Preprocessing Chains for Fast Dihedral Rotations Is Hard or Even Impossible

Michael Soss* Jeff Erickson† Mark Overmars‡

Submitted to *Computational Geometry: Theory and Applications*: April 19, 2002
Revised and resubmitted September 26, 2002

Abstract

We examine a computational geometric problem concerning the structure of polymers. We model a polymer as a polygonal chain in three dimensions. Each edge splits the polymer into two subchains, and a *dihedral rotation* rotates one of these subchains rigidly about the edge. The problem is to determine, given a chain, an edge, and an angle of rotation, if the motion can be performed without causing the chain to self-intersect. An $\Omega(n \log n)$ lower bound on the time complexity of this problem is known.

We prove that preprocessing a chain of n edges and answering n dihedral rotation queries is 3SUM-hard, giving strong evidence that $\Omega(n^2)$ preprocessing is required to achieve sublinear query time in the worst case. For dynamic queries, which also modify the chain if the requested dihedral rotation is feasible, we show that answering n queries is by itself 3SUM-hard, suggesting that sublinear query time is impossible after *any* amount of preprocessing.

1 Introduction

During the past several decades, questions regarding polymer structure have received widespread interest in the physics community. Throughout the literature, a polymer is often modeled as a self-avoiding chain of line segments in three-space, where the vertices represent atoms and the edges represent bonds. Due to the constraints of a chemical bond, the valence angles—angles between adjacent bonds to the same atom—are often fixed to attain a more realistic model [2, 11, 17, 26], resulting in a limited range of motion.

The most common method to sample the configuration space of polymers is to randomly reconfigure the chain in a Monte Carlo simulation [6, 7, 12, 20, 24, 25]. An edge of the chain is chosen at random, and a *dihedral rotation* is performed. Any edge $\overline{uv}$ splits the chain into two subchains A and B, where $u \in A$ and $v \in B$. A dihedral rotation at $\overline{uv}$ rotates the subchain B rigidly by some angle ϕ (or equivalently, rotates A by angle $-\phi$) around $\overline{uv}$, keeping the angles at u and v fixed. See Figures 1 and 2.

Before each dihedral rotation, the simulation must check whether the motion is *feasible*, that is, whether or not the chain collides with itself at any time during the motion. Because self-intersections

*School of Computer Science, McGill University. Present Address: Chemical Computing Group, Montreal; soss@chemcomp.com

†Department of Computer Science, University of Illinois at Urbana-Champaign; jeffe@cs.uiuc.edu; http://www.cs.uiuc.edu/~jeffe/. Research partially supported by a Sloan Fellowship and by NSF CAREER award CCR-0093348.

‡Institute of Information and Computing Sciences, Utrecht University; markov@cs.uu.nl; http://www.cs.uu.nl/people/markov/.

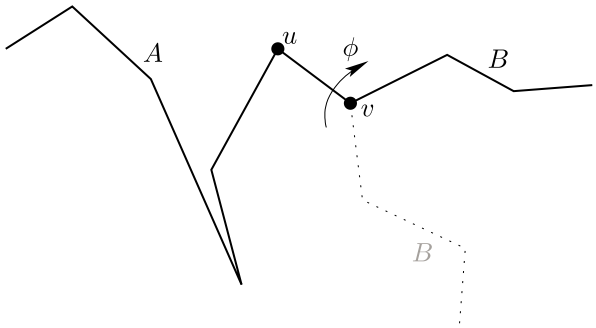

Figure 1. A dihedral rotation.

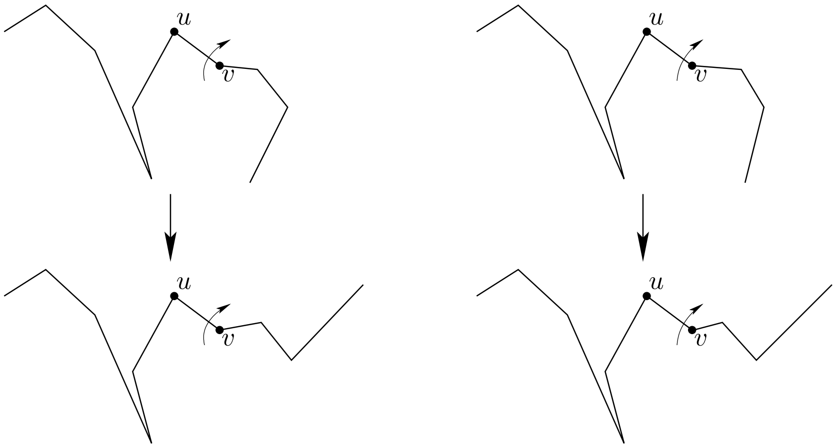

Figure 2. A dihedral rotation, shown as a stereogram. The image can be viewed in stereo by crossing one's eyes until the arrows coincide.

are not allowed in the model, if a rotation is deemed infeasible the resulting configuration must be discarded and another motion randomly chosen. The probability that a randomly selected motion is feasible decreases rapidly as larger polymers are considered. Thus, it is important to determining whether or not a dihedral rotation is feasible as quickly as possible. Soss and Toussaint [23] formalized this problem as follows.

