SANDIA REPORT

SAND94–3124 • UC–705
Unlimited Release
Printed April 1995

Assembly Sequencing with Toleranced Parts

Jean-Claude Latombe, Randall H. Wilson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

DISCLAIMER

SANDIA REPORT

SAND94–3124 • UC–705
Unlimited Release
Printed April 1995

Assembly Sequencing with Toleranced Parts

Jean-Claude Latombe, Randall H. Wilson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

SAND94-3124
Unlimited Release
Printed April 1995

Distribution
Category UC-705

Assembly Sequencing with Toleranced Parts

Jean-Claude Latombe
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

Randall H. Wilson
Intelligent Systems and Robotics Center
Sandia National Laboratories
Albuquerque, NM 87185

February 21, 1995

Abstract

The goal of assembly sequencing is to plan a feasible series of operations to construct a product from its individual parts. Previous research has thoroughly investigated assembly sequencing under the assumption that parts have nominal geometry. This paper considers the case where parts have toleranced geometry. Its main contribution is an efficient procedure that decides if a product admits an assembly sequence with infinite translations that is feasible for all possible instances of the components within the specified tolerances. If the product admits one such sequence, the procedure can also generate it. For the cases where there exists no such assembly sequence, another procedure is proposed which generates assembly sequences that are feasible only for some values of the toleranced dimensions. If this procedure produces no such sequence, then no instance of the product is assemblable. Finally, this paper analyzes the relation between assembly and disassembly sequences in the presence of toleranced parts. This work assumes a simple, but non-trivial tolerance language that falls short of capturing all imperfections of a manufacturing process. Hence, it is only one step toward assembly sequencing with toleranced parts.

Keywords: assembly planning, assembly sequencing, solid modeling, tolerancing, non-directional blocking graph.

Acknowledgments: This work was supported by NSF/ARPA grant IRI-9306544, a grant from the Stanford Integrated Manufacturing Association (SIMA), and Sandia National Laboratories under DOE contract DE-AC04-94AL85000. Part of this work was done while J.C. Latombe was a University Summer Faculty employee at Sandia National Laboratories. The work has benefited from early comments by Randy Brost, Matt Mason, and Vijay Srinivasan.

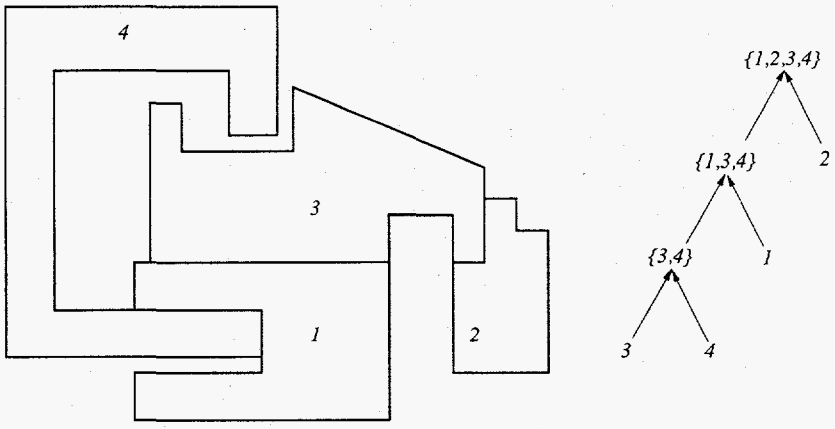


Figure 1: Example of an assembly sequence

1 Introduction

An assembly product is described by a geometric model of its parts and spatial relations defining their relative placement in the product. The goal of assembly sequencing is to plan a feasible partial ordering of operations to construct this product from its individual parts. Each operation generates a new subassembly by merging individual parts and/or subassemblies constructed by previous operations. The outcome of the final operation is the desired product. Each operation is specified by the subassemblies it merges and their relative motions. These motions should not lead any two parts to collide (contacts are allowed, however). Figure 1 shows an imaginary product made of four parts and the ordering (here, a total one) of an assembly sequence for this product. When parts are rigid, which is assumed here, assembly and disassembly sequences are the inverse of one another. Hence, in most of this paper, the two terms are used interchangeably.

In assembly sequencing it is usually assumed that parts and subassemblies are free-flying objects that execute perfect motions. Issues such as grasping, fixturing, stability, sensing and control are ignored. Despite these assumptions, an assembly sequencer can be very useful. For instance, it allows the detection of undesired geometric interferences among parts while a product is being designed. As efficient products require more parts to be densely packed together, the risk of such interferences, along with the need to automatically detect them at an early stage, increases.

There has been a significant amount of research on assembly sequencing during the past decade (e.g., [4, 5, 6, 11, 19, 20, 21, 22, 26, 27, 39, 40, 43, 44, 45]). Early assembly sequencers were mainly interactive sequence editors; geometric reasoning was supplied by a human who answered questions asked by the system [6, 11]. Automated geometric reasoning was then added to answer these questions automatically [4, 20, 22, 44]. This development first resulted in generate-and-test sequencers, with a module guessing candidate sequences and geometric reasoning modules checking their feasibility [22, 39]. More efficient techniques were later proposed to replace time-consuming generate-and-test [3, 40]. Research on "separability problems" in Computational Geometry is also related to assembly sequencing [9, 17, 29, 31, 34, 36].

Various forms of assembly sequencing have been shown to be computationally intractable [23, 24,

28, 44]. This result has led researchers to consider restricted, but still interesting subsets of assembly sequences, such as *monotone* sequences (where each operation generates a final subassembly) and *two-handed* sequences (every operation merges exactly two subassemblies). Often motions are also limited to translations. Though such restrictions vary slightly among the various assembly sequencers proposed so far, one is made in all of them: Parts are uniquely defined by their nominal geometry. In this paper we depart from this assumption by investigating assembly sequencing when parts have toleranced geometry. This work has been motivated by the fact that for many products tolerances have a crucial effect on assembly sequences and manufacturing costs.

Part tolerancing addresses the fact that manufacturing processes are inherently imprecise and produce parts of variable shapes [33, 37]. A large body of work has been aimed at developing tolerance languages (e.g., the American Y14.5 standard [1, 38]) to provide designers with symbolic means to specify acceptable variations. One important goal is to guarantee part interchangeability in an assembly product [37]: Given any set of parts manufactured according to the specified tolerances, they should assemble satisfactorily. Unfortunately, existing tolerance languages, which have evolved from shop practice, suffer from informality. They also allow tolerances that interact among themselves. As a result, the basic tolerance analysis problem, which is to accurately determine where the boundary of a part might be located in a given coordinate system [8, 13, 18, 30], is intractable. Proposed solutions to this problem are incomplete or approximate, and often deal with an idealized subset of the tolerance language. Checking part interchangeability is even harder. Previous work on this problem has focused on checking the geometric feasibility of the assembled state (i.e.: Does there exist an assembled state in which no two parts overlap?), using stack-up, optimization, constraint propagation, statistical analysis, and/or Monte Carlo techniques [2, 10, 12, 14, 30].

In this paper we go beyond the mere existence of an assembled state. We propose an efficient procedure that decides whether a product made of toleranced parts admits a guaranteed assembly sequence, i.e., a sequence that is feasible for all possible instances of the parts. This procedure can also generate all such sequences. The existence of an assembled state is not explicitly tested, but is implied by the existence of an assembly sequence (the converse is not true, however). For the cases where no guaranteed assembly sequence exists, we also propose another procedure that generates non-guaranteed assembly sequences, i.e., sequences that are only feasible for some instances of the parts. This procedure returns no such sequence if and only if the product is never assemblable. Our procedures assume a simple, but non-trivial tolerance language which does not model some important imperfections of manufacturing processes. The work reported in this paper is therefore only one limited step toward assembly sequencing with toleranced parts. Nevertheless, we believe it contributes to the much-needed understanding of what sort of tolerance language is suitable for assembly sequencing. Such understanding is of major interest to the community of researchers who are trying to improve the mathematical foundations of tolerancing [32, 35].

Section 2 describes the assembly-description language accepted by our algorithms. Section 3 gives appropriate technical background for the rest of the paper. It summarizes results previously reported in [40, 41, 42], including the concept of the non-directional blocking graph (NDBG) of a nominal product, an algorithm to compute NDBGs, and a procedure to generate assembly sequences from an NDBG. Section 4 develops the concept of a strong NDBG for products made of toleranced parts; this NDBG represents all possible blocking interferences between parts when their dimensions span the tolerance zones. It can be used in the same way as a "classical" NDBG to generate guaranteed assembly sequences. Section 5 describes in detail the algorithm enabling the construction

of the strong NDBG. The main difficulty faced here is that variations in the dimensions of the parts also cause the relative positions of the parts in the products to vary. Section 6 proposes the concept of a weak NDBG, which represents necessary blocking interferences between parts; this NDBG can be used to generate non-guaranteed assembly sequences. Section 7 analyzes some subtle aspects of the relation between assembly and disassembly sequences when parts have toleranced geometry. Finally, Section 8 presents several extensions of the algorithms described in Sections 4, 5, and 6.

2 Description of an Assembly Product

There are many ways to specify an assembly with toleranced parts. As we will see, some reduce assembly sequencing to the case where all parts have nominal geometry, but they usually yield a poor representation of reality. Others provide a comprehensive account of the imperfections of the manufacturing processes, but then assembly sequencing has daunting complexity. Here we propose an assembly-description language that lies between these two extremes; we also discuss shortcomings of this language. In Section 8 we will extend our planning algorithms and eliminate some of these shortcomings.

We consider a planar assembly product A made of N parts $P_1, \ldots, P_N$. It is described by a geometric model of the parts and spatial relations defining their relative placements.

We assume that each part P_i is a polygon manufactured such that all instances of P_i have perfectly straight edges, the same topology, i.e., the same sequence of edges, and the same angles between edges; but each edge may have different lengths in the various instances. The geometry of P_i is defined by its sequence of edges, with each edge specified by the orientation of its supporting line relative to some coordinate system and the interval of acceptable distances from the origin of this system to the supporting line.

We will refer to the coordinate system used to specify the geometry of P_i as the *coordinate system of* P_i. We denote its origin by p_i. The distance between p_i and the line supporting an edge of P_i is called a *variational parameter* and the interval of acceptable values for this distance a *tolerance zone*. The tolerance zones of the variational parameters of each part P_i should be small enough to guarantee that all instances of P_i have the same topology. A sufficient condition is that no vertex falls into the intersection of more than two stripes swept by edge-supporting lines when the variational parameters span the tolerance zones.

Figure 2 illustrates the description of a part with seven edges. It shows a particular instance of the part in bold contour, the variational parameters $d_1, \ldots, d_7$, and the extreme positions of the edge-supporting lines.

The orientation of an edge-supporting line is defined by its angle in $[0, \pi)$ with the x-axis of P_i's coordinate system. The distance from p_i to this line is a signed real; the sign is set as follows: If the outer normal to the corresponding edge points in the direction of p_i, the distance is negative; otherwise it is positive. For example, in Figure 2 all variational parameters, except d_5, are positive. This convention has two advantages:

- It allows p_i to lie without ambiguity in the stripe swept by an edge-supporting line when the corresponding variational parameter spans its tolerance zone.
- There is one instance of P_i that contains all other (called MMP, for Maximal Material Part). With our convention, it is obtained when all variational parameters are maximal.

