T U M

I N S T I T U T F Ü R I N F O R M A T I K

Detecting Geometric Infeasibility

Achim Schweikard

TUM-I9626
Juni 1996

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-06-1996-I9626-350/1.-FI

Typescript: ---

Druck: Mathematisches Institut und
Institut für Informatik der
Technischen Universität München

Detecting Geometric Infeasibility

Achim Schweikard
Informatik,
Technische Universität München
D-80290 München
schweika@informatik.tu-muenchen.de

June 27, 1996

Abstract

An exact and practical method for translational motion planning with many degrees of freedom is derived. It is shown that certain $D-$dimensional arrangements of hyperplanes can be searched in the following way: only a single connected component is traversed during the search, and the arrangement is searched as an arrangement of surface patches rather than full hyperplanes. This reduction in search effort allows for polynomial time bounds in appropriate cases. Heuristic and randomized planners cannot return an information about infeasibility of planning problems. Experiments with an implementation of the new methods suggest that translational infeasibility can be detected in practical cases.

Keywords: geometric reasoning, assembly planning, motion planning, complete algorithms, arrangement computation in D dimensions.

CR-Classification: I.2 Artificial Intelligence, I.2.9 Robotics, I.2.10 Vision and Scene Understanding, J.6 Computer-Aided Engineering

Introduction

It is conjectured that few basic principles suffice to represent human abilities in geometric reasoning and motion planning. However, human reasoning is

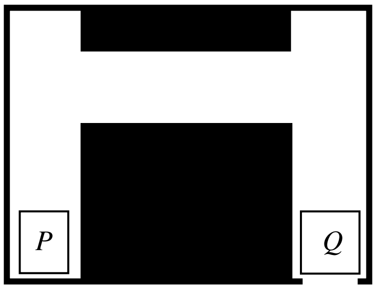

Figure 1: Detecting whether a part can be removed from other parts. Parts P and Q are movable, the container is fixed

capable not only of finding a solution to a given motion planning problem rapidly, but also of quickly recognizing that no feasible solution exists. Intuitively, recognizing infeasibility seems more difficult than finding one motion, if one exists. To detect infeasibility, a proof is required.

Heuristic and approximate motion planning methods are often sufficiently fast to solve practical problems, despite several results showing NP-completeness for large classes of problem instances. Typically, heuristic planners search an approximate decomposition or a graph of random nodes until a motion has been found, or a preset running time limit has been reached.

We are interested in practical methods which will not only report feasible motions, but are equally capable of reporting an exact information about infeasibility.

In this context we will address the following problem:

- Given an assembly of polygonal or polyhedral parts, decide whether one or more parts can be removed from the remaining set of parts by an arbitrary sequence of translations.

Note that this includes arbitrary lock-and-key configurations, where several groups of parts must move simultaneously into distinct directions, and/or change direction during motion. The number of translations in such a sequence is not limited.

Example: Parts P and Q in fig. 1 are movable, the container is fixed. The goal is to decide whether or not part P is removable, after an appropriate series of translational motions of both P and Q.

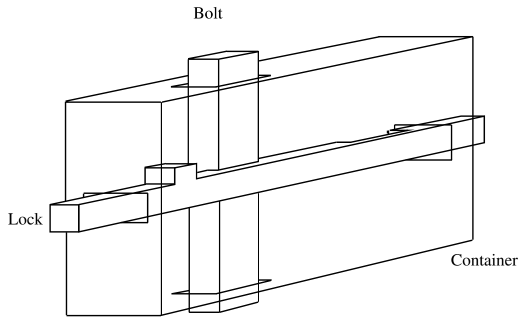

Figure 2: To test whether or not the bolt is removable, intermediate placements of the lock must be searched.

To explain why P is not removable, one would enumerate certain critical intermediate placements of both P and Q, and test for removability in each such placement. The main problem with reproducing such an intuitive approach in an algorithm is the decision about which of the possible intermediate placements are indeed critical.

In the second example (figure 2), intermediate positions of lock must be tested to decide that the bolt is not removable. There are comparatively few critical placements of the lock. In contrast, the complete enumeration of all distinct relative placements of vertices and bounding planes of parts will quickly become impossible even for simple cases.

We are thus interested in methods for reducing the set of potential relative placements of parts, *while retaining completeness*. To obtain a practical method, it is necessary to recognize the essential contacts between parts. But notice that we would detect infeasibility in error, if the set of essential contacts thus chosen was too small.

Our methods are based on comparatively simple principles for analyzing $D-$dimensional arrangements of hyperplanes. This analysis finds and enumerates critical placements without computing the entire arrangement.