Dihedral Rotation. *Given a polygonal chain, an edge $\overline{uv}$ of the chain, and an angle ϕ, is the dihedral rotation of angle ϕ at $\overline{uv}$ feasible?*

Maintaining polygonal chains under dihedral rotations and checking for self-intersections also has applications in robotics, especially in motion planning for snake-like [5] and self-reconfiguring modular robots [4]. Probabilistic roadmaps [16] and rapidly-exploring random trees [18] are powerful techniques for exploring the configuration spaces of objects with many degrees of freedom; collision detection is a major bottleneck in both of these methods.

Soss and Toussaint proved an $\Omega(n \log n)$ bound on the time complexity of this problem and described a brute force algorithm that runs in $O(n^2)$ time and $O(n)$ space, where n is the number of edges in the chain. For the special case where $\phi \geq 2\pi$ (a full rotation), they constructed a faster algorithm with the help of results by Agarwal and Sharir [1] and by Guibas, Sharir, and Sifrony [15] on arrangements of curves. This algorithm runs in expected time $O(n2^{\alpha(n)} \log n)$, where $\alpha(n)$ is the slowly-growing inverse Ackermann function. The general case can be solved in in $O(n^{5/3+\varepsilon})$ time and space, for any constant $\varepsilon > 0$, using an algorithm by Schömer and Thiel [22]. In practice, this problem is normally solved by tracking the motion of the chain over several discrete time steps, checking for self-intersections at each step using (for example) a hierarchy of bounding volumes; see, for example, recent results of Guibas *et al.* [14] and Lotan *et al.* [19]. None of these methods is efficient in the worst case.[1]

These results apply to single motions, but the simulation of a polymer is a complex process. A typical simulation might have hundreds or thousands of attempted motions. In this paper we examine the complexity of computing the feasibility of a sequence of dihedral rotations. We will refer to each such determination as a *dihedral rotation query.* We will distinguish between *static* queries, which do not modify the chain, and *dynamic* queries, which actually perform the dihedral rotation if it is feasible. To compute each motion as if it were a separate problem seems inefficient as the chain always maintains its edge lengths and vertex-angles. Thus, an intuitive goal is to preprocess the chain so that each ensuing dihedral rotation query can be solved in $o(n \log n)$ time.

We show two problems concerning multiple dihedral rotations to be *3SUM-hard*. A problem is 3SUM-hard if there is a subquadratic reduction from the following problem.

3SUM. *Given a set of integers, do any three elements sum to zero?*

3SUM-hardness was introduced by Gajentaan and Overmars [13] to provide evidence in support of conjectured $\Omega(n^2)$ lower bounds for several problems. The best known algorithm for 3SUM runs in time $\Theta(n^2)$. Quadratic lower bounds have been proven for 3SUM and a few other 3SUM-hard problems in restricted models of computation [8, 9, 10], but the strongest lower bound for any of these problems in a general model of computation is $\Omega(n \log n)$, which follows from results of Ben-Or [3].

In Section 2, we consider *static* dihedral rotation queries, which determine whether a given dihedral rotation is feasible or not, without modifying the chain. We show that preprocessing the chain and answering n static dihedral rotation queries is 3SUM-hard. Thus, $\Omega(n^2)$ preprocessing is almost certainly required to achieve sublinear query time.

In Section 3, we consider *dynamic* dihedral rotation queries, which either modify the chain by performing a dihedral rotation or report that the desired rotation is infeasible. We show that dynamic dihedral rotation queries cannot be answered in sublinear time after *any* amount of preprocessing, unless there is a *nonuniform* family of algorithms for 3SUM with subquadratic running time. Since this seems unlikely, especially in light of existing lower bounds [8], answering a single dynamic dihedral rotation query almost certainly requires $\Omega(n)$ time in the worst case. Even if such a nonuniform family of algorithms does exist, the preprocessing time would be at least the time required to construct the nth algorithm in the family. In contrast, if we do not need to check for feasibility, we can perform any dihedral rotation in $O(\log n)$ time, after only $O(n)$ preprocessing.

2 Static dihedral rotation queries

In this section, we consider the problem of preprocessing a chain of n segments so that we can quickly determine whether an arbitrary dihedral rotation is feasible. We refer to such tests as *static* dihedral rotation queries because they only test feasibility; performing a query does not actually modify the chain. We consider *dynamic* queries, which either modify the chain or report a collision, in the next section.

