Cutting Cycles of Rods in Space: Hardness and Approximation

Boris Aronov*  Mark de Berg†  Chris Gray†  Elena Mumford†

Abstract
We study the problem of cutting a set of rods (line segments in \( \mathbb{R}^3 \)) into fragments, using a minimum number of cuts, so that the resulting set of fragments admits a depth order.

We prove that this problem is NP-complete, even when the rods have only three distinct orientations. We also give a polynomial-time approximation algorithm with no restriction on rod orientation that computes a solution of size \( O(\tau \log \tau \log \log \tau) \), where \( \tau \) is the size of an optimal solution.

1 Introduction

Motivation. We say that an object \( o \) in \( \mathbb{R}^3 \) is below an object \( o' \) in \( \mathbb{R}^3 \) if there is a vertical line \( \ell \) such that there are points \( p \in o \cap \ell \) and \( p' \in o' \cap \ell \) with \( p < p' \).

A depth order on a set \( S \) of disjoint objects in \( \mathbb{R}^3 \) is a linear ordering of \( S \) such that if an object \( o \in S \) is below an object \( o' \in S \), then \( o \) comes before \( o' \) in the ordering. Depth orders play an important role in various problems arising in computer graphics. For example, one way to perform hidden-surface removal is with the so-called Painter’s Algorithm [9]: compute a depth order on the objects in the scene and draw them in this order onto the screen. Various other algorithms for hidden-surface removal [13, 14] and data structures for vertical ray shooting [4] need a depth order as well. Hence, the problem of computing a depth order on a given set of objects has received considerable attention in computational geometry [2, 3, 5, 12].

Unfortunately not every set of objects admits a depth order, because there can be cyclic overlap in the scene—see Figure 1(a). (In the presence of cyclic overlap many algorithms that compute a depth order simply report an incorrect order; thus there are also algorithms [2, 3] to verify whether a given order is a valid depth order.) If the original set of objects does not admit a depth order, one can try to cut the objects in such a manner that all cyclic overlap is removed, so that the resulting set of fragments does admit a depth order—see Figure 1(b). This naturally leads to the question: how can we cut the given objects into fragments using the minimum number of cuts so that no cyclic overlap remains? Another question that arises is: For a given class of objects—triangles, for example, or lines, or rods (that is, line segments)—how many cuts are needed in the worst case to remove all cyclic overlap? Already for the case of lines this appears to be a quite difficult problem, with intimate connections to the realizability of weaving patterns of lines [6].

Related results. For a given set of \( n \) lines (or rods) one can remove all cyclic overlap by simply cutting each line at every point that is directly below another line. Thus we spend one cut for every intersection point in the projection, showing that \( O(n^2) \) cuts always suffice to eliminate all cycles in a set of lines. Currently no subquadratic upper bound on the worst-case number of cuts is known. The only known lower bound is by Chazelle et al. [6], who observed that a grid-like pattern formed by three families of axis-parallel lines requires \( \Omega(n^{3/2}) \) cuts. Chazelle et al. [6] also show that \( O(n^{3/5}) \) cuts suffice in the special case of bipartite weavings.1

Aronov et al. [1] show that with \( O(n^{2-1/69}) \log^{16/69} n \) cuts one can remove all cycles formed by triples of lines.

Figure 1: (a) Cyclic overlap in a set of three rods. (b) One of the rods has been cut so that the resulting set of four objects admits a depth order, namely \( s_{3,1}, s_2, s_1, s_{3,1} \).

---

*Department of Computer and Information Science, Polytechnic University, Six MetroTech Center, Brooklyn, NY 11201-3840 USA; http://cis.poly.edu/~aronov. Research supported in part by NSA MSP Grant H98230-06-1-0016 and by a grant from the US-Israel Binational Science Foundation. Part of the research was performed when B.A. visited TU Eindhoven in January 2007.

†Department of Computing Science, TU Eindhoven. P.O. Box 513, 5600 MB Eindhoven, the Netherlands. Email: {mdberg,gray,emumford}@win.tue.nl. This research was supported by the Netherlands’ Organisation for Scientific Research (NWO) under project no. 639.023.301.

