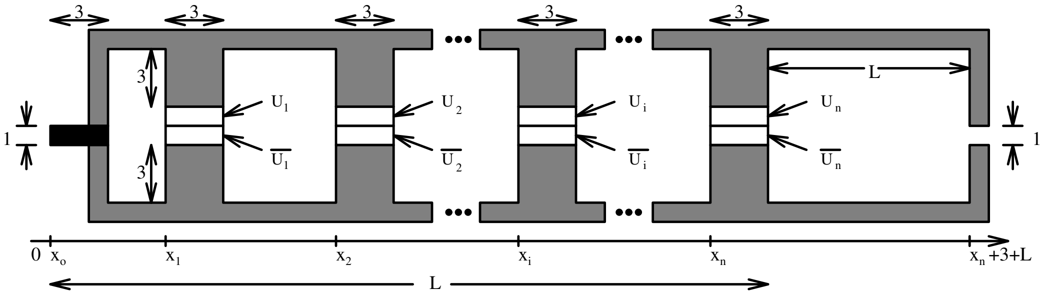


Figure 3: The assignment mechanism (not drawn to scale).

One of the two rectangles that correspond to the variable u_i is labeled with U_i, and the other with $\overline{U_i}$. Notice that U_i is placed always on top of $\overline{U_i}$ in the assignment mechanism. If U_i, ($\overline{U_i}$ respectively) is a member of S, we consider that the truth assignment *true* (*false* resp.), has been chosen for the variable u_i. We indicate with x_0 the initial position of the key and with x_i, the initial position of the assignment rectangles for the variable u_i, $i = 1, \ldots, n$. The choice of the x_i's is crucial and is described below.

From Figure 2 it is clear that the key initially can move only to the right. We observe that the key will not be able to pass through the assignment mechanism if no other parts of the mechanism are moved. In addition, the parts removed must translate rigidly. Hence, the only subassembly S that stands a chance to move out of the assignment mechanism, and eventually out of the total assembly, is a subassembly that consists of the key and at least one of U_i or $\overline{U_i}$, for each i. Because the exit gate of the assignment mechanism has a height of only 1, exactly one of U_i or $\overline{U_i}$, for each i, must be selected in S. Let L be the length of the moving subassembly S.

The collision-free motion of S out of the assignment mechanism is possible only if the x_i's are selected carefully. We show in the Appendix how to choose the x_i's to guarantee the following: for any k, when U_k or $\overline{U_k}$ (whichever of them belongs

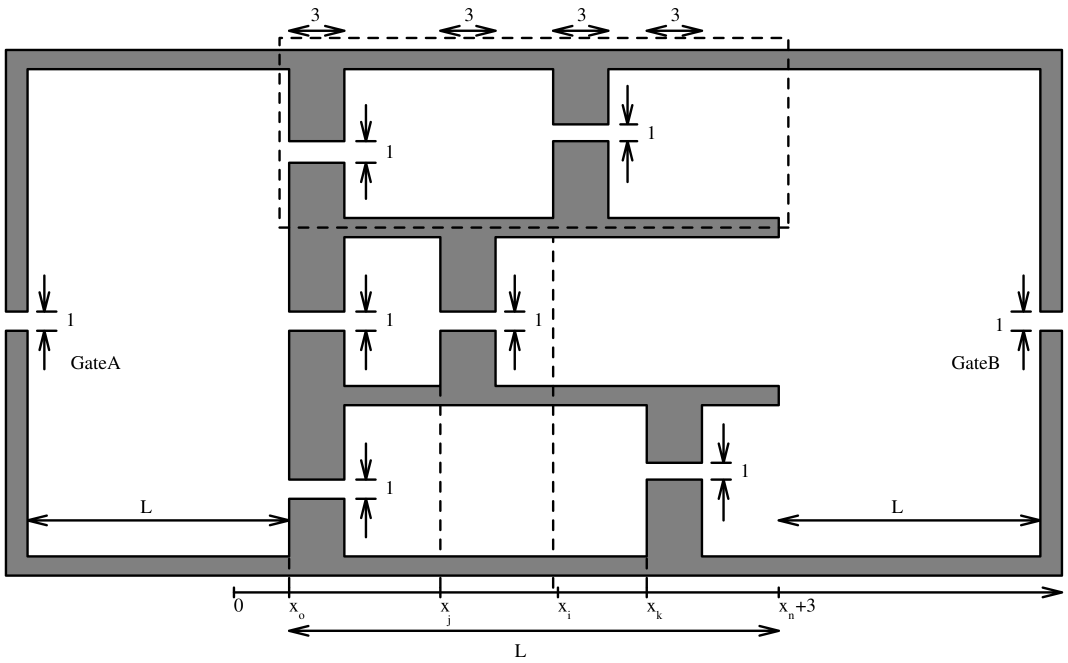


Figure 4: The OR gate for the clause $u_i \vee \overline{u_j} \vee u_k$ (not drawn to scale).

to S) is close to position x_p, $p \neq k$, and must go through the hole that has been created at this position, none of the other assignment rectangles or the key is close to a position x_i, $i \neq p$. In other words, all the other parts of S are in the wide free sections of the assignment mechanism and can follow the constrained motion of the part that is close to x_p. Hence, the above mechanism ensures independent selection of the variable assignments of the 3-SAT instance and allows the resulting subassembly to translate out of the mechanism.

Once S translates out of the exit gate of the assignment mechanism, it must pass through the AND/OR mechanism. This mechanism is a sequence of OR gates, one for each clause of the 3-SAT instance. The OR gate for the clause $u_i \vee \overline{u_j} \vee u_k$ is shown in Figure 4. We observe from the figure that there are three possible ways for S to go from GateA to GateB. Each of these enforces a truth assignment for one of the terms of the clause. Suppose, for example, that S goes through the section enclosed in the dashed box in Figure 4. This section enforces the truth assignment *true* for the variable u_i. Here is why: When the key is at x_0, the rectangle chosen for the truth assignment of u_i is at position x_i. Unless U_i is selected in S, it is impossible for S to go through the dashed box of Figure 4. Notice, however, that the rest of S can be threaded through the gates at x_0 and x_i without problems: because of the property of the x_i's mentioned above, when a part of S needs to go through the above gates, none of the other parts of S is close to a narrow passage. Hence, S can follow the motion of its constrained part without being obstructed by the walls of the OR gate.