1 Related Work

To find a plan for assembling an industrial product from its components, it is useful to compute a disassembly motion, i.e. a motion for separating or removing parts from a given assembly.

Natarajan [11] derives lower bounds on the number of simultaneous translations necessary for separating objects.

Agarwal, de Berg, Halperin and Sharir [1] consider sequences of translations for separating polyhedra. In [1] the set of allowed motion directions is assumed to be given in advance.

Kavraki [7] shows that the problem considered here is NP-complete even for a very restricted class of planar objects.

In [13] we describe an algorithm for computing a *subassembly* S of a given assembly, such that S is removable by a single translation, if such a subassembly exists. The output consists of S and an appropriate removal direction d. Interestingly this computation is possible in polynomial time, even though the number of removable subassemblies is exponential in general. The algorithm in [13] will thus compute a valid subassembly and a removal direction if there is such a subassembly, but does not enumerate all possible subassemblies.

In [14] we describe techniques for analyzing arrangements defined by radiation beams in radiosurgery. These techniques are based on searching so-called minimal and maximal cells in three dimensions and are related to the methods developed here. Specifically, we find a constraint set consisting of constraints for minimal and maximal cells only, allowing for reducing the original constraint set in such a way that it becomes solvable in practice. The minimal and maximal cells do not form a connected component, and for the application in [14] it is sufficient to search the minimal and maximal cells after computing the entire arrangement.

Guibas and Halperin et al. [4] describe methods for partitioning three-dimensional assemblies under infinitesimal motions. The algorithm in [4] allows for computing one allowed infinitesimal motion in polynomial time, and combines basic methods in [13] and [14]. An infinitesimal motion may not give a valid motion for removing parts, since parts may still collide during the motions extending the computed infinitesimal motion. The number of distinct infinitesimal motions is exponential, so that a valid partitioning may not be found even if such a partitioning exists. However, the techniques in [4] include rotations.

Wilson et al. [15] describe variants and improvements of the basic techniques in [13].

The basic techniques in [13] are not used here, since these techniques inherently require that all moving parts follow the same motion path.

Instead, the analysis of 3-dimensional arrangements given by pairwise Minkowski-differences of parts is used to guide the search of a $D-$dimensional arrangement.

The next section gives an informal description of the basic principles in this context. The main idea of the present approach is described in section 3. Indeed, there is a remarkably simple way to compute certain multi-dimensional arrangements as arrangements of *surface patches*, while avoiding the computation of the entire underlying arrangement of *hyperplanes*. However, this is only possible for a particular class of arrangements. The arrangements to be considered here all belong to this class. Section 4 derives an algorithm from these principles. Section 5 gives the analysis of the mentioned reductions. Interestingly, output-sensitive improvements over direct methods can be obtained for the most effective reductions. Section 6 describes experimental results obtained with an implementation. The experiments suggest that it is possible to decide about infeasibility in a complete way. The experimental evaluation compares the derived methods to approximate and randomized motion planners. It is shown that the above principles lead to an exact motion planning algorithm of surprising performance in experiments.

2 Basic Concepts

Let $P_1, \ldots, P_k$ be an assembly of three-dimensional parts. Thus $P_1, \ldots, P_k$ are non-intersecting polyhedra in given spatial placement.

We allow for all parts to translate in space independently. The position of each part is given by three parameters. A space E^D of dimension D describes all simultaneous placements of all parts. Here $D = 3k$. A point in this space is called forbidden, if two or more parts intersect in the corresponding placement.

We consider pairwise Minkowski-differences ($C-$obstacles, [9]) for each pair of parts P_i, P_j. The $C-$obstacle for P_i and P_j is a three-dimensional region, defined by $C(P_i, P_j) = \{p_j - p_i \mid p_i \in P_i, p_j \in P_j\}$.

The $C-$obstacles determine a set of halfspaces $H_1, \ldots, H_s$ in E^D, such that the forbidden regions are bounded by the corresponding hyperplanes in the following sense.

$H_1, \ldots, H_s$ determine an *arrangement* A_D of halfspaces in E^D, and partition E^D into *cells*. A cell is a maximal connected region not containing any points on any of the hyperplanes bounding $H_1, \ldots, H_s$. Cells are thus open $D-$dimensional sets. Cells are regular, i.e. for each cell c either all or no points in c are forbidden. The partitioning given by $H_1, \ldots, H_s$ also defines cells of dimension less than D. Lower dimensional cells are regular as well.