†A bipartite weaving is a collection of rods \( S = S_1 \cup S_2 \), where every rod from \( S_1 \) intersects all rods from \( S_2 \) in the projection, but no two rods from \( S_1 \) (or \( S_2 \)) intersect in the projection.
In this paper we do not study these combinatorial problems, but we focus on the algorithmic question: for a given set of $n$ rods, how can we compute a small set of cuts such that the resulting set of fragments does not have a cycle? This problem was also studied by Solan [16], who gave an algorithm for the special case of bipartite weavings that makes at most $O(n^{9/5} \log n)$ cuts and runs in $O(n^{47/15+\varepsilon})$ time. Solan also considered the case of $c$-oriented rods—that can take one of $c$ possible orientations, for some constant $c$—presenting an algorithm that makes $O(n^{3/2+\varepsilon})$ cuts and runs in $O(n^{1/2+\varepsilon})$ time. Note that these algorithms do not try to compute an optimal set of cuts for the given input—they focus on obtaining a set of cuts whose size is close to optimal in the worst case. Solan also gave an algorithm for arbitrary rods that is output-sensitive: its running time and number of cuts depend on $\tau$, the minimum number of cuts needed for the input instance. More precisely, the algorithm has a running time of $O(n^{3/3+\varepsilon} \tau^{1/3})$ and the number of cuts it performs is $O(n^{1+\varepsilon} \tau^{1/2})$. Even though the algorithm is output-sensitive, it does not have a very good approximation ratio: if a single cut suffices, the algorithm may actually produce $O(n^{1+\varepsilon})$ cuts. More recently, Har-Peled and Sharir [11] gave a randomized improvement to this result: the expected number of cuts produced by their algorithm is $O(n\sqrt{\tau} \alpha(n) \log n)$ and it runs in expected time $O(n^{2/3+\varepsilon} \tau^{1/3})$. This is the topic of our paper: are there efficient algorithms with a better approximation ratio, or perhaps even algorithms that produce an optimum number of cuts?

Our results. We start by investigating the computational complexity of the problem of computing a minimum-size collection of cuts that removes all cycles in a given set of rods. We show in Section 2 that this is an NP-complete problem, even when the rods have only three distinct orientations, by a reduction from VERTEX COVER. We turn our attention to approximation algorithms in Section 3, where we present a polynomial-time algorithm with no restrictions on rod orientation for computing a collection of cuts of size $O(\tau \log \tau \log \log \tau)$, where $\tau$ is the minimum number of cuts needed.

Notation and definitions. We denote the input set of $n$ rods by $S$ and assume that rods are pairwise disjoint. We consider rods to be relatively open, that is, a rod does not include its endpoints. We denote the vertical projection of a rod $s$ onto the $xy$-plane by $\pi(s)$. For two rods $s$ and $t$ we say that $s$ is below $t$, denoted $s \prec t$, if there are points $p \in s$ and $q \in t$ such that $p_x = q_x$, $p_y = q_y$, and $p_x < q_x$. In other words, $s \prec t$ if $\pi(s)$ intersects $t$ and “at the intersection point” $s$ is below $t$. Instead of saying that $s$ is below $t$ and writing $s \prec t$, we sometimes say that $t$ is above $s$ and write $t \succ s$. We use $G_\succ(S)$ to denote the directed graph with vertex set $S$ that has an edge $(s, t)$ whenever $s \prec t$. A depth order on $S$ corresponds to a topological order on the graph $G_\succ(S)$, and cyclic overlap in $S$ corresponds to a cycle in $G_\succ(S)$.

A cut of a rod $s = pq$ is an operation replacing $s$ by two rods $s_1 = pr$ and $s_2 = rq$, where $r \in s$ is called the cut point. Since rods do not include their endpoints, the cut point $r$ is neither a part of $s_1$ nor of $s_2$. Let $s$ and $t$ be two rods such that the projections of $s$ and $t$ intersect in a single point $r$. With a slight abuse of terminology, we say that we cut $s$ at its intersection point with $t$ when we cut $s$ at the point projecting onto $r$. Note that the projections $\pi(s)$ and $\pi(t)$ of the two fragments resulting from the cut are disjoint from $\pi$, since neither of them contains the cut point.

A cut set $C$ for $S$ is a collection of cuts on the rods in $S$. We use $C(S)$ to denote the set of rods resulting from cutting the rods in $S$ at the points in $C$. We say that $C$ is complete if $C(S)$ admits a depth order. Thus we are interested in finding a complete cut set of small size for a given set $S$ of rods.

2 NP-Completeness

The decision version of our problem, which we call CYCLIC OVERLAP REMOVAL, is defined as follows.

