Polyhedral Assembly Partitioning Using Maximally Covered Cells in Arrangements of Convex Polytopes*

Leonidas J. Guibas† Dan Halperin†‡ Hirohisa Hirukawa§
Jean-Claude Latombe† Randall H. Wilson¶

Abstract

We study the following problem: Given a collection A of polyhedral parts in 3D, determine whether there exists a subset S of the parts that can be moved as a rigid body by infinitesimal translation and rotation, without colliding with the rest of the parts, $A \setminus S$. A negative result implies that the object whose constituent parts are the collection A cannot be taken apart with two hands. A positive result, together with the list of movable parts in S and a direction of motion for S, can be used by an assembly sequence planner (it does not, however, give the complete specification of an assembly operation). This problem can be transformed into that of traversing an arrangement of convex polytopes in the space of directions of rigid motions. We identify a special type of cells in that arrangement, which we call the *maximally covered cells*, and we show that it suffices for the problem at hand to consider a representative point in each of these special cells rather than to compute the entire arrangement. Using this observation, we devise an algorithm which is complete (in the sense that it is guaranteed to find a solution if one exists), simple, and improves significantly over the best previously known solutions. We describe an implementation of our algorithm and report experimental results obtained with this implementation.

1 Introduction

In this paper we study an instance of the *assembly partitioning problem* [21]: Given a collection A of non-overlapping polyhedral parts, does there exist an infinitesimal motion (translation and rotation) that can be applied to a subset S of the parts in A, that will

---

*Work on this paper by L.J. Guibas, D. Halperin and J.-C. Latombe has been supported by NSF/ARPA Grant IRI-9306544, and by a grant from the Stanford Integrated Manufacturing Association (SIMA). Work on this paper by L.J. Guibas and D. Halperin has been also supported by NSF Grant CCR-9215219. Work on this paper by R.H. Wilson has been supported by Sandia National Laboratories, Laboratory Directed Research and Development Program, under DOE contract DE-AC04-94AL85000. A Preliminary version of the paper appeared in *Proc. IEEE International Conference on Robotics and Automation*, Nagoya, 1995, pp. 1585–1592.

†Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305.

‡Author's current address: Department of Computer Science, Tel Aviv University, Tel Aviv 69978, ISRAEL.

§Electrotechnical Laboratory, Agency of Industrial Science and Technology, Ministry of International Trade and Industry, 1-1-4 Umezono, Tsukuba 305 JAPAN. Work on this paper was carried out while H. H. was visiting Stanford University.

¶Intelligent Systems and Robotics Center, Sandia National Laboratories, Albuquerque, NM 87185.

move S as a rigid body without colliding with the rest of the parts, $A \setminus S$? A collection of parts is *two-handed* if it can be assembled by a sequence of operations, each of which moves exactly one rigid subassembly relative to the rest of the parts, which do not move during the operation. A negative answer to the above question means that A is not two-handed. It may be the case that A can be assembled by more than two hands, but then the assembly process usually becomes more costly, and indeed most industrial products are two-handed. It may also be the case that A cannot be assembled.

A positive answer to the question, together with a list of the parts in S and a direction of motion for S, can be used in an assembly sequence planner. It does not provide a complete specification for an assembly operation, but it points out a plausible direction of motion. Further computation is required to produce a feasible finite motion of the subassembly S, if one exists. However, infinitesimal motions are attractive in assembly planning because they represent a necessary constraint on the existence of an assembly operation, and furthermore their analysis translates to handling linear constraints, even when allowing rotation (see, e.g., [10],[14],[22]). For more information on assembly planning see, e.g., [11],[21],[23].

In 1988, in his paper "On Planning Assemblies" [17], Natarajan conjectured that "two hands suffice to assemble any composite comprised of convex polyhedra in 3-space". In a surprising result, Snoeyink and Stolfi [19] have recently been able to disprove this conjecture: They gave an example consisting of thirty convex polyhedral parts that cannot be taken apart with two hands. The proof of the validity of the construction relies on a computer program that exhaustively tries every subset of the collection of parts against the rest of the parts for infinitesimal separation (that is, by showing that there is no possible infinitesimal translation and rotation for any division of the parts). Thus their algorithm for checking infinitesimal separability is exponential in the number of parts. This is typical of several existing assembly planning techniques that rely on a "generate-and-test" approach; see, e.g., [12]. We remark that the work by Snoeyink and Stolfi continues a long line of research, whose objective was to construct composites that are interlocked under various types of motions (see, e.g., [3], [6], [17]).

An efficient procedure for the infinitesimal partitioning problem was proposed by Wilson and Matsui [22] who devised a polynomial-time algorithm to solve this problem. Their solution is based on the non-directional blocking graph (NDBG) concept [20] (see also Section 2 below).

In this paper we take a similar approach to that of Wilson and Matsui, but we derive a considerably more efficient algorithm. The problem can be transformed into that of traversing an arrangement of convex polytopes in the space of directions of rigid motions. We identify a special type of cells in that arrangement, which we call the *maximally covered cells*, and we show that it suffices for the problem at hand to consider a representative point in each of these special cells rather than to compute the entire arrangement. We devise an efficient algorithm to access these cells directly and thus obtain a solution to the partitioning problem that improves considerably over the best previously known algorithms and is at the same time complete, namely, if there exists a solution our algorithm will find it.

Our method is not restricted in dimension and it can be applied to various problems involving any number of degrees of freedom. It does, however, rely on the fact that the number of degrees of freedom is not too large. For infinitesimal rigid motions in threedimensional space this number is five.

We have implemented the algorithm and we present details of the implementation and experimental results.

The rest of the paper is organized as follows. In Section 2 we supply more background which is needed to explain our algorithm. We expose the ideas underlying our new approach in Section 3. These are then used in the algorithm which we present in Section 4, where we also analyze its running time. In Section 5 we give details of our implementation of the algorithm and then we present experimental results produced with the implementation. Some concluding remarks and open problems are given in Section 6.

2 Preliminaries

In this section we review related results and background material necessary to understand our approach. In Subsection 2.1 we discuss contact analysis and describe the approach to infinitesimal partitioning by Wilson and Matsui. In Subsection 2.2 we survey some combinatorial results concerning arrangements in general and arrangements of convex polytopes in particular.

2.1 Blocking Graphs and Contact Analysis

The starting point of our new approach is similar to that of Wilson and Matsui [22]. In this section we briefly review some of the ingredients of their analysis that are needed here as well. We refer the reader to their paper [22] for more details.