The specific method for obtaining the equations for $H_1, \ldots, H_s$ is not important in this context, but will be considered in some more detail below. The origin in E^D represents the initial placement of all parts, and is not forbidden.

Notice that A_D contains *unbounded* cells. An unbounded cell is a cell entirely containing a ray in its interior. A valid removal motion for one or more parts is a sequence of non-forbidden cells connecting the origin to one of the unbounded cells.

A direct way to obtain an answer to the problem stated above is the following: We compute a graph representation of the arrangement A_D ([2]). Graph nodes for unbounded cells are labelled. Searching this graph will give an exact solution to the above problem.

However, this direct method cannot be used in practice. The reason is that the graph representation of A_D has an exponential number of nodes, and it is not possible to store this graph even for a small number of polyhedra each with few faces.

To see whether parts are removable, it suffices to search a single connected component of the arrangement, namely the component containing the origin. It is obvious that searching only a single connected component will often reduce storage requirements to some extent, but this reduction is insufficient in practice. However, this simple idea provides a basis for a more effective reduction of the storage requirements and search effort, described in the next section.

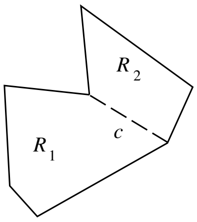

Figure 3: Door cell c. All points in both R_1, R_2 are visible from points in c.

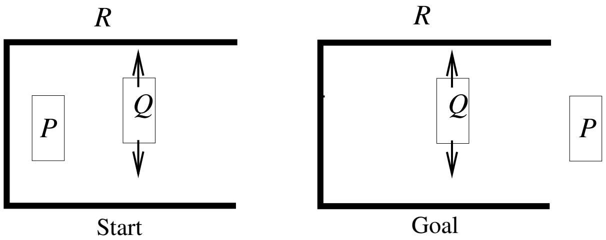

Figure 4: Deciding whether a part can be removed from other parts. Part Q is constrained to move in the vertical direction only

3 Floorgraphs

Constraints for pairs of parts. For two parts P_i and P_j, the bounding planes of $C(P_i, P_j)$ define an arrangement of planes $A(P_i, P_j)$ in three-dimensional space.

We partition the complement of $C(P_i, P_j)$ into convex regions. Each convex region thus obtained is given as the intersection of halfspaces. Let $H_1^{ij}, \ldots, H_r^{ij}$ be the halfspaces stemming from P_i and P_j. These halfspaces are given as inequalities in the three parameters x, y and z.

Let R_1 and R_2 be two adjacent convex regions in the partitioning of the exterior of $C(P_i, P_j)$. Then there is a two-dimensional cell c in the (three-dimensional) arrangement $A(P_i, P_j)$ such that all points in R_1 are visible from any point in c, and all points in R_2 are visible from any point in c (fig. 3). c is a planar patch in $A(P_i, P_j)$. We call c a *passage* or *door cell.*

A *floorgraph* is defined in the following way. Each convex region in the above partitioning of the exterior of $C(P_i, P_j)$ corresponds to one node in the floorgraph. If c is a door cell shared by two regions R_1 and R_2, then c defines

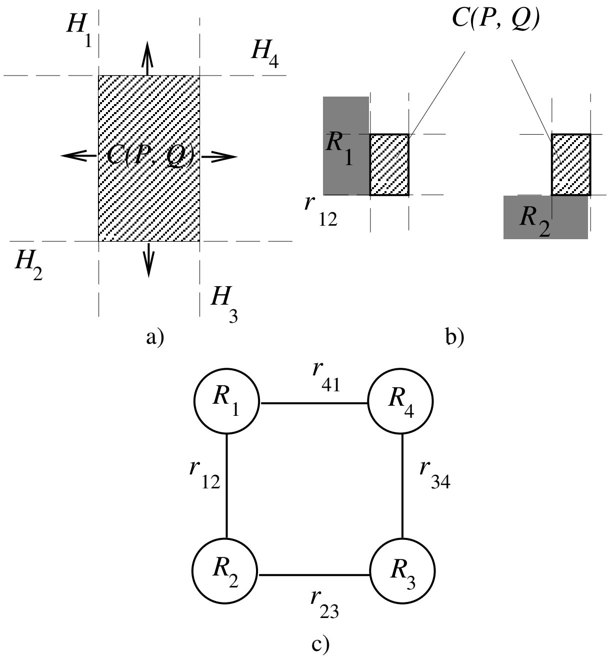

Figure 5: a) Halfspaces $H_1, ..., H_4$ bounding $C(P,Q)$. b) Regions R_i partitioning the exterior of $C(P,Q)$ into convex regions. Door cell r_{12}. c) Floorgraph for pair (P,Q).

an edge in the floorgraph. To each node/edge we assign a set of defining inequalities and equalities.