**Cyclic Overlap Removal**

Input: A set $S$ of $n$ rods in $\mathbb{R}^3$ and a natural number $k$.

Output: YES if there is a cut set for $S$ of size at most $k$, NO otherwise.

A subset $V^*$ of vertices of an undirected graph $G = (V, E)$ is called a vertex cover of $G$ if for every edge $(v, w) \in E$ we have $v \in V^*$ or $w \in V^*$ (or both). We will show that CYCLIC OVERLAP REMOVAL is NP-complete by giving a polynomial-time reduction from the problem of deciding whether a planar graph of maximum degree 3 admits a vertex cover of a given size:

**Planar Vertex Cover**

Input: An undirected planar graph $G = (V, E)$ of maximum degree 3 and a natural number $k$.

Output: YES if there is a vertex cover for $G$ of size at most $k$, NO otherwise.

**Planar Vertex Cover** is known to be NP-complete [10]. A feedback vertex set of a directed graph $G = (V, E)$ is a subset $V^* \subset V$ that hits—that is, contains a vertex from—every cycle in $G$. Thus removing all vertices in a feedback vertex set eliminates all cycles in the graph. Our reduction will be similar to (but much more involved than) the reduction that is used to
prove the NP-hardness of deciding if there is a feedback vertex set of a given size in a directed graph.

**Feedback Vertex Set**

Input: A directed graph $G = (V,E)$ and a natural number $k$.

Output: YES if there is a feedback vertex set for $G$ of size at most $k$, NO otherwise.

It is easy to see that Planar Vertex Cover can be reduced to Feedback Vertex Set: take an instance $(G, k)$ of Planar Vertex Cover and transform it to an instance $(G', k)$ of Feedback Vertex Set by replacing the undirected graph $G$ by its directed analogue $G'$, with two oppositely directed edges replacing each undirected edge. Now there is a feedback vertex set of size at most $k$ in $G'$ if and only if there is a vertex cover of size at most $k$ in $G$. $G'$ will also play a role in our reduction.

### 2.1 The reduction

The input to our reduction is a planar undirected graph $G$ of maximum degree 3. The idea is to construct a specific embedding for the directed graph $G'$ obtained by doubling and directing the edges of $G$. From this embedding we then construct a set $S$ of rods with the property that all cyclic overlap can be removed from $S$ by $k$ cuts if and only if $G$ has a vertex cover of size $k$.

Our construction uses several intermediate embeddings, as illustrated in Figure 2. We start by constructing a rectilinear planar embedding $E_0(G)$ of $G$ using the polynomial-time algorithm of Tamassia [17]. Next we turn this into a fat embedding $E_1(G)$ by replacing each vertex by a small disk, and slightly “fattening” each edge. Then we replace each fat edge by two directed edges, one along each side of the fat edge. This way we obtain an embedding $E_2(G')$ of the directed graph $G'$. As in the reduction from Planar Vertex Cover to Feedback Vertex Set, we direct the two edges corresponding to the same edge in $G$ in opposite ways. However, we need one additional property. Let $v$ be a degree-6 vertex in $G'$ and consider the directions of its incident edges (incoming or outgoing) listed in clockwise order around $v$. We say that $v$ is admissible if at least two edges coming into $v$ are consecutive. (Therefore there must also be two consecutive outgoing edges.) This property is equivalent to the following: not all three 2-cycles incident to $v$ are oriented the same way.

**Lemma 2.1.** The edges in $E_2(G')$ can be oriented such that for each edge in $G$ the two corresponding edges in $G'$ have opposite directions and all degree-6 vertices are admissible. The orientations can be computed in linear time.

**Proof.** Consider a 4-edge-coloring of the graph $G$, that is, a coloring of its edges by four colors—red, green, blue, yellow—such that no two edges incident to a common vertex have the same color. Skulrattanakulchai [15] has shown that any graph of maximum degree 3 admits such a coloring and that the coloring can be found in linear time.