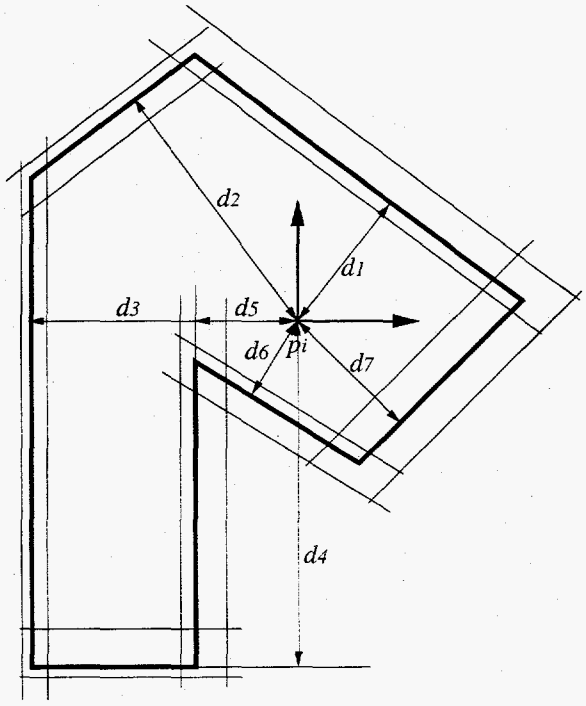


Figure 2: Part description

The relative placement of the N parts in A is defined by a set of spatial relations. Each relation R uniquely defines the relative placement of two particular parts. This means that for every possible instance of these two parts, a single relative position of their coordinate systems achieves R. We assume that R consists of two elementary relations: one stating that two edges, one from each part, are parallel, with their outer normals pointing in either the same or opposite directions and a signed distance between the lines supporting the two edges; the other stating that a vertex of one part is at some signed distance of the line supporting an edge of the other part. This definition of spatial relations subsumes normal contact relationships between parts: One specifies a contact between two edges by setting the distance between them to zero. We assume zero tolerances in the distance values of the spatial relations.

Figure 3 illustrates a spatial relation between two parts P_i and P_j. Edges e and f are parallel, with their normals pointing in the same direction, at some distance of each other (the distance, not given in the figure, is negative to indicate that e is ahead of f along the direction of the outer normals). The vertex v is at some distance of the edge g (again, the value of this distance has been omitted in the figure).

The set of relations must be complete and non-redundant. By complete, we mean that if one randomly picks a geometry for every component of A within the tolerance zones, the relations determine a unique geometry for A (such an assembly is said to be "static" [30]). By non-redundant, we mean that removing any one of the relations makes the set incomplete. In order for the set of relations to be complete and non-redundant, it is necessary and sufficient that the undirected graph whose nodes are the components of A and whose links are the spatial relations be connected

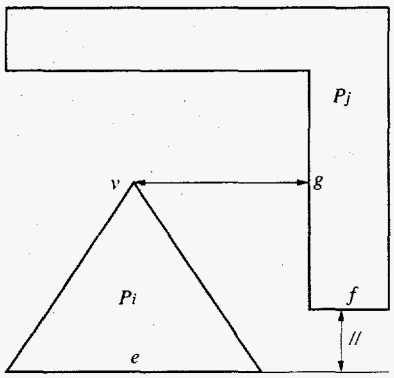


Figure 3: Spatial relation between two parts

and without cycles. We call this graph the *relation graph* of A.

The set of elementary relations used to specify the spatial relation between two parts could easily be enriched. For example, we could allow relations stating that two edges are perpendicular, that two edges are parallel (without specifying any distance), or that two vertices are at some distance from each other. It suffices that each elementary relation completely removes one or several degrees of freedom between the two parts and that the set of elementary relations forming a spatial relation be complete (i.e., remove all three degrees of freedom between the two related parts) and non-redundant. The first condition means in particular that a relation stating that two edges are perpendicular should also specify the angle (either $\frac{\pi}{2}$ or $-\frac{\pi}{2}$) between their outer normals. The second condition implies that there can be no more than three elementary relations in a spatial relation. Allowing all such elementary relations is straightforward, but would lengthen several sections of this paper.

We refer to the achievement of a spatial relation between two parts P_i and P_j as the relative *placement* of these two parts, and to the position of p_j in the coordinate system of P_i as their relative *position*. While the relative placement of any pair of parts in two instances of A is unique, the relative positions of these parts may differ due to variations in the lengths of the edges of the parts. The same variational parameter may influence both the shape of a part and the position of this part relative to other parts. On the other hand, the relative orientations of the coordinate systems are fixed.

Let q_i designate the number of edges of P_i ($i = 1$ to N). $Q = q_1 + \ldots + q_N$ is the total number of edges in the parts of A. The geometry of any particular instance of A is defined by a single value of the tuple $(d_1, \ldots, d_Q)$ of variational parameters. The space spanned by this tuple is a Q-dimensional hyper-parallelepiped $\mathcal{V}$, the Cartesian cross-product of the tolerance zones. We call $\mathcal{V}$ the *variational space* of A. In the following, the same notation P_i (resp. A) will be used to designate both the variational class of parts (resp. assemblies) determined by $\mathcal{V}$ and any instance in that class. Whenever some ambiguity may arise, we will explicitly mention to which we refer.

At this point, there is no requirement that an assembly product specified as above be feasible. Perhaps no assembly sequences allow the product to be put together. Or the product may even be impossible, because some parts overlap in the assembled state. It will be the task of an assembly

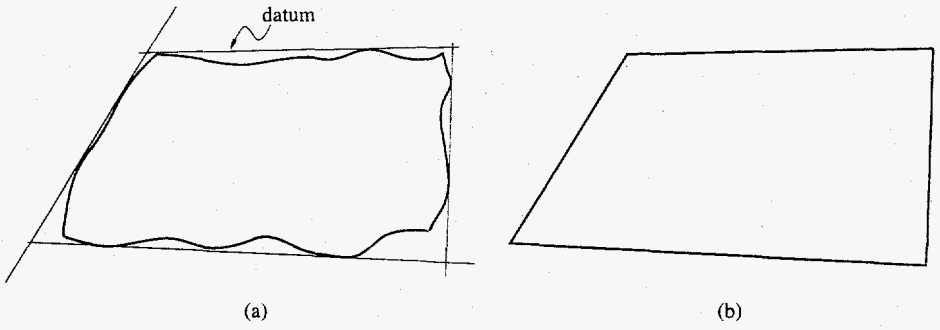


Figure 4: Part with imperfect edges

sequencer to determine that.

Discussion: We now briefly discuss some of the shortcomings of our assembly-description language. We focus on tolerancing, since this is the main theme of this paper.

First, let us remark that the assembly sequencing problem depends intimately on how we describe an assembly A. We noticed before that each part admits an MMP. Suppose that, instead of using spatial relations, we had defined the relative placement of every two parts in A by the relative position of their coordinate systems. Then assembly sequencing would trivially reduce to assembly planning with MMPs. This does not seem to make much sense, however. Indeed, contacts and/or clearances between parts are crucial in assemblies. When the relative positions of the coordinate systems are directly provided in the description of the product, contacts can only be achieved at the ends of tolerance zones (otherwise parts could overlap); similarly clearance constraints are only met for some values of the variational parameters.

The most blatant assumption in our language is that edges are perfectly straight. Such edges are impossible to manufacture. However, the assumption is not really needed. Consider a part with imperfectly shaped edges as illustrated in bold contour in Figure 4.a. We can bring a straight line, called a *datum* [30], into two-point contact with each edge and replace the imperfect edges by the perfect ones defined by the datums (Figure 4.b). Our algorithms apply to the parts defined by these virtual edges.

In the Y14.5 standard, specifying a distance between two edges e and f leads to associating a datum with one edge, say e. The tolerance zone defines the region (a stripe in 2D) within which the other edge, f, should lie. In our case, the tolerance zone defines the locus of the virtual edge. The constraint expressed in Y14.5 entails ours, but the converse is not true. Although the relative weakness of our constraint would matter if we wanted to ensure that parts be interchangeable in function, it does not affect their interchangeability in assembly, which is our only concern in this paper. Said otherwise, the constraint expressed in Y14.5 can be translated into our language without affecting part interchangeability in assembly.

Another important limitation is that edges are cut with perfect angles between them (which now only means that the virtual edges make perfect angles). Perfect angles are not possible in practice, even between datums. Tolerancing angles would allow the relative orientation of parts to vary, thus seriously complicating assembly sequencing. In Section 8 we will show that our algorithms can

handle toleranced distances in spatial relations between parts, but by no means does this ability completely eliminate the above limitation. See also the conclusion for a short discussion of how this limitation could be removed.

The coordinate system of a part P_i can be located anywhere. In practice, dimensions are specified relative to datums associated with edges. Then we could choose P_i's coordinate system such that one of its axes is aligned with an edge and its origin coincides with one extremity of that edge. But using a single "central" coordinate system may be a limitation, since it often happens that datums in a single part are "chained" by distance specifications. We will see in Section 8 that we can adapt our algorithms to handle multiple coordinate systems per part.

The fact that we only consider planar assemblies is one important limitation not directly related to tolerancing. Again, in Section 8 we will see that some of our algorithms extend to 3D polyhedral assemblies.

3 Background

Let the assembly A be described as above, but with zero-length tolerance zones. Hence, all parts and subassemblies are nominal. In this section we briefly review previous techniques that generate monotone two-handed assembly sequences for A. In particular we present the concept of the NDBG of A for infinite translations. This concept is central to the work presented in the following sections.

An assembly sequence is a partial ordering on operations of the form: "Merge S_1 and S_2 into S by translating S_1 along t." Its *inverse*, a disassembly sequence, is obtained by reversing the ordering and replacing each operation, such as the above, by: "Break S into S_1 and S_2 by translating S_1 along $t+\pi$." In fact, under the assumption that parts are rigid (which is the case here), the inverse map is a bijection between the assembly and the disassembly sequences. Any assembly sequence can thus be produced by first generating a disassembly sequence and then inverting it. A disassembly sequence is intuitively easier to produce since it starts from the highly constrained assembled state, in which spatial relations may directly suggest candidate disassembly motions.

Let `partition` be a procedure that takes the description of an assembly S as input and generates two subassemblies S_1 and S_2 ($S_1 \cup S_2 = S$), along with a direction t such that S_1 can be removed from S and translated arbitrarily far along t without colliding with S_2. Whenever such subassemblies and direction don't exist, the procedure returns failure. Disassembly sequences are generated by applying `partition` to A and, recursively, to the generated subassemblies that are not individual parts. Let `disassemble` designate this recursive procedure.

In several early assembly sequence planners, `partition` was based on a generate-and-test technique. Given S, this technique enumerates all candidate partitions $\{S_1, S_2\}$ of S, until it finds a direction t that separates S_2 from S_1 without disturbing S_1. Finding t often consists of inferring it from spatial relations between parts (mainly from contacts), computing the region that will be swept by S_2, and checking that this region does not intersect S_1. But this technique can be very slow. Indeed, the number of candidate partitions is always exponential in the number of parts in S, while the number of feasible partitions is usually much smaller. The non-directional blocking graph, NDBG for short, was introduced to avoid the combinatorial trap of generate-and-test [40, 41, 42]. The idea is to precompute a structure, the NDBG, that represents all blocking interferences among the parts in A, and to query this structure to generate one, several, or all disassembly sequences. As we will see, the NDBG yields a more efficient implementation of `partition` than generate-and-test.