The *non-directional blocking graph* (NDBG, for short) is a subdivision of the space of all allowable motions of separation into a finite number of cells such that inside any single cell the blocking relation between all pairs of parts is fixed. These blocking relations for a fixed motion (and hence for a cell in the subdivision) are gathered in a directed graph, the directional blocking graph (DBG), whose nodes $v_1, v_2, \ldots, v_n$ represent the n parts $P_1, P_2, \ldots, P_n$ in the assembly, and a directed arc from node v_i to node v_j means that the part P_i will collide with the part P_j if this motion is applied to P_i. A partitioning of the full assembly into two subassemblies under a specific motion is possible if and only if the DBG for that motion is not *strongly connected.*[1] For more details on the NDBG concept, see [20],[21].

Contacts between polyhedra consist of points, line segments, and planar polygonal contacts. The infinitesimal motion constraints arising from line segments and planar polygonal contacts can be reduced to equivalent finite sets of point contact constraint. For example, the contact between a convex edge e of one polyhedron and a face f of another polyhedron is equivalent to two point-plane constraints, one at each end of the intersection segment of e and f. (For more details, see [10],[18],[22].) We therefore concentrate on point contact constraints.

An infinitesimal motion ΔX of a polyhedron P_i can be described as a vector with three

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

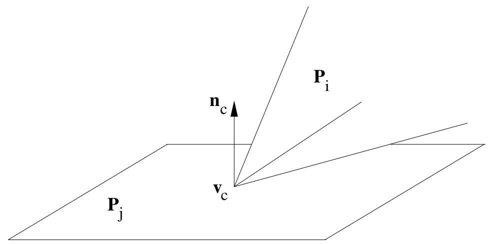


Figure 1: A point–plane contact between two polyhedra

parameters for translation and three for rotation:

$$\Delta X = (\Delta x, \Delta y, \Delta z, \Omega_x, \Omega_y, \Omega_z) \; ,$$

where Ω_x, Ω_y, and Ω_z are the components of the angular velocity around the x, y, and z axes, respectively. Now, consider the point-plane contact c between the vertex v_c of a polyhedron P_i and the face of a polyhedron P_j with outward normal n_c (see Figure 1). The infinitesimal motion ΔX causes the vertex v_c of P_i to undergo a translation $J_c \Delta X$, where J_c is the 3×6 Jacobian matrix that relates the differential motion of P_i to the motion of v_c.

An infinitesimal motion ΔX such that $n_c^T J_c \Delta X < 0$ will cause the part P_i to penetrate into P_j at the contact point v_c. Therefore this point-plane contact will allow only infinitesimal motions ΔX such that $n_c^T J_c \Delta X \geq 0$, and when this inequality holds we say that ΔX *obeys* the contact c. Equality (i.e., when $n_c^T J_c \Delta X = 0$) means that the infinitesimal motion causes a sliding of one polyhedron relative to the other at the contact. The set of infinitesimal motions allowed by all the point constraints involving one polyhedron P_i is the intersection of the motions that obey each constraint individually.

Since two motions ΔX_1 and ΔX_2, with $\Delta X_1 = s \cdot \Delta X_2$ for a positive scalar s, differ only in velocity, we restrict ourselves to motions ΔX such that $|\Delta X| = 1$. Hence our motions are all represented by points on the unit sphere $\mathcal{S}^5$ in six-dimensional space. Each point contact defines a hyperplane that divides $\mathcal{S}^5$ in half, and determines a closed hemisphere (whose boundary is a great circle on $\mathcal{S}^5$) of infinitesimal motions that obey that specific contact.

For convenience, we choose a hyperplane Π tangent to $\mathcal{S}^5$ and centrally project the great circles introduced by the constraints on $\mathcal{S}^5$, onto that hyperplane. Now our constraints are transformed into closed halfspaces in R^5. This way we have only projected a hemisphere of $\mathcal{S}^5$ onto the special hyperplane Π. However, it is easily verified that with a little caution we do not lose any information by this transformation. Let Π' be the hyperplane parallel to Π and passing through the origin. We choose Π such that Π' does not cross vertices on $\mathcal{S}^5$ (i.e., points where 5 constraint hyperplanes or more meet), and then the projected hemisphere contains all the information necessary to find a partitioning if one exists, because the other hemisphere has a symmetric subdivision on it. In other words, if a point p on the sphere represents a motion that will separate S from $A \setminus S$, then the antipodal point of p will represent the separation of the same two subassemblies in precisely the opposite direction.

2.2 Arrangements of Polytopes

Let L be a given collection of n lines in the plane. We denote by $\mathcal{A}(L)$ the *arrangement* of L, i.e., the decomposition of the plane into vertices, edges, and faces, induced by the lines in L. A vertex of $\mathcal{A}(L)$ is an intersection point of two lines, an edge is a maximal connected relatively open portion of a line that does not meet any vertex, and a face is a maximal connected open region of the plane not meeting any edge or vertex. Similarly, an arrangement of hyperplanes in d-dimensional space is the subdivision of that space into relatively open cells of dimensions $0, 1, \ldots, d$, each being a maximal connected set contained in the intersection of a fixed subcollection of the hyperplanes and not meeting any other hyperplane.

Arrangements of hyperplanes have been studied extensively in combinatorial and computational geometry [4]. The combinatorial complexity of an arrangement is defined to be the overall number of cells of various dimensions in the arrangement. It is known that the maximum combinatorial complexity of an arrangement of n hyperplanes in d-dimensional space, for d a small constant, is $O(n^d)$ and that this bound is tight in the worst case. Furthermore, there is an algorithm with running time $O(n^d)$ to compute all the cells in the arrangement and their adjacency relations—see [4, Chapter 7], [5]. Wilson and Matsui use this algorithm in their solution of the partitioning problem [22].

In our solution we look at a slightly different arrangement: *an arrangement of convex polytopes*. Let Q be a collection of K convex polytopes in d-dimensional space, with a total of N facets. The arrangement $\mathcal{A}(Q)$ is the subdivision of d-space induced by the polytopes in Q. An obvious upper bound on the complexity of such an arrangement is $O(N^d)$—we extend each facet of every polytope in Q into a full hyperplane. However, Aronov et al. [1] have shown that the actual complexity of such an arrangement is in general much lower. They presented a bound $O(N^{\lfloor \frac{d}{2} \rfloor} K^{\lceil \frac{d}{2} \rceil})$ and showed that it is tight in the worst case. We note that we are not aware of an algorithm that takes advantage of this reduced complexity in arbitrary dimension (this is trivial to do in two-dimensional space).

One way of achieving efficiency when computing with arrangements has been to focus on portions of an arrangement that are relevant to a specific problem instead of computing the entire arrangement. A typical example of such saving is in robot motion planning when one is often interested in a single cell in the arrangement of *constraint surfaces* (i.e., boundaries of configuration space obstacles); see [8], [9]. In the next section we present a special substructure in arrangements of convex polytopes that is relevant to the partitioning problem.