[1]Guibas *et al.* [14] and Lotan *et al.* [19] independently proved that a hierarchy of tight bounding spheres can be used to detect self-intersections in a *static*, *well-behaved* polygonal chain in $O(n^{4/3})$ time. This does not imply an efficient dihedral rotation algorithm, even for well-behaved chains. Simulating a single rotation may require many time steps, and choosing appropriate time steps to detect instantaneous self-intersections is nontrivial.

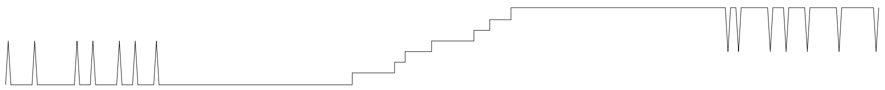

Figure 3. Reducing 3SUM′ to a series of static dihedral rotation queries.

We are interested in tradeoffs between the preprocessing time and the worst-case query time. For example, using the algorithm of Soss and Toussaint [23], we can compute the degrees of freedom for every possible dihedral rotations in $O(n^3)$ time; if we store the results in a table, then any query can be answered in $O(1)$ time simply by looking up the result. On the other hand, with no preprocessing, the optimal query time lies somewhere between $\Omega(n \log n)$ and $O(n^2)$ [23].

The remainder of this section provides strong evidence for the following conjecture.

Conjecture 1. *In any scheme to preprocess a chain of n edges to answer static dihedral rotation queries, either the preprocessing time is $\Omega(n^2)$ or the worst-case query time is $\Omega(n)$.*

We provide strong support for this conjecture by proving that preprocessing a chain of n segments and performing n static dihedral rotation queries is 3SUM-hard. To simplify our reduction, rather than using 3SUM directly, we will instead use the following closely related problem.

3SUM′. *Given three sets of integers, are there elements, one from each set, whose sum is zero?*

Using 3SUM′ instead of 3SUM poses no additional complication, since the two problems are reducible to one another in linear time, only changing the complexity of the input by a constant factor [13]. Therefore a reduction from 3SUM is equivalent to a reduction from 3SUM′.

Because the time complexity of 3SUM′ is unknown, we will use the notation 3SUM(n) to denote the time complexity of the 3SUM′ problem, where n is the total size of the three sets.

Theorem 2. *Preprocessing a chain of n edges and performing n static dihedral rotation queries is 3SUM-hard.*

Proof: Given any instance of 3SUM′, we create a polygonal chain of n segments in $O(n \log n)$ time, such that a sequence of $O(n)$ dihedral rotation queries solves the 3SUM′ problem. Thus, if we spend $P(n)$ time preprocessing the chain and $Q(n)$ time answering each query, then $P(n) + nQ(n) = \Omega(3\text{SUM}(n))$.

Let A, B, and C be three sets of integers; our goal is to determine if there are elements $a \in A$, $b \in B$, and $c \in C$ such that $a + b + c = 0$. If necessary, we modify the sets so that each set lies in an interval far from the other sets. Specifically, we replace A and C with two new sets $A' = \{a - 2m \mid a \in A\}$ and $C' = \{c + 2m \mid c \in A\}$, where m is the maximum absolute value of any element in $A \cup B \cup C$. This replacement clearly does not affect the outcome of 3SUM′. To simplify the reduction, we also sort the three sets in $O(n \log n)$ time. (There is a more complicated $O(n)$-time reduction that avoids sorting by exploiting the third dimension.)

We create a planar chain as illustrated in Figure 3. The chain consists of two *combs* joined by an axis-parallel *staircase*. For each element $a' \in A'$, the left comb contains a very slim upward tooth centered on the line $x = a'$. For each element $c' \in C'$, the right comb contains a very slim downward tooth centered on the line $x = c'$. Finally, for each element $b \in B$, the staircase contains a vertical edge on the line $x = -b/2$.

We now ask a series of $O(n)$ static dihedral rotation queries; namely, can a dihedral rotation of angle 2π be performed at each vertical edge in the orthogonal staircase? Since the edge is vertical,

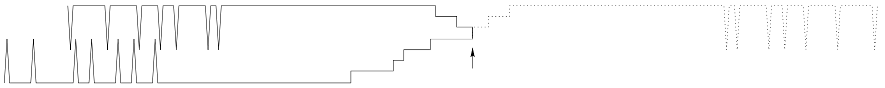

Figure 4. A dihedral rotation at a vertical staircase edge.

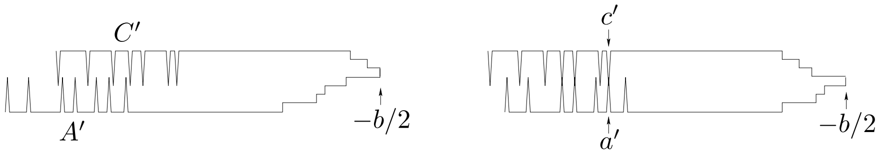

Figure 5. A dihedral rotation at $x = -b/2$. *Left:* No collision implies that $a' + b + c' \neq 0$ for all $a' \in A'$ and $c' \in C'$. *Right:* A collision implies that $a' + b + c' = 0$ for some $a' \in A'$ and $c' \in C'$.

and the chain is planar, the only possibility for an intersection is when the rotation has reached π. At this point, one comb and part of the staircase have been reflected across the vertical edge, as in Figure 4.