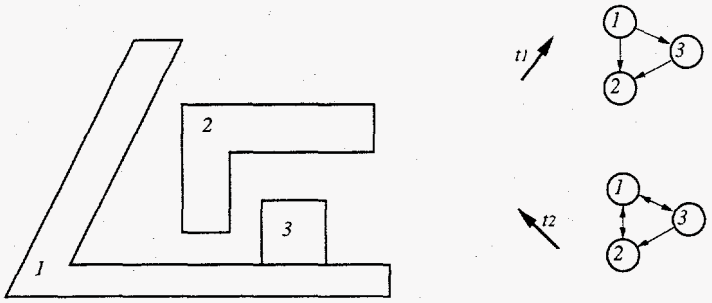


Figure 5: Examples of directional blocking graphs

Consider two parts P_i and P_j in their relative position in A. Ignore all other parts. The direction t is a *feasible infinite translation for P_i relative to P_j* if one can translate P_i to infinity along t without colliding with P_j. Now consider the full assembly A and a direction t. The *directional blocking graph*, or DBG, *of A for t* is the directed graph whose nodes are the parts of A and whose arcs are all pairs of parts (P_i, P_j) such that t is not a feasible infinite translation of P_i relative to P_j. Figure 5 shows DBGs of a simple assembly for two directions t_1 and t_2.

In two dimensions the set of all directions is represented by the unit circle S^1. The set of feasible infinite translations of P_i relative to P_j is a cone C_{ij} that determines an arc in S^1. Hence, all the cones C_{ij}, $i, j \in [1, N]$, $i \neq j$, partition S^1 into $O(N^2)$ arcs such that the DBG of A remains constant over each arc. The sequence of arcs and their DBGs form the *non-directional blocking graph* of A.

We assume for simplification that there are no tight insertions in A. Each cone C_{ij} can then be constructed by erecting the two extreme rays originating at p_i (the origin of the coordinate system of P_i) and tangent to $P_j \ominus P_i$ (the Minkowski difference of P_j and P_i). See Figure 6, where the polygon in bold contour is $P_j \ominus P_i$. If P_i and P_j touch each other, then p_i lies on the boundary of $P_j \ominus P_i$. If they overlap (in which case, A is not a possible assembly), p_i lies in the interior of $P_j \ominus P_i$. (If we allowed P_i to be tightly inserted into P_j, we would have to be more careful, since the set of positions where P_i touches P_j would then be a superset of the boundary of $P_j \ominus P_i$.)

If P_i and P_j are non-convex polygons with q_i and q_j edges, we can decompose them into convex components, which we denote by P_i^k $(k = 1, 2, \ldots)$ and P_j^l $(l = 1, 2, \ldots)$, respectively. We have: $P_j \ominus P_i = \bigcup_{k,l} P_j^l \ominus P_i^k$. A trapezoidalization of P_i and P_j yields $O(q_i)$ and $O(q_j)$ components, each of constant complexity, in times $O(q_i \log q_i)$ and $O(q_j \log q_j)$ [29]. Each region $P_j^l \ominus P_i^k$ is a convex polygon of constant complexity that takes constant time to compute. Let C_{ij}^{kl} be the cone formed by the two rays stemming from p_i and tangent to $P_j^l \ominus P_i^k$. We have: $C_{ij} = \bigcap_{k,l} C_{ij}^{kl}$. All cones C_{ij}^{kl} are computed in time $O(q_i q_j)$. They determine $O(q_i q_j)$ arcs in S^1. The computation of the arc where C_{ij} intersects S^1 is thus done in total time $O(q_i q_j + q_i \log q_i + q_j \log q_j)$.

Let q be the maximal number of edges in a single part of A. The $O(N^2)$ cones C_{ij} are computed in time $O((Nq)^2)$. They determine $O(N^2)$ points in S^1 that are sorted in time $O(N^2 \log N)$. The DBG in any arc can be obtained in time $O(N^2)$. However, between any two adjacent arcs, the DBG undergoes a small number of changes that can be computed in constant time. Thus, once a DBG has been computed, all other DBGs can be computed in total time $O(N^2)$ by scanning the sequence of arcs in S^1 and, for each arc, modifying the DBG constructed for the previous arc [40]. The complete

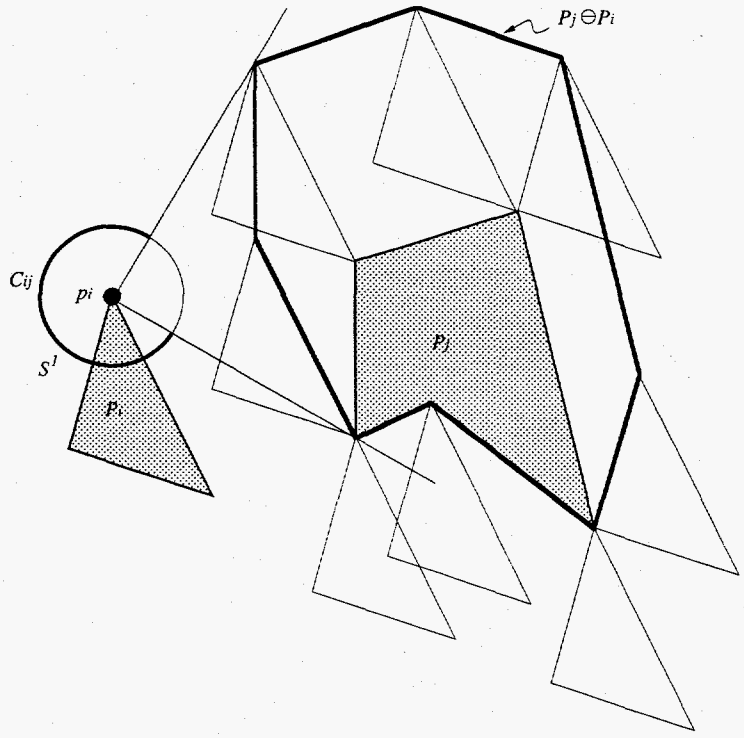


Figure 6: Construction of a cone of feasible infinite translations

NDBG takes time $O(N^2(\log N + q^2))$ to compute.

Consider now the DBG G of A for some direction t. A can be partitioned into two subassemblies S_1 and S_2 by translating S_1 along t if and only if there exists no arc in G connecting a part of S_1 to a part of S_2. Hence, A can be partitioned by a translation along t if and only if G is not strongly connected. The strong components of G yield all possible partitionings of A. Notice also that the NDBG of any subassembly S of A is obtained by restricting every DBG to the parts of S and merging adjacent arcs of S^1 having the same DBGs. Hence, given the NDBG of A, `partition` can be implemented as follows:

```
procedure partition(S);
    for every arc c in the NDBG of S do:
        if the DBG associated with c is not strongly connected
            then return c and a feasible partition of S;
    return failure;
```

Computing the strong components of a DBG takes time $O(N^2)$. (A better bound, $O(N^{1.38})$, can be obtained by taking advantage of the fact that any two successive DBGs differ by a small amount [25].) Hence, `partition` runs in time $O(N^4)$ and `disassemble` generates an assembly sequence in time $O(N^5)$.

The procedures `partition` and `disassemble` can easily be modified to generate all feasible assembly sequences [40]. In the worst case, however, the number of these sequences is exponential in N.

Remark: The above presentation has focused on planar assemblies and infinite translations. However, NDBGs have been successfully extended both to deal with 3D assemblies and to generate more complicated motions (e.g., rotational motions [42, 15] and multiple extended translations [43]). This requires adapting the definition of a feasible motion of P_i relative to P_j. Another planning approach, based on "monotone paths," has been proposed to avoid the combinatorial trap of generate-and-test for assemblies of polygons in the plane [3]. But, so far, this approach has only been proposed to generate translational assembly sequences for planar polygonal assemblies. Attempts to efficiently generalize it to 3D assemblies and/or rotational motions have failed.

4 Strong NDBG

From now on let the assembly A be made of toleranced parts, as described in Section 2. While the question "Does there exist an assembly sequence to construct A?" had only two possible answers, "yes" and "no", when parts in A had nominal geometry, it now has three possible answers, "yes", "no" and "maybe". Moreover, if the answer is "yes", two cases are possible: There may, or may not exist an assembly sequence that is feasible for all values of the variational parameters. We call such a sequence a *guaranteed* assembly sequence, and a sequence that is only feasible in a non-empty subset of the variational space $\mathcal{V}$ a *non-guaranteed* sequence.

In this section we focus on guaranteed assembly sequences. We propose an extension of the NDBG concept to represent all possible blocking interferences among parts of A for infinite translations, when the variational parameters span $\mathcal{V}$. We call this extension the strong NDBG. The procedures `partition` and `disassemble` apply to this NDBG without modification. The procedure `disassemble` now produces guaranteed assembly sequences, whenever such sequences exist; it returns failure otherwise.

Consider any two parts in A. Due to possible variations in their geometry and relative position, the cone of feasible infinite translations of one part relative to the other is not constant. Therefore, at each point in the variational space $\mathcal{V}$, one may compute a distinct NDBG. To be sure that A can be partitioned into two subassemblies by translating one to infinity along some direction, this partitioning must be feasible in all NDBGs over $\mathcal{V}$.

Since computing all NDBGs over $\mathcal{V}$ is generally impractical, we propose to "project" these NDBGs onto S^1. This projection associates a set of DBGs with every direction t of S^1. This set consists of all distinct DBGs for direction t, when the variational parameters span $\mathcal{V}$. Usually, if two directions t_1 and t_2 are very close to each other, the same set of DBGs is associated with both directions. Indeed, translating each component of A along t_1 is likely to yield collisions with the same parts as a translation along t_2. This is not true, however, for some directions where a translated part collides with a new part or stops colliding with a part. These critical directions partition S^1 into arcs such that a single set of DBGs is associated with every arc. We call this structure the *multi-valued* NDBG of A.

