# Assembly Sequencing for Arbitrary Motions

Tomás Lozano-Pérez

MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139
tlp@ai.mit.edu

Randall H. Wilson

Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305
rwilson@cs.stanford.edu

## Abstract

We consider the following problem that arises in assembly planning: given an assembly, identify a subassembly that can be removed without disturbing the rest of the assembly. Solutions to this problem have been presented when the motions allowed for the separation are of certain restricted types. In this paper, we generalize these solutions to allow arbitrary relative motions between the two subassemblies. The generalization is based on a configuration space construction that makes explicit the spatial interferences between every pair of parts for every relative motion. Based on this construction, we can simultaneously determine (1) the path by which a subassembly can be removed and (2) the parts contained in the subassembly. While the algorithms resulting from this construction may in the worst case require time exponential in the length of the removal path, the expected complexity for realistic assemblies is an open question.

## Introduction

This paper addresses a geometric problem in assembly sequencing posed by Wilson [5]: given an assembly of parts, identify subassemblies that can be removed from the assembly. That is, identify a subset of the parts that can be moved (as a single rigid object) to infinity without disturbing the other parts. This is the *partitioning problem* for the assembly. Figure 1 shows a simple example. In this example, the box can be moved away from the assembly, each screw can be removed and the subassembly composed of the lid and the screws can also be removed.

In this paper, we retain the assumptions that the assembly sequences are *binary* and *monotone*, that is,

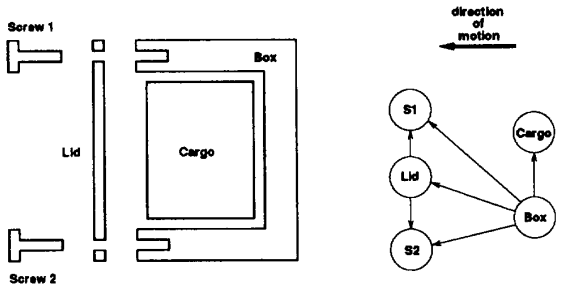


Figure 1: Simple assembly example with a directional blocking graph for one direction of motion [5].

only one group of parts moves at a time, and the motion completely separates the moved parts from the rest of the assembly. In other words, no parts are placed in intermediate positions. For example, Figure 2(b) is not a monotone, binary assembly.

Wilson solves the partitioning problem when the disassembly motions are restricted to either (a) instantaneous translations or rotations or (b) unbounded translations. The solution involves constructing graphs that identify which parts collide, given an instantaneous displacement in a given direction. These graphs are called the *directional blocking graphs (DBGs)* for the assembly (see Figure 1). Any subgraph of the DBG with no outgoing links represents a subassembly that is free to move (incrementally) in the indicated direction. Wilson points out that only a finite set of DBG's need to be built. The space of allowed motions can be partitioned into regions having the same graph. This partitioned space of motions and the associated DBGs constitute the *non-*



1050-4729/93 $3.00 © 1993 IEEE

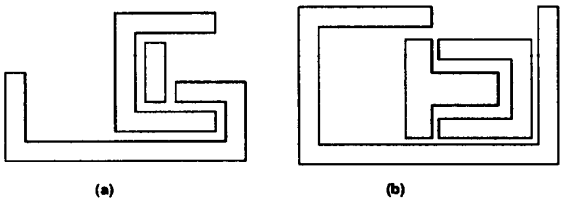


Figure 2: Examples of assemblies requiring non-straight-line motions for disassembly. (a) is a monotone binary assembly while (b) is not.

*directional blocking graph (NDBG)* for the assembly.

The solution proposed by Wilson handles many assemblies but does not handle situations where a non-straight-line path is required to disassemble some parts or subassemblies. Figure 2(a) shows such an assembly. In this paper, we describe a construction that, in principle, eliminates this restriction.

# 1 The Interference Diagram

Our solution will involve constructing a somewhat complicated configuration-space diagram, so we can illustrate it graphically only in simple examples such as the one in Figure 2(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 C-space regions is full-dimensional. The construction generalizes to more general motions and to situations involving contacts.

Since we do not know *a priori* which parts will be moving and which are stationary, we must determine, for every pair $(X, Y)$ of parts, the set of placements of $X$ in which it intersects with $Y$. This set of placements is the configuration space (or C-space) obstacle for the pair of parts [4]. The C-space obstacle for part $A$ moving with $B$ as an obstacle is labeled $A/B$ while the C-space obstacle for $B$ moving with $A$ as an obstacle is labeled as $B/A$. 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 $B/A$ is simply $A/B$ rotated by $\pi$ radians or, equivalently, every forbidden translation $x$ of $A$ has a corresponding forbidden translation $-x$ of $B$. Figure 3 shows the 6 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. 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.