Now consider the two directed edges $e_1$ and $e_2$ in $G'$ corresponding to some edge $e$ in $G$. If $e$ is green or red, we direct $e_1$ and $e_2$ such that they form a clockwise cycle, otherwise we direct them such that they form a counterclockwise cycle. Since every degree-3 vertex in $G$ is incident to at least one red or green edge and at least one blue or yellow edge, this implies that every degree-6 vertex in $G'$ is admissible. □

Finally, we construct the required embedding $E_3(G')$ by slightly adjusting the embedding $E_2(G')$ in order to meet the following requirement: each degree-6 vertex has either two consecutive incoming edges that are collinear (that is, both horizontal or both vertical), or two consecutive outgoing edges that are collinear. Note that the existence of consecutive incoming (and outgoing) edges is guaranteed by Lemma 2.1. Hence, we only need to make sure that two such consecutive edges are collinear. Consider a vertex $v$ for which this is not the case. We can satisfy the requirement by re-routing one of its incident fat edges using one extra bend—see for example vertex $u$ in Figure 2(e). We also make sure...
that the incident edges of each degree-4 vertex are not collinear, so that there is a bend at each degree-4 vertex.

From the embedding $E_3(G')$ we construct our set $S$ of rods. We represent every vertex of $G'$ by a configuration of rods we call a vertex gadget, every directed edge by a configuration of rods called an edge gadget, and connect the gadgets together in a special way such that every directed cycle in $G'$ is represented by a depth order cycle in $G_\prec(S)$. Figure 3 shows the result for the graph of Figure 2. We construct $S$ using rods with only three orientations: parallel to the $x$-axis, parallel to the $y$-axis, and nearly parallel to the $z$-axis. We call the rods that are parallel to the $x$- or $y$-axis horizontal rods. Next we describe our various gadgets in more detail.

The vertex gadget. Each vertex gadget consists primarily of three horizontal rods—$A$, $B$, and $M$—arranged so that we have rod $M$ above rod $B$ and below rod $A$. The projection of the vertex gadget on $xy$-plane lies inside the disk that represents vertex $v$ of $G'$. The projections $\overrightarrow{A}$ and $\overrightarrow{B}$ are parallel and $\overrightarrow{M}$ is perpendicular to them. Each vertex gadget also has a number of horizontal auxiliary rods, one for each incident edge in $G'$. We call these rods connectors as they are used to connect vertex gadgets to edge gadgets. The connector corresponding to an incoming edge is called an incoming connector, the connector corresponding to an outgoing edge is called an outgoing connector.

A gadget for vertices of degree six thus has six connectors: two placed above $A$ at either end, two placed below $B$ at either end, one placed either above or below the end of $M$, and one placed either below $B$ or above $A$ in the middle. The exact placement of the connectors and whether they are above or below $A$, $B$, or $M$ depends on the directions of the edges incident to the vertex. Figure 4(b.1) and (b.2) show the two possibilities. Because all degree-6 vertices have two consecutive incoming (or outgoing) edges that are collinear—made possible by Lemma 2.1—these are indeed all possibilities, up to symmetries.

For vertices of degree four, the vertex gadget has four connectors—see Figure 4(c.1) and (c.2) for their placement. Recall that we made sure that the incident edges of a degree-2 vertex are not collinear, so the figure indeed shows all possibilities, up to symmetries.

Finally, in the case of vertices of degree two, the two connectors are placed as shown in Figure 4(d). The following lemma is easily verified.

**Lemma 2.2.** Let $S(v)$ be the set of rods in the gadget of a vertex $v$. Then there are no cycles in $G_\prec(S(v))$. Moreover, there is a path in $G_\prec(S(v))$ from every incoming connector to every outgoing connector, and all such paths can be destroyed by cutting the rod $M \in S(v)$ at a single point.

The single cut on $M$ that destroys all paths from incoming to outgoing connectors can always be made at the intersection of $M$ with either $A$ or $B$. Notice that this works because rods are 1-dimensional and do not include their endpoints, so that after cutting $M$ at its intersection with $B$, say, the two resulting pieces $M_1$ and $M_2$ are neither above nor below $B$. Also note that we can make all rods horizontal—and, hence, parallel to the $x$- or $y$-axis—because the rods in $S(v)$ do not make a cycle.

The edge gadget. Edge gadgets are used to connect vertex gadgets and create depth orders between them. A directed edge $e = (u, v)$ embedded as a rectilinear polyline consisting of segments $e_1, \ldots, e_k$ is represented by an edge gadget consisting of rods $s_1, \ldots, s_{k+2}$. Each $s_i$ has an $xy$-projection that contains $e_i$, but $s_i$ is slightly longer at each end. Note that all these rods are parallel to the $x$- or $y$-axis. We pick one rod $s_1$ arbitrarily and replace it by three rods arranged as the first three rods in Figure 4(a). One of these rods, whose $xy$-projection has an orientation of 45°, is very steep (meaning that it is almost parallel to the $z$-axis). In our figures, the steep rod has an elevation that increases from left to right. This steep rod allows us to “change height” along an edge. At each bend where $e_i$ intersects $e_{i+1}$ we place $s_i$ above $s_{i+1}$. Thus the edge $(u, v)$ is represented by the path $s_1 \prec \cdots \prec s_{k+2}$. We denote the set of rods in the gadget for $e$ by $S(e)$.

