Assembly Sequencing with Toleranced Parts[*]

Jean-Claude Latombe[†], Randall H. Wilson[§], and Frédéric Cazals[†]

[†] Robotics Laboratory, Department of Computer Science,
Stanford University, Stanford, CA 94305, USA

[§] Intelligent Systems and Robotics Center,
Sandia National Laboratories, Albuquerque, NM 87185, USA

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 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 (i.e., translations that can be extended arbitrarily far along a fixed direction) 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. These two procedures are described for two-dimensional assemblies made of polygonal parts and for three-dimensional assemblies made of polyhedral parts. So far, only the first has been implemented (for the planar case). This work assumes a simple, but non-trivial tolerance language that falls short of capturing all imperfections of a manufacturing process. In particular, it assumes that faces and edges have perfect relative orientations. Thus, it is only one step toward dealing with tolerances in assembly sequencing.

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

[*]This paper is an extended version of an earlier paper presented at the Third Symposium on Solid Modeling and Applications, Salt Lake City, Utah, May 17-19, 1995 [27].

1 Introduction

An assembly is described by a geometric model of its parts and their relative placement. The goal of assembly sequencing is to plan a partial ordering of operations to construct this product from its parts. Each operation generates a new subassembly by merging individual parts and/or subassemblies constructed by previous operations. It is specified by the subassemblies it merges and their relative motions.

There has been considerable research in assembly sequencing during the past decade (e.g., [5, 10, 18, 19, 20, 25, 28, 40, 41, 43, 44]). Early assembly sequencers were mainly interactive sequence editors; geometric reasoning was supplied by a human who answered questions asked by the system [10]. Automated geometric reasoning was then added to answer these questions automatically [5, 20, 44]. This development first resulted in generate-and-test sequencers, with a module guessing candidate sequences and geometric reasoning modules checking their feasibility [20, 40]. More efficient techniques were later proposed to replace time-consuming generate-and-test [4, 41]. Research on "separability problems" in Computational Geometry is also related to assembly sequencing [8, 32, 35, 37].

Assembly sequencing has been shown to be intractable [21, 22, 23, 29, 44], leading researchers to consider restricted, but still interesting subsets of assembly sequences, e.g.: *monotone* sequences, where each operation generates a final subassembly, and *two-handed* sequences, where every operation merges exactly two subassemblies. Often motions are also limited to translations. Though restrictions vary slightly among the 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 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 [34, 38]. A large body of work has been devoted to the development of tolerance languages (e.g., Y14.5 [1, 39]) providing designers with symbolic means to specify acceptable variations. One important goal is to guarantee part interchangeability in an assembly product [38]: given any set of parts manufactured according to the specified tolerances, they should assemble satisfactorily. The basic tolerance analysis problem – determining where the boundary of a part might be located in a given coordinate system – has attracted considerable interest (e.g., [7, 12, 16, 31]). But verifying part interchangeability is much harder, and previous work 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 [3, 9, 11, 13, 31].

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. 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 [33, 36]. As of today, we have only implemented the first procedure, for polygonal assemblies.

Section 2 describes the assembly-description language accepted by our algorithms. Section 3 gives technical background for the rest of the paper. It summarizes results previously reported in [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 blocking interferences between parts when their dimensions span the tolerance zones. It is 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 generalizes the algorithms of Section 4 and 5 (presented for planar polygonal assemblies) to polyhedral assemblies. Section 8 analyzes some subtle aspects of the relation between assembly and disassembly sequences when parts have toleranced geometry.

2 Description of an Assembly

We consider a planar assembly product A made of N parts $P_1, \dots, P_N$. 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 a unique coordinate system and the interval of acceptable distances from the origin of this system to the supporting line.

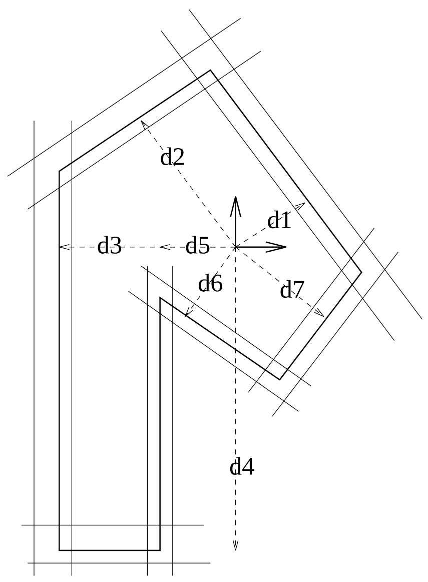

Figure 1: Part description

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.

Fig. 1 illustrates the description of a part with seven edges. It shows a particular instance of the part, 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 real whose 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 Fig. 1 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.