If we are interested in moving one part away from the assembly, it is clear that these pairwise C-spaces give us the information we need to plan these motions. We can test if part $B$, for example, can be moved by examining whether a motion exists in the space outside the union of $B/A$ and $B/C$. Similarly, one can test for legal motions of $A$ relative to the obstacles $A/B$ and $A/C$ and for motions of $C$ relative to $C/A$ and $C/B$.

The question is how to deal with the motions of subassemblies. In principle, we could construct all subassemblies and compute the C-space obstacles for them, but there are an exponential number of subassemblies to consider. Interestingly, we can plan the motions of subassemblies without computing any additional C-space obstacles. The reason is that in a subassembly the parts all move rigidly and so the C-space obstacle of a subassembly relative to a stationary part is simply the union of the pairwise C-space obstacles of the parts comprising the subassembly.

Conceptually, we can label the region occupied by each C-space obstacle with its label, e.g. $A/B$, indicating that the placement of $A$ in this region is forbidden when $B$ is in its original position. Now, since all of the pairwise C-spaces were computed with the same reference point, we can simply superimpose all the C-space obstacles. That is, all these obstacles are embedded in the same configuration space. This may be a little confusing if one is used to asking "This is the configuration space of what?" If it helps, one can think of these as a set of parallel C-spaces. Note that any point in the combined C-space can represent a displacement of any of the parts in the assembly. The set of superimposed C-spaces is called the *interference diagram* for the assembly.

Figure 4(a) shows the interference diagram for the assembly of Figure 2(a), obtained by combining the pairwise C-spaces. Each region is labeled with the pairs that collide. Note that the whole diagram has the expected symmetry: every free displacement $x$ for a subassembly has a corresponding free displacement $-x$ for the rest of the assembly.

The interesting question regards the semantics of the regions in the interference diagram. Clearly, each region will inherit a set of labels corresponding to the obstacles that include it. Each label is a constraint on the subassemblies that may be placed in the region. A label $X/Y$ means that if part $X$ is in that region, it

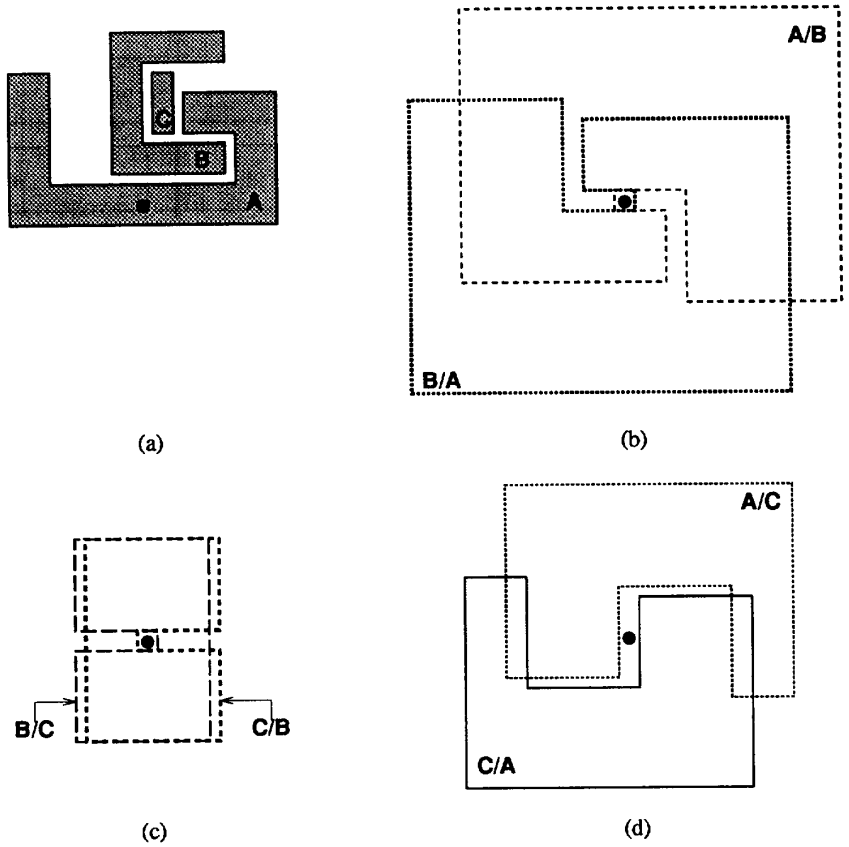


Figure 3: C-space diagrams for each pair of parts in the assembly. The notation X/Y indicates that X collides with Y in that region.

collides with part $Y$ in its initial position. Therefore, if a subassembly in that region includes $X$, it must also include $Y$, or the subassembly will collide with the stationary parts.