Suppose there exists a satisfying truth assignment for the 3-SAT instance. Let S be the subassembly that consists of the key and encodes this truth assignment. S can go through the AND/OR mechanism, because it can translate through each of its OR gates. Then S can translate to the upper left corner of the assembly, rotate there by 90 degrees, and exit through the 2-unit-wide gate that was initially blocked by the key.

Conversely, assume that the assembly in Figure 2 can be partitioned, and let S be the subassembly that is removed from it. S clearly contains the key. As we argue above, the key can be removed only in a subassembly that contains a truth assignment for the variables of the 3-SAT instance. Because S can pass through the AND/OR mechanism, it represents a satisfying truth assignment for the 3-SAT instance. Finally, it can be shown easily that the reduction presented above is polynomial in the size of the 3-SAT problem. It follows that:

Theorem 1 *PP is NP-complete.*

Variants of the above proof establish that the following problems are NP-complete [31; 18; 17]:

- Partitioning of a planar assembly when rotation is not allowed.
- Partitioning of an assembly of polyhedra when rotation is allowed and when rotation is not allowed.
- Partitioning of an assembly with parts of constant complexity (i.e., each part is limited to a constant number of vertices).
- Partitioning of assemblies of polygons with the additional requirement that the two subassemblies produced are connected. A subassembly is connected when the union of its parts is connected.
- Partitioning of assemblies that can be fully disassembled. The reader may note that the assembly in Figure 2 cannot be fully disassembled. This result proves that monotone two-handed assembly sequencing is an NP-complete problem.

4 The Interference Diagram

In this section we present a general framework for reasoning about assembly motions; this framework is called the *interference diagram*. The interference diagram is motivated by the desire to reason about a trajectory for a subassembly S, without knowing *a priori* which parts of the assembly are members of S. Once a path is known, or as it is constructed, constraints on which parts of the assembly may be in S are analyzed to determine the members of S.

The interference diagram is a somewhat complicated configuration-space diagram, so we can illustrate it graphically only in simple examples such as the one in Figure 5(a). To simplify the presentation, we will limit ourselves to pure translations. Furthermore, we have drawn the parts with substantial clearances so that each of the relevant regions in configuration space would be full-dimensional. We sketch the generalization to higher dimensions, rotations, and situations involving contacts in Section 4.4.

4.1 Placing a Part

Because we do not know which parts will be moving and which are stationary, the interference diagram represents the constraints on motion between any pair of parts in the assembly. For every pair (X, Y) of parts, the set of placements of X in which it intersects with Y is the configuration space (or C-space) obstacle for the pair of parts [22]. The C-space obstacle for part X moving with Y as an obstacle is labeled X/Y, while the C-space obstacle for Y moving with X as an obstacle is labeled Y/X. In translation, the C-space obstacle X/Y is given by

$$X/Y = Y \ominus X = \{y - x \mid x \in X,\ y \in Y\};$$

i.e., the Minkowski difference of the two sets of points Y and X. Note that Y/X is simply X/Y rotated by π radians or, equivalently, every forbidden translation x of X has a corresponding forbidden translation $-x$ of Y.

Figure 5 shows the six C-space obstacles for the simple example we are considering. A crucial point about these C-spaces is that they are all constructed using the *same reference point*, indicated by a solid circle in the figure; the reference point of an object need not be inside that object. That is, treat this point as being rigidly attached to each part in turn. Then the C-space obstacles represent the positions of this reference point for which two parts collide.

To remove one part from the assembly, these pairwise C-spaces clearly give us the information we need to plan the motion. We can test if part B, for example, can be moved by examining whether a motion exists in the region outside the union of B/A and B/C. However, we wish to reason about the motion of subassemblies. In principle, we could enumerate all subassemblies and compute the C-space obstacles for them, but there is an exponential number of subassemblies to consider.

4.2 Placing a Subassembly

Instead, we reason about the motion of an unknown rigid subassembly while gathering constraints on which parts can follow that motion. Because the parts of the moved subassembly must remain in fixed relative position, the reference points of its parts will coincide throughout its motion. When the reference point is in C-space obstacle X/Y, the following constraint is in effect: If X is moving and Y is station-