Because the rotation is performed at a vertical edge, no edge changes height. This immediately implies that the staircase cannot self-intersect. Each comb stays individually rigid, so neither comb can self-intersect. Furthermore, because each vertical edge in the staircase is at distance at most m from every other staircase edge, but at distance at least $3m/2$ from any edge of a comb, a dihedral rotation cannot cause a comb and the staircase to intersect. Therefore, the only possible intersection during the rotation occurs between the two combs. Since the height of an edge is maintained throughout the motion, intersections are only possible at the teeth.

Suppose we perform a dihedral rotation of angle π at a vertical staircase edge on the line $x = -b/2$. This rotation reflects the right comb across this vertical line, moving each tooth of the right comb from x-coordinate c' to x-coordinate $-c' - b$. This rotation causes two teeth to collide if and only if $a' = -c' - b$, or equivalently, $a' + b + c' = 0$, for some elements $a' \in A$ and $c' \in C$.

We perform a dihedral rotation query for each vertical staircase edge. If any of these rotations is infeasible, the infeasible rotation identifies three elements $a' \in A'$, $b \in B$, and $c' \in C'$ such that $a' + b + c' = 0$. Conversely, if every dihedral rotations are feasible, there is no such triple. Thus, by performing at most n dihedral rotation queries, we solve the original instance of $3\text{SUM}'$.

Let $P(n)$ denote the time to preprocess a chain of n segments for static dihedral rotation queries, and let $Q(n)$ be the worst-case time for a single query. Our reduction solves any instance of $3\text{SUM}'$ of size n in time $O(n \log n) + P(n) + nQ(n)$. Results of Ben-Or [3] imply that $3\text{SUM}(n) = \Omega(n \log n)$. It follows that $P(n) + nQ(n) = \Omega(3\text{SUM}(n))$, as desired. $\square$

3 Dynamic dihedral rotation queries

We now switch our attention to the case of *dynamic* dihedral rotation queries. Given an edge e and an angle ϕ, a dynamic dihedral rotation query determines whether the dihedral rotation at edge e by angle ϕ is feasible, and if it is, modifies the chain by performing the rotation. These queries allow us to determine the feasibility of an arbitrary sequence of rotations. For example, we might ask, “Can we rotate at edge e_1 by angle ϕ_1, then edge e_2 by angle ϕ_2, then edge e_3 by angle ϕ_3, without any collisions at any time?”

Dynamic dihedral rotation queries are more general than the static queries considered earlier, since we can simulate any static query using at most two dynamic queries. Specifically, if a rotation at edge e by angle ϕ is feasible, a second rotation at edge e with angle $-\phi$ restores the chain to its original configuration. Thus, any lower bound for static queries automatically applies (up to a constant factor) to dynamic queries as well. However, we conjecture that dynamic queries are much harder.

Conjecture 3. *In any scheme to preprocess a chain of n edges to answer dynamic dihedral rotation queries, the worst-case query time is $\Omega(n)$, regardless of the preprocessing time.*

One might reasonably ask why Conjecture 3 is in any way nontrivial; after all, a dihedral rotation can change the locations of up to $n-1$ vertices of the chain. However, there is no reason *a priori* that we need to modify these locations explicitly. In fact, if we do not care about collisions, we can perform any sequence of dihedral rotations, each in $O(\log n)$ time, using a simple, linear-size data structure. Essentially the same data structure was independently proposed by Lotan *et al.* [19].

Theorem 4. *Given a chain of n edges and a sequence of k dihedral rotations, all assumed to be feasible, we can compute the resulting chain in $O(n + k \log n)$ time and $O(n)$ space.*

Proof: We maintain a balanced binary tree T whose leaves represent the vertices of the chain and whose internal nodes represent contiguous subchains. At each leaf ℓ, we store a set of (x, y, z)-coordinates for the corresponding chain vertex p_ℓ. At each node v, we store some representation for a rigid motion $M_v : \mathbb{R}^3 \to \mathbb{R}^3$. The actual coordinates of each vertex are computed by composing all the transformations stored on the corresponding root-to-leaf path. Specifically, for each tree node v, we define the function $\overline{M}_v : \mathbb{R}^3 \to \mathbb{R}^3$ as follows. If v is the root, then $\overline{M}_v = M_v$; otherwise, $\overline{M}_v = M_v \circ \overline{M}_u$, where u is the parent of v. Finally, if leaf ℓ stores the coordinates (x, y, z), then the actual location of the corresponding chain vertex p_ℓ is $\overline{M}_\ell(x, y, z)$.