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 geometry of these two parts, a single relative position of their coordinate

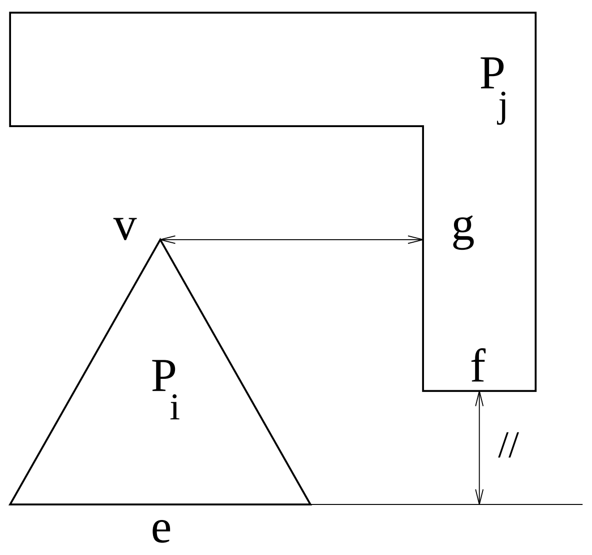

Figure 2: Spatial relation between two parts

systems achieves R. We assume that R consists of two more elementary relations: one states 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 states that a vertex of one part is at some signed distance of the line supporting an edge of the other part. (In addition, the three edges involved in R must not be parallel.) 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. For simplifying our presentation, we assume zero tolerances in the distance values of the spatial relations; removing this assumption in our algorithms is straighforward.

Fig. 2 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 in the description of A must be complete and non-redundant. By complete, we mean that if one randomly picks a geometry for every component of A, the relations determine a unique geometry for A (such an assembly is said to be "static" [31]). 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 and without cycles. We call this graph the *relation graph* of A.

For any two parts, P_i and P_j, in their relative placement in A, we refer to the position of p_j

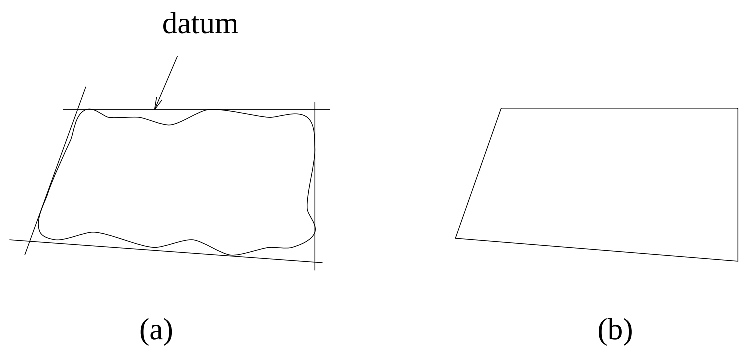

Figure 3: Part with imperfect edges

in the coordinate system of P_i as the *position* of P_j *relative* to P_i. This relative position may vary over several instances of A, due to variations in the geometry of the parts composing A. On the other hand, the relative orientation of the two coordinate systems is 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.

There is no requirement that an assembly product specified as above be feasible.

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 Fig. 3.a. We can bring a straight line, called a *datum* [31], into two-point contact with each edge and replace the imperfect edges by the perfect ones defined by the datums (Fig. 3.b). Our algorithms apply to the parts defined by these virtual edges and the vertices created by these 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, and this assumption is the main limitation of our language. See, however, the conclusion for a discussion of how it 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. In [26] we show that a simple preprocessing allows our algorithms to handle multiple coordinate systems per part. This consists of partitioning a part into a collection of subparts and assigning a distinct coordinate system to each part. The relative placements of the subparts in the part are defined by spatial relations having the same form as above. Our assembly sequencing algorithms then treats subparts as parts with a single coordinate system, except that they cannot be separated from one another.

The fact that we only consider planar assemblies is one important limitation not directly related to tolerancing. In Section 7 we show that the algorithms of Sections 4 and 5 are easily generalized to 3D polyhedral assemblies.

Another generalization of our algorithms discussed in [26] is the use of spatial relations linking more than two parts.

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 review previous techniques that generate monotone two-handed assembly sequences for A. We present the NDBG of A for infinite translations, i.e., translations that can be extended arbitrarily far along a fixed direction.

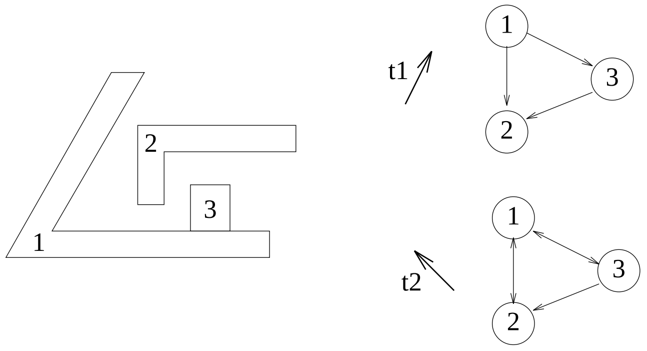

Figure 4: Examples of directional blocking graphs

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 $t + \pi$." When parts are rigid, this inverse map is a bijection between assembly and 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 early sequence planners, `partition` was based on generate-and-test: given S, enumerate all candidate partitions $\{S_1, S_2\}$ of S, until a direction t is found 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 the number of candidate partitions is exponential in the number of parts in S, while the number of feasible partitions is usually much smaller. The NDBG was introduced to avoid this combinatorial trap [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.