3 Maximally Covered Cells

Our novel and more efficient approach is based on several observations that we explain in this section.

First, we group the contact constraints for each ordered pair of parts (P_i, P_j). We denote the collection of closed halfspaces that represent constraints on the motion of P_i relative to P_j by Q_{ij}. It follows from the discussion in the previous section, that the intersection of all the constraints in Q_{ij} is the convex polytope representing the motion directions in which P_i will *not* collide into P_j. With a slight abuse of notation we will refer to Q_{ij} both as a collection of closed halfspaces and as the convex polytope that is the intersection of these halfspaces.

Next, we consider the arrangement in 5-space induced by all the constraints Q_{ij} for all the ordered pairs of parts in our assembly, which is an arrangement of convex polytopes as described in Subsection 2.2. All the points inside each cell of any dimension in this arrangement are contained in the same set of polytopes. Therefore, the blocking relation, and hence the blocking graph (the DBG, see Subsection 2.1) inside a single cell is fixed. The crux of our new technique is the observation that we need to consider only some of the cells in this arrangement and that there is a way to access these cells directly without computing the entire arrangement.

Since our approach seems to be applicable in other settings as well, we describe it more generally from this point. (In the examples that we present below, we will refer to arrangements in two- and five-dimensional space.) Let Q denote the collection of K polytopes in d-dimensional space (in our application, these are the at most $n(n-1)$ polytopes Q_{ij} in 5-dimensional space, where n is the number of parts in the assembly). Let $\mathcal{A}(Q)$ denote the arrangement in d-space induced by Q, namely the collection of relatively open cells of dimensions $0, 1, \ldots, d$ induced by the boundaries of all the polytopes in Q. For example, a d-dimensional cell in the arrangement is a maximal portion of d-space not meeting any boundary of any polytope in Q. Let N be the total number of facets in all the polytopes together (in our application, N is the overall number of constraint halfspaces).

To discuss *maximally covered cells*, we will use the following definition

Definition 3.1 *The* **covering set** *of a point p in R^d is the subset of polytopes in Q that contain p.*

Informally, a cell is maximally covered, if it is covered by more polytopes than its immediate neighbors. In other words, a cell C is maximally covered if every point outside the closure $\overline{C}$ of the cell and infinitesimally close to $\overline{C}$ is covered by only a proper subset of the polytopes that cover the cell.

Since we deal with *closed polytopes* it is equivalent to require that any point on the relative boundary of the cell will have the same covering set as its interior. Since a maximally covered cell and its relative boundary have the same covering set, dealing with both the cell and each of its bounding faces is redundant. Hence, we require that the closure of a maximally covered cell is a maximal connected region of d-space with that covering set. We summarize the above discussion in the following

Definition 3.2 *A cell C in the arrangement $\mathcal{A}(Q)$ is called a* **maximally covered cell** *if the covering set of any point on the relative boundary of C is the same as the covering set of its interior, and the closure of C is a maximal connected region of d-space with that covering set.*

To get a feeling for what maximally covered cells are, consider Figure 2, which depicts a collection of convex polygons in the plane. There are four maximally covered cells in the arrangement defined by the six polygons in the figure. Three of the maximally covered cells are two-dimensional cells and they are shaded in the figure: the shaded hexagon is covered by one polygon and every point outside the hexagon and in its immediate neighborhood

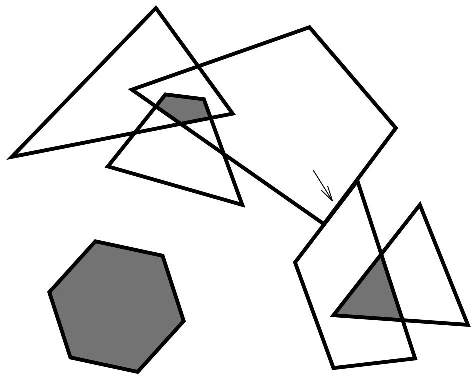


Figure 2: Four maximally covered cells in a planar arrangement

is not covered by any polygon. The shaded triangle is covered by two polygons, and the shaded pentagon is covered by three polygons. There is also one maximally covered cell that is one-dimensional. This is the segment of intersection between two polygon boundaries pointed to by the arrow. Every point along this intersection segment is covered by the two polygons (recall that we deal with closed polytopes) and every point in the neighborhood of this segment but not lying on it is either covered by a single polygon or by none.

Back to the original problem, we argue that if there is a solution (S, p) to our partitioning problem, namely there is a subset S of the parts in A, and a direction p of motion that will separate S infinitesimally from the rest of the parts, then there is a solution (S, p') such that p' is a point inside a maximally covered cell. To see why this claim is true, consider a partitioning (S, p) such that p is not inside a maximally covered cell. Let C_x denote the cell of the arrangement containing the point x. Since C_p is not maximally covered, there is a point q on the boundary of C_p that is covered by more polytopes than its interior. Since we deal with closed polytopes, by moving to the boundary of C_p we cannot get out of any polytope covering the interior of C_p. We move to the point q on the boundary. If C_q is not maximally covered, we can either move to its boundary or inside any of the cells having C_q on its boundary. We keep on moving as long as we can augment the covering set without losing any of the polytopes which are already in the current covering set. Thus the covering set of p' contains the covering set of p. It follows that the edges of the DBG at p' are a subset of the edges of the DBG at p, namely the DBG at p' is less constrained than the DBG at p'. Since the DBG at p is already not strongly connected, (S, p') is a valid partitioning.

This process will stop only when we have reached a maximally covered cell, and since we never leave a polytope we are in, the process is guaranteed to stop after a finite number of steps.

What do we gain from the above observation? We show next that the number of maximally covered cells is potentially smaller than the overall number of cells in the entire arrangement $\mathcal{A}(Q)$.

Theorem 3.3 *The maximum number of maximally covered cells in $\mathcal{A}(Q)$ is $O(K^d)$.*

Proof: We choose an arbitrary direction in R^d, say the X_d direction (the positive direction along the X_d coordinate), and we look for the minimum point in the X_d direction in each maximally covered cell. We assume here, without loss of generality, that every bounded cell in the arrangement $\mathcal{A}(Q)$ has a unique minimum point in the X_d direction; otherwise we can rotate all the input constraints slightly to guarantee this property. The minimum point inside a cell is a vertex v of the arrangement where at least d facets of polytopes in Q meet.