To make this concept clearer, consider two parts P_i and P_j in A. For every value of the variational parameters, we can compute a cone C_{ij} of feasible infinite translations of P_i relative to P_j. Let us intersect all cones C_{ij} when the variational parameters span their tolerance zones. The result is the possibly empty cone SC_{ij} of infinite translations that are feasible for all values of the variational parameters. We call it the *small cone* of feasible translations of P_i relative to P_j. Similarly, the union LC_{ij} of all cones C_{ij} is the cone of all infinite translations that are feasible for

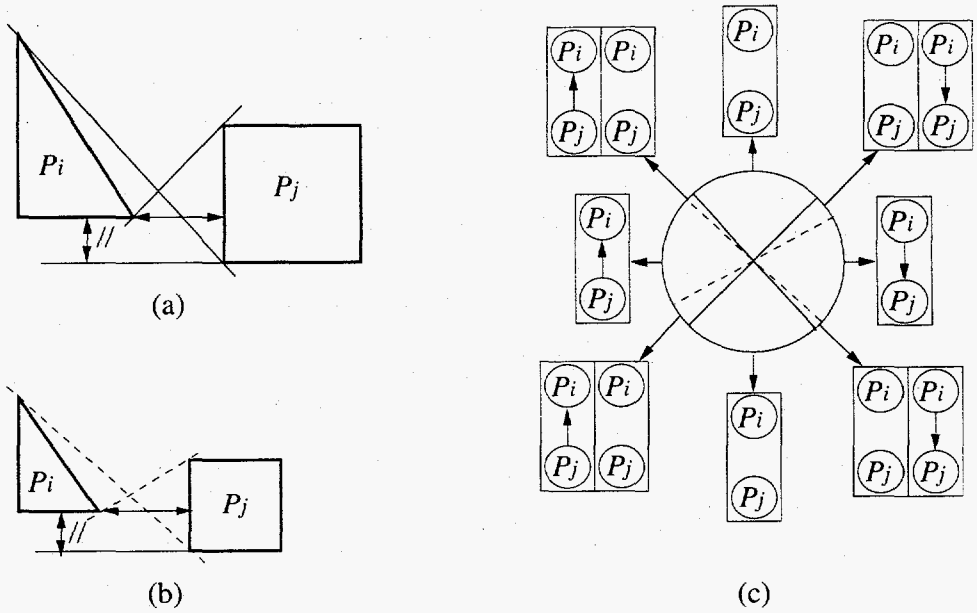


Figure 7: Multi-valued NDBG for two parts

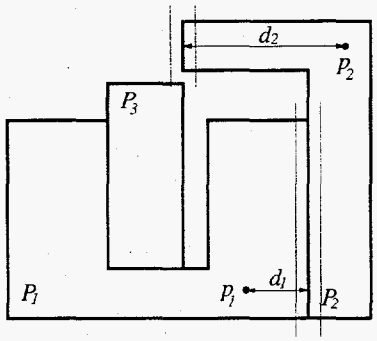


Figure 8: Influence of variational parameters on feasible translations

at least one value of the variational parameters (LC_{ij} may have a 2π angle). We call it the *large cone*. Inverting SC_{ij} and LC_{ij} yields SC_{ji} and LC_{ji}, respectively. The four cones SC_{ij}, LC_{ij}, SC_{ji}, and LC_{ji} partition S^1 into at most 8 arcs, such that a single set of DBGs reduced to P_i and P_j is associated with each cell. The set of arcs and the associated sets of DBGs form the multi-valued NDBG of the subassembly made of P_i and P_j. See Figure 7, where the small (resp. large) cones are bounded by plain (resp. dashed) lines.

In the example of Fig. 7, the small and large cones are respectively obtained for the maximal material parts (Fig. 7.a) and least material parts (Fig. 7.b). But this is not always the case. For example, Fig. 8 shows a 3-part assembly with two variational parameters d_1 and d_2. When d_1 is minimal and d_2 maximal, the peg P_3 can't be translated vertically. When d_1 is maximal and d_2

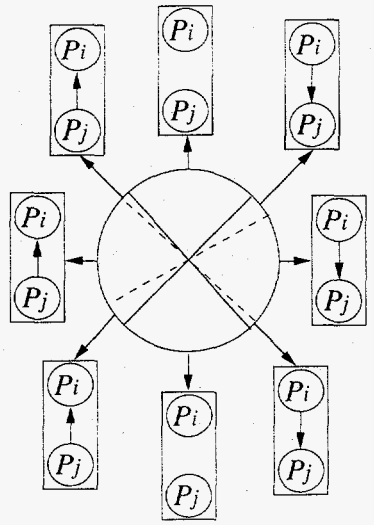


Figure 9: Strong NDBG for two parts

minimal, this translation is feasible.

Scanning all pairs of parts in A leads to partitioning S^1 into $O(N^2)$ arcs. But in the worst case the set of DBGs associated with one arc has size $O(2^{N^2})$. For this reason, we replace this set by the union of the DBGs it contains. The result is called the strong DBG. The NDBG whose cells are labeled by strong DBGs is called the *strong* NDBG. As mentioned above, it describes all possible blocking interferences among the parts of A when the variational parameters span $\mathcal{V}$. Figure 9 shows the strong NDBG derived from the multi-valued NDBG of Figure 7. Notice that several adjacent DBGs are identical; so, the corresponding arcs should be merged. As a result, only the small cones are needed to construct the strong NDBG.

At the core of the computation of the strong NDBG is the algorithm that generates the small cone SC_{ij} for any two parts P_i and P_j. In the next section we will propose an algorithm that computes all cones SC_{ij} in time $O(N^2 n(q^2 + \log n))$, where $n < N$ is the maximal length of a path in the relation graph of A and $q < Q$ is the maximal number of edges in a part of A. In general, $n \ll N$ and $q \ll Q$. As in the nominal-geometry case, the DBGs associated with two adjacent arcs in the strong NDBG differ by a small amount. Hence, the DBG for one arc can still be computed in constant time by slightly modifying the DBG computed for the previous arc. The total time to construct the strong NDBG is $O(N^2 \log N + N^2 n(q^2 + \log n))$. In most practical cases, this time is $O(N^2 n q^2)$.

When applied to the strong NDBG, the procedure `disassemble` generates guaranteed sequences, whenever such sequences exist. If it returns failure, then the product may still be assemblable in all cases, but with several sequences depending on the values of the variational parameters, or it may be assemblable only for some values of these parameters, or it may never be assemblable.

Discussion: An alternative to the computation and exploitation of the strong NDBG would be to compute the nominal NDBG of the assembly A and then perform sensitivity analysis on a nominal assembly plan. However, our approach gives a much stronger result: While sensitivity analysis

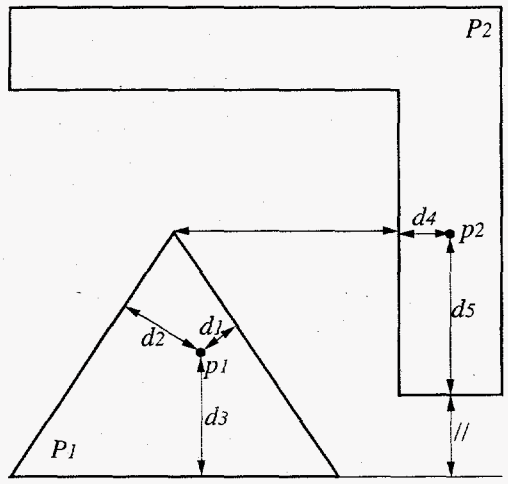


Figure 10: Parameters influencing the relative position of two parts

would usually not be able to formally prove that a particular sequence is feasible for all instances of the parts, our approach checks the existence of a guaranteed sequence and, if one exists, produces it. Moreover, sensitivity analysis could be very time consuming. Indeed, the number of variational parameters is often large and the number of feasible nominal sequences can be exponential in the number of parts. Instead, the time complexity of our method is both well-bounded and reasonable.

In this paper the only assembly motions we consider are infinite translations. As indicated earlier, "classical" NDBGs have been applied to other types of motions. We hope that the work reported here will also be eventually extended to produce assembly sequences with various motions. Notice, however, that it is often desirable that products be manufacturable with translations only. The algorithms described here are directly relevant to that case.

5 Computation of Small Cones

In this section we describe an algorithm to compute the small cone SC_{ij} of feasible infinite translations of P_i relative to P_j.

Recall from Section 3 that, if the geometry and relative position of P_i and P_j are uniquely defined, then the cone of feasible infinite translations of P_i relative to P_j is identical to the cone of feasible translations of the point p_i (the origin of the coordinate system of P_i) relative to $P_j \ominus P_i$. Here, both the geometry and the relative position of P_i and P_j are functions of the variational parameters $d_1, \ldots, d_Q$. Thus, the small cone SC_{ij} is the cone of feasible translations of p_i relative to the region $\bigcup_{\mathcal{V}}(P_j \ominus P_i)$ swept by $P_j \ominus P_i$ when $(d_1, \ldots, d_Q)$ spans the variational space $\mathcal{V}$.

To compute $\bigcup_{\mathcal{V}}(P_j \ominus P_i)$, we first remark that it only depends on a subset of variational parameters. Indeed, the geometries of P_i and P_j depend on q_i and q_j parameters, respectively. On the other hand, recall that the spatial relation between two parts consists of two elementary relations: one states that two edges, one in each part, are parallel at a given distance; the other states that a vertex of one part is at some distance from an edge of the other part. Hence, the relative position of two parts linked by a spatial relation depends on at most 5 variational parameters: 2

are contributed by the distance between two parallel edges, and are the variational parameters of these two edges; the other 3 are contributed by the distance between an edge and a vertex, and are the variational parameters of this edge and the two edges intersecting at this vertex. For example, in Figure 10, the relative position of P_1 and P_2 depends on $d_1, \ldots, d_5$. The relative position of P_i and P_j thus depends on $5r_{ij}$ variational parameters, at most, where r_{ij} is the number of spatial relations defining the relative placement of P_i and P_j (i.e., r_{ij} is the length of the path between P_i and P_j in the relation graph of A). Moreover, among the r_{ij} relations, one defines the relative placement of P_i and another part. Out of the 5 (or less) variational parameters that influence the relative position of these two parts, 2 or 3 also affect the geometry of P_i. The same remark holds for P_j. Therefore, a maximum of $q_i + q_j + 5r_{ij} - 4$ variational parameters influence the cone of feasible translations of P_i relative to P_j.

We divide these remaining parameters into three disjoint subsets, J, K, and L:
- J contains the variational parameters of P_i and P_j that do not influence the relative position of the two parts. We call them the *shape parameters.*
- K contains all parameters that are not variational parameters of P_i or P_j; hence, they only affect the relative position of P_i and P_j. We call them the *position parameters.*
- L contains the variational parameters of P_i and P_j that do influence the relative position of the two parts; it contains at most 6 parameters. We call them the *shape-position parameters.*
(We strongly advise the reader to memorize these definitions, since the sets J, K, and L will be used extensively in the rest of this section, as well as in the next section.)

We now consider the above three sets in sequence:

Shape parameters (J): Assume that we fix the parameters in $K \cup L$ to some arbitrary value in their tolerance zones, while we let the parameters in J span their domains. Let $\bigcup_J(P_j \ominus P_i)$ denote the region swept by $P_j \ominus P_i$.

The value of the parameters in J affects the shapes of P_i and P_j, but not their relative position. Let $\mathcal{P}_i$ and $\mathcal{P}_j$ stand for the regions swept by P_i and P_j, respectively, in the coordinate systems of P_i and P_j. $\mathcal{P}_i$ (resp. $\mathcal{P}_j$) is exactly equal to P_i (resp. P_j) when the parameters in J have their maximal values, the parameters in $K \cup L$ being set as above. Thus, we have:

$$\bigcup_J(P_j \ominus P_i) = \mathcal{P}_j \ominus \mathcal{P}_i.$$

We can also interpret $\bigcup_J(P_j \ominus P_i)$ as follows: When the parameters in J vary, the parameters in $K \cup L$ being fixed, the small cone of feasible translations of P_i relative to P_j is the cone of feasible translations of $\mathcal{P}_i$ relative to $\mathcal{P}_j$.

Position parameters (K): Now let the parameters in $J \cup K$ span their domains, while the parameters in L keep the arbitrary value given above. $\bigcup_{J,K}(P_j \ominus P_i)$ denotes the region swept by $\bigcup_J(P_j \ominus P_i)$ as the parameters in K vary.