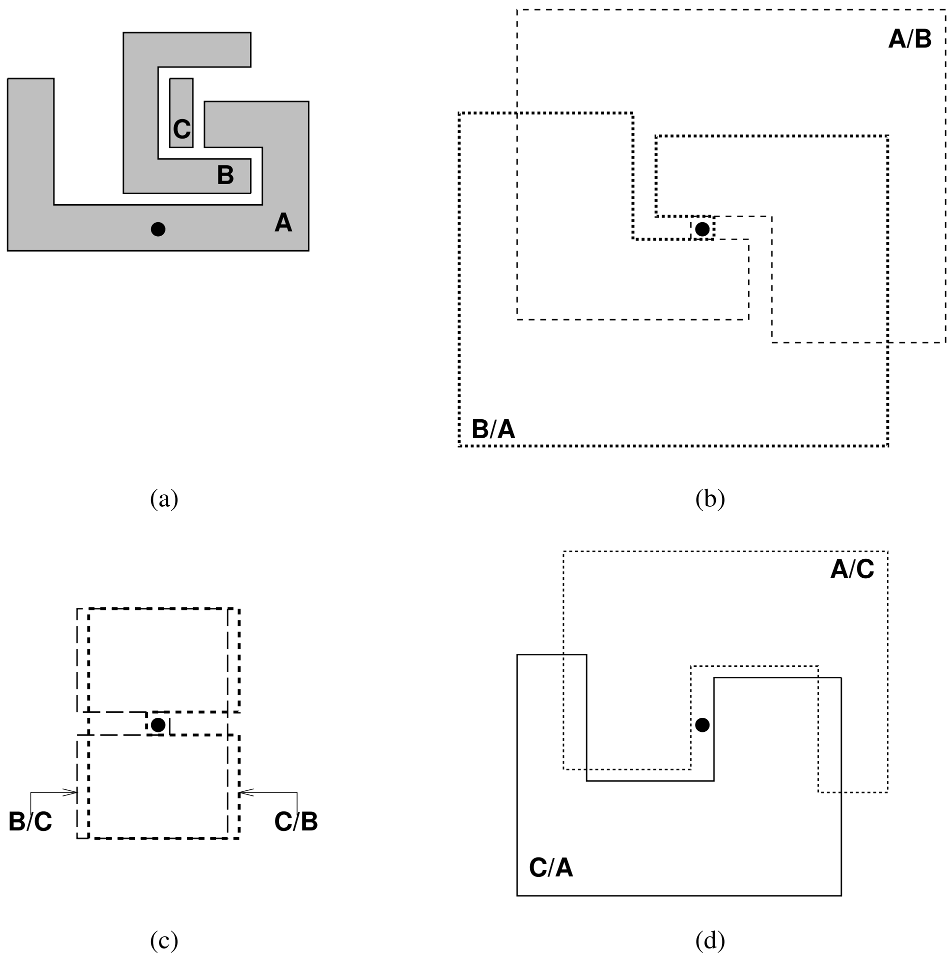


Figure 5: C-space diagrams for each pair of parts in the assembly. The notation X/Y indicates that X (moving) collides with Y (stationary) in that region.

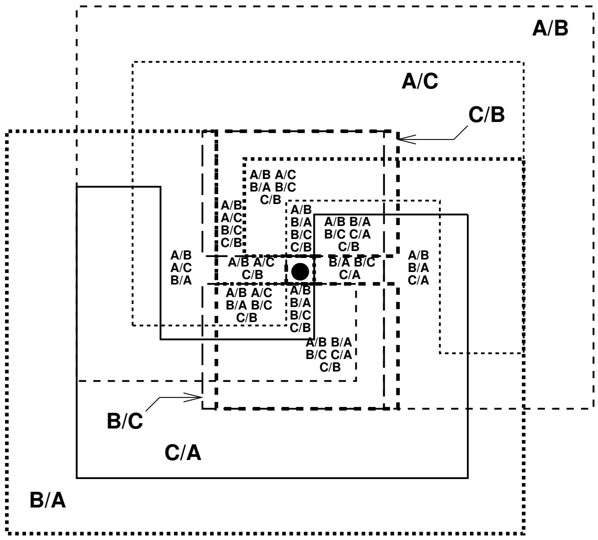


Figure 6: The interference diagram for the assembly of Figure 5(a). The small labels indicate the C-space obstacles that share that cell.

ary, a collision has occurred. In other words, if a path passes through X/Y, then either X must be stationary or Y must be moving.

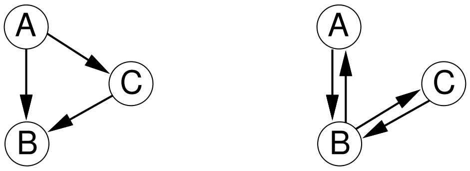
Because all of the pairwise C-spaces were computed with the same reference point, we now superimpose all the C-space obstacles. That is, all the obstacles are embedded in the same coordinate system. A point in this space represents a displacement of any subset of the parts in the assembly, and the C-space obstacles containing that point give the constraints on which parts may be members of the moving subassembly. The boundaries of the obstacles then divide the plane into cells such that all points in any one cell are in the same set of C-space obstacles. The set of superimposed C-spaces is the interference diagram for the assembly. Figure 6 shows the interference diagram for the assembly of Figure 5(a), obtained by superimposing all six pairwise C-spaces.

Each cell in the interference diagram is labeled with the C-space obstacles that include the cell. Each label is a constraint on any subassembly placed in that cell. For example, the cell to the left of the origin in Figure 6 has the labels: A/B, A/C and C/B. If the reference point attached to A is placed anywhere in that cell, A will collide with B and C. Similarly, placing C in that cell will cause a collision with part B. Importantly, since the labels B/A and B/C are missing from the cell, B can safely be placed there, as shown in Figure 7(a). Finally, the subassembly $\{B, C\}$ is safe, since neither B nor C collides with the remaining objects (Fig. 7(b)).

The constraints in a cell can be represented and analyzed using a *blocking*

(a) (b)

Figure 7: Two subassemblies that may be placed in a cell of the interference diagram.



(a) (b)

Figure 8: Two blocking graphs from the interference diagram of Figure 6.

graph [29; 30]. A blocking graph is a directed graph with a node for each part; a label X/Y in the cell induces a directed arc from node X to node Y in the graph. If label X/Y is present, part Y is said to *block* part X in that cell. If a proper subgraph S of the blocking graph has no outgoing arcs to the rest of the graph, then S represents a subassembly that may be placed in the cell, and S is called *free*. Such a subgraph exists if and only if the blocking graph is not strongly connected.[1] If a blocking graph is not strongly connected, then one of its strong components has no outgoing arcs and represents a subassembly that may be placed in the cell. The reduced graph of strong components can be analyzed in a straightforward way to identify all free subassemblies in the blocking graph.

Figure 8(a) shows the blocking graph for the cell we considered above. It is not strongly connected, and because the subassemblies $\{B\}$ and $\{B, C\}$ both have no outgoing arcs, each may be placed anywhere in the cell without collision (see Fig. 7). Figure 8(b) shows the blocking graph for the cell just above the origin in Figure 6;

[1] A *strongly connected component* (or *strong component*) of a directed graph is a maximal subset of nodes such that for any pair of nodes (n_1, n_2) in this subset, a path connects n_1 to n_2. A graph is strongly connected if it has only one strong component.

it is strongly connected, so no proper subassembly can be placed in that position.