For example, consider the region just to the left of the origin in Figure 4(a) with the labels: $A/B$, $A/C$ and $C/B$. If the reference point of $A$ is placed anywhere in that region, $A$ will collide with $B$ and $C$. Similarly, placing $C$ in that region will cause a collision with part $B$. Importantly, since the labels $B/A$ and $B/C$ are missing from the region, $B$ can safely be placed there. Likewise, since $C/A$ is missing, we can conclude that $C$ would not collide with $A$. Finally, note that we can also conclude that $B$ and $C$ treated as a subassembly are safe, since neither collides with the remaining objects, in this case, only $A$.

In short, the labels in a region define one of Wilson's *blocking graphs*. A blocking graph is a directed graph with a node for each part; a label $X/Y$ in the region induces a directed arc from node $X$ to node $Y$ in the graph. 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 there. As we know from [5], such a subgraph exists if and only if the blocking graph is not strongly connected.[1] For the region we considered above, $B$ and $C$ form such a subgraph and may be placed in the region without collision. Figure 4(b) shows the subassemblies that may be placed in each of the labeled regions in Figure 4(a).

[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.

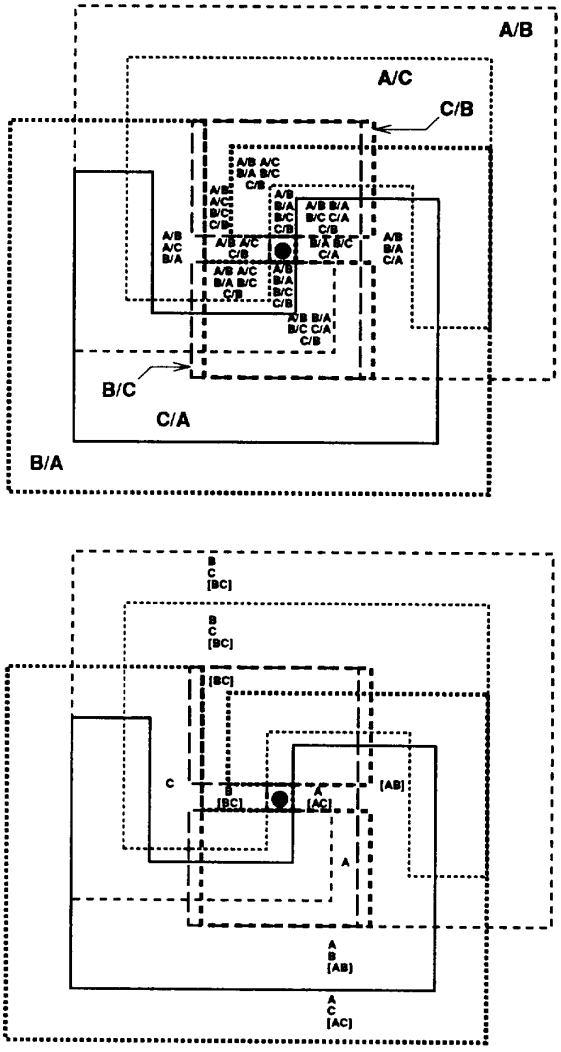


Figure 4: (a) The interference diagram for the assembly of Figure 2(a). The small labels indicate all the C-space obstacles that share that region. (b) Interference diagram in which regions are labeled by which subassemblies are free to be placed within the region.

# 2 Disassembly Paths

To search the interference diagram for a removable subassembly, we determine the subassembly simultaneously with the path it will follow. A *path* in the interference diagram is a sequence of regions, each a neighbor of the previous one. We are interested in finding a path that connects the *initial region* (the region including the origin) to the free, outermost region of the interference diagram (call this the *final region*). In addition, a subassembly must be able to follow the path without colliding with the rest of the parts. Such a subassembly must satisfy all the constraints encountered in regions along the path.