Initially, every M_v is the identity transformation, and each leaf stores the actual coordinates of its chain vertex. We can easily create the initial tree in $O(n)$ time.

Now suppose we want to perform a dihedral rotation at some edge e by angle ϕ. Let $R(\phi, e)$ denote the rigid motion that rotates space around the line through e by angle ϕ. We want to apply this transformation to the subchain on one side of the edge e. To do this, we first find a set of $O(\log n)$ maximal subtrees of T containing the vertices of this subchain. These subtrees can be found using a binary search for one endpoint of e in $O(\log n)$ time. Then, for each root v of one of these maximal subtrees, we replace M_v with the composition $R(\phi, e) \circ M_v$; this has the effect of replacing $\overline{M}_v$ with $R(\phi, e) \circ \overline{M}_v$. We emphasize that the tree T itself does not change.

Finally, after all k rotations have been performed, we can recover the actual coordinates of the chain vertices in $O(n)$ time by a simple tree traversal. □

In support of Conjecture 3, we prove in this section that sublinear dynamic dihedral rotation queries are impossible unless there is a *nonuniform* family of algorithms for 3SUM with subquadratic running time. A nonuniform family of algorithms consists of an infinite sequence of algorithms, one for each possible input size, not necessarily described by a single efficient procedure. The existence of such a family seems unlikely in light of Erickson's $\Omega(n^2)$ lower bound for 3SUM, although in a restricted model of computation [8]. Even if such a nonuniform family of algorithms does exist, our preprocessing time would be at least the time required to construct the nth algorithm in the family.

The distinction between uniform and nonuniform algorithms is best illustrated by a result of Meyer auf der Heide [21], who proved that for each input length n, there is a linear decision tree of depth $O(n^4 \log n)$ that solves the (NP-complete) KNAPSACK problem: Given a set of n real numbers, does any subset sum to 1? These linear decision trees exploit 'hardwired' information that

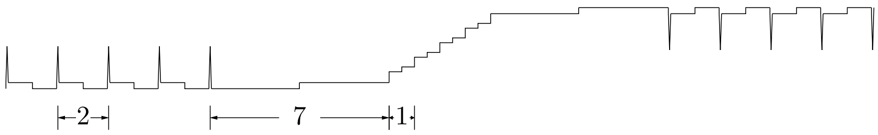

Figure 6. The canonical chain for $n = 5$; see the proof of Theorem 5.

any uniform algorithm would require superpolynomial time to compute on the fly, unless P=NP.[2] Of course, one could precompute all this hardwired information if the input size n is known in advance, but this would require exponential time and space (even if P=NP).

We now demonstrate the link between nonuniform algorithms for 3SUM′ and the dynamic dihedral rotation query problem. Let 3SUMPREP(n) denote the time required to construct, given the input size n, an algorithm that can solve any instance of 3SUM′ of size n in $o(n^2)$ time. For example, if there is a nonuniform family of linear decision trees of subquadratic depth, 3SUMPREP(n) is (at most) the time to construct the nth tree in the family. If there is no subquadratic nonuniform algorithm for 3SUM′, then 3SUMPREP$(n) = \infty$.

Theorem 5. *Suppose we have a data structure that can answer dynamic dihedral rotation queries in $Q(n)$ time, after $P(n)$ preprocessing time, for any chain of length n. Then either $Q(n) = \Omega(n)$, or $P(n) = \Omega(\text{3SUMPREP}(n))$, or $\text{3SUM}(n) = o(n^2)$.*

Proof: We reduce the construction of a subquadratic nonuniform algorithm for 3SUM′ on sets of size n to a series of dynamic dihedral rotation queries as follows. Suppose we are given the integer n and asked to construct an algorithm for any 3SUM′ problem where each set has n elements. We create a chain whose structure is determined solely by the number n, and spend time $P(n)$ preprocessing it to answer dynamic dihedral rotation queries. When the preprocessing has finished, the sets A, B, and C are revealed. We then perform a sequence of $O(n)$ dihedral rotations, each in time $Q(n)$, that move the chain into a configuration similar to the one in Figure 3 for the three sets. After $O(n)$ additional rotations, as in the proof of Theorem 2, the given instance of 3SUM′ is solved. If $Q(n) = o(n)$, then the 3SUM′ instance has been solved in subquadratic time. Thus, constructing a subquadratic algorithm for instances of 3SUM′ of size n has been reduced to constructing and preprocessing the chain. It follows that $P(n)$ must be $\Omega(\text{3SUMPREP}(n))$. In particular, if there is no subquadratic nonuniform algorithm for 3SUM′, then $Q(n)$ must be $\Omega(n)$.

For any positive integer n, we construct a *canonical* planar chain as follows. We begin by building a chain consisting of a left comb pointing up, a staircase, and a right comb pointing down, exactly as in the previous section. See Figure 6. Each comb consists of n teeth, each of height 1, where adjacent pairs of teeth are distance 2 apart. The staircase consists of $n + 1$ steps, each with width 1 and height $2/n$. The distance between the staircase and either comb is 7.