Connecting the gadgets. Consider a directed edge $(u, v)$ with corresponding edge gadget consisting
of rods $s_1, \ldots, s_{k+2}$. We need to connect the vertex gadget for $u$ to the vertex gadget for $v$ using this edge gadget. Namely, we place $s_1$ so that it is above the appropriate connector of the vertex gadget of $u$, and we place $s_{k+2}$ so that it is below the appropriate connector of the vertex gadget of $v$—see Figure 3. Placing $s_1$ above its connector and $s_{k+2}$ below its connector is possible because we can change the height along the edge connecting them, as explained above.

Let $S$ be the set of all rods used in the construction of all the gadgets. Clearly $|S| = O(|V| + |E|) = O(|V|)$.

**Lemma 2.3.** The set $S$ admits a complete cut set of size at most $k$ if and only if the graph $G = (V, E)$ admits a vertex cover of size at most $k$.

**Proof.** Let $C$ be a complete cut set for $S$. We construct a vertex cover $V^*$ for $G$ by putting one vertex into $V^*$ for each cut point in $C$. Namely, if a cut point lies on a rod of the vertex gadget of some vertex $v$ we put $v$ into $V^*$, and if the cut point lies on a rod of the edge gadget of some edge $(u, v)$ we put $u$ into $V^*$. Now consider an arbitrary edge $(v, w) \in E$. Then there is a 2-cycle $(v, w, v)$ in the directed graph $G'$, and by construction there is a cycle $\Psi$ in $G_\infty(S)$ consisting of rods from $S(v) \cup S((v, w)) \cup S(w) \cup S((w, v))$. Since there must be a cut point on a rod in the cycle $\Psi$, this implies we have put $v$ or $w$ into $V^*$. Hence, $V^*$ is a vertex cover of size at most $|C|$. Conversely, a vertex cover $V^*$ of $G$ can be turned into a complete cut set for $S$ of size $|V^*|$ by putting for each vertex $v \in V^*$ a cut point on the rod $M \in S(v)$ such that all paths from incoming connectors to outgoing connectors are destroyed. Since $V^*$ is a vertex cover, this gives us a complete cut set for $S$. \hfill \Box

Since PLANAR VERTEX COVER is NP-hard, we conclude that CYCLIC OVERLAP REMOVAL is NP-hard as well. In fact, CYCLIC OVERLAP REMOVAL is NP-complete because it is easy to verify in polynomial time [5] whether a given cut set is complete.

**Corollary 2.1.** CYCLIC OVERLAP REMOVAL is NP-complete, even if the rods only have three distinct orientations.

### 3 Approximation algorithm

Recall that the FEEDBACK VERTEX SET problem on a directed graph $G = (V, E)$ asks for a subset $V^* \subseteq V$ such that every cycle in $G$ is hit by (that is, contains) at least one vertex from $V^*$. We obtain our approximation algorithm for CYCLIC OVERLAP REMOVAL by transforming the set $S$ of input rods into a directed graph $\hat{G}$ and then running an existing approximation algorithm for FEEDBACK VERTEX SET on $\hat{G}$. The approximation factor we obtain is twice as high as the approximation factor given by the FEEDBACK VERTEX SET approximation algorithm.

To assist in constructing the graph $\hat{G}$ we first construct an intermediate graph $\tilde{G} = (\tilde{V}, \tilde{E})$, as follows—see Figure 5(a)–(c). Vertically project all rods in $S$ onto the $xy$-plane. Let $\overline{\mathfrak{S}}$ denote the set of projected rods that intersect at least two other projected rods—the rods with zero or one intersection can safely be ignored\(^2\).