Example 1: The assembly in fig. 4 contains three parts P, Q, R. R is the (fixed) container. P and Q are rectangular. $C(P,Q)$ is a rectangle (fig. 5-a), and the exterior of $C(P,Q)$ is a union of halfspaces $H_1, \ldots, H_4$. The orientations of $H_1, \ldots, H_4$ are indicated by arrows in the figure.

By convention, the halfspace obtained by reversing the inequality for H_i is denoted by H_i^-. Similarly, the bounding plane of H_i is denoted by $H_i^=$. The exterior of $C(P,Q)$ is thus partitioned into four regions $R_1, \ldots, R_4$, where $R_1, \ldots, R_4$ are given by

$R_1 : H_1^+ \cap H_2^-$, $R_2 : H_2^+ \cap H_3^-$,
$R_3 : H_3^+ \cap H_4^-$, $R_4 : H_4^+ \cap H_1^-$.

The floorgraph for $C(P,Q)$ is shown in fig. 5-c. The partitioning gives

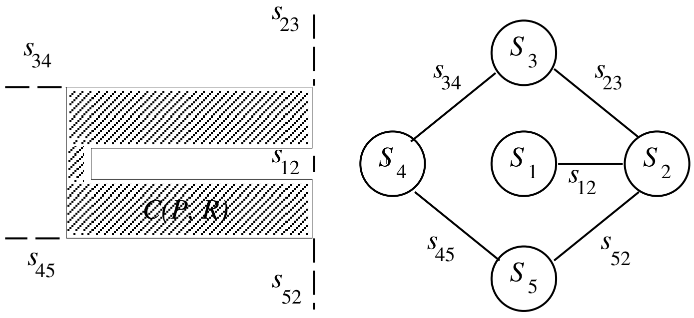

Figure 6: a) C-obstacle $C(P,R)$ for parts P,R in example 1. b) Floorgraph for pair (P,R).

four door cells:

$r_{12} : H_1^+ \cap H_2^=$, $r_{23} : H_2^+ \cap H_3^=$,
$r_{34} : H_3^+ \cap H_4^=$, $r_{41} : H_4^+ \cap H_1^=$.

The floorgraph for $C(P,R)$ is shown in fig. 6. Since we constrain Q to move in the vertical direction only, the floorgraph for (Q,R) can be represented by a single node T_1.

Simultaneous motions of all parts. The $D-$dimensional arrangement A_D is constructed from the halfspace inequalities $H_1, \ldots, H_s$ in the following way.

The position of each part P_i is given by three parameters $p_x^{(i)}, p_y^{(i)}, p_z^{(i)}$. These parameters describe the placement of P_i with respect to a fixed initial placement. Thus a point $(p_x^{(1)}, p_y^{(1)}, p_z^{(1)}, \ldots, p_x^{(k)}, p_y^{(k)}, p_z^{(k)})$ describes a simultaneous placement of all parts $P_1, \ldots, P_k$.

Each halfspace inequality H_i contains the three variables x, y and z. For an inequality H stemming from a pair (P_i, P_j), we replace the variable x by the expression $p_x^{(i)} - p_x^{(j)}$. Similarly, substitute $p_y^{(i)} - p_y^{(j)}$ for y and $p_z^{(i)} - p_z^{(j)}$ for z.

After this substitution, a pair (P_j, P_i) gives the same inequalities as the pair (P_i, P_j). (This follows from the fact that $C(P_j, P_i) = -C(P_i, P_j)$ by definition). Thus it is sufficient to consider each pair (P_i, P_j), where $i < j$.

A point $(p_x^{(1)}, p_y^{(1)}, p_z^{(1)}, \ldots, p_x^{(k)}, p_y^{(k)}, p_z^{(k)})$ in E^D is forbidden, if one of the

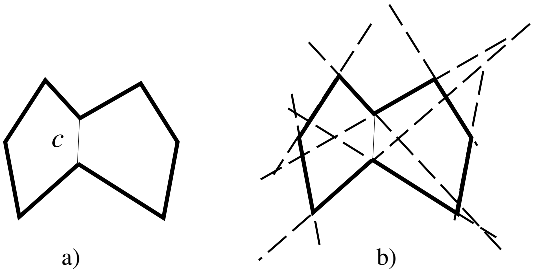

Figure 7: An arrangement of surface patches has smaller number of cells than the corresponding arrangement of (extended) hyperplanes. c is a door cell.

points $(p_x^{(i)} - p_x^{(j)}, p_y^{(i)} - p_y^{(j)}, p_z^{(i)} - p_z^{(j)})$ for $i < j$ is in $C(P_i, P_j)$.