We then replace every horizontal segment in the chain with a *hinge* consisting of five segments, as shown in Figure 7. Each hinge allows us to bring any adjacent pair of vertical segments arbitrarily close together by a short sequence of dihedral rotations. Specifically, referring to the left side of Figure 7, we can bring teeth a_1' and a_2' to any desired distance by performing a dihedral rotation at α_1 by some angle $0 < \theta < \pi/2$, a dihedral rotation at uv by -2θ, and a dihedral rotation at α_2 by angle θ. After the three rotations, the teeth are at any desired distance less than 1, and the rest of the chain is unaffected except for a translation. We easily verify that if the portion of the chain on one side of the hinge is coplanar, then we can perform these rotations without collisions.

[2] Specifically, the computation path for any input implicitly depends on which subset of a set of 2^n hyperplanes intersects a cell in a grid of hypercubes in $\mathbb{R}^n$. Although we can locate the appropriate cell on the fly in polynomial time, calculating the subset of hyperplanes that intersect it is NP-hard.

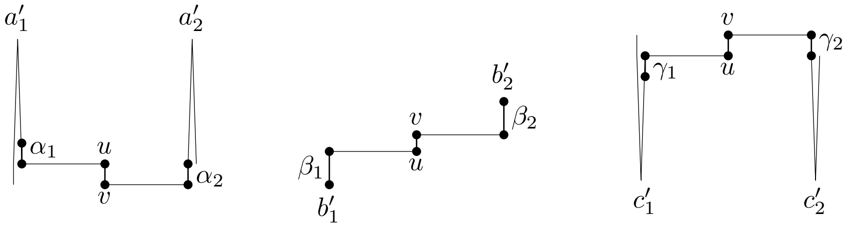

Figure 7. Hinges for the left comb, the staircase, and the right comb.

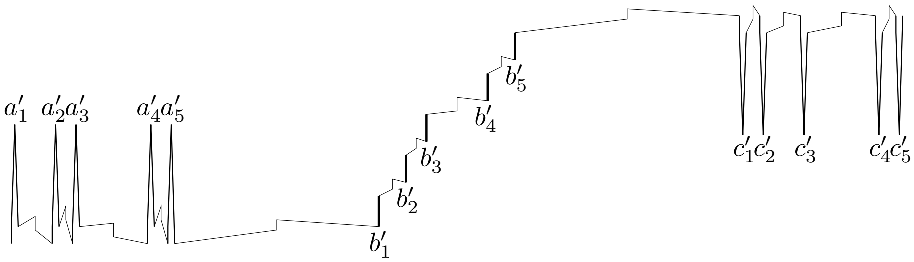

Figure 8. The canonical chain, manipulated to encode A', B', and C'. *(Shown in stereo in Figure 9.)*

Once we construct the canonical planar chain, we preprocess it in time $P(n)$ for dynamic dihedral rotation queries.

Now suppose we are given three sets A, B, and C, each containing n integers, and are asked if they respectively contain three elements a, b, and c whose sum is zero. To solve this instance of 3SUM$'$, we perform a sequence of $O(n)$ dynamic dihedral rotation queries; a triple of elements summing to zero exists if and only if some dihedral rotation in this sequence is infeasible.

Our reduction will be easier if we assume that the three input sets A, B, and C have the same number of elements. If some set has fewer elements than another, then we can augment the smaller set with elements of the form $i/4n$, for some small integer i, without affecting the outcome of 3SUM$'$. (Because the other elements of the sets are integers, none of these fractions can contribute to a triple of elements that sum to zero). We will also assume, as in the proof of Theorem 2, that the sets are given in sorted order.

Let m be the maximum absolute value of any element in $A \cup B \cup C$. We define three new sets A', B', and C' as follows:

$$A' = \{a/m - 5 \mid a \in A\}, \qquad B' = \{b/2m \mid b \in B\}, \qquad C' = \{c/m + 5 \mid c \in C\}.$$

Clearly, the original sets contain elements a, b, c such that $a + b + c = 0$ if and only if these new sets contain corresponding elements a', b', c' such that $a' + c' = 2b'$.

To encode these sets into our chain, we manipulate the hinges in order from left to right so that the x-coordinates of the left comb's teeth are the elements of A', the x-coordinates of the vertical staircase edges are the elements of B', the x-coordinates of the right comb's teeth are the elements of C'. This manipulation is always possible, because the required distance between the ends of any hinge is no more than their distance in the original canonical chain. An example of the final configuration is shown in Figures 8 and 9.

We observe that the chain does not self-intersect during these dihedral rotations by examining the hinges in Figure 7. As described earlier, each hinge is manipulated using a sequence of three rotations. Because we manipulate the hinges in order from left to right, whenever we flex a hinge, the portion of the chain to the right of that hinge is coplanar.