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

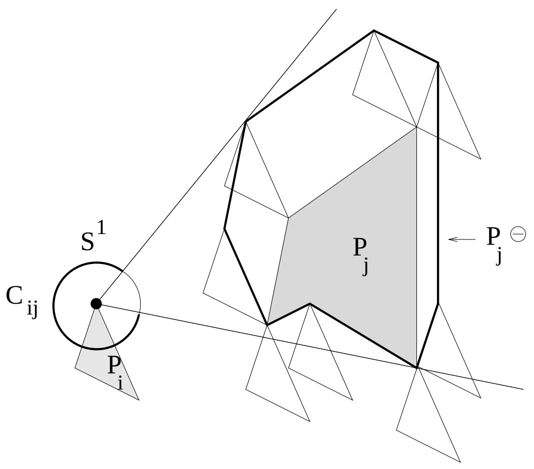

Figure 5: Construction of a cone of feasible infinite translations

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. Fig. 4 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.

Assume that there are no tight insertions in A. Each cone C_{ij} can 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[1] of P_j and P_i). See Fig. 5, 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 in the contour 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 decompose them into convex components denoted by P_i^k and P_j^l $(k, l = 1, 2, \ldots)$, 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 $O(q_j \log q_j)$ [30]. 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 $O(q_i q_j + q_i \log q_i + q_j \log q_j)$.

[1]Given two sets of points, X and Y, we have: $X \ominus Y = \{x - y | x \in X, y \in Y\}$ and $X \oplus Y = \{x + y | x \in X, y \in Y\}$.

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 [41]. The complete 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.[2] 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, $O(N^{1.46})$, can be obtained by taking advantage of the fact that any two successive DBGs differ by a small amount [24].) 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 [41]. 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 [14, 42] and multiple extended translations [17, 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 [4]. But, so far, this approach has only been introduced to generate translational assembly sequences for planar polygonal assemblies. Attempts to efficiently generalize it to 3D assemblies and/or rotational motions have failed.

[2] A *strongly connected component* (or *strong component*) of a directed graph is a maximal subset of nodes such that for any pair of nodes in this subset there exists a path connecting them in the graph. A directed graph is *strongly connected* if it has only one strong component.

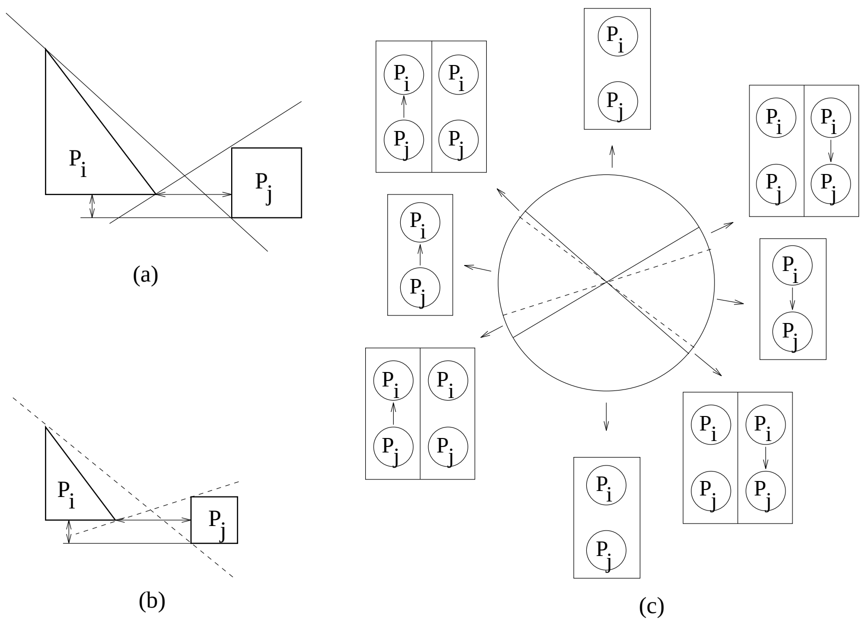

Figure 6: Multi-valued NDBG for two parts

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 unique 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 extend the NDBG concept to represent all 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 $\mathcal{V}$.

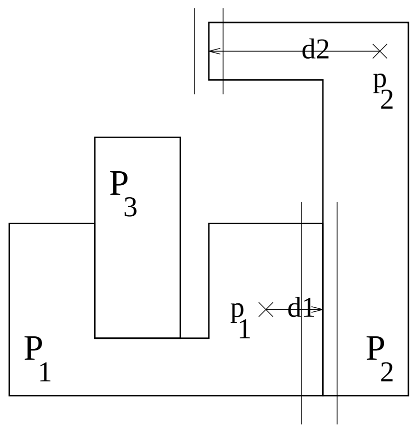

Figure 7: Influence of variational parameters on feasible translations

As computing all NDBGs over $\mathcal{V}$ is impractical, we project these NDBGs onto S^1: with every direction t of S^1, we associate the set 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. But this is not true for some isolated directions where a translated part collides with a new part or stops colliding with a part. These 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 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 Fig. 6, where the small (resp. large) cones are bounded by plain (resp. dashed) lines.