4.3 Removing a Subassembly

When a subassembly moves, the common reference point moves through a sequence of cells in the interference diagram, each a neighbor of the previous one. Such a sequence of connected cells is called a *path*, and a blocking graph can be associated with it. A subassembly that can follow a path must be collision free at each point along the path, so the blocking graph for a path is the union of the blocking graphs associated with its cells. If the blocking graph for the path is not strongly connected, then some subassembly can follow it, and the path is called *feasible*.

Let the *initial cell* be the cell containing the origin of the interference diagram. Similarly, the *final cell* is the free, outermost cell of the interference diagram; any subassembly in this cell is completely separated from its complement. To partition the assembly, a feasible path must be found that connects the initial cell to the final cell. Such a feasible path is called a *disassembly path*, and the free subassemblies of the corresponding blocking graph may be removed along the disassembly path.

Figure 9 shows a disassembly path for the assembly of Figure 5(a), and its blocking graph. The removable subassembly is $\{B, C\}$, which as required has no outgoing arcs in the blocking graph.

A number of strategies might be used to search for a disassembly path. For instance, the search might proceed by extending feasible paths from the initial cell outward. The initial cell has no constraints, because in any valid assembly all parts are disjoint in their initial positions. When a path is extended from a cell to one of its neighbors, the constraints for any C-space obstacles whose boundaries are crossed are added to the blocking graph for the path. If at any point the blocking graph becomes strongly connected, the path is discarded. A feasible path that reaches the final cell is a disassembly path, and the free subassemblies of its blocking graph can be removed along that path.

The complexity of the interference diagram itself is clearly polynomial in the number of edges of the parts, and the diagram can be constructed in similar time. The problem is that the number of paths to consider might be very large. Not all of these paths need be considered; for example, when two paths pass through the same cell, and the blocking graph of one path is a subset of the blocking graph of the other, the more constrained path may be discarded. However, such simplifications do not reduce the complexity of the problem, which is exponential in the general case.

4.4 Generalizations

The interference diagram extends directly to more general C-spaces (hence, motions of the subassemblies). In particular, the generalization to three-dimensional translations of polyhedra is clear; the interference diagram is an xyz C-space, and

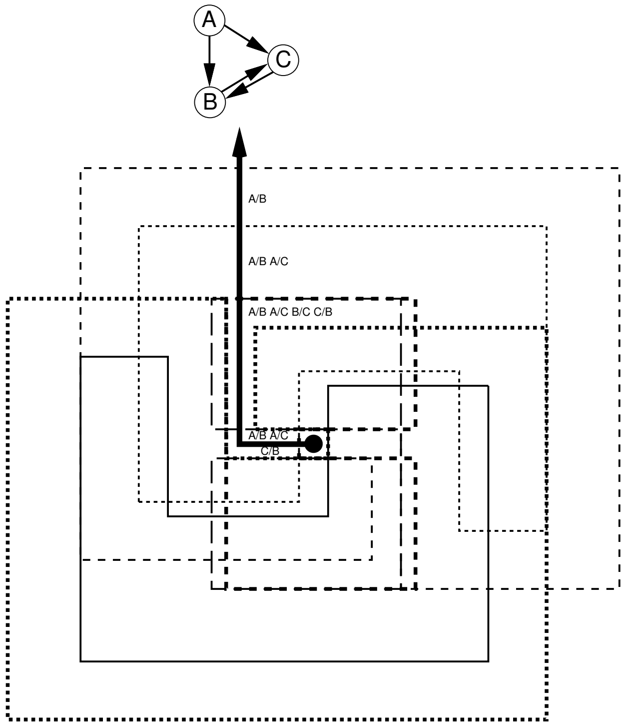


Figure 9: A disassembly path for the assembly in Figure 5(a) and the blocking graph it induces. Only the cells of the interference diagram traversed by the path are labeled.

the regions are polyhedral. This is quite manageable in practice. In theory, one can carry out this construction in arbitrary C-spaces, including the six-dimensional one of general rigid motion, and for parts with arbitrary curved surfaces. For rigid motions, the reference point attached to each part is replaced by a reference frame that coincides with the origin in the initial configuration. A motion of this reference frame can then refer to any subset of the parts. As before, a disassembly path is a path connecting the initial cell (containing the original placement of the frame) to the final cell surrounding the rest of the cells. In each case the interference diagram is of polynomial complexity and can be constructed in polynomial time. However, implementing such constructions is difficult.

To allow contacts, we model the parts as open sets. Then the boundaries of the C-space obstacles are separate regions, in which the two parts do not collide. The constraint X/Y is only added to the blocking graph for a path as that path is extended into the *interior* of the C-space obstacle X/Y.

Although exponential in the worst case, a partitioning algorithm for arbitrary disassembly paths that searches an explicit interference diagram may well prove practical for typical industrial assemblies. We have not implemented the method to test this hypothesis. Instead, the next sections describe special cases of the interference diagram that yield polynomial-time partitioning for restricted assembly motions. These cases prove very useful in practice.

5 Partitioning with Single Translations

Very few assembly operations in industry require the kind of convoluted paths that the general partitioning problem allows; in fact, an assembly that needs such a complicated motion is a prime candidate for redesign [4]. Instead, the vast majority of operations can be accomplished by relatively simple motions, such as single translations to infinity or extended twisting motions. In this section we examine the effect on the partitioning problem when disassembly paths are limited to single translations.

A single-translation disassembly path is a ray with its base at the origin of the interference diagram. As in the general case, the cells intersected by the ray determine a set of constraints on subassemblies that may be removed along the ray. A ray is a disassembly path if the blocking graph induced by these constraints is not strongly connected.