We must fix the position of one (arbitrary) part, since we are testing for removability. Otherwise all ($D-$dimensional) cells in A_D would be unbounded. We thus set the parameters $p_x^{(k)}, p_y^{(k)}$ and $p_z^{(k)}$ of part P_k to zero.

Each of the inequalities for the halfspaces $H_1, \dots, H_s$ contains at most six of the variables $p_x^{(i)}, p_y^{(i)}, p_z^{(i)}$. Specifically each halfspace inequality is of the form

$$a(p_x^{(i)} - p_x^{(j)}) + b(p_y^{(i)} - p_y^{(j)}) + c(p_z^{(i)} - p_z^{(j)}) \leq d.$$

We regard all inequalities thus obtained as $D-$dimensional inequalities, where at most six variables have non-zero coefficients.

Notice that one floorgraph corresponds to each pair of parts P_i, P_j where $i < j$. In the next section it will be shown that floorgraphs allow for searching A_D in the following way:

- Only a single connected component within A_D is traversed. We do not need to enumerate all free cells in A_D.
- A_D is searched as an arrangement of *surface patches* (fig. 7), rather than an arrangement of full hyperplanes.

4 Searching floor graphs

In the following we assume that all parts $P_1, \dots, P_k$ are open sets. In this way none of the inequalities defined above is a strict inequality, i.e. all inequalities are $\leq$-inequalities. We thus allow configurations in which two parts are in contact, but do not overlap. Motions can consist of segments where two or more parts slide along each other.

Let $G_1, \ldots, G_f$ be the floorgraphs for all pairs of parts. For a node n in one of G_i, let $C(n)$ be the set of defining constraint inequalities. Similarly $C(e)$ gives the defining constraints for an edge e in one floorgraph.

Let $n_1, \ldots, n_f$ be nodes in the floorgraphs, where n_i is a node of G_i. A tuple $S = (n_1, \ldots, n_f)$ will be called a $D-$node.

Thus, a $D-$node contains exactly one node of each floorgraph. If $x \epsilon E^D$ satisfies the constraints $C(n_1), \ldots, C(n_f)$ then S is called a feasible $D-$node.

The origin in E^D defines a $D-$node. Indeed, the origin is in free space, and we can find a node n_i in each floorgraph, such that the origin satisfies $C(n_i)$.

We define a *successor* of a $D-$node in the following way.

Let

$S = (n_1, ..., n_{i-1}, n_i, n_{i+1}, ..., n_f)$

be a $D-$node. Then

$S' = (n_1, ..., n_{i-1}, n_i', n_{i+1}, ..., n_f)$

is a successor of S, if n_i' is a successor of n_i in the floorgraph G_i.

Similarly, we define a *successor-edge* of a $D-$node:

$S' = (n_1, ..., n_{i-1}, e, n_{i+1}, ..., n_f)$

is a successor edge of

$S = (n_1, ..., n_{i-1}, n_i, n_{i+1}, ..., n_f)$

if e is an edge emerging from n_i in the floorgraph G_i.

In general, a single $D-$node has several successors. Thus the successor relation on $D-$nodes defines a graph. We search this graph in depth first order, where all $D-$nodes previously visited are marked and not visited again:

1. Compute all floorgraphs for pairs P_i, P_j with $i < j$.

2. Compute a $D-$node S for the origin in E^D.

3. Set $L = \{S\}$.

4. Repeat until $L =$ empty

(a) If L = empty return result 'infeasible', and stop. Set P = first node in L. Remove P from L.

(b) If P is unbounded, return the path from O to P and stop. Set L' = successor edges of P.

(c) For each S in L', test whether S is feasible. If so, store a point x satisfying the constraints in S.

(d) Remove infeasible nodes from L'. Remove all previously visited nodes from L'. Replace each successor edge in L' by the corresponding successor node. Move all entries in L' to the front of L.

Notice that each intermediate point x stored with a $D-$node S (except the origin) is a *point in a door cell.* Thus each point x computed in 4-c satisfies the constraints in one floorgraph *edge* and $f-1$ floorgraph *nodes*. In step 4-d the constraints for this edge are replaced by the constraints for the successor node in the corresponding floorgraph. New nodes generated at the initialization of the list L' in 4-b again have constraints for $f-1$ floorgraph nodes and one floorgraph edge. Segments of the paths can be contained entirely in cells of dimension less than D.