A path is *feasible* if some proper subassembly can follow it without collision, and a feasible path connecting the initial to the final region is called a *disassembly path*. A subassembly can follow a path exactly when it is collision-free at every point along the path. This means that all constraints in regions along the path must be satisfied by the subassembly. To capture this, we associate to any path a blocking graph whose arcs are given by the union of all constraints encountered in regions along the path. Then clearly a path is feasible exactly when its blocking graph is not strongly connected.

The search for a disassembly path proceeds by extending feasible paths from the initial region outward. The initial region has no constraints, since in any valid assembly all parts are disjoint in their initial positions. When a path is extended from a region 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 may be discarded. A feasible path that reaches the final region is a disassembly path, and its blocking graph gives the subassemblies that can be removed along the path.

Figure 5 shows a disassembly path for the assembly of Figure 2(a), and its blocking graph. The removable subassembly is composed of parts B and C, which as required has no outgoing arcs in the blocking graph.

In general a region may be the endpoint of many feasible paths. For instance, consider the assembly in Figure 6 consisting of three lettered parts $a, b, c$ and a large part, call it $d$, consisting of the (disconnected) gray regions. Similar assemblies can be built in 3D of connected parts. Subassembly $ac$ can be moved through the passage marked 1, and subassembly $ab$ can be moved through passage 2. Thus the region in the interference diagram corresponding to point 3 is reached by at least two distinct paths, one having a constraint $b/d$ and one having $c/d$. Which of these

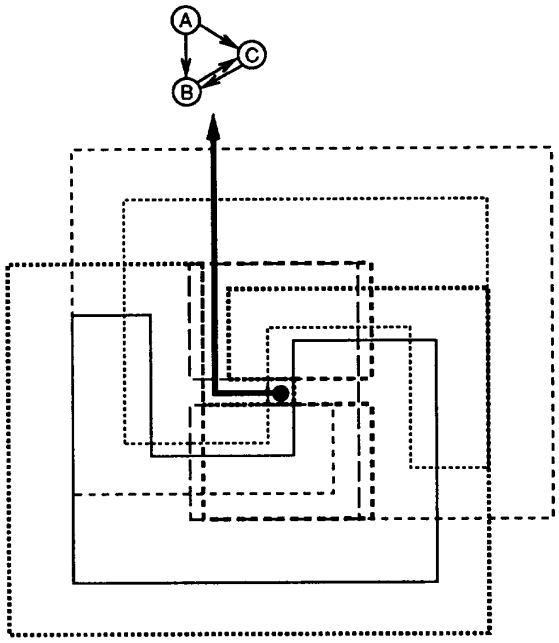


Figure 5: A disassembly path for the assembly in Figure 2(a) and the blocking graph it induces.

paths is extended determines whether passage 4 or 5 must be followed for disassembly. With additional parts and passages, the number of paths to a single region may become very large.

However, not all paths that reach a region must be extended: some of the paths are subsumed by others. Consider two paths $P_1$ and $P_2$ ending in the same region, and their respective blocking graphs $G_1$ and $G_2$. If the arcs of $G_1$ are a subset of those of $G_2$, then the constraints accumulated along path $P_1$ are a subset of those along $P_2$. In other words, any subassembly that could follow $P_2$ could also follow $P_1$. Therefore if a disassembly path begins with $P_2$, another could be constructed by substituting $P_1$ for $P_2$. Hence path $P_2$ need not be extended. In fact, let $G^*$ denote the transitive closure of a graph $G$; then $P_2$ can be pruned whenever the arcs of $G_1^*$ are a subset of those of $G_2^*$. Note that this is not the case for the two paths discussed above for Figure 6.

The search for a disassembly path may follow one of several strategies, none of which is clearly superior. Since a single region in the interference diagram may be the endpoint of many feasible paths, a breadth-first search would allow maximum use of the above

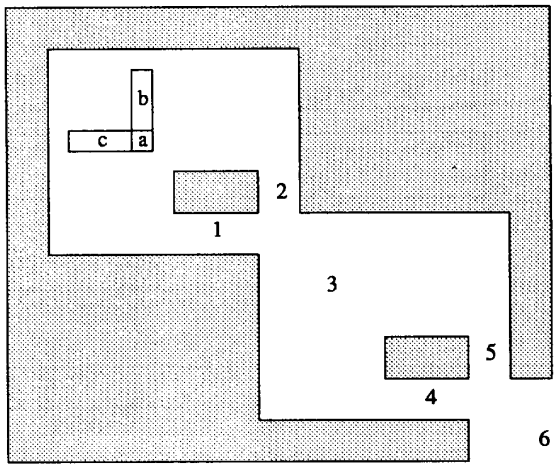


Figure 6: An assembly with several paths leading to a single region in its interference diagram

pruning rules, but the number of blocking graphs in each region may grow very large. Depth-first search, or best-first search with an appropriate heuristic, might be preferable in some situations.