Once the chain is set for A', B', and C', we perform dihedral rotations of angle 2π at every vertical edge in the staircase that corresponds to an element of B'. Just as in the proof of Theorem 2,

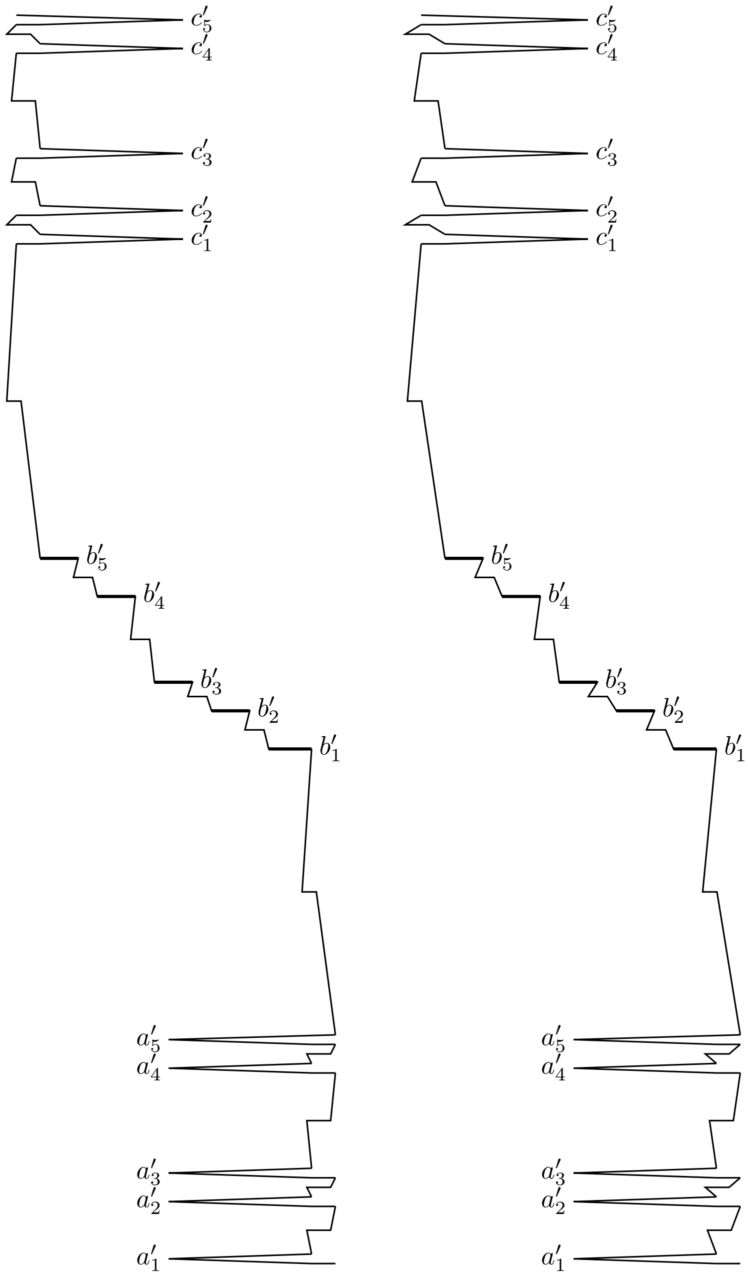

Figure 9. In stereo: The canonical chain, manipulated to encode A', B', and C'.

the chain self-intersects if and only if there exists a triplet $a' - 2b' + c' = 0$. Thus, the sequence of n dynamic queries solves the original $3\textsc{sum}'$ problem.

We spent time $P(n)$ preprocessing the chain before the sets were revealed, and time $10nQ(n)$ for $10n$ dihedral rotations—$9n$ rotations to set the $3n$ hinges, and n more to test for collisions. Thus, after $P(n)$ preprocessing time, we can answer any instance of $3\textsc{sum}'$ of size n in time $O(nQ(n))$. The theorem now follows immediately. □

4 Conclusions

We have shown that two problems related to dihedral rotations are 3SUM-hard. Our results imply that processing a sequence of dihedral rotations almost certainly requires at least linear time per rotation in the worst case.

Our proofs use polygonal chains with very sharp angles and edges of very different lengths, which is rather unrealistic. However, it is possible to modify our canonical chain construction so that the lengths of the shortest and longest edges differ by only a small constant factor and all angles are larger than some constant. The modification requires replacing the staircase with a square wave, replacing the teeth of each comb with triangles parallel to the yz-plane, and moving the hinges into the third dimension. We omit the (tedious) details. We conjecture that processing a sequence of dihedral rotations is 3SUM-hard even for chains where every edge has unit length and every joint angle is equal to an arbitrary constant $0° < \theta < 180°$—in particular, if θ is one of the common bond angles $90°$, $109.47°$, and $120°$.