We must show that the sequence of points thus computed indeed yields a feasible path.

Lemma 1 *Let S_1 and S_2 be two feasible $D-$nodes and S_2 be a successor edge of S_1. Let x_1 and x_2 be points satisfying the constraints in S_1 and S_2. Then all points on the line segment connecting x_1 to x_2 in E^D are in free space.*

Proof: x_1 is in a region defined by nodes $n_1, \ldots, n_f$. x_2 has floorgraph nodes $n_1, \ldots, n_{i-1}, n'_i, n_{i+1}, \ldots, n_f$. The nodes n_i and n'_i are connected by an edge e in G_i. By construction x_2 satisfies the constraints

$$C(n_1) \cup \ldots \cup C(n_{i-1}) \cup C(e) \cup C(n_{i+1}) \cup \ldots \cup C(n_f).$$

Since all inequalites are non-strict x_2 also satisfies

$$C(n_1) \cup \ldots \cup C(n_i) \cup \ldots \cup C(n_f),$$

namely the constraints for x_1. Each constraint set $C(n_i)$ defines a convex set in E^D, and their intersection is convex. Thus the entire line segment joining x_1 and x_2 is in free space. •

For each $D-$node S, we store a pointer to its immediate predecessor. This allows for returing a path from any feasible $D-$node to the origin (step 4-b). This path connects the points x_i stored with feasible $D-$nodes by straight line segments.

To obtain a practical algorithm, we must implement the test for feasibility in step 4-c of the above sketch. This test is implemented as a simplex feasibility test ([10]). If positive, the test returns a point x satisfying the given constraints. In the simplex test, we must ensure that variables may become negative. Here standard methods apply.

Furthermore, we must implement the test for boundedness in step 4-b. We must thus decide whether the current $D-$node contains a ray u. Let S be the current $D-$node. u may have points in the boundary of S, i.e. on the defining hyperplanes of S, but must not cross these hyperplanes.

A direct approach uses a simplex minimization along an appropriate vector v, followed by a maximization along the same direction. However, this test can fail to detect unboundedness if S does indeed contain rays, but the set of these rays has dimension lower than D. I.e. the test may fail if all such rays are contained in a hyperplane, and v is orthogonal to this hyperplane. In this case the result of the minimization/maximization is undefined and depends on the implementation. An implementation-independent (and more rapid) test is the following: Let

$$H_1 : n_1 x - d_1 = 0, \ldots, H_r : n_r x - d_r = 0$$

be the equations of the hyperplanes defining S. $n_1, \ldots, n_r$ are the normal vectors of the hyperplanes $H_1, \ldots, H_r$. Let x_0 be the point in S computed in step 4-c. Assume the above hyperplane equations are oriented such that x_0 is above (or in) all hyperplanes $H_1, \ldots, H_r$, i.e. $n_1 x_0 - d_1 \geq 0, \ldots, n_r x_0 - d_r \geq 0$. To decide whether there is a vector u such that $x_0 + tu$ does not cross one of $H_1, \ldots, H_r$ for any $t > 0$ it suffices to test whether there is a u with $n_1 u \geq 0, \ldots, n_r u \geq 0$. Thus a single simplex phase 1 test will find out whether S is unbounded.

To illustrate the search process, we return to example 1 above (see fig. 5 and 6). The first $D-$node is given by the three floorgraph nodes (R_1, S_1, T_1). It is then tested whether the constraints in $C(R_1), C(S_1), C(T_1)$ determine an unbounded cell. The first cell is bounded and a successor edge is tested next. Assume this successor edge is (R_1, s_{12}, T_1). The corresponding constraints define an empty set. The next successor edge is (r_{12}, S_1, T_1), which is nonempty. We thus move to the new current $D-$node (R_2, S_1, T_1). This node is bounded. The next step tests one of the open successor edges (R_2, s_{12}, T_1) or (r_{23}, S_1, T_1). The latter edge is found to be feasible. From the third $D-$node (R_3, S_1, T_1), we move to (R_3, S_2, T_1) which is unbounded.

5 Analysis

The above method outputs a path - if one exists - as a sequence of line segments in E^D. Each segment represents a simultaneous translation of one or more parts, possibly in distinct directions. To find such a path, a tree of line segments is expanded internally.

We will first consider the number of node expansion steps taken by the algorithm. We decompose the faces of each part into triangles. Let n be the maximum number of triangles in each of the k parts. To compute the pairwise Minkowski-differences, it suffices to compute the Minkowski-differences for pairs of triangles on faces, and we will obtain $O(n^2)$ inequalities for each pair of parts. We obtain a total of $s = O(k^2n^2)$ halfspaces in A_D.