There may be more than d facets meeting at this point, in which case we choose exactly d facets lying in d distinct hyperplanes and we denote by Q' the set of up to d polytopes that contain these facets on their boundary. The point v is clearly the minimum point in the direction X_d in the intersection of the polytopes in Q', and this intersection is a unique convex polytope. Hence we can charge the maximally covered cell to this intersection polytope. There are at most $\binom{K}{i}$ polytopes that are the intersection of i polytopes in Q. The sum $\Sigma_{i=1}^{d}\binom{K}{i} = O(K^d)$ is therefore also a bound on the number of maximally covered *bounded* cells.

Next we bound the number of *unbounded* maximally covered cells. Let B be a hyperbox that contains all the vertices of the arrangement $\mathcal{A}(Q)$ in its interior. Every unbounded cell U in $\mathcal{A}(Q)$ now has a representative cell inside B, namely the cell $U \cap B$. If U is maximally covered then so is $U \cap B$. We repeat the argument that we have used for bounded cells, but look only for polytopes that are the intersection of B and up to $d-1$ polytopes from Q. The number of such intersection polytopes is $O(K^{d-1})$, which is a bound on the number of unbounded maximally covered cells, and is asymptotically dominated by the bound for bounded maximally covered cells. □

For infinitesimal separation, Wilson and Matsui consider the entire arrangement induced by the hyperplanes supporting the facets of polytopes in Q, therefore they consider $\Theta(N^5)$ cells. The overall number of cells in the arrangement $\mathcal{A}(Q)$ is $\Theta(N^2K^3)$ in the worst case [1]. The number of cells that our algorithm examines is $\Theta(K^5)$. Note that K is never bigger than N; in practical situations K is often much smaller than N.

Interestingly, the discussion above shows that the number of directions that need to be checked depends on the number of parts and not on the shape of the parts or the shape of the contact areas between the parts. In the next section we show how we directly access these cells without computing the entire arrangement.

4 The Algorithm

As explained above, we need only examine maximally covered cells in the arrangement $\mathcal{A}(Q)$ in order to find a possible solution to the partitioning problem. Furthermore, we only need one sample point inside each cell, because all the points inside any single cell in the arrangement represent a fixed blocking graph (and hence a fixed set of blocking relations).

4.1 Finding Representative Points

The idea behind the algorithm is the same as in the proof of Theorem 3.3, namely, the minimal vertex of any maximally covered cell in a fixed direction D is a minimal vertex in direction D of the convex polytope which is the result of intersecting at most d polytopes from the given collection Q.

We choose the positive direction along the coordinate X_d as the fixed direction. (This choice is arbitrary.) Our algorithm therefore looks for the X_d minimal vertex of the intersection of any set of up to d polytopes in Q. Since we are only interested in a single representative point, we do not have to compute the intersection of the polytopes. Instead, we use linear programming (LP, for short), where the constraints are the halfspaces determining the polytopes, and the objective function we wish to minimize is X_d.

Remark. The set of points produced by this approach contains all *candidate* representative directions. The produced points are not necessarily inside maximally covered cells. Whether a cell is maximally covered cannot be determined by checking just one point inside it. We pay for the simplicity of obtaining all the candidate directions by producing possibly spurious points. A central open problem that our work raises is whether one can avoid producing spurious cells.

4.2 Overall Algorithm

Even if a representative point belongs to a maximally covered cell, we still need to check if it actually represents a feasible direction of collision-free infinitesimal motion. We do this by constructing the directional blocking graph (DBG) at each direction and checking it for strong connectivity.

Since we do not construct the entire arrangement in the space of possible motion directions, we cannot use adjacency relations between cells of the arrangement to incrementally update the DBG as we move from one cell to another, as in [22]. Rather, we construct the DBG from scratch at each point. To do this efficiently we build, for every polytope in our collection, a data structure that will enable us to determine in logarithmic time whether a point is contained in the polytope or not. If a given point g is *not* contained in a given polytope Q_{ij} this means that we have to put an arc directed from the node v_i to the node v_j (corresponding to the parts P_i and P_j respectively) in the blocking graph. Otherwise, there is no arc between the two nodes.

To summarize, here is a sketch of the algorithm for the partitioning problem with infinitesimal motions where the space of directions has dimension d. The input is a set of K constraint sets Q_{ij} for each pair of ordered parts (P_i, P_j) in contact, and the output is a subset of parts and a direction to move it, if one exists, or INTERLOCKED otherwise. In the variable H we collect all the constraints defining a given subset R of the polytopes of one subproblem. The variable e will contain the answer point from the LP program or NULL if there is no feasible solution. The function $DBG(e)$ returns the DBG at the point e. The procedure $LP(H, X_d \downarrow)$ runs a linear program for the constraint halfspaces H and the objective function X_d.

1. for $i = 1, \ldots, d$

    2. for each subset R of input polytopes such that $|R| = i$

        3. $H \leftarrow \bigcup_{r \in R}$(Halfspaces in r)
        4. $e \leftarrow LP(H, X_d \downarrow)$
        5. If $e =$ NULL goto 2, else
        6. If DBG(e) is not strongly connected

            7. Output e and movable subassembly
            8. exit

9. Report "INTERLOCKED"

A preliminary step, omitted in the algorithm description above, derives the point contacts that are used to define the input hyperplane constraints. For that purpose, we use the algorithm devised by Hirukawa et al. [10]; see there for details.

Lines 1–8 of the algorithm should be run twice. Once with subsets of polytopes from Q only, and another time where each subset includes the bounding hyperbox B (see proof of Theorem 3.3).

Note that the set of candidate points produced may change if we choose a different direction D along which we look for minimal vertices. However, the algorithm is guaranteed to produce a point inside each maximally covered cell.

4.3 Complexity Analysis

The main loop of our algorithm performs $O(K^d)$ iterations, and in each iteration the major steps performed are: (i) solving a linear programming problem, (ii) computing a DBG, and (iii) checking the DBG for strong connectivity. In this subsection we analyze the worst-case running time of the algorithm using the best known procedures for each step. We remark that our implementation, with which the experimental results reported below were obtained, uses different procedures to implement steps (i) and (ii) for reasons of software availability and simplicity.