Now consider what happens when the ray rotates to new disassembly directions. The blocking graph will not change as long as the rotating ray intersects the same set of cells. Indeed, the blocking graph changes only when the ray begins to intersect a new C-space obstacle or stops intersecting one of the obstacles. We could partition the assembly by constructing the interference diagram and then sweeping a ray through it, checking the blocking graph induced at each change. However, the same computation can be performed more efficiently by considering the orientations of the ray directly.

For planar assemblies, we represent the set of all disassembly directions by the points on the unit circle S^1. Projecting each C-space obstacle X/Y onto the circle by a central projection, we obtain a single arc; this arc consists of exactly those directions in which the ray will intersect X/Y.[2] For example, Figure 10 shows two polygonal parts from Figure 5(a), their C-space obstacle, and the projection onto the unit circle. The C-space obstacles give rise to $n(n-1)$ overlapping arcs on S^1; the blocking graph only changes at the endpoints of these arcs. Checking each graph for strong connectedness, a disassembly path can be detected if one exists.

In three dimensions, we represent the disassembly directions as points on the unit sphere S^2. Again, we do not need to construct the interference diagram explicitly, but only its projection on S^2. The C-space obstacle for two polyhedral parts is a polyhedron, and when projected onto the sphere, this becomes a region bounded

[2] If X and Y are in contact in their initial positions, this projection must be handled more carefully, but the result is still a single arc.

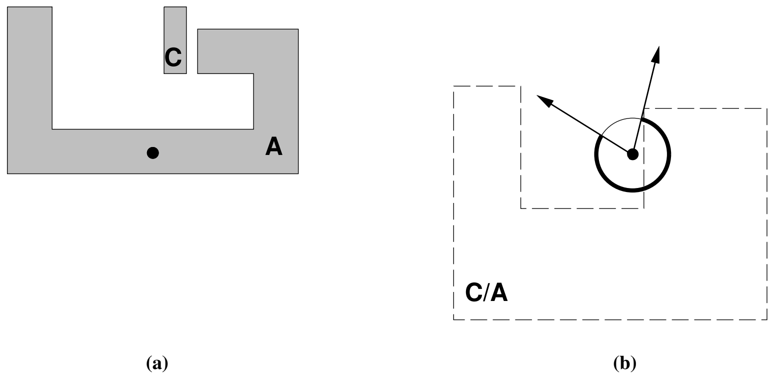


Figure 10: (a) Two polygonal parts with common reference point, and (b) their C-space obstacle projected onto the unit circle (the bold arc).

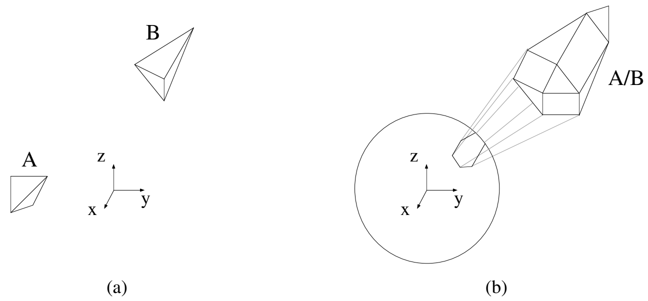


Figure 11: (a) Two simple polyhedral parts, and (b) their C-space obstacle, projected onto the sphere.

by arcs of great circles (Fig. 11). This region is the set of all translation directions in which the two parts will collide. The edges of the regions for all pairs of parts determine an arrangement of faces on S^2; the blocking graph for a translational disassembly path only changes at the boundaries of these regions. Each face is labeled with the labels that project onto it. In practice, the C-space obstacles need not be constructed explicitly; their projections can be built by triangulating the faces of the polyhedra, computing and projecting C-space obstacles for each pair of triangles, and taking their union on the sphere.

For an assembly of n polyhedra with a total of v vertices, the $n(n-1)$ regions on S^2 are bounded by a total of $O(v^2)$ arcs. The arrangement they determine has complexity $O(v^4)$ and can be calculated in the same time bound [5]. The blocking graphs can be found by traversing the arrangement, making small changes as each cell boundary is crossed. Checking a blocking graph for strong connectedness may require $O(n^2)$ time, for a total of $O(n^2v^4)$ time to partition a polyhedral assembly with single translations. If all partitionings are desired, they can be found by analyzing the reduced graph of strong components for each blocking graph. To find an assembly sequence of operations with single translations requires $n-1$ applications of the partitioning algorithm, or $O(n^3v^4)$ time.

6 Partitioning with Infinitesimal Rigid Motions

This section describes another special case of the interference diagram that is useful when an assembly cannot be partitioned using single translations or when other partitionings are desired to optimize the assembly sequence in some way. It is motivated by the following observation: Any disassembly path, no matter how convoluted, must begin with the initial cell (the origin of the interference diagram) followed by one of its neighbors. Hence a necessary condition on removable subassemblies can be computed by exploring the local topology of the neighborhood of the initial cell in the interference diagram. The subassemblies that can follow a path from the initial cell to one of its immediate neighbors are exactly those that can move a very small distance in the assembly. Each subassembly that satisfies this *local freedom* constraint can then be passed to a path planner or other evaluation technique.

The topology of the initial cell's neighborhood is determined wholly by the contacts between parts in their initial placement; no other constraints come into play until a subassembly moves a finite distance. Two parts are in contact when their boundaries intersect. In experiments, industrial assemblies have proven to be quite constrained by their contacts [2; 15; 32], resulting in very few subassemblies that must be further tested for removability.

Let us consider disassembly motions consisting of arbitrary rigid motions (combining translation and rotation) in 3D for polyhedral parts; the 2D and translation cases are just simpler. The interference diagram in this case is a 6D C-space construction. The neighborhood of interest is an infinitesimal 5D sphere surrounding the origin of the interference diagram. This sphere is cut into faces of dimension 0 (a vertex) to 5 by the surfaces of the interference diagram that neighbor the initial cell, and each face is labeled with the blocking graph of the cell containing it. By checking these blocking graphs for strong connectedness, we can determine the subassemblies that are free to move a small distance in the assembly.