Let m be the maximum number of nodes in each floorgraph. Then $m = O(n^2)$. At each step, we will reach a new node in one floorgraph, while the pointer to the current node in all other floorgraphs is not moved. At each step will reach a new cell in A_D, and cells will not be visited again. A_D has at most $O((k^2n^2)^D)$ cells [2].

It will now be shown that the number of steps in the above method can be reduced to account only for the maximum number m of nodes in the floorgraphs, and not for the actual number of bounding planes.

Lemma 2 *The number of node expansions in the above algorithm is bounded by $O((k^2m)^D)$, where m is the number of nodes in each floorgraph and k is the number of parts.*

Proof. Each node in each floorgraph represents a convex (generalized) cylinder in E^D. (The cylinders are called generalized cylinders because their bounding hypersurfaces are linear, not curved). The set of feasible points for a $D-$node is the intersection of $O(k^2)$ such cylinders. There are $O(k^2m)$ cylinders in our arrangement. At each step, we proceed from one *convex* cell of this cylinder arrangement to the next *convex* cell. Notice that the cylinder arrangement contains non-convex cells as well. At most one unbounded $D-$node will be examined, so we must only account for bounded nodes. There is a vector v in E^D such that each bounded convex cell has exactly one extremal vertex in direction v. We are in D dimensions, and each vertex is the intersection of exactly D hyperplanes, except in cases, where hyperplanes are not in general position. After a sufficiently small displacement of all hyperplanes, we can reach a placement, in which each $v-$extremal vertex is the intersection of exactly D distinct hyperplanes. Of course this displacement is only done for accounting purposes, not in the program. The displacement of each hyperplane can be made sufficiently small such that none of the convex cells of the cylinder arrangement will vanish. Each v-extremal vertex of each convex cell is the intersection of at most D cylinders. There are $O(k^2m)$ cylinders. We consider the number of subsets with at most D elements in this set of size $O(k^2m)$. This number is bounded by

$$\binom{k^2m}{1} + \ldots + \binom{k^2m}{D} \leq \sum_{i=1}^{D} \frac{(k^2m)^i}{i!}$$
$$\leq (k^2m)^D \sum_{i=1}^{D} \frac{1}{i!} \leq (k^2m)^D e,$$

where e is the Euler constant.

The first step in this chain of inequalities is the least obvious, but will become clear if we observe that for any $a \geq i$

$$\binom{a}{i} = \frac{a!}{i!(a-i)!} = \frac{1 \cdot \ldots \cdot (a-i)(a-i+1) \cdot \ldots \cdot a}{i! \cdot (1 \cdot \ldots \cdot (a-i))} \leq \frac{a^i}{i!}.$$

Thus there are $O((k^2m)^D)$ extremal vertices of convex cells in the cylinder arrangement. Each such convex cell has one $v-$extremal vertex, i.e. there are $O((k^2m)^D)$ convex cells in the cylinder arrangement.•

Here $D = 3k - 3$ for polyhedral assemblies and $2k - 2$ in the planar case. To see why $D = 3k - 3$ rather than $D = 3k$ notice that the position of one part must be fixed. Otherwise all ($D-$dimensional) cells would be unbounded.

Simple examples show that the last reduction (which is the analysis of the main step in the above method) yields a substantial reduction in practice. Indeed, many floorgraphs (such as the one for the lock/container in fig. 2) consist of a single node.

From the number of expansion steps, one can directly obtain a bound for the running time of the algorithm:

Let $LP(s, D)$ be the number of steps required for solving a linear pro-

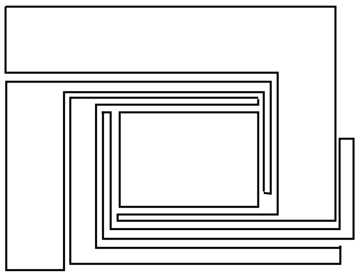

Figure 8: An assembly with interlocking parts.

gram with s constraints and D variables. Then each node expansion will require at most $O((k^2r)LP(k^2r, D))$ steps, where r is the maximum number of constraints in each floorgraph node or floorgraph edge. The total running time is thus bounded by $O((k^2m)^D)k^2rLP(k^2r, D))$.

In this bound we have not accounted for the precomputation of pairwise floorgraphs. The analysis of this preprocessing step is straightforward and follows methods in [12] and [13].

Assuming $LP(s, D)$ is polynomial, one can obtain a polynomial time bound for cases in which the number of node expansions remains constant. This is the case for problems in which parts interlock (fig. 8). Here the number of node expansions is 1, since all successors of nodes in floorgraphs are unreachable.