Lemma 4.1 *Solving all the linear programs together takes time $O(K^{d-1}N)$.*

Proof: We have to solve $O(K^d)$ linear programming problems. Recall that we assume that the dimension d is a small constant. We will focus on the problems that arise when we take all possible combinations of exactly d polytopes; this will dominate the running time of solving all the other LP problems. We use Megiddo's algorithm that runs in time linear in the number of constraints [13]. Thus asymptotically the overall running time for solving all the LP problems equals the overall number of constraints given to all these problems. Let us fix one constraint h, belonging to one polytope T (i.e., to one set of constraints). This constraint will participate in $O(K^{d-1})$ LP problems, which is all the possibilities to choose the other $d-1$ polytopes that are not T. Since there are N constraints in all the polytopes together, the bound follows. $\square$

Computing the DBG for one fixed direction can be reduced to a collection of polytope membership queries as explained above. For one polytope with m facets in dimension d, the preprocessing takes $O(m^{\lfloor \frac{d}{2} \rfloor + \epsilon})$ expected time, and allows for query time $O(\log m)$ [15]. Let m_i be the number of facets of the ith polytope in the given collection. The total preprocessing time is

$$\sum_{i=1}^{K} O(m_i^{\lfloor \frac{d}{2} \rfloor + \epsilon}) = O(N^{\lfloor \frac{d}{2} \rfloor + \epsilon}) ,$$

and the overall query time is

$$O(K^d) \sum_{i=1}^{K} O(\log m_i) = O(K^{d+1} \log N) .$$

The time to check for strong connectivity is linear in the number of nodes and arcs in the DBG and hence it is dominated by the construction time of the DBG.

We summarize

Theorem 4.2 *The partitioning of a polyhedral assembly under infinitesimal motion, where any direction of motion is defined by d parameters, can be computed in $O(K^{d-1}N + K^{d+1} \log N)$ time after preprocessing in $O(N^{\lfloor \frac{d}{2} \rfloor + \epsilon})$ expected time, where K is the number of ordered pairs of polyhedral parts in contact in the assembly (thus K is at most $n(n-1)$ for an assembly with n parts), and N is the total number of the point contact constraints among the parts.*

Rephrased in our original setting, namely for partitioning under infinitesimal rigid motions we have

Theorem 4.2$'$ *The partitioning of a polyhedral assembly under infinitesimal rigid motion (translation and rotation), can be computed in $O(K^4 N + K^6 \log N)$ time after preprocessing in $O(N^{2+\epsilon})$ expected time, where K is the number of ordered pairs of polyhedral parts in contact in the assembly (thus K is at most $n(n-1)$ for an assembly with n parts), and N is the total number of the point contact constraints among the parts.*

As for the storage requirements. The data structures for polytope membership queries require an expected total of $O(N^{\lfloor \frac{d}{2} \rfloor + \epsilon})$ space. All other procedures require $O(N)$ space. Every linear program handles at most N constraints and hence requires $O(N)$ storage. Checking a graph with n nodes and at most K arcs for strong connectivity requires $O(n+K)$ space. It is obvious that $K \geq N$. We also assume that $K \geq n-1$—otherwise there is at least one part P_i that does not touch any other part in the assembly and the partitioning problem has a trivial solution. In summary the algorithm requires expected $O(N^{\lfloor \frac{d}{2} \rfloor + \epsilon})$ storage, for $d \geq 2$.

5 Experimental Results

5.1 Implementation

The algorithm described above has been implemented, and here we describe how this has been done. The implemented program consists of three modules. The contact constraints are computed by the first module, the representative points of the maximally covered cells are found by the second module, and the DBG is constructed at each representative point and checked for strong connectivity by the third module.

The contact constraints are computed by the algorithm described in [10]. Let us very briefly review this algorithm. The contact constraints between polyhedra for infinitesimal motions are equivalent in general to those at a finite number of point contacts. So the algorithm finds all pairs of vertices and planar faces in contact and those of pairs of edges in contact. Then the contact constraint is computed at each point by finding separating planes between the two neighborhoods of the point. An example of the constraints found by the algorithm is shown in Figure 3. In the figure, each arrow shows the normal n_c as

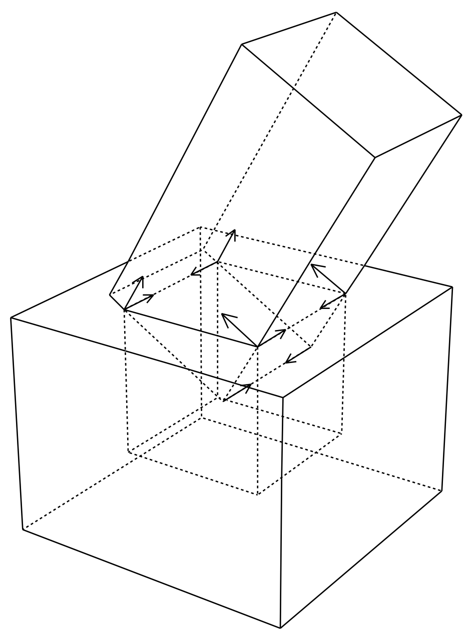


Figure 3: Example of contact constraints

in Subsection 2.1. The input of the first module is geometric models of assemblies, and the output is linear inequalities describing the contact constraints for all pairs of parts in contact. The first module is implemented in CommonLisp with object-oriented primitives on a *Sun Sparc Station* 1+.

As to the second module of the implementation, the representative points are found by linear programming. This linear programming part is solved using the MINOS package [16]. Here, a random hyperplane is used for the central projection (the hyperplane Π in Subsection 2.1), whose parallel hyperplane passing through the origin should not cross the vertices on $\mathcal{S}^5$. In the MINOS package, this hyperplane is given as one of the linear constraints to the LP problem. That is, it is written as $1 \leq N_r^T \Delta X$, where N_r is a 6×1

random vector. The input of the second module is the output of the first module. The output of the second module is a superset of representative points of the maximally covered cells. The second module is implemented in *C* and *Fortran* on a *DEC Station* 5000/240.

Currently the third module is an implementation of a naive algorithm querying the polytope membership by checking the inequalities one by one, because, as it turns out, the actual number of representative points is not too large in most cases. The strong connectivity of the DBG is checked by an algorithm due to Tarjan and Hopcroft [2]. The input of the third module is the output of the second module, and the final output is the separability of the given assembly. This module is implemented in the same environment as the first module.