Consider a rod $s$ with projection $\pi \in \overline{\mathfrak{S}}$. We let $v_1(s), \ldots, v_5(s)$ denote the set of intersection points

---

\(^2\)We assume there are no degeneracies in the projections of the rods. That is, no three rods project to the same point and no two rod projections overlap. These assumptions can be removed by a simple modification of the algorithm.
on \( \overline{\pi} \) with the other projected rods, ordered along \( \overline{\pi} \). We create a vertex \( v_i(s) \) for each \( v_i(s) \), and let \( \hat{V}(s) = \{v_1(s), \ldots, v_k(s)\} \) denote the set of vertices created for \( s \). The vertex set \( \hat{V} \) of \( \hat{G} \) is now defined as \( \hat{V} := \bigcup_{s \in S} \hat{V}(s) \).

Next we define the edge set \( \hat{E} \) of \( \hat{G} \). To this end, let \( s, t \) be a pair of rods with \( s < t \). Then \( \overline{\pi} \) and \( \overline{\tau} \) intersect, and there are points \( v_i(s) \) and \( v_j(t) \) such that \( v_i(s) = v_j(t) = \pi \cap \tau \). We put the edge \( (v_i(s), v_j(t)) \) into \( \hat{E} \). Since this edge connects vertices corresponding to different rods, we call it an external edge. We call \( v_i(s) \) an up vertex and \( v_j(t) \) a down vertex, because we can step up from \( s \) to \( t \) (or down from \( t \) to \( s \)) where \( \pi \) intersects \( \tau \). For every rod \( s \), we add to \( \hat{E} \) edges from each of its down vertices to each of its up vertices. These edges are called internal edges.

**Lemma 3.1.** There is a one-to-one correspondence between the cycles in \( G_<(S) \) and the cycles in \( \hat{G} \).

**Proof.** Let \( \Psi \) be a cycle in \( G_<(S) \). Let \( s, t, u \) be three consecutive rods in \( \Psi \). Hence \( s < t < u \). Then there exist \( v_i(s) \) and \( v_j(t) \) in \( \hat{G} \) corresponding to the intersection \( \overline{\pi} \cap \overline{\tau} \), and there is an edge \( (v_i(s), v_j(t)) \) in \( \hat{G} \) by construction. Similarly, there are vertices \( v_j(t) \) and \( v_k(u) \) such that \( (v_j(t), v_k(u)) \) is an edge in \( \hat{G} \). Finally, since \( v_j(t) \) is an down vertex and \( v_k(t) \) is an up vertex, we also have an edge from \( v_j(t) \) to \( u \). Since \( s, t, \) and \( u \) are arbitrary vertices in \( \Psi \), we have shown that \( \hat{G} \) has a cycle \( \Psi' \) corresponding to the cycle \( \Psi \) in \( G_<(S) \).

Conversely, consider a cycle \( \Psi' \) in \( \hat{G} \). Note that internal edges always go from down vertices to up vertices, while external edges always go from up vertices to down vertices. Hence, \( \Psi' \) consists of an alternating cycle of internal and external edges. For every external edge \( (v_i(s), v_j(t)) \) we have \( s < t \), so indeed \( \Psi' \) corresponds to a unique cycle \( \Psi \) in \( G_<(S) \). \( \square \)

Now consider a vertex \( v_i(s) \). By cutting the rod \( s \) at the point projecting onto \( v_i(s) \), we destroy all cycles in \( G_<(S) \) that correspond to cycles in \( \hat{G} \) containing \( v_i(s) \). Combined with Lemma 3.1 this implies that a subset \( V^* \subset \hat{V} \) that hits every cycle in \( \hat{G} \) corresponds to a complete cut set for \( G_<(S) \). Unfortunately this is not good enough, because it may take many fewer cuts to obtain a complete cut set than it takes to hit every cycle in \( \hat{G} \). The reason is that cutting a rod \( s \) at the intersection with some other rod \( t \) destroys not only the cycles involving the edge \( (s, t) \), but also the cycles containing a triple \( u < s < v \) where \( \pi \cap \tau \) and \( \pi \cap \tau \) lie on opposite sides of \( \pi \cap \tau \).