We represent the infinitesimal sphere by the unit sphere S^5. We could construct this arrangement on the sphere by building the 6D interference diagram for rigid motions, then cutting S^5 by the tangent hyperplanes to surfaces through the origin. However, the tangent hyperplanes and arrangement on S^5 can be computed more directly from the contacts of the assembly.

The constraints that contacts impose on the relative motion of the two contacting

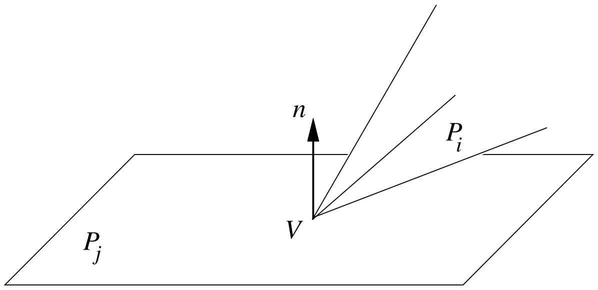


Figure 12: A point–plane contact.

bodies have been studied in depth (see for instance [23; 25; 27]). An infinitesimal rigid motion can be described by a 6D vector $dX = (dx, dy, dz, d\alpha, d\beta, d\gamma)$ giving the components of translation and rates of infinitesimal rotation about the coordinate axes, respectively. Consider two parts P_i and P_j in contact, and let V be a vertex of P_i (Fig. 12). A motion dX applied to P_i causes V to undergo a translation $t = J_V dX$, where J_V is a constant 3×6 Jacobian matrix relating the motion of P_i to the translation of V. Assume that P_i and P_j are in contact such that V is contained in a face of P_j with outward normal n. Then dX causes V to penetrate P_j when $n \cdot t < 0$, to break the contact when $n \cdot t > 0$, and to slide in F when $n \cdot t = 0$. The hyperplane $n \cdot t = 0$ (the sliding case) is the tangent hyperplane at the origin to a surface of the interference diagram. Hence the motions dX *allowed* by the contact are those satisfying $n \cdot (J_V dX) \geq 0$.

The motions allowed by other contacts between polyhedra can be expressed with conjunctions of point–plane contact constraints. For instance, consider a contact between a face of P_i and a face of P_j with outward normal n (Fig. 13). The set of motions dX allowed by this contact is the intersection of the closed half-spaces $n \cdot (J_{V_k} dX) \geq 0$ computed for each of the vertices V_k of the convex hull of the intersection of the two faces. In Figure 13 the vertices V_k are circled. (See [12; 29] for the constraints arising from other polyhedral contact types, and [27] for the constraints from curved surfaces in contact.)

The intersection of the half-spaces of point–plane constraints between each pair of parts P_i and P_j is a closed convex 6D polytope in the space of infinitesimal motions. P_j blocks the motion of P_i for all motions outside this polytope and for no motions inside the polytope. The intersection of these polytopes with the sphere S^5 determines an arrangement of regions of dimensions $0, \ldots, 5$ on S^5. These regions capture the local topology of the origin of the interference diagram.

Let there be $m \leq \binom{n}{2}$ pairs of parts in contact in the assembly, described by a total of $c \in O(v^2)$ point–plane constraints,[3] where $c \geq m$. Then the arrangement

[3] In most real assemblies, $m \approx n$, $c \ll v^2$, and $c \gg m$.

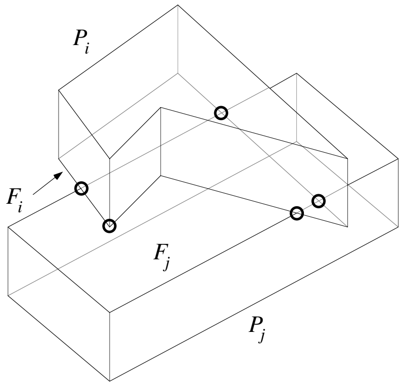


Figure 13: Plane–plane contact expressed as point–plane constraints.

of polytopes on S^5 has $O(c^5)$ cells and can be constructed in $O(c^5)$ time using a topological sweep [7]. Because each cell's blocking graph can have at most $2m$ edges, checking it for strong connectedness can be performed in time $O(m)$. Hence, identifying a subassembly that is locally free has total running time of $O(mc^5)$. This bound will improve if and when better algorithms to calculate arrangements in high dimensions are developed. Note that in practice, many free subassemblies will have to be generated from the reduced graph of strong components for each blocking graph, because some locally free subassemblies may not satisfy global constraints.

A version of this algorithm has been implemented and tested on a number of industrial assemblies. The experimental system allows parts with planar, cylindrical, and some helicoidal surfaces. The implemented version of the algorithm considers the set of all infinitesimal translations $dX = (dx, dy, dz)$, plus a finite set of "suggested motions" inferred from nonplanar contacts in the assembly. For example, a cylindrical contact suggests pure rotation about its axis, and a threaded contact between two helicoidal surfaces suggests a screwing motion, etc. The set of suggested rigid motions is incomplete but accounts for most motions required by actual assemblies. Figure 14 shows a model aircraft combustion engine with 42 parts; the planner identifies all 33 subassemblies of the engine that satisfy local freedom constraints (25 of which satisfy global constraints) in 8.5 seconds on a DECstation 5000/200.

7 Discussion

We have shown that the partitioning problem is NP-complete for assemblies of simple polygons in the plane when arbitrary motions are allowed to separate the two subassemblies. On the other hand, when the motions allowed for assembly are limited to single translations or infinitesimal rigid motions, partitioning is in P.

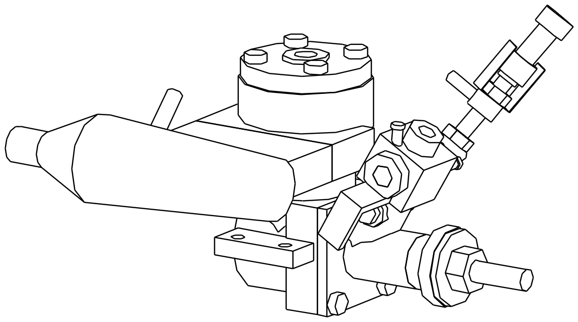


Figure 14: Model aircraft engine

This raises the question: under what conditions can partitioning be accomplished in polynomial time?