In the example of Fig. 6, the small and large cones are respectively obtained for the maximal material parts (Fig. 6.a) and least material parts (Fig. 6.b). But this is not always the case. For example, Fig. 7 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 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

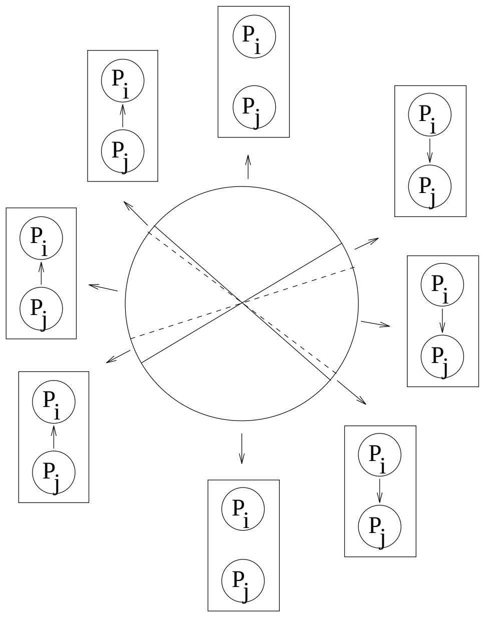

Figure 8: Strong NDBG for two parts

case the set of DBGs associated with one arc has size $O(2^{N^2})$. We thus 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; it describes all blocking interferences among the parts of A when the variational parameters span $\mathcal{V}$. Fig. 8 shows the strong NDBG derived from the multi-valued NDBG of Fig. 6. Since several adjacent DBGs are identical, the corresponding arcs should be merged. Clearly, 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. Considering all combinations of maximal and minimal values of the variational parameters would yield an algorithm exponential in the number of variational parameters. In the next section a different approach allows us to 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 NDBGs 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, the product may still be always assemblable, 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

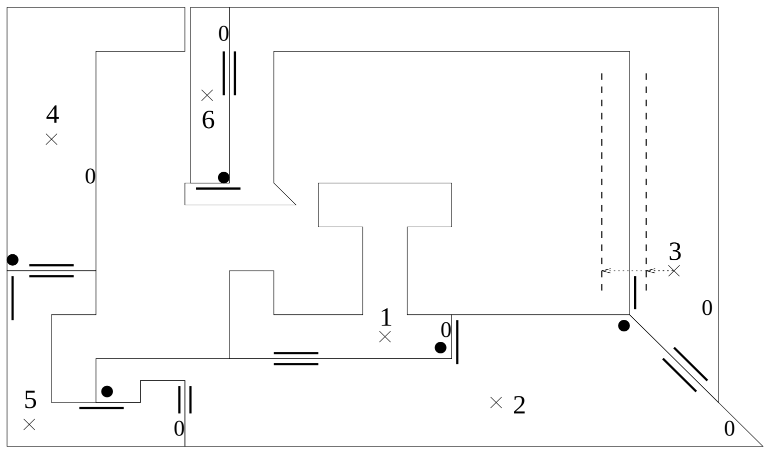

Figure 9: Engine example: Nominal model and tolerance zone

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 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 consumming. 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 infinite translations only. The algorithms described here are directly relevant to that case.

Implementation: We have implemented an assembly sequencer that computes and queries the strong NDBG of a product, using the algorithms described above and in the next section. This implementation was done within the CAS.CADE environment of Matra-Datavision [2], which offers an extensive C++ geometric library. Although this environment does not include Minkowski operations and NDBG computation, it provides more elementary data structures and operations that make their implementation easier than in plain C++. The implementation consists of 11000 lines of code.

We have run our sequancer on several examples, including the simplified six-part "engine" shown in Fig. 9. In this figure, for any two parts that jointly participate in a spatial relation

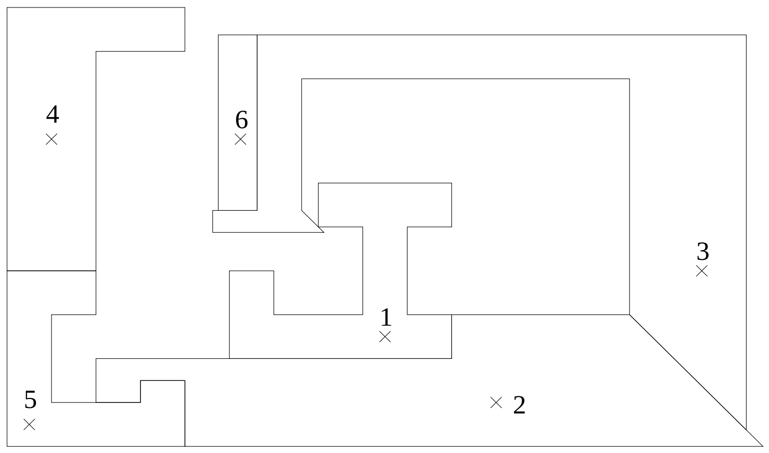

Figure 10: Engine model: Extreme case

as specified in Section 2, the edges involved in the edge-edge relation are marked with short bold line segments, while the vertex and the edge involved in the vertex-edge relation are marked with a bullet and a line segment, respectively. Also, each part is identified by a number. Its first edge is tagged with the number 0, while its other edges are labelled 1, 2, ..., in counterclockwise order.