By definition, the parameters in K do not affect the shapes of P_i and P_j. However, when the parameters in K vary, the origin p_j of the coordinate system of P_j spans a region W in the coordinate system of P_i. The geometry of W is independent of the values of the parameters in $J \cup L$. If $r_{ij} = 1$, K is empty and W reduces to a single point. If $r_{ij} = 2$, W is a convex polygon of constant complexity, which also takes constant time to compute (see Figure 11, where W is shown gray). If $r_{ij} > 2$, W is a convex polygon of complexity $O(r_{ij})$, which is computed in time

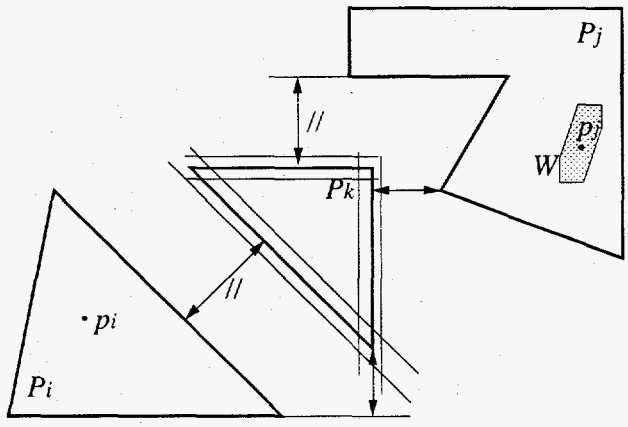


Figure 11: Locus of p_j with respect to p_i when $r_{ij} = 2$

$O(r_{ij} \log r_{ij})$ by a divide-and-conquer technique as the convolution of $r_{ij} - 1$ convex polygons of constant complexity [16]. Thus, while $\mathcal{P}_i$ is fixed, $\mathcal{P}_j$ sweeps the region $W \oplus \mathcal{P}_j$. We have:

$$\bigcup_{J,K}(P_j \ominus P_i) = W \oplus \mathcal{P}_j \ominus \mathcal{P}_i.$$

We can also interpret $\bigcup_{J,K}(P_j \ominus P_i)$ as follows: When the parameters in $J \cup K$ vary, the variational parameters in L being fixed, the small cone of feasible translations of P_i relative to P_j is the cone of feasible translations of $\mathcal{P}_i$ relative to $W \oplus \mathcal{P}_j$.

Shape-position parameters (L): We now obtain $\bigcup_{\mathcal{V}}(P_j \ominus P_i)$ by letting the parameters in L span their domain and constructing the region swept by $\bigcup_{J,K}(P_j \ominus P_i)$. The difficulty here is that the parameters in L affect both the relative position and the shapes of $\mathcal{P}_i$ and $\mathcal{P}_j$.

For any value of the parameters in L, $\bigcup_{J,K}(P_j \ominus P_i)$ is exactly the region bounded by the outer contour of the union of the polygons $\phi^{kl} = W \oplus e_j^l \ominus e_i^k$, where e_i^k $(k = 1, 2, \ldots)$ and e_j^l $(l = 1, 2, \ldots)$ denote the edges of $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively.

As the parameters in L vary, $\mathcal{P}_i$ and $\mathcal{P}_j$ keep the "same" edges. Therefore the region swept by $\bigcup_{J,K}(P_j \ominus P_i)$ is bounded by the outer contour of the union of the regions swept by the sets ϕ^{kl}. Since the geometry of W and the orientations of the edges of $\mathcal{P}_i$ and $\mathcal{P}_j$ are independent of the parameters in L, each ϕ^{kl} also keeps the "same" edges with the same orientations. Moreover, the coordinates of every vertex v in every ϕ^{kl} are linear functions of the parameters in L, whose domain is a hyper-parallelepiped. Hence, v spans a convex polygon whose vertices are attained when the parameters in L take extreme values (i.e., are at vertices of their domain). Consider two consecutive vertices v_1 and v_2 of any ϕ^{kl}. The region swept by the edge connecting v_1 and v_2 is exactly the convex hull of the two polygons spanned by v_1 and v_2. It follows that the region swept by any ϕ^{kl} is the convex hull of the polygons spanned by its vertices.

To obtain SC_{ij}, however, we do not need to explicitly compute $\bigcup_{\mathcal{V}}(P_j \ominus P_i)$. Indeed, let SC_{ij}^{kl} be the cone of feasible translations of p_i relative to the region swept by ϕ^{kl}. We have: $SC_{ij} = \bigcap_{k,l} SC_{ij}^{kl}$.

Each ϕ^{kl} has $O(r_{ij})$ vertices and is computed in time $O(r_{ij})$. The number of extreme values of the parameters in L is exponential in the size of L, which is at most 6; hence, this number is $O(1)$.

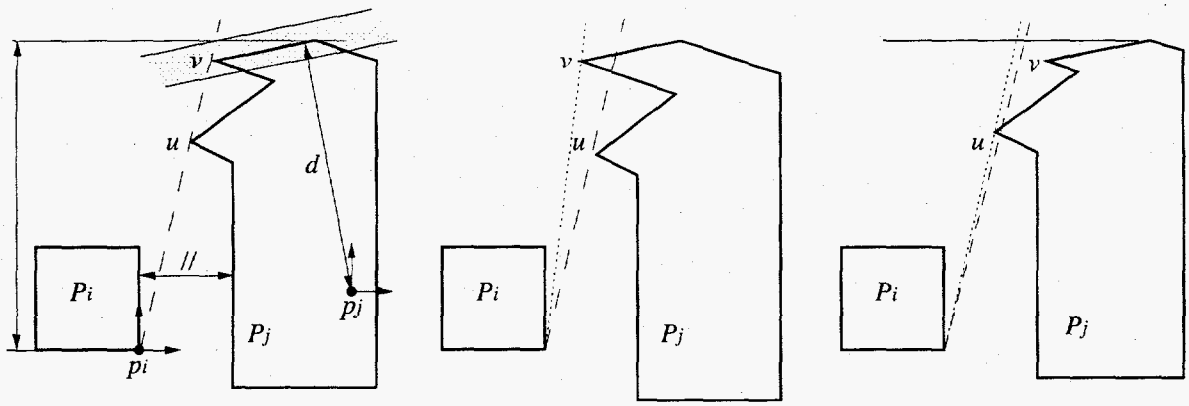


Figure 12: Large cone and extreme rays

By exploiting the fact that the edges of ϕ^{kl} keep constant orientations, we compute the convex hull of the polygons spanned by the vertices of ϕ^{kl} in time $O(r_{ij})$. Thus, SC_{ij} is computed in total time $O(r_{ij}(q_i q_j + \log r_{ij}))$. The logarithmic term comes from the computation of W.

Let q denote the maximal number of vertices in a part of A and n the length of the longest path in the relation graph of A. The $O(N^2)$ small cones necessary to the construction of the strong NDBG of A are computed in time $O(N^2 n(q^2 + \log n))$.

6 Weak NDBG

In Section 4 we defined the strong NDBG by replacing the set of DBGs associated with each arc of the multi-valued NDBG by the union of these DBGs. We now replace this set by the intersection of the DBGs. We get another NDBG, which we call the *weak* NDBG. It describes blocking interferences that necessarily occur between the parts of A, whatever the value of the variational parameters.

Assume that the strong NDBG yields no guaranteed assembly sequence. Then the procedures `partition` and `disassemble` applied to the weak NDBG generate non-guaranteed assembly sequences whenever there exists an instance of A that can be assembled. A failure of `disassemble` now means that no instance of A can be assembled.

The weak NDBG is interesting in several ways. For example:
- There exists no guaranteed sequence: One may wish to generate non-guaranteed sequences to estimate their probability of success using, say, Monte Carlo techniques.
- Some parts in an assembly are sealed together: For safety purposes (e.g., the product is a toy), one may wish to check that the resulting assembly cannot be disassembled.

To construct the weak NDBG, we must first compute the large cones LC_{ij} of feasible translations of P_i relative to P_j, for all pairs of parts in A. In general, if P_i and P_j are not convex, LC_{ij} is not equal to the cone of feasible translations of p_i relative to the intersection of all the regions $P_j \ominus P_i$ when the parameters in $J \cup K \cup L$ span their domain. This leads us to directly form the union of the cones C_{ij} of feasible translations of p_i relative to $P_j \ominus P_i$ when the parameters in $J \cup K \cup L$

vary. But there is another, more basic difficulty: *Neither of the two rays bounding LC_{ij} may be passing through a vertex of $P_j \ominus P_i$, at a position attained by this vertex when the parameters in L have extreme values.* This point is illustrated in Figure 12, where we assume for simplification that $d \in L$ is the only variational parameter (i.e., the tolerance zone of every other parameter has length zero). We consider two rays erected from p_i, one passing through vertex u, the other through vertex v. In Figure 12.a, the value of d is chosen in its tolerance zone (depicted by the gray area) so that both rays are aligned. Figures 12.b and 12.c show the rays (dotted lines) with the most counterclockwise orientations when d takes its extreme values (the ray with long dashes is identical to the one in Figure 12.a; it is reproduced to facilitate comparison with the other rays). As d varies, the two rays rotate in opposite directions. They form one side of LC_{ij} for the value of d where they coincide; this value is neither maximal, nor minimal. More generally, let the parameters in L vary linearly. The vertices of $P_j \ominus P_i$ then move along straight-line segments (some may remain fixed, however), but these segments may have different orientations and different lengths. Consequently, the rays erected from p_i and passing through the vertices of $P_j \ominus P_i$ rotate in different directions at different rates. Each side of LC_{ij} may be obtained when two rays coincide.

In the rest of this section we present an algorithm to compute LC_{ij}. Since dealing with the parameters in J and K is relatively easy, we first consider the parameters in L.

Shape-position parameters (L): We assume here that the parameters in J and K are fixed to some arbitrary value. Let $\bigcup_L C_{ij}$ be the union of all cones C_{ij} when the parameters in L span their domain (which we will designate by $\mathcal{V}_L$). Without loss of generality, we assume that the parameters in L are $d_1, \ldots, d_6$ (though there may be less than 6).

We denote the edges of P_i and P_j by f_i^k ($k = 1, 2, \ldots$) and f_j^l ($l = 1, 2, \ldots$), respectively. Let C_{ij}^{kl} be the cone of feasible translations of p_i relative to $\psi^{kl} = f_j^l \ominus f_i^k$. For any value of the variational parameters, we have: $C_{ij} = \bigcap_{k,l} C_{ij}^{kl}$. We call a ray erected from p_i and passing through a vertex v of a region ψ^{kl} a *vertex ray* and we denote it by $\rho(v)$. We refer to v as the *defining vertex* of $\rho(v)$. When the parameters in L span $\mathcal{V}_L$, every ψ^{kl} keeps the "same" edges with the same orientations, and the coordinates of its vertices are linear functions of $d_1, \ldots, d_6$.

Our goal is to select a finite set of points in $\mathcal{V}_L$ such that each of the two sides of $\bigcup_L C_{ij}$ is a side of the cone C_{ij} computed at one of these points. We generate this set as the union of two sets, H_1 and H_2. H_1 is the set of all points where a vertex ray achieves an extreme orientation. We initially define H_2 as the set of all points where two or more coinciding vertex rays achieve an extreme orientation (we will trim this conservative definition later). Note that H_2 is not a subset of H_1: An extreme orientation for one ray, while it coincides with other rays, is usually not an extreme orientation for that same ray, when no coincidence is required. No point in $\mathcal{V}_L \backslash (H_1 \cup H_2)$ can contribute a side of $\bigcup_L C_{ij}$ that is not already contributed by the points in $H_1 \cup H_2$.

Since the coordinates of the vertices of the regions ψ^{kl} are linear functions of $d_1, \ldots, d_6$, the extreme orientations of every vertex ray are obtained at vertices of $\mathcal{V}_L$. These vertices are all the points of H_1. They are constant in number.