A more serious issue is that our arguments apply only to the worst-case complexity of the problem. Monte Carlo molecular simulations apply a series of *random* dihedral rotations; the edge to rotate is chosen uniformly, and the rotation angle is typically chosen from a uniform or Gaussian distribution [6, 7, 12, 20, 24, 25]. It would be interesting to analyze the average-case complexity of checking a random dihedral rotation for a random chain (with a given sequence of lengths and angles) under these distributions.

Acknowledgments

We would like to thank Godfried Toussaint for organizing the Workshop on Computational Polygonal Entanglement Theory on February 4-11, 2000, at which this research was initiated. We also thank the other participants of the workshop, Oswin Aichholzer, David Bremner, Carmen Cortés, Erik Demaine, Vida Dujmović, Ferran Hurtado, Henk Meijer, Belén Palop, Suneeta Ramaswami, Vera Sacristán, and Godfried Toussaint for stimulating conversations. Finally, we thank the anonymous referees for their helpful comments.

References

[1] Pankaj K. Agarwal and Micha Sharir. Red-blue intersection detection algorithms, with applications to motion planning and collision detection. *SIAM J. Comput.*, 19(2):297–321, 1990.

[2] H. Benoit. Calcul de l'écart quadratique moyen entre les extrémités de diverses chaînes moléculaires de type usuel. *Journal of Polymer Science*, 3(3):376–388, 1948.

[3] Michael Ben-Or. Lower bounds for algebraic computation trees. *Proc. 15th Annu. ACM Sympos. Theory Comput.*, 80–86, 1983.

[4] Arancha Casal and Mark Yim. Self-reconfiguration planning for a class of modular robots. *Sensor Fusion and Decentralized Control in Robotic Systems II*, pages 246–257. Proc. SPIE 3839, 1999.

[5] Gregory S. Chirikjian and Joel W. Burdick. Kinematics of hyper-redundant robot locomotion. *IEEE Trans. Robotics Automation* 11:781–793, 1995.

[6] John G. Curro. Computer simulation of multiple chain systems—the effect of density on the average chain dimensions. *Journal of Chemical Physics*, 61(3):1203–1207, 1974.

[7] John G. Curro. Computer simulation of multiple chain systems—equation of state of hard sphere chains. *Journal of Chemical Physics*, 64(6):2496–2500, 1976.

[8] Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.* 1998(8), 1998.

[9] Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.* 28(4):1198–1214, 1999.

[10] Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discr. Comput. Geom.* 13(1):41–57, 1995. Erratum in *Discr. Comput. Geom.* 18(2):239–240, 1997.

[11] Henry Eyring. The resultant electric moment of complex molecules. *Physical Review*, 39:746–748, 1932.

[12] J. J. Freire and A. Horta. Mean reciprocal distances of short polymethylene chains. Calculation of the translational diffusion coefficient of n-alkanes. *Journal of Chemical Physics*, 65:4049–4054, 1976.

[13] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

[14] Leonidas Guibas, An Nguyen, Daniel Russel, and Li Zhang. Collision detection for deforming necklaces. *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 33–42, 2002.

[15] Leonidas J. Guibas, Micha Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete Comput. Geom.*, 4:491–521, 1989.

[16] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* 12:566-580, 1996.

[17] Werner Kuhn. Über die Gestalt fadenförmiger Moleküle in Lösungen. *Kolloid Zeitschrift*, 68(1):2–15, 1934.

[18] Steven M. LaValle and James J. Kuffner, Jr. Randomized kinodynamic planning. *Internat. J. Robot. Res.* 20(5):378–400, 2001.

[19] Itay Lotan, Fabian Schwarzer, Dan Halperin, and Jean-Claude Latombe Efficient maintenance and self-collision testing for kinematic chains. *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 43–52, 2002.

[20] D. S. McKenzie. Polymers and scaling. *Physics Reports (Section C of Physics Letters)*, 27(2):35–88, 1976.

[21] Friedhelm Meyer auf der Heide. A polynomial time linear search algorithm for the n-dimensional knapsack problem. *J. ACM*, 31:668–676, 1984.

[22] Elmar Schömer and Christian Thiel. Efficient collision detection for moving polyhedra. *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 51–60, 1995.

[23] Michael Soss and Godfried Toussaint. Geometric and computational aspects of polymer reconfiguration. *Journal of Mathematical Chemistry*, 27(4), 2001.

[24] Steven D. Stellman and Paul J. Gans. Computer simulation of polymer conformation, II. Distribution function for polymers with exluded volume. *Macromolecules*, 5(6):720–729, 1972.

[25] Steven D. Stellman and Paul J. Gans. Efficient computer simulation of polymer conformation, I. Geometric properties of the hard-sphere model. *Macromolecules*, 5(4):516–526, 1972.

[26] William J. Taylor. Average length and radius of normal paraffin hydrocarbon molecules. *Journal of Chemical Physics*, 16(4):257–267, 1948.