5.2 Examples

5.2.1 Preliminary Examples

The first example, shown in Figure 4, is a puzzle consisting of six parts, inducing 352 point contacts, and having 12 ordered pairs of parts in contact, i.e., $K = 24$ and $N = 352$, using the notation introduced in Section 3. Five parts are identical and each of them has two

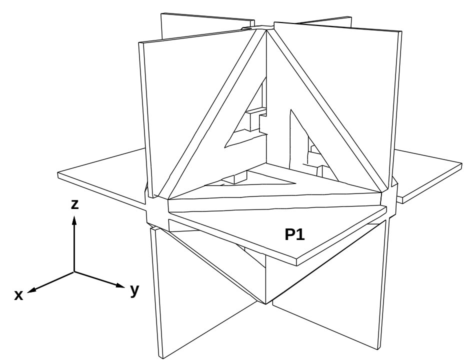


Figure 4: A puzzle consisting of six parts, with $K = 24$ and $N = 352$

small tabs at the bottom (see Figure 5 for an illustration). The remaining part has only one tab. For this example we solve the infinitesimal partitioning problem under translation only, so $d = 2$ in this case. Some of the contact constraints are illustrated in Figure 5, where each arrow corresponds to the outward facing normal of the plane in the point contact (see Section 2). Note that some of the constraints in the figure are redundant and we can eliminate them in a preprocessing stage (See [10] for details). The representative points in the maximally covered cells found by the algorithm are $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$, where the coordinate system is shown in Figure 4. The DBGs corresponding to these points are illustrated in Figure 6, where P1 is the part with one tab. The DBGs are strongly connected for the directions $(1, 0, 0)$ and $(0, 0, 1)$, and P1 is separable from the other parts by infinitesimal translation in the direction $(0, 1, 0)$.

The object of the next example, depicted in Figure 7, cannot be separated without rotation, so we need to consider the five-dimensional space here. In this example there are three parts, $K = 6$ and $N = 140$. Examples of the constraints are illustrated by arrows in Figure 8. The representative points in the maximally covered cells for this example

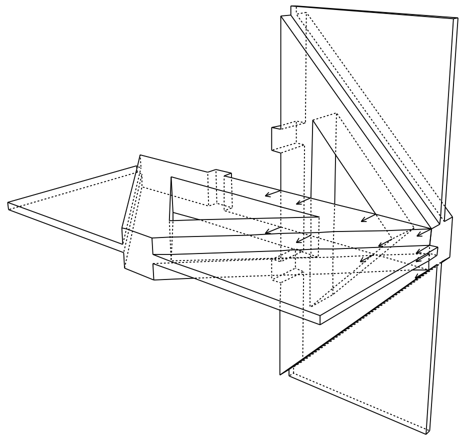


Figure 5: Two parts in contact and the constraints between them

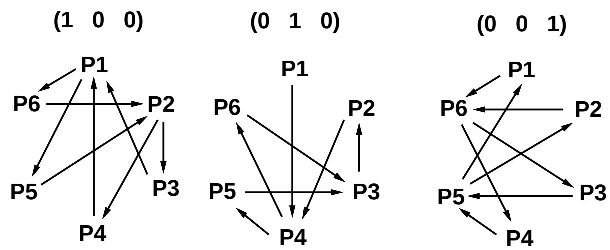


Figure 6: DBGs for the puzzle example

are $(1,0,0,0,0,0)$, $(0,1,0,0,0,0)$, $(0,0,1,0,0,0)$ and $(0,0,0,0,0,1)$; see Figure 9, where P1 is the biggest part including the big horizontal plate, P3 is the smallest part, and the coordinate system is located at the center of the upper face of P1. Note that the location of the origin should be specified to represent an infinitesimal motion including the rotation. The DBGs are strongly connected for the first, second and third directions, none of which involves rotation, and P2 is separable from P1 and P3 by infinitesimal motion along the fourth direction, which involves rotation. Figure 10 shows a rotational motion in that direction.

5.2.2 Convex Objects in Contact

The next example, given by Snoeyink and Stolfi [19], consists of six identical tetrahedra in contact and shown in Figure 11. They proved that no proper subset is separable by infinitesimal translation. We revisit this example, confirm their result with our program, and show that if we allow general infinitesimal motion (i.e., including rotation), then this construction can be partitioned. In this example, $K = 24$ and $N = 96$.

In the case of translation only (where $d = 2$), 22 representative points are found by the algorithm, and all of the corresponding DBGs are confirmed to be strongly connected. In the case of rigid motions (where $d = 5$), 190 representative points are found, and only 118 of the corresponding DBGs are strongly connected. That is, some proper subset is separable in 72 selected directions of infinitesimal motion with rotation. Examples of the DBGs for $d = 2$ and $d = 5$ are shown in Figure 12. The DBG for $d = 2$ is strongly connected, and

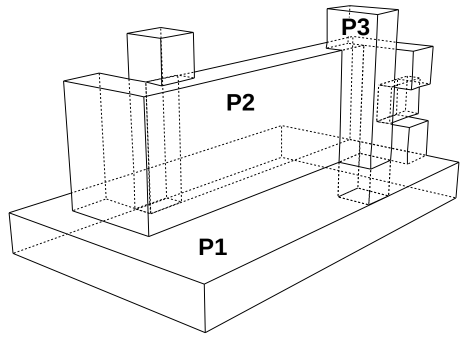


Figure 7: An assembly consisting of three parts, with $K = 6$ and $N = 140$

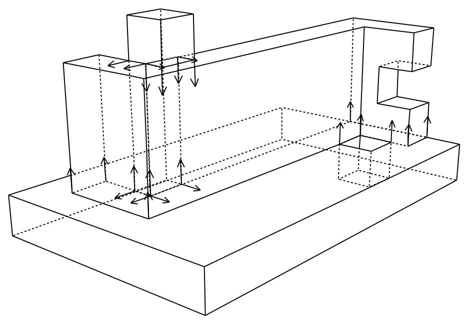


Figure 8: Two parts P1 and P2 in contact and the constraints between them

that for $d = 5$ is not. In the latter case, Part P2 is separable by the infinitesimal motion shown in the figure, where the direction of the motion is represented in the coordinates in Figure 11, whose origin is located at the center point of the set of tetrahedra.

The next example, also given by Snoeyink and Stolfi [19], consists of thirty convex polyhedral parts in contact. Each part is in contact with 10 parts, and so $K = 300$. Each contact region is a quadrangle, thus $N = 1,200$. Unfortunately, the complexity of our algorithm for this example with $d = 5$ seems to be too high for solving in reasonable time without parallelization, because of the large K. So our algorithm is applied to reconfirm that any proper subset is not separable by any infinitesimal translation. Then, representative points are found by our algorithm, and all of the corresponding DBGs are confirmed to be strongly connected.

5.2.3 Industrial Example

Here we consider the application of our algorithm to an industrial example. The shapes of industrial parts are relatively complicated in general, so N, the number of the constraints between parts, tends to be relatively large. But this is not an impediment for our algorithm, since the time complexity of our algorithm is linear in N.