The construction of H_2 is more involved. Consider any two vertices v_1 and v_2. The vertex rays $\rho(v_1)$ and $\rho(v_2)$ are aligned (i.e., either coincide or points in exactly opposite directions) when the coordinates (x_1, y_1) of v_1 and (x_2, y_2) of v_2 satisfy the equation:

$$x_1 y_2 - x_2 y_1 = 0. \tag{1}$$

Let us pose:

$$x_i = \sum_{j=1}^{j=6} \alpha_{ij} d_j + \alpha_{i0} \quad \text{and} \quad y_i = \sum_{j=1}^{j=6} \beta_{ij} d_j + \beta_{i0}, \quad \text{for} \quad i = 1, 2,$$

where all coefficients α_{ij} and β_{ij} are constants. Equation (1) becomes:

$$F(d_1, \ldots, d_6) = 0, \tag{2}$$

where F is a second-degree multivariate polynomial. Equation (2) describes a hyper-surface S.

The extreme orientations of $\rho(v_1)$, while it coincides with $\rho(v_2)$, can be attained in the interior or the boundary of $\mathcal{V}_L$:

- In the interior of $\mathcal{V}_L$, they are obtained when:

$$\frac{\partial(y_1/x_1)}{\partial d_k} = 0, \quad k = 1, \ldots, 5, \tag{3}$$

where d_6 is an implicit function of $d_1, \ldots, d_5$ defined by Equation (2). Thus, we must solve a system of six polynomial equations in $d_1, \ldots, d_6$: Equation (2), which has degree 2, and the five Equations (3), which have degree 4 each. This takes time exponential in the number of variables and polynomial in the maximal degree of the polynomials [7]. Here, this time is $O(1)$.

- To get the extreme orientations of $\rho(v_1)$ when S intersects a face of $\mathcal{V}_L$ of dimension $p \in [1, 5]$, we must also solve a system of six polynomial equations. This system consists of: Equation (2), the $6 - p$ equations defining the face, and $p - 1$ equations of the form of Equations (3), in which $6 - p$ variational parameters are determined by the equations of the face and one other parameter is an implicit function of the remaining $p - 1$ parameters through Equation (2). When $p = 1$, the face is a one-dimensional edge and there is no equation of this last type. Each of these systems also takes time $O(1)$ to solve.

In total, there is a constant number of systems to solve. Hence, the computation of the points of $\mathcal{V}_L$ where $\rho(v_1)$ achieves extreme orientations while coinciding with $\rho(v_2)$ has constant complexity.

Up to 6 vertex rays may coincide simultaneously. The alignment of m rays ($m \in [2, 6]$) yields the intersection of m hyper-surfaces such as S. The extremal orientations of these rays while they coincide are still solutions of systems each having 6 polynomial equations of constant degree in $d_1, \ldots, d_6$. Again, such a system can be solved in constant time.

By considering all combinations of $m \in [2, 6]$ vertices of the regions ψ^{kl}, we obtain a set H_2 of size $O((q_i q_j)^6)$. This size can be reduced as follows: We notice that, when $d_1, \ldots, d_6$ vary, the supporting lines of at most 3 edges of P_i translate; hence, at most $O(1)$ vertices move; we refer to them as the *special* vertices of P_i. Similarly, the supporting lines of all edges in P_j, except a maximum of 3, translate by the same amount relative to P_j; hence, all vertices, except $O(1)$ of them, to which we refer as the special vertices of P_j, move in the same way. Every vertex of a region ψ^{kl} is of the form $v_j \ominus v_i$, where v_i and v_j are vertices of P_i and P_j, respectively. We divide the vertices of all the regions ψ^{kl} into two subsets: One contains all vertices $v_j \ominus v_i$ where neither v_i nor v_j is special; its size is $O(q_i q_j)$. The other contains all the other vertices; its size is $O(q_i + q_j)$. All vertices in the first subset move the same. So, if v and v' are two vertices of this subset and the rays $\rho(v)$ and $\rho(v')$ coincide, this coincidence cannot create a side of L_{ij}. Therefore, to construct H_2, it is sufficient to consider all combinations of $m \in [2, 6]$ vertices such that at most one belongs

to the first subset. Posing $q = \max\{q_i, q_j\}$, this remark reduces the size of H_2 and the time to compute it to $O(q^7)$.

H_2 still contains too many points. For example, a point need not be included in H_2 if, at this point:
- Two coinciding rays have opposite directions. (Equation (1) allows that to happen.)
- One of the coinciding rays is not tangent to the region ψ^{kl} of its defining vertex.
- Two of the coinciding rays achieve different contacts, clockwise and counterclockwise, with the regions of their defining vertices.

However, these additional reductions do not change the worst-case asymptotic size of H_2.

We now have H_1 and H_2. We can compute the union $\bigcup_{H_1,H_2} C_{ij}$ of all the cones obtained at the points of $H_1 \cup H_2$. If this union is a connected cone, it is $\bigcup_L C_{ij}$. If, instead, it consists of several disjoint cones (only connected at their common apex), it remains to determine which are the two sides of these cones that also bound $\bigcup_L C_{ij}$. For that purpose, we consider the complement of the union in S^1 and we slightly perturb the points of $H_1 \cup H_2$ within $\mathcal{V}_L$, along the axes of $\mathcal{V}_L$. Eventually, all cones in $S^1 \setminus \bigcup_{H_1,H_2} C_{ij}$ will shrink a bit, except one, which is $S^1 \setminus \bigcup_L C_{ij}$. Hence, $\bigcup_L C_{ij}$ is computed in time $O(q^9)$.

Shape parameters (J): Now, let the parameters in J vary. Let P_i (resp. P_j) stand for P_i (resp. P_j) when all the parameters in J are minimal. For any value of the parameters in L, P_i is included in every other instance of P_i. The same holds for P_j. Hence, when the parameters in both J and L vary, we obtain the union $\bigcup_{J,L} C_{ij}$ of all cones C_{ij} by performing the same computation as above, with P_i and P_j substituted for P_i and P_j, respectively.

Position parameters (K): When the parameters in K vary, the polygon $\mathsf{P}_i \ominus \mathsf{P}_j$ keeps a constant shape, but p_j spans the constant-shape polygon W (see previous section). For any value of the parameters in J and L, the extreme orientations of the vertex rays are obtained when p_j is at vertices of W. Hence, when all parameters in $J \cup K \cup L$ vary, we perform the above computation with p_j successively located at every vertex of W. Since W has size $O(r_{ij})$, we obtain $O(r_{ij})$ cones $\bigcup_{J,L} C_{ij}$. If their union consists of a single cone, this cone is LC_{ij}; otherwise, we identify LC_{ij} by slightly perturbing the position of p_j at each vertex of W (within W). LC_{ij} is thus obtained in total time $O(r_{ij}q^9)$.

Computing the $O(N^2)$ large cones needed to construct the weak NDBG of A takes time $O(N^2 n(q^9 + \log n))$. While polynomial, this bound is too large for a practical implementation. Further effort is needed to reduce it, either by a tighter count of H_2 (which we believe is possible), or by finding a suitable approximation algorithm.

7 Inverting Disassembly Sequences

So far, our presentation has assumed that assembly and disassembly sequences are inverse of one another. This is clearly true when parts have nominal geometry (and are rigid). When parts have toleranced geometry the relation between the two types of sequences is less obvious. In this section we analyze this relation.

Consider an assembly A of two parts P_1 and P_2 linked by a relation R. Let "Break A into $\{P_1\}$ and $\{P_2\}$ by translating $\{P_1\}$ along t" be a feasible disassembly operation. The inverse operation,

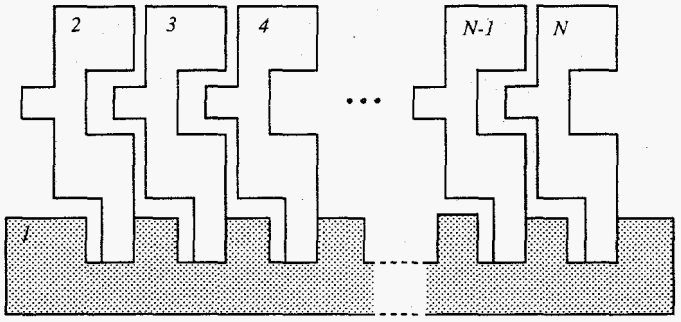


Figure 13: An N-part product requiring N hands to assemble

as defined in Section 3, is not truly an assembly operation: We do not know where to exactly position P_1 prior to translating it, since we do not accurately know in advance the relative position that P_1 and P_2 will have in A. In the disassembly operation, this issue does not arise, because P_1 and P_2 are *de facto* appropriately positioned prior to the translation. It does not arise either under the nominal-geometry assumption, since the unique relative position of the two parts in A and the direction $d + \pi$ then determine the line on which p_1 (the origin of P_1's coordinate system) should lie prior to the translation of P_1.

We can get rid of this difficulty by assuming that some sensing operation locates the edges and vertices of P_1 and P_2 involved in R with sufficient accuracy. We can then compute where to place p_1 prior to translating P_1. In doing so, we make an assumption that is not required by the disassembly operation. Because P_1 and P_2 are available prior to merging them, this assumption seems reasonable. Furthermore, whatever assumptions are made in assembly sequencing, the actual execution of the assembly operation will require some sensing and/or passive compliance to actually succeed.

However, this sensing assumption may not be sufficient for assemblies made of more than two parts. Let D be a disassembly sequence of A that produces a subassembly S whose relation graph (the subgraph obtained by restricting the relation graph of A to the parts in S) contains two connected components, or more. We call the subset of parts contained in each such component a *float.* While the relative positions of parts in a float are uniquely determined by the parts in that float, the relative position of any two floats in S depends on parts in $A \backslash S$. Here, we do not extend the above sensing assumption, because it would require the availability of parts that are not involved in the merging operation that produces S. Therefore, this operation does not have a uniquely defined outcome: It can only produce a temporary S. The relative positions of the floats in S will later have to be re-adjusted, when S is merged with other subassemblies. D still induces an assembly sequence, but this assembly sequence is non-monotone in the sense that it produces non-final subassemblies. It is also multi-handed since some assembly operations require more than two hands to hold the various floats and adjust their relative positions.

To illustrate this discussion, consider the product of Figure 1, with spatial relations between parts 1 and 3, 1 and 4, and 2 and 4. The interlocking between parts 3 and 4 requires disassembly to eventually produce the subassembly $\{3, 4\}$, whose geometry depends on part 1. The corresponding assembly sequence generates a temporary subassembly $\{3, 4\}$ with two floats (3 and 4), and later merges this subassembly with part 1. This merging operation requires three hands to hold the

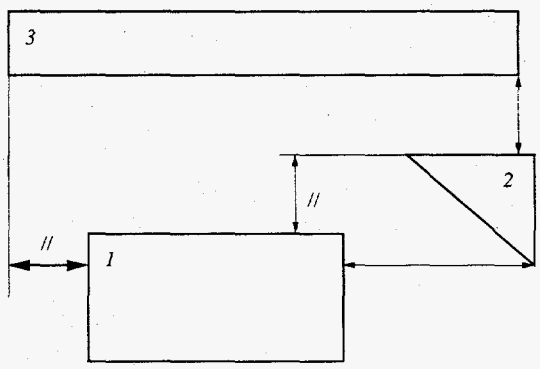


Figure 14: Ternary spatial relation

three floats (3, 4, and 1) and adjust their relative positions. As another example, consider the product of Figure 13, with spatial relations between 1 and 2, 1 and 3, ... , and 1 and N. The first operation in any disassembly sequence partitions the product into 1 and $\{2, \ldots, N\}$. The second subassembly consists of $N - 1$ floats. The corresponding assembly operation requires N hands to separately adjust the positions of parts $2, 3, \ldots, N$ relative to part 1.