6 Experiments

To find practical limitations, the above methods were implemented on a Unix-workstation HP 700 in C. We consider the example shown in fig. 4. In this case there are three floorgraphs (for the pairs (P, Q), (P, R) and (Q, R)) with at most five nodes. Three $D-$nodes are expanded during the search, leading to 8 LP-calls each with 7 inequalities. The computing time required to find a removal motion is 1 second. For appropriately enlarged part Q, removal of P becomes impossible, which is detected in the same amount of time.

For comparison, a random planner based on the principles in [5, 8] was implemented for the mentioned hardware environment. Both methods were then applied to the example in figure 9. The running time of approximate

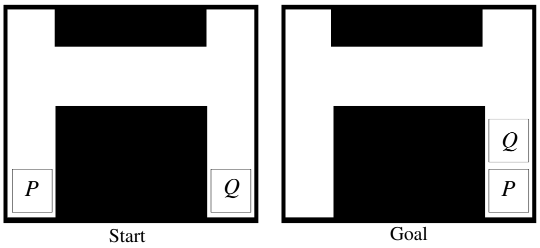

Figure 9: Moving part P from start to goal position.

planners depends largely upon the width of the smallest passage the planner must find. Here the ratio between edge length of moving parts and width of passages was set to 9/10. In this case the random planner takes on the order of 30 seconds to find a solution. This computing time increased if the width of the passages was reduced. The running time for the exact method was not affected by this width, and a path was found in less than 2 seconds. Infeasibility (after appropriate changes of the container) was detected in the same amount of time. Preliminary experience with the exact method suggests that it is capable of reaching or surpassing the performance of advanced random planning schemes. Infeasibility can be detected. But notice that the exact method is limited to translational motion planning, which is not the case for random planners. The implementation of the exact method is comparatively simple and consists of very little program code given standard packages for arrangement computation and linear programming.

A simple bisection method can be used in combination with the above methods to find extremal values of parameters (length/width of objects) to allow for feasibility. However, since the methods are still restricted to translations, maximum surface area problems ([3]) cannot be addressed with this automatic planner.

7 Conclusions

The described methods allow for computing minimum cost paths. Here the cost of a path is number of $D-$nodes along the path, rather than the total Euclidean length in E^D. This cost measure addresses practical considerations, since the number of $D-$nodes along a path bounds the number of direction/velocity changes during the motions. Assembly motions requiring many changes of direction increase assembly costs, and are often impractical due to fixturing and stability problems.

References

[1] P. K. Agarwal, M. de Berg, D. Halperin, M. Sharir. Efficient Generation of k-Directional Assembly Sequences. *Proc. 7th ACM-SIAM Symp. Discr. Algorithms (SODA)*, 122-131, 1996.

[2] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer, Heidelberg, 1987.

[3] J. L. Gerver. On moving a sofa around a corner. *Geometriae Dedicata*, 42:267–283, 1992.

[4] L. Guibas, D. Halperin, H. Hirukawa et al. Polyhedral assembly partitioning using maximally covered cells in arrangements of convex polytopes. *To appear: Intl. J. Comp. Geometry and Applications.*

[5] L. Kavraki, J.-C. Latombe. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5), 229–235, 1993.

[6] L. Kavraki, M. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-Complete. *Information Processing Letters*, 55(3), 159–165, 1995.

[7] L. Kavraki. Randomized preprocessing of configuration space for fast path planning. *IEEE Intl. Conf. Rob. Automation*, 2138–2145, 1994.

[8] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

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

[10] J. J. More', S. J. Wright. Optimization Software Guide. *Frontiers in Applied Mathematics* , 14, SIAM, 1993. See also: "http://www.mcs.anl.gov/home/otc/Guide/SoftwareGuide/".

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

[12] J. O'Rourke. Computational Geometry in C. Cambridge University Press, Cambridge, 1994.

[13] A. Schweikard and R. H. Wilson. Assembly Sequences for Polyhedra. *Algorithmica*, 13(6): 539–552, June 1995.

[14] A. Schweikard, R. Tombropoulos, L. Kavraki, J. R. Adler, J.-C. Latombe. Treatment Planning for a Radiosurgical System with General Kinematics. *IEEE Conference Robotics and Automation*, 1720–1727, May 1994.

[15] R. H. Wilson, L. Kavraki, J.-C. Latombe, and T. Lozano-Pérez. Two-Handed Assembly Sequencing. *Intl. J. of Robotics Research* 14(4): 335–350, 1995.