Our algorithm is applied to an engine of a model-aircraft whose parts are shown in Figure 13. Figure 14 shows the assembled engine. In this example, the number of the parts is 12, the total number of their faces is $1,066$, $K = 24$, and $N = 1,112$. Note that the cylindrical surfaces are approximated by polygonal ones, and that K is not much larger

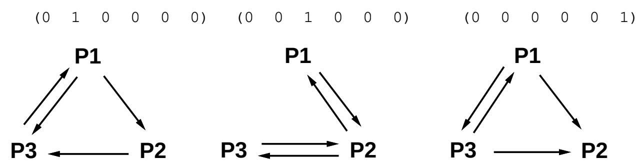


Figure 9: DBGs for the example with three parts

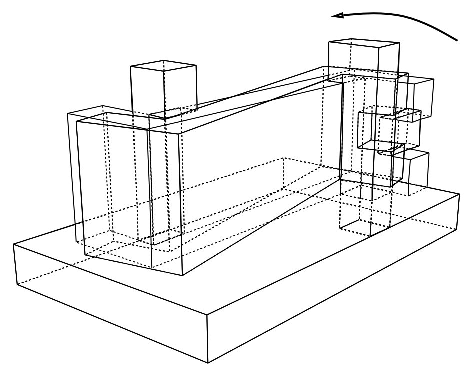


Figure 10: Possible disassembly rotational motion

than the number of the parts.

The case $d = 2$ is summarized in Table 1. In the case of $d = 5$, six representative points are found, and none of the DBGs is strongly connected. One of the DBGs, corresponding to an infinitesimal rotation, and another, corresponding to an infinitesimal translation are shown in Figure 15.

Table 1 summarizes the CPU time in seconds to solve the examples given in the paper. The CPU time is given for the three modules of the implementation separately, the first for computing the constraints between the parts, the second for finding the representative points, and the third for constructing the DBGs and checking them for strong connectivity. The first and third modules were run on *Sparc Station* 1+ (17.5MIPS), and the second one on *DEC* 5000/240 (42.9MIPS). In the table, the time for constructing the DBGs and checking for strong connectivity is very long for the Snoeyink-Stolfi example with thirty parts, since our current implementation of the polytope membership queries is naive.

This example also points to some limitations of our formulation from a practical point of view and suggests open problems for further study. Briefly the two major problems are: (i) infinitesimal motions may not lead to a full separation of subassemblies, because the separability is checked locally under the linearized contact constraints, and (ii) approximating curved objects by polyhedra may lead to false results, because extra contact constraints are added by the polyhedral approximation.

6 Conclusions

We have presented a new, simple and efficient approach to the partitioning problem of polyhedral assemblies under general infinitesimal motions. The algorithm improves considerably over the best previously known algorithms for this problem (for a comparison of the

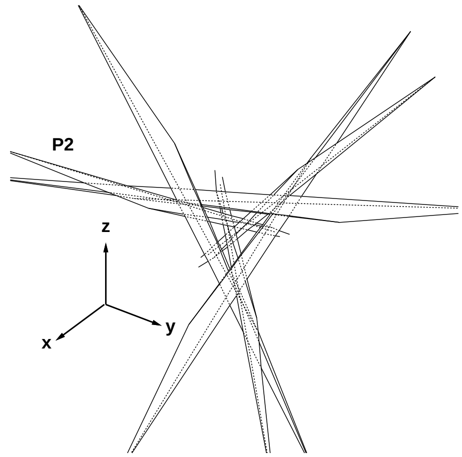


Figure 11: Six tetrahedra in contact [19]

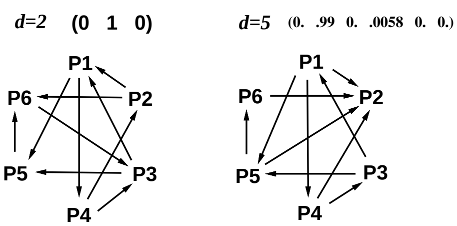


Figure 12: Examples of the DBGs for the six tetrahedra

number of cells checked see the last paragraph of Section 4), and appears to perform well in practice. The new solution seems to be of independent interest; in [7] we have shown how it can be used to solve a different problem arising in the context of object recognition. Are there other applications where maximally covered cells in arrangements of convex polytopes may be of use?

Our algorithm was implemented and we have reported experimental results. Currently, our main goal is to further improve the running time of our program so that it can handle larger and more difficult examples. We are also interested in extending our analysis and implementation to parts that are not necessarily polyhedral, such as spheres and cylinders.

Acknowledgment

The authors thank Michael A. Saunders for letting us use the MINOS package, and Rhea Tombropoulos for her assistance with the linear programming code. We thank Jack Snoeyink and Jorge Stolfi for valuable discussions concerning the contents of the paper. We also thank Jack Snoeyink for supplying us with data of the constructions in [19].

References

[1] B. Aronov, M. Bern and D. Eppstein, Arrangements of polytopes, *manuscript*, 1991.

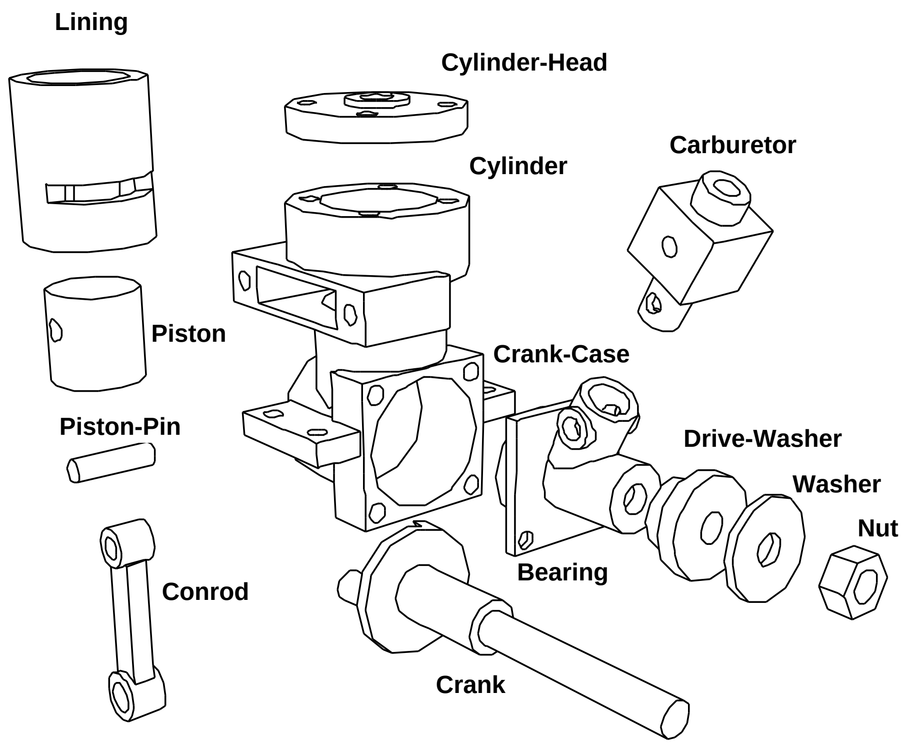


Figure 13: Parts of a model-aircraft engine

[2] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge MA, 1990.