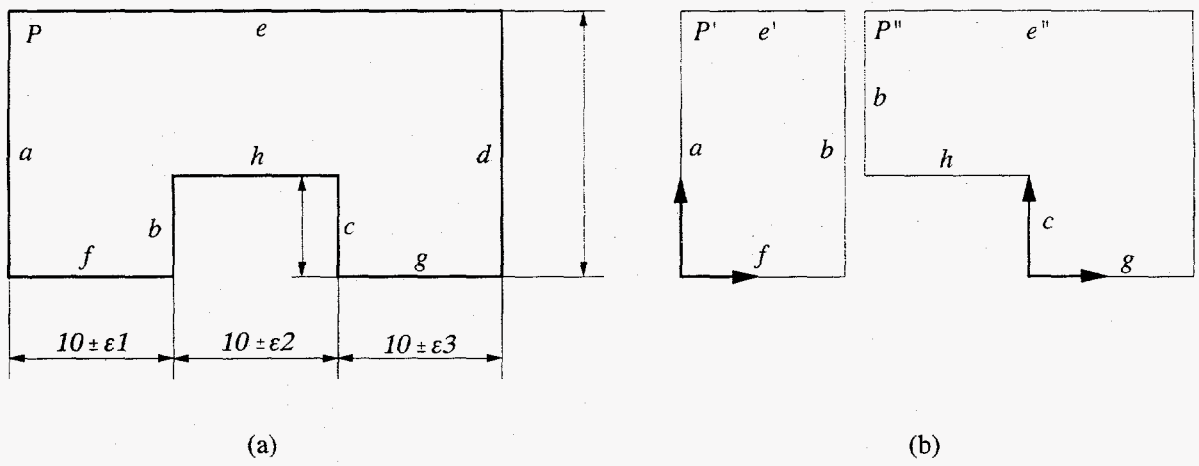
In general, one would like to generate a two-handed disassembly sequence that yields a minimally-handed assembly sequence, since this assembly sequence is likely to be more easily executed than one that is not minimally-handed. If A admits a disassembly sequence D yielding an m-handed assembly sequence, every subset of A (not just those produced by D) admits a disassembly sequence yielding a p-handed assembly sequence with $p \leq m$. Therefore, we can modify `partition` into a greedy algorithm so that `disassemble` only constructs disassembly sequences yielding minimally-handed assembly sequences. The new `partition` always scans the entire NDBG of the (sub)assembly S passed as argument, and selects a partitioning of S into two subassemblies such that the total number of floats in these subassemblies is minimal. On the average, this variant takes more time to run, but it has the same worst-case complexity as the original procedure.

8 Generalizations

The algorithms of Sections 4, 5, and 6 can be extended to handle several generalizations of the problem stated in Section 2.

Spatial relations involving more than two parts: To define some assemblies we may need spatial relations involving more than two parts. Figure 14 shows such an assembly: While the relative placement of parts 1 and 2 is defined by a spatial relation between them, the placement of part 3 is defined relative to parts 1 and 2 by a relation involving features of both parts 1 and 2. We call the relation between 1 and 2 a binary relation and the relation between 3 and $\{1, 2\}$ a ternary relation.

Although we could be even more general, let the relative placements of the parts in A be described by binary and ternary spatial relations such that, if a part P_i is connected to two other parts P_j and P_k by a ternary relation, then P_j and P_k are connected by a chain of binary relations.



(a) (b)

Figure 15: Part with multiple coordinate systems

Let B be the undirected graph whose nodes are the parts of A and whose links are the binary relations. Let T be the undirected graph whose nodes are the connected components of B and whose links are the ternary relations (more precisely, let X_1 and X_2 be any two nodes of T; if a ternary spatial relation defines the placement of a part in X_1 relative to two parts in X_2, then X_1 and X_2 are connected by a link). The set of all spatial relations is complete if and only if T is connected. It is non-redundant if and only if neither B nor T contain cycles and no two nodes of T are connected by more than one ternary relation.

Allowing ternary spatial relations requires little change in our algorithms. The only difference is in the identification of the variational parameters that affect the relative position of P_i and P_j, prior to computing each cone SC_{ij} and LC_{ij}. But this identification remains straightforward and the generated parameters can still be separated between the two sets J and L defined in Section 5. The maximal size of L remains 6 (this is important since the algorithms take time exponential in this number). The size of J is now $O(N)$, instead of $O(n)$, but previously n could be as large as $N-1$ in the worst case. The locus of p_j relative to p_i when the parameters in J vary is thus a polygon W of complexity $O(N)$. The rest is unchanged. The computation of the strong and weak NDBGs takes time $O(N^3(q^2+\log N))$ and $O(N^3(q^4+\log N))$, respectively.

Assigning multiple coordinate systems to a part: Consider the part P of Figure 15.a. The distance between edges a and b is $10 \pm \epsilon_1$, the distance between edges b and c is $10 \pm \epsilon_2$, and the distance between edges c and d is $10 \pm \epsilon_3$. (Ignore the other distances for a while.) To express such tolerance constraints by assigning a single coordinate system to P, we would have to rewrite them into, say, three constraints relating a to b, a to c, and a to d. But these constraints would be stronger than the original ones. Instead, we can decompose P into two fictitious components P' and P'', and assign a coordinate system to each of these two "parts" (Figure 15.b). The origin of the system of P' (resp. P'') lies at the intersection of edges a and f (resp. c and g). The tolerance zone defining the distance between a and b is specified in the system of P', while the tolerance zones defining the distances between b and c, c and d, and g and h are described in P''. (Continue ignoring the distance between f, g, and e.) Moreover, a spatial relation is defined between P' and

P'' to achieve their relative placement in P. In more complicated cases, the decomposition of a part may yield overlapping fictitious components.

The strong and weak NDBGs of an assembly A containing part P (or several such parts) can be generated using the algorithms presented in Sections 4, 5, and 6. To compute the small and large cones, we replace P by its fictitious components. Identifying the variational parameters that affect the relative position of any two parts remains straightforward. For any pair of parts, the set of parameters that affect both their geometry and relative position (set L) is still of size 6, at most. Thus, the cost of computing the small and large cones is unchanged. (Note in passing that there is no need to compute the small and large cones for any pair of fictitious parts deriving from P, since these parts will not be moved separately.) Once the relevant cones have been computed, the strong and weak NDBGs are generated as before, with P yielding a single node in each DBG: Any arc that is not a set of feasible directions for some component of P is also not a set of feasible directions for P. Beyond this step, the decomposition of P is forgotten.

Finally, in the example of Figure 15, we deal with the fact that f and g are aligned, i.e., lie at the same distance of e, by simply requiring that the distances between f and e' in P' and between g and e'' in P'' be equal. This means that the two variational parameters are collapsed into one.

Tolerancing distances in spatial relations: Our algorithms can deal with toleranced distances in spatial relations, by treating these distances as additional variational parameters with their own tolerance zones. None of them influence the shapes of the parts, but they may affect the relative position of some pairs of parts. For any two parts P_i and P_j, the set J of parameters used in the computation of the cones SC_{ij} and LC_{ij} should include the toleranced distances involved in the spatial relations that determine the relative placement of P_i and P_j. Since there are at most two toleranced distances per spatial relation, the complexity of the algorithms is unchanged.

In the same way, our algorithms can deal with some "floating" assemblies. Such assemblies differ from the "static" assemblies considered so far by the fact that the relative placement of two or more subsets of parts is allowed to vary within some bounded range [30]. We can describe such assemblies by assigning intervals of values to distances in some spatial relations. Each such interval is then treated as a tolerance zone, though it has a completely different meaning and is usually larger.

Three-dimensional polyhedral assemblies: Let now A be an assembly made of N polyhedral parts. The language of spatial relations between parts is extended accordingly, but this raises no serious difficulty. There are only more ways to express spatial relations. The variational parameters of every part P_i in A are the distances between p_i and the planes supporting the faces of P_i. The tolerance zones are small enough to guarantee that any two instances of the same part have the same topology.

Let us first assume that A has nominal geometry. In 3D, directions span the unit sphere S^2. The cone C_{ij} of feasible infinite translations of a part P_i relative to a part P_j is still the cone of all translations erected from p_i and intersecting $P_j \ominus P_i$. The region $P_j \ominus P_i$ is a polyhedron. Hence, C_{ij} is a polyhedral cone whose intersection with the unit sphere centered at its apex is a "polygon" bounded by arcs of great circles. The arcs obtained with all the cones C_{ij} create an arrangement of regions in S^2 such that the DBG of A remains constant over each one. This arrangement and the associated DBGs form the NDBG of A. A system implementing this computation is presented in [40].

The computation does not require the explicit construction of the 3D region defined by $P_j \ominus P_i$. We need only project its edges into S^2, as follows: first, we compute the Minkowski difference of every pair of faces of P_i and P_j using the algorithm given in [16]; next, we project the edges of all computed differences into S^2. We get more arcs than actually needed, but in the worst case their asymptotic number is the same. Let q be the maximal number of vertices in a part of A. Each pair of parts contributes $O(q^4)$ arcs of the arrangement on S^2. The total arrangement has size $O(N^2q^4)$ and is computed in time $O(N^2q^4\log(Nq))$. In every region the DBG is computed in time $O(N^2)$. The total NDBG is constructed in time $O((Nq)^4 + N^2q^4\log(Nq))$. Each DBG has $O(N^2)$ arcs, so that finding its strong components takes time $O(N^2)$. Hence, `partition` has complexity $O((Nq)^4)$.

If A is made of toleranced parts, all small cones SC_{ij} can be computed as suggested in Section 5: $\bigcup(P_j \ominus P_i)$ is constructed by computing a finite number (more than 6, however) of regions $\bigcup_{J,K}(P_j \ominus P_i)$. None of these regions need to be explicitly constructed in 3D. For each of them, we decompose P_i and P_j into convex components P_i^k and P_j^l and we project the edges of $\bigcup_{J,K}(P_j^l \ominus P_i^k) = W \oplus P_j^l \ominus P_i^k$ into S^2. Each pair of parts yields $O(n^2q^4)$ arcs in the arrangement on S^2. The arrangement defining the strong NDBG has size $O((Nn)^2q^4)$ and is computed in time $O((Nn)^2q^4\log(Nq))$. The procedure `partition` has complexity $O(N^4n^2q^4)$. The computation of the large cones and therefore the weak NDBG seems much more problematic, however.

9 Conclusion

Previous research has thoroughly investigated assembly sequencing under the assumption that parts and products have nominal geometry. It has produced useful algorithms to detect undesirable geometric interferences among parts. But these algorithms cannot help designers analyze the effect of their tolerancing decisions on the assembly process. As product quality and manufacturing automation increase, such analysis becomes critical. This paper is a first attempt to fill this need. It describes algorithms to generate assembly sequences for products made of toleranced parts. These algorithms could be embedded in an interactive CAD environment to assist designers in the selection of appropriate tolerance values.