For simplicity, let us focus our attention on edge 9 of part 3 and assume that this is the only variational parameter with a non-zero tolerance zone. We consider two tolerance zones for this parameter: the "positive-only" and the "negative-only" zones. Figure 9 depicts the assembly at nominal size. The union of the two tolerance zones is shown between the two dashed lines. The positive-only zone is on the left of edge 9 at its nominal position, while the negative-only zone is on its right.

We have run our assembly sequencer in three cases derived from this example: (1) edge 9 of part 3 is at its nominal location; (2) it lies anywhere in the positive-only zone; and (3) it lies anywhere in the negative-only zone. In each case, we output two measures characterizing some aspects of the NDBG: the number of guaranteed assembly sequences and the number of parts that must be removed before part 1 can be removed.

In case (1), the strong NDBG coincides with the regular NDBG described in Section 3, since all tolerance zones reduce to a single line. The NDBG then encodes 55 different assembly sequences. The shortest sequence to access part 1 is by removing parts 3 and 6 (as a rigid subassembly) with a vertical translation.

In case (2), the number of distinct guaranteed assembly sequences encoded by the strong NDBG shrinks to 23. Accessing part 1 is also more involved. It is no longer possible to vertically translate parts 3 and 6. Instead, part 4 can be removed first; then parts 3 and 6 can be translated toward the left-top (see Fig. 10, in which edge 9 of part 3 lies at the extreme end of its tolerance zone). An alternative to this sequence is to remove 5 and 2 by a vertical downward translation. If fixturing issues are ignored, this is a valid sequence. But

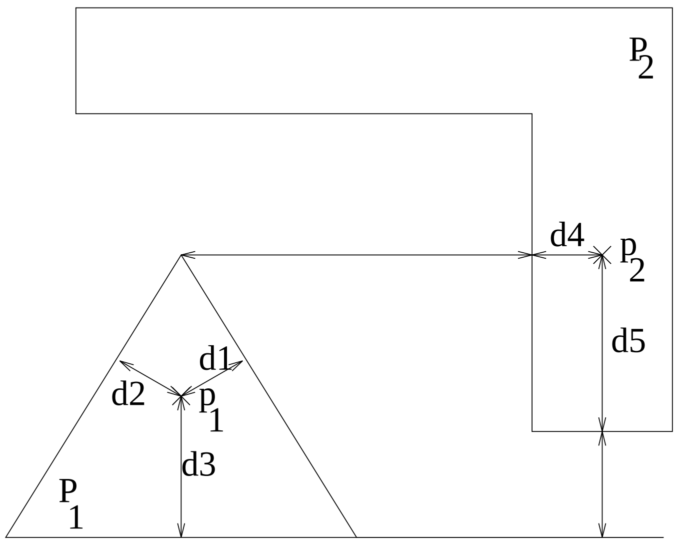

Figure 11: Parameters influencing the relative position of two parts

if they are not, parts 4, 1, and 3 will then have to be appropriately fixtured.

Finally, in case (3), the NDBG contains no guaranteed sequence. As can easily be observed, parts 4 and 6 can then overlap each other, which means that even the assembly state may not exist.

5 Computation of Small Cones

We now 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 more 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 Fig. 11, 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 (*shape parameters*) contains the variational parameters of P_i and P_j that do not influence the relative position of the two parts.

- K (*position parameters*) 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.

- L (*shape-position parameters*) 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 now consider these 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:

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

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_i^j in the coordinate system of P_i. The geometry of W_i^j is independent of the values of the parameters in $J \cup L$. If $r_{ij} = 1$, K is empty and W_i^j reduces to a single point. If $r_{ij} = 2$, W_i^j is a convex polygon of constant complexity, which may degenerate to a point or a line segment; Fig. 12 shows two examples: in the left example, W_i^j is a polygon, while in the right example, it is a line segment. W_i^j then takes constant time to compute. If $r_{ij} > 2$, W_i^j is a convex polygon

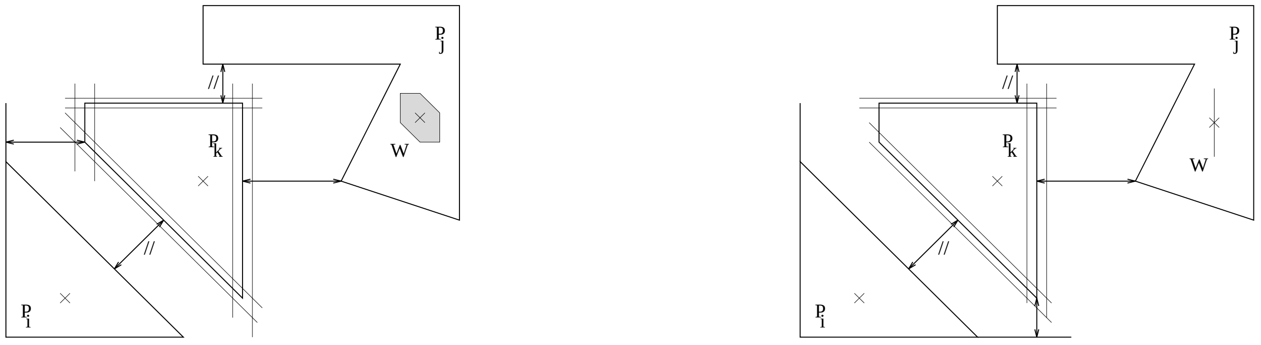

Figure 12: Locus W_i^j of p_j with respect to p_i

of complexity $O(r_{ij})$, which is computed in time $O(r_{ij} \log r_{ij})$. (The computation of W_i^j is discussed in the appendix; it is not difficult, but requires analyzing multiple cases.) Thus, while $\mathcal{P}_i$ is fixed, $\mathcal{P}_j$ sweeps the region $W_i^j \oplus \mathcal{P}_j$, where $X \oplus Y$ denotes the Minkowski sum of two sets X and Y. We have:

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

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 $\phi^{kl} = W_i^j \oplus e_j^l \ominus e_i^k$, where e_i^k and e_j^l $(k, l = 1, 2, \ldots)$ 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_i^j 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 $O(1)$. 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_i^j.

Let q be 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 needed 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, e.g.:

- There exists no guaranteed sequence: one may wish to generate non-guaranteed sequences to estimate their probability of success using, say, Monte Carlo sampling 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 subtle point is illustrated in Fig. 13, where we assume for simplicity 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 Fig. 13.a, the value of d is chosen in its tolerance zone so that both rays are aligned. Fig. 13.b and 13.c show the rays (dotted lines) with the most counterclockwise orientations when d takes its extreme

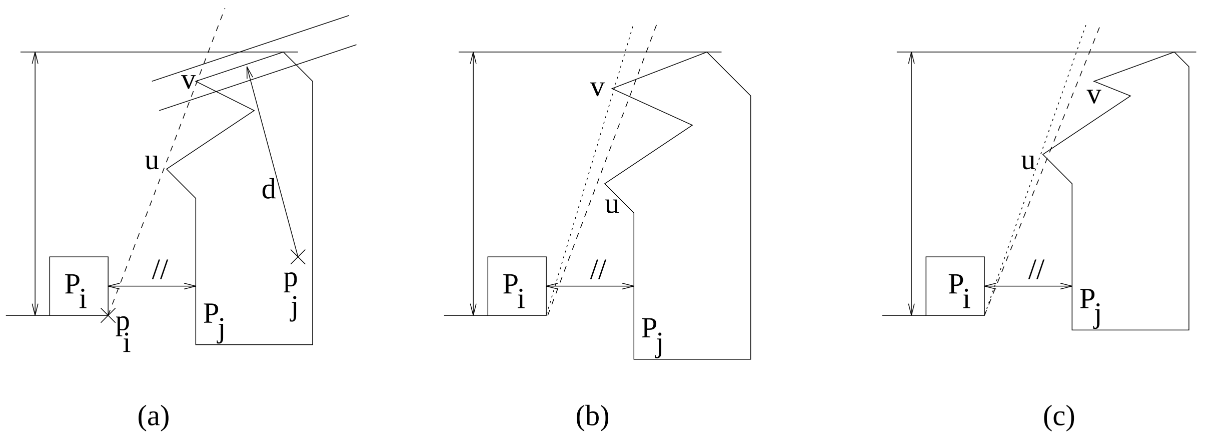

Figure 13: Large cone and extreme rays

values (the dashed rays are identical to the one in Fig. 13.a; they are reproduced to facilitate comparison betwen figures). 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 [6]. 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. 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 $O(q^7)$.

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_i^j (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_i^j. 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_i^j. Since W_i^j 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_i^j (within W_i^j). 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 Polyhedral Assemblies

In this section we generalize the algorithms of Sections 4 and 5 to the cases where A is an assembly made of N polyhedral parts. The language of spatial relations between parts must be 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 a unique 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 [41].

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 [15]; 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 $\bigcup_{J,K}(P_j^l \ominus P_i^k) = W_i^j \oplus P_j^l \ominus P_i^k$ 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.

8 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 unique 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, 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 unique-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. But, as P_i and P_j 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,

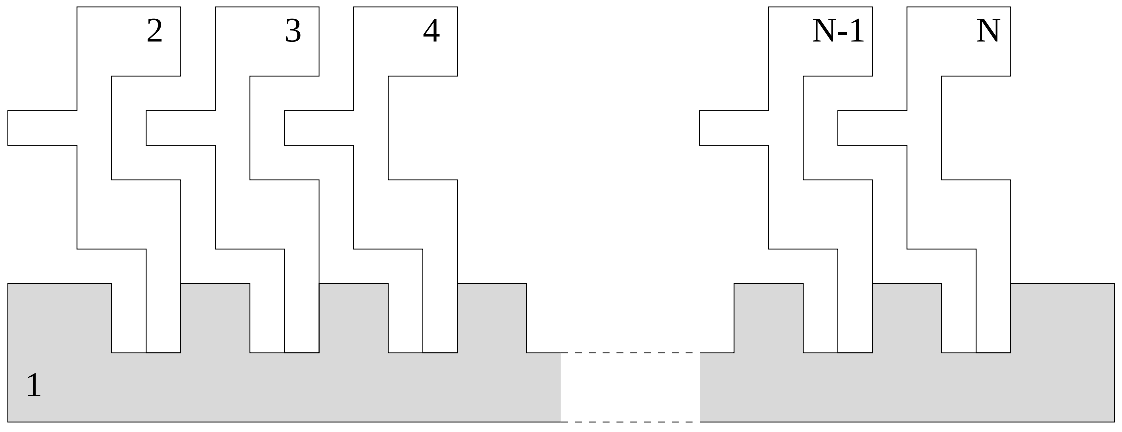

Figure 14: An N-part product requiring N hands to assemble

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 Fig. 14, with N parts denoted by $1, \ldots, N$. Spatial relations are 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 necessarily one produced by D) admits a disassembly sequence yielding a p-handed assembly sequence with $p \leq m$. Therefore, we can easily modify `partition` so that `disassemble` only construct 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.