# 3 Complexity

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. However, the simple search strategies mentioned above do not immediately yield a worst-case polynomial-time algorithm to find a disassembly path. This is because a large number of feasible paths might end in a given region in some cases, and it is unclear whether they must all be considered for continuation. This may lead to a large number of useless paths being extended, or to a large number of blocking graphs in one region. In general, the number of paths that must be considered is exponential in the number of regions each path may traverse.

In fact, analyzing the complexity of searching the interference diagram has very recently led to computational complexity results for the partitioning problem itself. In [6] Wilson, Latombe and Lozano-Pérez show that the partitioning problem for assemblies of polyhedra is NP-complete, and Kavraki and Latombe [2] strengthen this result to assemblies of polygons in the plane. In spite of these hardness results, searching the interference diagram may well prove practical for typi-

cal industrial assemblies, which do not have the maze-like qualities of the assemblies constructed in [6, 2].

# 4 Generalizations

The interference diagram extends directly to more general configuration spaces. In particular, generalization to three-dimensional translations of polyhedra is clear; the interference diagram is an $xyz$ C-space, and the regions are polyhedra. This is quite manageable in practice. In general, one can carry out this construction in arbitrary C-spaces, including the 6-dimensional one of general motion. 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 between parts, the boundaries of the C-space obstacles are considered separate regions, in which the two parts do not collide. Then 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 $X/Y$.

The most serious barriers to actual implementation are numerical accuracy problems. Once we start allowing contacts between the parts, we need to represent and label all the vertices and edges of the C-space regions. This cannot be done reliably with floating point arithmetic. This is a problem common to many geometric algorithms. There has been a great deal of progress in developing techniques for robust geometric computations (see for instance [1, 3]), and such methods could be employed here. Alternatively, one could shrink the parts to within the inner envelope of their tolerance so as to ensure that there are no glancing contacts in the assembly. In either case, this issue requires careful attention.

# 5 Relation to the NDBG

Wilson's *non-directional blocking graph* (NDBG) consists of a partition of the space of incremental (or indefinitely extended linear) motions each labeled with a distinct blocking graph. In the framework of this paper, the NDBG for incremental motions captures the local topology of the regions immediately bordering the initial region in the interference diagram for the corresponding type of motion. It embodies a necessary condition on the existence of a disassembly path. Because it only examines the local properties of the interference diagram around the initial region, the NDBG for infinitesimal motions can be constructed and analyzed in polynomial time [5].

On the other hand, the NDBG for indefinitely extended translations represents a central projection of the interference diagram regions onto a unit sphere centered at the origin. Each cell on the sphere has a blocking graph whose arcs are the union of the blocking graphs for the regions of the interference diagram projected onto that cell. This corresponds to limiting the paths through the interference diagram to straight lines. Again, this restriction on the type of paths allows a polynomial-time algorithm to analyze the NDBG [5, 7]. The interference diagram is, therefore, a natural generalization of the NDBG.

## Acknowledgments

The authors would like to thank Jean-Claude Latombe and Achim Schweikard for useful discussions and comments. The first author was supported by a grant from DARPA under ONR contract N00014-91-J-4038. The second author was supported by a grant from the Stanford Integrated Manufacturing Association.

# References

[1] C. M. Hoffman, J. E. Hopcroft, and M. S. Karasick. Towards implementing robust geometric computations. In *Proc. of the 4th ACM Symp. on Computational Geometry*, 1988.

[2] L. Kavraki and J.-C. Latombe. Partitioning a planar assembly is NP-complete. Manuscript, January 1993.

[3] Z. Li and V. Milenkovic. Constructing strongly convex hulls using exact or rounded arithmetic. In *Proc. of the 6th ACM Symp. on Computational Geometry*, 1990.

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

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

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

[7] R. H. Wilson and A. Schweikard. Assembling polyhedra with single translations. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 2392–2397, 1992.