One factor appears to be the "complexity" of the motions allowed to separate the two subassemblies. For instance, one way to measure the complexity of a translational path is by the number of straight-line translations it includes. In separate work, Halperin and Wilson [11] investigate the problem of partitioning an assembly with a small number t of translations. The case $t = 1$ is the single translation case described above. For the case $t = 2$, a disassembly path (in two dimensions) can be described by a triplet (x, y, ϕ), where (x, y) is the end point of the first translation, and ϕ is the orientation of the second translation to infinity. For instance, the disassembly path shown in Figure 9 is of this type.

As in the single translation case, the cells of the interference diagram cut by a two-translation path (x, y, ϕ) do not change when x, y, and ϕ vary slightly. The set of cells cut by the path changes only at certain critical values. These values determine surfaces in (x, y, ϕ)-space, dividing this space into regions; each region represents a set of two-translation paths that have the same blocking graph. Checking the graphs for strong connectedness yields a polynomial-time two-translation partitioning algorithm.

In theory, this easily extends to disassembly paths of more than two translations. A t-translation path is described by a tuple $(x_1, y_1, \ldots, x_{t-1}, y_{t-1}, \phi)$, where (x_i, y_i) is the end point of the ith translation in the path. The cells of the interference diagram now determine hypersurfaces in a $(2t{-}1)$-dimensional space of disassembly paths. For any fixed t, this approach yields a polynomial-time partitioning algorithm; however, the number of regions in the arrangement can be exponential in t, so the resulting algorithm will be practical only for very small values of t.

In experimental assembly planning systems, an additional constraint is often added to the partitioning problem, requiring that each of the two separated subassemblies be connected. As mentioned at the end of Section 3, a subassembly is considered *connected* if the union of its parts is a connected set. This constraint is useful in practice, because unconnected subassemblies are usually more difficult to grasp, fixture, and manipulate. As noted at the end of Section 3, partitioning assemblies of polygons in the plane into connected subassemblies is NP-complete [17]. Partitioning with infinitesimal rigid motions, as in Section 6, can be accomplished in polynomial time when connected subassemblies are required [29]. However, the computational complexity of the partitioning problem with single translations, as considered in Section 5, remains unknown when connected subassemblies are required. A practical algorithm to solve this last problem could have important applications in assembly planning systems.

Since this article was submitted, a more efficient approach has been developed to check each graph for strong connectedness in a long list of related blocking graphs. When disassembly motions are limited to single translations or infinitesimal motions, the interference diagram becomes an arrangement of regions, each associated with a blocking graph. The blocking graphs of neighbor regions differ only slightly: If a single surface separates two regions, then the blocking graphs for the regions differ by at most one arc. Checking a single graph for strong connectedness requires time linear in the number of edges, and the time to check all the blocking graphs dominates the running time of each algorithm. Khanna *et al.* [19] group consecutive graphs into phases that are checked together, achieving amortized time per graph of $O(n^{1.38})$, where n is the number of nodes in each graph. This reduces the running time of the algorithms in Sections 5 and 6 accordingly.

Finally, the techniques considered in this article consider only the constraints arising directly from the geometry of the parts. There are many other constraints on assembly plans, such as the need to grasp, fixture, and manipulate the assemblies. Constraints also arise from the effects of uncertainty or flexible parts on the assembly process. Much work remains to be done in these areas.

Appendix

We discuss here the choice of x_i's that are used in the assignment mechanism of our construction of Section 3. The x_i's indicate the relative horizontal positions of the parts of the moving subassembly S and are preserved throughout the motion of S.

Assume that $S = \{R_0, R_1, \ldots, R_n\}$, where R_0 denotes the key and R_i denotes either U_i or $\overline{U_i}$, $1 \leq i \leq n$. We say that R_i is at position x and write $pos(R_i) = x$ iff the x-coordinate of the bottom-left vertex of the rectangle R_i is x.

Lemma: *Let $x_i = 8 \cdot a_i$, $i = 0, 1, \ldots, n$, where $a_0, a_1, \ldots, a_n$ is an integer sequence such that all pairwise differences $a_i - a_j$ are distinct when $i \neq j$. When $|x_p - pos(R_i)| < 4$, $p \neq i$, then for all R_j, $j \neq i$, we have that $|x_q - pos(R_j)| > 4$.*

Proof At some time during the motion of S, R_i is 4-close to the position x_p, that is $|x_p - pos(R_i)| < 4$. Consider now any R_j, $j \neq i$, and any position x_q with $q \neq j$. Let us bound the quantity $|x_q - pos(R_j)|$ from below using the triangular inequality:

$$|x_q - pos(R_j)| \geq |(x_p - pos(R_i)) - (x_q - pos(R_j))| - |x_p - pos(R_i)|$$

or equivalently,

$$|x_q - pos(R_j)| \geq |(x_p - x_q) + (pos(R_j) - pos(R_i))| - |x_p - pos(R_i)|$$

However, because $|x_p - pos(R_i)| < 4$ and $pos(R_j) - pos(R_i) = x_j - x_i = 8 \cdot a_j - 8 \cdot a_i$ the above inequality becomes:

$$|x_q - pos(R_j)| \geq 8 \cdot |a_p - a_q - (a_j - a_i)| - 4.$$

From the hypothesis about a_n the quantity $|a_p - a_q + (a_j - a_i)| = |(a_j - a_q) - (a_i - a_p)|$ must be at least one, because $j \neq i$ and $i \neq p$. Thus we get:

$$|pos(R_j) - x_q| \geq 8 \cdot 1 - 4 \geq 4.$$

which completes the proof. □

The above lemma guarantees that for any k, if R_k is close to position x_p, $p \neq k$, and must go through the hole that has been created at this position, then none of the $R_0, R_1, \ldots, R_{k-1}, R_{k+1}, \ldots R_n$ is close to a position x_q. That is, all of $R_0, R_1, \ldots, R_{k-1}, R_{k+1}, \ldots R_n$ are in the wide parts of the assignment mechanism and can follow the constrained motion of R_k.