9 Conclusion

Previous research has thoroughly investigated assembly sequencing under the assumption that parts and products have unique 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 more 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 [41]. 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 observation 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 and small. It is crucial because the time complexity of the algorithms is exponential in this number. This observation remains valid in several generalizations presented in Section 7 and in [26].

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. Removing this limitation would result in assembly instances where parts do not have the same relative orientations. This would seriously complicate our algorithms. One *ad hoc* way to accept toleranced angles is to discretize the corresponding tolerance zones and treat each set of discrete values as perfect angles. One could also perform some Monte-Carlo-based sensitivity analysis of a guaranteed assembly sequence 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 well as motions combining translation and rotation. The computation of large cones in 3D seems a challenging issue as well.

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. Frédéric Cazals is supported by Matra-Datavision. Early comments by Randy Brost, Matt Mason, and Vijay Srinivasan helped us set up the initial framework of our research. This paper has directly benefited from valuable remarks by Dan Halperin.

References

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

[2] *The CAS.CADE Software Factory. Technical Overview*, Version 1.2/Dec. 94, CF100-1.2-GP-TEC-A-01-US, Matra Datavision, Les Ulis, France, 1994.

[3] 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.

[4] 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.

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

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

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

[8] 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.

[9] 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.

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

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

[12] Frants, L., *Automating Tolerance Analysis in Computer Aided Design*, PhD Thesis, Dept. of Computer Science, Stanford Univ., 1995.

[13] 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.

[14] 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. IEEE Int. Conf. on Robotics and Automation*, Nagoya, 2553-2560, 1995.

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

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

[17] Halperin, D. and Wilson, R.H., Assembly Partitioning Along Simple Paths: The Case of Multiple Translations, *Proc. IEEE Int. Conf.on Robotics and Automation*, Nagoya, 1585-1592, 1995.

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

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

[20] 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.

[21] 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.

[22] Kavraki, L. and Kolountzakism M.N., Partitioning a Planar Assembly into Two Connected Parts is NP-Complete, *Information Processing Letters*, 55, 159-165, 1995.

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

[24] Khanna, S., Motwani, R., and Wilson, R.H., On Certificates and Lookahead in Dynamic Graph Problems, *Proc. 7th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 1996 (to appear).

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

[26] Latombe, J.C. and Wilson, R.H., *Assembly Sequencing with Toleranced Parts*, Technical Report SAND94-3124, Sandia National Laboratories, Albuquerque, NM, 1994.

[27] Latombe, J.C. and Wilson, R.H., *Assembly Sequencing with Toleranced Parts*, *Proc. 3rd Symp. on Solid Modeling and Applications*, 83-94, 1995.

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

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

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

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

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

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

[34] 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.

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

[36] 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.

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

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

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

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

[41] Wilson, R.H., *On Geometric Assembly Planning*. PhD Thesis, Dept. of Computer Science, Stanford Univ., 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, Intl. J. of Robotics Research*, 14(4), 335-350, 1995.

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

Appendix: Computation of W_i^j

In this appendix we consider two parts P_i and P_j of the assembly A. Let r_{ij} be the number of spatial relations defining the relative placement of P_i and P_j. We let $P_k, \ldots, P_{k+r_{ij}-2}$ designate the (possibly empty) list of parts along the path connecting P_i to P_j in the relation graph of A, i.e., P_i and P_k are linked by a spatial relation; so are P_k and P_{k+1}, ..., and $P_{k+r_{ij}-2}$ and P_j. The variational parameters of $P_k, \ldots, P_{k+r_{ij}-2}$ that affect the relative position of P_i and P_j form the set K defined in Section 5.

We assume P_i fixed and we describe the computation of the locus W_i^j of the origin p_j of the coordinate system attached to part P_j, when the parameters in K span their tolerance zones.

As mentioned in Section 5, we distinguish among three cases:

(1) $r_{ij} = 1$, (2) $r_{ij} = 2$, (3) $r_{ij} > 2$.

In Case (1), P_i and P_j are directly linked by a spatial relation. Hence, K is empty, and W_i^j reduces to a single point, whose coordinates are easily computed from the spatial relation linking P_i and P_j.

In Case (2), the set K contains variational parameters of a single part P_k. Let us assume for an instant that we know how to compute W_i^j in this case.