[3] R. Dawson, On removing a ball without disturbing the others, *Mathematics Magazine* **57** no. 1 (1984), pp. 27–30.

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

[5] H. Edelsbrunner, R. Seidel and M. Sharir, On the zone theorem for hyperplane arrangements, SIAM J. Comput. **22** (1993), pp. 418–429.

[6] L. Fejes Toth and A. Heppes, Uber stabile Körpersysteme, *Compositio Mathematica* **15** no. 2 (1963), pp. 119–126.

[7] L. J. Guibas, D. Halperin, H. Hirukawa, J.-C. Latombe, and R. H. Wilson, A simple and efficient procedure for polyhedral assembly partitioning under infinitesimal motions, *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 2553–2560, 1995.

[8] L.J. Guibas, M. Sharir and S. Sifrony, On the general motion planning problem with two degrees of freedom, *Discrete Comput. Geom.* **4** (1989), 491–521.

[9] D. Halperin and M. Sharir, Almost tight upper bounds for the single cell and zone problems in three dimensions, *Discrete Comput. Geom.* **14** (1995), pp. 385–410.

[10] H. Hirukawa, T. Matsui and K. Takase, Automatic determination of possible velocity and applicable force of frictionless objects in contact, *IEEE Trans. Robotics and Automation* **10** no. 3 (1994), pp. 309–322.

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

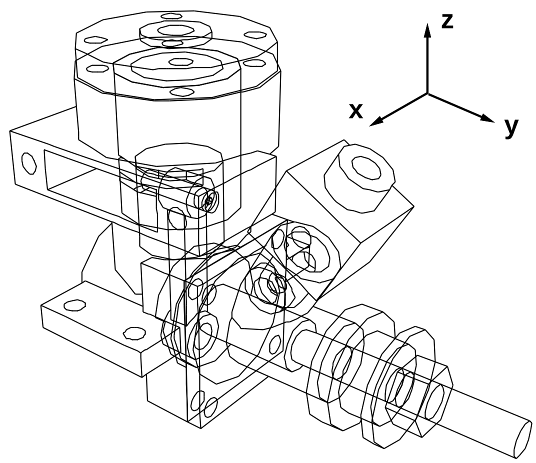
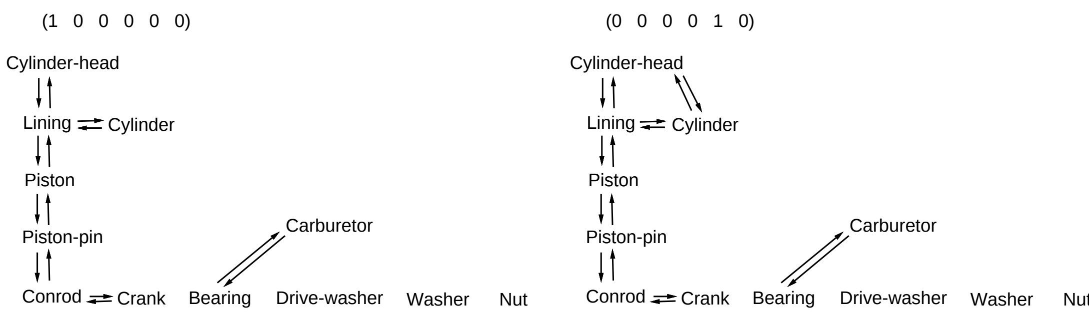


Figure 14: Assembled engine



(a) (b)

Figure 15: Examples of the DBG for $d = 5$

[12] L.S. Homem de Mello and A.C. Sanderson, A correct and complete algorithm for the generation of mechanical assembly sequences, *IEEE Trans. Robotics and Automation* **7** no. 2 (1991), pp. 228–240.

[13] N. Megiddo, Linear programming in linear time when the dimension is fixed, J. ACM **31** (1984), pp. 114-127.

[14] R.S. Mattikalli and P.K. Khosla, Motion constraints from contact geometry: Representation and analysis, *Proc. IEEE International Conference on Robotics and Automation*, Nice, 1992, pp. 2178–2185.

[15] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall, New York, 1993.

[16] B.A.Murtagh and M.A.Saunders, MINOS 5.4 USER'S GUIDE, Technical Report SOL 83-20R, Department of Operations Research, Stanford University, 1993.

[17] B.K. Natarajan, On planning assemblies, *Proc. 4th ACM Symposium on Computational Geometry*, 1988, pp. 299–308.

[18] M.S. Ohwovoriole and B. Roth, An extension of screw theory, *Trans. ASME, J. Mechanical Design* **103** (1981), pp.725-735.

[19] J. Snoeyink and J. Stolfi, Objects that cannot be taken apart with two hands, *Discrete and Computational Geometry*, **12** (1994), pp. 367–384.

	Puzzle	Assembly	Snoeyink-Stolfi 6	Snoeyink-Stolfi 6	Snoeyink - Stolfi 30	Engine	Engine
d	2	5	2	5	2	2	5
K	24	6	24	24	300	24	24
N	352	140	96	96	1,200	1,112	1,112
# of Rep. Pts.	3	3	22	190	2,838	4	6
# of S.C. DBGs	2	2	22	118	2,838	0	0
# of Parts	6	3	6	6	30	12	12
Time for the Constraints	104.8	12.3	4.0	4.0	808.0	1881.7	1881.7
Time for the Rep. Pts.	4.7	2.8	2.7	1286.9	509.0	12.6	7880.1
Time for the DBGs	12.7	4.7	27.0	240.5	53,021.3	59.9	120.3

Table 1: Summary of experiments, CPU time reported in seconds

[20] R.H. Wilson, *On Geometric Assembly Planning*, Ph.D. Dissertation, Computer Science Department, Stanford University, March 1992.

[21] R.H. Wilson and J.-C. Latombe, Geometric reasoning about mechanical assembly, *Journal of Artificial Intelligence*, **71** no. 2, 1994, pp. 371–396.

[22] R.H. Wilson and T. Matsui, Partitioning an assembly for infinitesimal motions in translation and rotation, *Proc. IEEE International Conference on Intelligent Robots and Systems*, 1992, pp. 1311–1318.

[23] J.D. Wolter, *On the automatic generation of plans for mechanical assembly*, Ph.D. Thesis, The University of Michigan, Ann Arbor, 1988.