The key element in the above proof is the property of the sequence a_i; namely, that the pairwise differences of its terms are all distinct. The "8" is just a scaling factor that gives extra space for the width of the parts. It is not hard to choose the a_i's; a straightforward example of sequence that has all its terms distinct is $a_i = 2^i$. However, this sequence yields an exponential length for the assignment mechanism. Using a result of Erdös [10], it is possible to select the a_i's in such a way that $max\{a_0, \ldots, a_n\} = O(n^2)$. Then the physical length of the assignment mechanism becomes a polynomial of n.

Acknowledgments

The authors thank D. Halperin for helpful comments. R. Wilson was supported by a grant from the Stanford Integrated Manufacturing Association. L. Kavraki and J. C. Latombe were partially supported by ARPA grant N00014-92-J-1809, NSF grant IRI-9306544, and a grant from the Stanford Integrated Manufacturing Association. T. Lozano-Pérez was partially supported by a grant from DARPA under ONR contract N00014-91-J-4038.

References

[1] E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles, with applications to planning assemblies. In *Proc. of the 5th ACM Symp. on Computational Geometry*, pages 334–343, 1989.

[2] D. F. Baldwin. Algorithmic methods and software tools for the generation of mechanical assembly sequences. Master's thesis, Massachusetts Institute of Technology, 1990.

[3] D. F. Baldwin, T. E. Abell, M.-C. M. Lui, T. L. De Fazio, and D. E. Whitney. An integrated computer aid for generating and evaluating assembly sequences for mechanical products. *IEEE Trans. on Robotics and Automation*, 7(1):78–94, 1991.

[4] G. Boothroyd. *Assembly Automation and Product Design.* Marcel Dekker, 1991.

[5] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39(1):1–54, 1992.

[6] A. Delchambre and P. Gaspart. KBAP: An industrial prototype of knowledge-based assembly planner. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 2404–2409, 1992.

[7] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[8] H. A. Ernst. *MH–1: A Computer-Operated Mechanical Hand.* Sc.D. thesis, Massachusetts Institute of Technology, 1961.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, 1979.

[10] H. Halberstam and K. R. Roth. *Sequences.* Springer-Verlag, 1983. Page 90.

[11] D. Halperin and R. H. Wilson. Assembly partitioning with a constant number of translations. Technical Report SAND94–1819, Sandia National Labs, 1994.

[12] H. Hirukawa, T. Matsui, and K. Takase. Automatic determination of possible velocity and applicable force of frictionless objects in contact from a geometric model. *IEEE Trans. on Robotics and Automation*, 10(3):309–322, 1994.

[13] R. L. Hoffman. A common sense approach to assembly sequence planning. In L. S. Homem de Mello and S. Lee, editors, *Computer-Aided Mechanical Assembly Planning*, pages 289–314. Kluwer Academic Publishers, 1991.

[14] L. S. Homem de Mello and S. Lee, editors. *Computer-Aided Mechanical Assembly Planning.* Kluwer Academic Publishers, 1991.

[15] L. S. Homem de Mello and A. C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Trans. on Robotics and Automation*, 7(2):228–240, 1991.

[16] H. Inoue. Force feedback in precise assembly tasks. Technical Report AIM–308, MIT AI Lab, 1974.

[17] L. Kavraki and M. N. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*, 1995. To appear.

[18] L. Kavraki and J.-C. Latombe. Complexity of partitioning a planar assembly. Technical Report STAN-CS-93-1467, Dept. of Computer Science, Stanford Univ., 1993.

[19] S. Khanna, R. Motwani, and R. H. Wilson. Graph certificates, lookahead in dynamic graph problems, and assembly planning in robotics. Technical Report SAND94-3128, Sandia National Labs, 1994.

[20] H. Ko and K. Lee. Automatic assembling procedure generation from mating conditions. *Computer Aided Design*, 19(1):3–10, 1987.

[21] S. S. Krishnan and A. C. Sanderson. Path planning algorithms for assembly sequence planning. In *Intl. Conf. on Intelligent Robotics*, pages 428–439, 1991.

[22] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[23] B. Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2:541–558, 1987.

[24] B. K. Natarajan. On planning assemblies. In *Proc. of the 4th ACM Symp. on Computational Geometry*, pages 299–308, 1988.

[25] M. S. Ohwovoriole. *An Extension of Screw Theory and its Application to the Automation of Industrial Assemblies.* PhD thesis, Stanford Univ., April 1980.

[26] R. S. Palmer. *Computational Complexity of Motion and Stability of Polygons.* PhD thesis, Cornell Univ., 1989.

[27] E. Rimon and J. Burdick. Mobility of bodies in contact—I: A new second order mobility index for multiple-finger grasps. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 2329–2335, 1994.

[28] J. T. Schwartz and M. Sharir. On the piano movers' problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[29] R. H. Wilson. *On Geometric Assembly Planning.* PhD thesis, Stanford Univ., March 1992. Stanford Technical Report STAN-CS-92-1416.

[30] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1994.

[31] R. H. Wilson, J.-C. Latombe, and T. Lozano-Pérez. On the complexity of partitioning an assembly. Technical Report STAN-CS-92-1458, Dept. of Computer Science, Stanford Univ., 1992.

[32] R. H. Wilson and J.-F. Rit. Maintaining geometric dependencies in assembly planning. In L. S. Homem de Mello and S. Lee, editors, *Computer-Aided Mechanical Assembly Planning*, pages 217–242. Kluwer Academic Publishers, 1991.

[33] J. D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly.* PhD thesis, Univ. of Michigan, 1988.