In Case (3), we have:

$$W_i^j = W_i^{k+1} \oplus W_k^{k+2} \oplus \ldots \oplus W_{k+r_{ij}-3}^j.$$

Indeed, W_i^{k+1} is the locus of p_{k+1} relative to the coordinate system of P_i, when the variational parameters of P_k span their tolerance zones, all the other parameters being fixed. W_k^{k+2} is the locus of p_{k+2} relative to the coordinate system of P_k, when the variational parameters of P_{k+1} vary, all the other parameters being fixed. When we let the variational parameters of both P_k and P_{k+1} vary, all the other parameters being fixed, we obtain $W_i^{k+2} = W_i^{k+1} \oplus W_k^{k+2}$. And so on. Therefore, the computation of W_i^j can be done by performing the computation done in Case (2) $r_{ij} - 1$ times.

We now focus on Case (2). We classify the variational parameters of P_k that influence the relative position of P_i and P_j into two types: the *input parameters*, which affect the position of P_k relative to P_i, and the *output parameters*, which affect the position of P_j relative to P_k. If a parameter is both an input and an output parameter, we call it an *input-output parameter*.

We break Case (2) into two subcases:

(2.1) There exist no input-output parameters.

(2.2) There exist input-output parameters.

Fig. 15 illustrates Subcase (2.1): here, the input parameters are d_1 and d_2, while the output parameters are d_3 and d_4. Fig. 16 shows two examples of (2.2). In (a), d_2 is an input parameter, d_4 an output parameter, and d_3 an input-output parameter. In (b), d_1 is an input

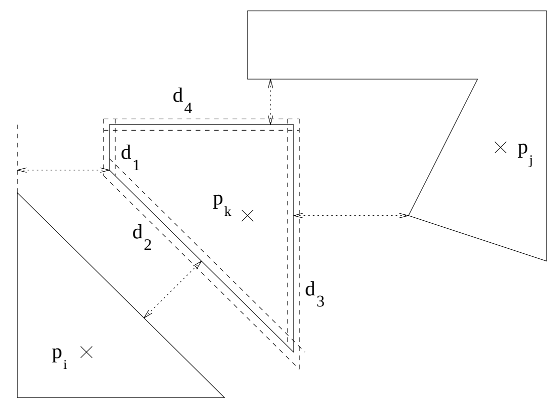

Figure 15: Illustration for Subcase (2.1)

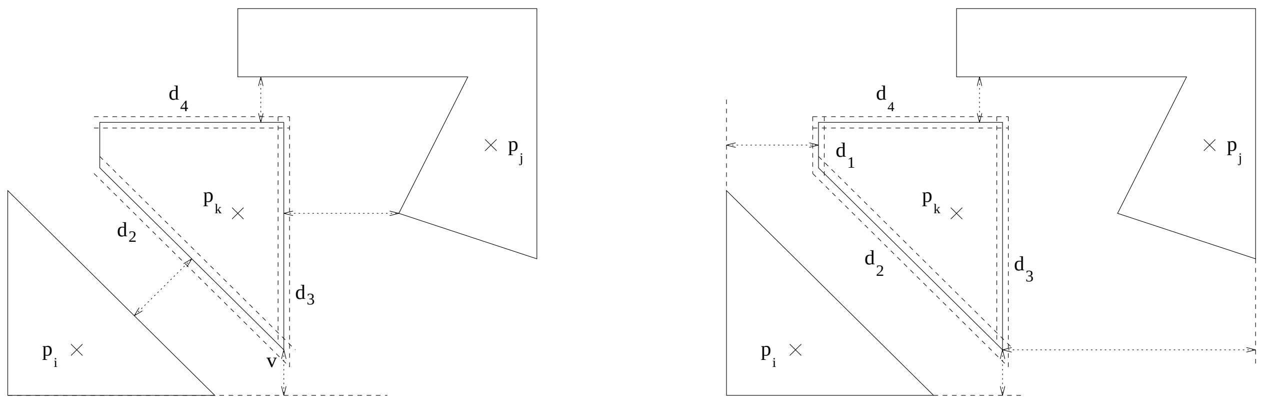

Figure 16: Illustration for Subcase (2.2)

parameter and d_4 an output parameter, while both d_2 and d_3 are input-output parameters.

In Subcase (2.1), the variation of each input or output parameter d yields p_j to span a line segment. Computing this segment is done by considering several cases, depending on whether the variation of d affects the location of an edge of P_k involved in the relation between P_i and P_k (if d is an input parameter) or between P_k and P_j (if d is an output parameter), or the location of a vertex, or the locations of both an edge and a vertex. The analysis of each case is fastidious, but straightforward. When all parameters vary, p_j spans a polygon which is the Minkowski sum of all the segments defined above. W_i^j has constant complexity and is computed in constant time.

In Subcase (2.2), we handle the input-output parameters separately from the other input and output parameters. The variation of the non input-output parameters (the input-output parameters being fixed) leads p_j to scan a polygon computed as W_i^j in Subcase (2.1). The variation of the input-output parameters (the other parameters being fixed) also leads p_j to scan a polygon. Computing this second polygon also requires analyzing multiple cases. W_i^j is the Minkowski sum of the two polygons. It has constant complexity and is computed in constant time.

Finally, let us return to Case (3). It involves the computation of the Minkowski sum of $r_{ij} - 1$ polygons, each having constant complexity. The resulting polygon has complexity $O(r_{ij})$. A classical divide-and-conquer technique computes it in time $O(r_{ij} \log r_{ij})$.