Our approach to assembly sequencing with toleranced parts derives from the NDBG-based approach previously proposed in [40]. Two non-directional blocking graphs, the strong and the weak, are precomputed. They respectively represent possible and necessary blocking interferences among parts in an assembly. These NDBGs are then exploited in a query phase to generate assembly sequences. Using the strong NDBG we determine if a product accepts an assembly sequence that is always feasible, independent of the values of the variational parameters in their tolerance zones. Using the weak NDBG we determine if a product is never assemblable, or if it accepts non-guaranteed assembly sequences. One may use Monte Carlo techniques to estimate the probability of success of non-guaranteed sequences.

At the core of this approach are two algorithms to compute cones of feasible infinite translations of one part P_i relative to another P_j, when both parts have toleranced geometry and their relative position varies due to the toleranced geometry of parts lying between them. The key fact underlying these two algorithms is that the number of variational parameters that affect both the shapes of P_i and P_j and their relative position is constant, i.e., independent of the complexity of P_i, P_j, and the total assembly. This fact is crucial because the time complexity of the algorithms depends

exponentially on this number, while it is polynomial in all other measures of the size of the input data. It is preserved in several generalizations presented in the last section of this paper.

Another finding reported in this paper is that, unlike in the nominal-geometry case, assembly sequences are not the strict inverses of disassembly sequences. Although our algorithms generate monotone two-handed disassembly sequences, the corresponding assembly sequences may be non-monotone and multi-handed due to the presence of multiple floats in some subassemblies. The relative position of two floats in a subassembly S is not entirely determined by the parts in S. The initial construction of this subassembly achieves a temporary relative position of the two floats (arbitrarily selected in the set of possible positions). This temporary position is later adjusted when S is merged with other subassemblies. We have proposed a simple algorithm to generate disassembly sequences that minimize the number of floats at any one step.

The tolerance language used to describe assemblies is simple and falls short of modeling all imperfections of a manufacturing process. It nevertheless captures several important features of the Y14.5 standard. Its main limitation is that it assumes perfect angles between edges. To remove this limitation, we could associate a tolerance zone with the orientation of every line supporting an edge of a part. However, this would lead parts to have different relative orientations in various instances of the assembly. One *ad hoc* way to deal with this difficulty would be to discretize the angular tolerance zones and treat each set of discrete values as perfect angles. One could also perform some sort of Monte-Carlo sensitivity analysis on guaranteed assembly sequences around the nominal orientations of the edges. However, we believe that additional research should make it possible to provide an exact solution (at least for planar assemblies).

Another topic for future research is to go beyond infinite translations and allow motions made of several extended translations (as in [43]), as well as motions combining translation and rotation. The computation of large cones in 3D seems a challenging issue as well.

References

[1] *Dimensioning and Tolerancing, ANSI Y14.5M-1982*, ASME, United Engineering Center, New York, NY, 1982.

[2] Altschul, R.E. and Scholz, F.W., Statistical Tolerancing: A Case Study, *Proc. Intl. Forum on Dimensional Tolerancing and Metrology*, ASME, New York, NY, 1993.

[3] Arkin, E.M., Connelly, R., and Mitchell, J.S.B., On Monotone Paths Among Obstacles with Applications to Planning Assemblies, *Proc. 5th ACM Symp. on Computational Geometry*, 334-343, 1989.

[4] Baldwin, D.F., *Algorithmic Methods and Software Tools for the Generation of Mechanical Assembly Sequences*, Master's Thesis, MIT, 1990.

[5] Baldwin, D.F., Abell, T.E., Lui, M.C., De Fazio, T.L., and Whitney, D.E., An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products, *IEEE Tr. on Robotics and Automation*, 7(1), 78-94, 1991.

[6] Bourjault, A., *Contribution à une Approche Méthodologique de l'Assemblage Automatisé: Elaboration Automatique de Séquences Opératoires.* Thèse d'Etat, Faculté des Sciences et des Techniques de l'Université de Franche-Comté, Besançon, France, 1984.

[7] Canny, J.F. *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

[8] Chase, K.W. and Greenwood, W.H., Design Issues in Mechanical Tolerance Analysis, *Manufacturing Review*, 1, 50-59, 1988.

[9] Chazelle, B., Ottmann, T.A., Soisalon-Soininen, E., and Wood, D., The Complexity and Decidability of SEPARATION, *Lecture Notes in Computer Science*, Vol. 172, Springer Verlag, New York, NY, 119-127, 1984.

[10] Clément, A., Desrochers, A., and Rivière, A., Theory and Practice of 3D Tolerancing for Assembly, *Proc. CIRP Seminar on Computer Aided Tolerancing*, Penn State Univ., May 1991.

[11] De Fazio, T.L. and Whitney, D.E., Simplified Generation of All Mechanical Assembly Sequences, *IEEE Tr. on Robotics and Automation*, 3(6), 640-658, 1987.

[12] Fleming, A., *Analysis of Uncertainties and Geometric Tolerances in Assemblies of Parts*, PhD Thesis, Dept. of Computer Science, Univ. of Edinburgh, 1987.

[13] Frants, L., *Automating Tolerance Analysis in Computer Aided Design*, PhD Thesis, Dept. of Computer Science, Stanford Univ., 1995 (to appear).

[14] Giordano, M. and Duret, D., Clearance Space and Deviation Space, Application to Three-Dimensional Chain of Dimensions and Positions, *Proc. 3rd CIRP Seminar on Computer Aided Tolerancing*, Editions Eyrolles, Paris, April 1993.

[15] Guibas, L., Halperin, D., Hirukawa, H., Latombe, J.C., and Wilson, R.H., A Simple and Efficient Procedure for Polyhedral Assembly Partitioning under Infinitesimal Motions, *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, Nagoya, Japan, May 1995 (to appear).

[16] Guibas, L. and Seidel, R., Computing Convolution by Reciprocal Search, *Proc. ACM Symp. on Computational Geometry*, Yorktown Heights, NY, 90-99, 1986.

[17] Guibas, L. and Yao, F., On Translating a Set of Rectangles, in *Computational Geometry*, Preparata, F.P. (ed.), Advances in Computing Research 1, JAI Press, London, 61-77, 1983.

[18] Guilford, J. and Turner, J., Advanced Analysis and Synthesis for Geometric Tolerances, *Manufacturing Review*, 6(4), 305-313, Dec. 1993.

[19] Hoffman, R.L., A Common Sense Approach to Assembly Sequence Planning, in [21], 289-314, 1991.

[20] Homem de Mello, L.S., *Task Sequence Planning for Robotic Assembly*, PhD Thesis, Carnegie Mellon University, 1989.

[21] Homem de Mello, L.S. and Lee, S. (eds.), *Computer-Aided Mechanical Assembly Planning*, Kluwer Academic Publishers, Boston, 1991.

[22] Homem de Mello, L.S. and Sanderson, A.C., A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences, *IEEE Tr. on Robotics and Automation*, 7(2), 228-240, 1991.

[23] Hopcroft, J.E., Schwartz, J.T., and Sharir, M., On the Complexity of Motion Planning for Multiple Independent Objects: PSPACE-Hardness of the 'Warehouseman's Problem', *The Intl. J. of Robotics Research*, 3(4), 76-88, 1984.

[24] Kavraki, L., Latombe, J.C., and Wilson, R.H., On the Complexity of Assembly Partitioning, *Information Processing letters*, 48, 229-235, 1993.

[25] Khanna, S., Motwani, R., and Wilson, R.H., Dynamic Strong Connectivity with a Look-Ahead, in preparation.

[26] Krishnan, S.S. and Sanderson, A.C., Path Planning Algorithms for Assembly Sequence Planning, *Proc. IEEE Intl. Conf. on Intelligent Robotics*, 428-439, 1991.

[27] Lee, S. and Shin, Y.G., Assembly Planning Based on Geometric Reasoning, *Computation and Graphics*, 14(2), 237-250, 1990.

[28] Natarajan, B.K., On Planning Assemblies. *Proc. ACM Symp. on Computational Geometry*, 299-308, 1988.

[29] O'Rourke, J., *Computational Geometry in C*, Cambridge Univ. Press, New York, NY, 1994.

[30] Parratt, S.W., *A Theory of One-Dimensional Tolerancing for Assembly*, PhD Thesis, Sibley School of Mechanical and Aerospace Eng., Cornell Univ., 1993.

[31] Pollack, R., Sharir, M., and Sifrony, S., Separating Two Polygons by a Sequence of Translations, *Discrete and Computational Geometry*, 3, 123-136, 1988.

[32] Requicha, A.A.G., Mathematical Definition of Tolerance Specifications, *Manufacturing Review*, 6(4), 269-274, 1993.

[33] Requicha, A.A.G, Representation of Tolerances in Solid Modeling: Issues and Alternative Approaches, in *Solid Modeling by Computers: From Theory to Applications*, Pickett, M.S. and Boyse, J.W. (eds.), Plenum Press, New York, 3-22, 1984.

[34] Snoeyink, J. and Stolfi, J., Objects that Cannot be Taken Apart with Two Hands, *Proc. 9th ACM Symp. on Computational Geometry*, 247-256, 1993.

[35] Srinivasan, V., Recent Efforts in Mathematization of ASME/ANSI Y14.5M Standard, *Proc. 3rd CIRP Seminars on Computer Aided Tolerancing*, Editions Eyrolles, Paris, 223-232, April 1993.

[36] Toussaint, G.T., Movable Separability of Sets, in *Computational Geometry*, Toussaint, G.T. (ed.), North-Holland, Amsterdam, Netherlands, 335-375, 1985.

[37] Voelcker, H., A Current Perspective on Tolerancing and Metrology, *Manufacturing Review*, 6(4), 258-268, Dec. 1993,

[38] Walker, R.K. and Srinivasan, V., Creation and Evolution of the ASME Y14.5.1 Standard. *Manufacturing Review*, 7(1), 1994.

[39] Wilson, R.H. and Rit, J.F., Maintaining Geometric Dependencies in an Assembly Planner. *Proc. IEEE Intl. Conf. on Robotics and Automation*, Scottsdale, AZ, 890-895, 1990.

[40] Wilson, R.H., *On Geometric Assembly Planning*. PhD Thesis, Dept. of Computer Science, Stanford Univ., 1992.

[41] Wilson, R.H. and Latombe, J.C., On the Qualitative Structure of a Mechanical Assembly. *Proc. AAAI Nat. Conf. on Artificial Intelligence*, San Jose, CA, 697-702, 1992.

[42] Wilson, R.H. and Latombe, J.C., Geometric Reasoning About Mechanical Assembly, *Artificial Intelligence*, 71(2), 371-396, 1994.

[43] Wilson, R.H., Kavraki, L., Lozano-Pérez, T., and Latombe, J.C., *Two-Handed Assembly Sequencing*, Tech. Rep. No. STAN-CS-93-1478, Dept. of Computer Science, Stanford Univ., June 1993. To appear in *Intl. J. of Robotics Research*.

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

[45] Wolter, J., Chakrabarty, S., and Tsao, J., Mating Constraint Languages for Assembly Sequence Planning. *Proc. IEEE Intl. Conf. on Robotics and Automation*, Nice, France, 2367-2374, 1992.

Distribution for SAND94-3124:

MS9018	Central Technical Files, 8523-2	(1)
MS 0899	Technical Library, 13414	(5)
MS 0619	Technical Publications, 12615	(1)
MS 0100	Document Processing for DOE/OSTI, 7613-2	(2)
MS 0951	Randy Wilson, 2121	(10)