To overcome this problem we modify \( \hat{G} \) to obtain the final graph \( G \). Let \( V(s) = \{v_1(s), \ldots, v_k(s)\} \) be the set of vertices created for \( s \). We create two more sets of vertices for \( s \), namely \( V^-(s) := \{v_1^-(s), \ldots, v_{k-1}^-(s)\} \) and \( V^+(s) := \{v_1^+(s), \ldots, v_{k-1}^+(s)\} \). Next we replace the internal edges we had for \( s \)—the edges which connected down vertices in \( \hat{V}(s) \) to up vertices in \( \hat{V}(s) \)—by a collection of edges \( E(s) \), as explained next and illustrated in Figures 6 and 5(d). First, we put the edges \( (v_i^+(s), v_{i+1}^-(s)) \) into \( E(s) \), for \( 1 \leq i < k-1 \). Next, we put the edges \( (v_i^+(s), v_{i-1}^-(s)) \) into \( E(s) \), for \( 1 < i < k-1 \). Finally, for each \( v_i(s) \) we add at most two more edges to \( E(s) \): If \( v_i(s) \) is an up vertex we add \( (v_{i-1}^-(s), v_i(s)) \) (provided \( i > 1 \)) and \( (v_i^-(s), v_i(s)) \) (provided \( i < k \)); if \( v_i(s) \) is a down vertex we add \( (v_i(s), v_i^+(s)) \) (provided \( i < k \)) and \( (v_i(s), v_{i-1}^-(s)) \) (provided \( i > 1 \)). We repeat this procedure for every rod \( s \). This gives us our final graph \( G \). (The external edges in \( G \) are the same as in \( \hat{G} \).)

**Lemma 3.2.** There is an internal edge \( (v_i(s), v_j(t)) \) in \( \hat{G} \) if and only if there is a path from \( v_i(s) \) to \( v_j(t) \) consisting of internal edges in \( G \).

**Proof.** There is a directed edge \( (v_i(s), v_j(t)) \) in \( \hat{G} \) if and only if \( v_i(s) \) is a down vertex and \( v_j(t) \) is an up vertex. There are two cases: \( i < j \) and \( j < i \). If \( i < j \) then there is an edge from \( v_i(s) \) to \( v_j^+(s) \), a path from \( v_j^+(s) \) to \( v_{j-1}^+(s) \), an edge from \( v_{j-1}^+(s) \) to \( v_j(t) \). If \( j < i \) then there is an edge from \( v_i(s) \) to \( v_{i-1}^-(s) \), a path from \( v_{i-1}^-(s) \) to \( v_j^-(s) \), an edge from \( v_j^-(s) \) to \( v_j(t) \).

Assume there is a directed path from \( v_i(s) \) to \( v_j(t) \) in \( G \) made up of internal edges. Note that in \( G \), all down
vertices are sources and all up vertices are sinks with respect to internal edges. This implies that a directed path from \( v_i(s) \) to \( v_j(s) \) can only occur if \( v_i(s) \) is a down vertex and \( v_j(s) \) is an up vertex. Therefore, there must be an edge from \( v_i(s) \) to \( v_j(s) \) in \( \tilde{G} \).

This, along with Lemma 3.1, implies that all cycles in \( G_{\prec}(S) \) correspond to cycles in \( G \) and vice versa—see Figure 7. The idea is that removing \( v_i^+(s) \) and \( v_i^-(s) \) from \( G \) corresponds to making a cut on the rod \( s \) at the point projecting to \( v_i(s) \).

**Lemma 3.3.** Let \( \tau \) be the minimum size of a complete cut set \( C \) for \( S \). Then there is a set \( V^* \) containing at most \( 2\tau \) vertices of \( \tilde{G} \) that hits every cycle in \( G \).

**Proof.** We will construct a set \( V^* \) with the required properties. Let \( p \) be a cut point in \( C \) and let \( s \) be the rod on which \( p \) lies. We may assume that \( p \) lies at the intersection of \( \pi \) with some other projected rod \( \tilde{t} \)—indeed, a cut point can always be moved to the nearest intersection point without creating a cycle. Thus we have a node \( v_i(s) \) in \( G \) corresponding to \( p \). If \( v_i(s) \) is an up vertex we put the vertices \( v_i^+(s) \) and \( v_i^+(s) \) (provided \( i > 1 \)) into \( V^* \), and if \( v_i(s) \) is a down vertex we put the vertices \( v_i^+(s) \) (provided \( i > 1 \)) and \( v_i^+(s) \) into \( V^* \). Doing this for all cut points \( p \in \hat{C} \), we obtain a set \( V^* \) with at most \( 2\tau \) vertices.

We claim that \( V^* \) hits every cycle in \( \tilde{G} \). To see this, consider a cycle \( \Psi \) in \( \tilde{G} \). By Lemmas 3.1 and 3.2 there is a corresponding cycle \( \tilde{\Psi} \) in \( G_{\prec}(S) \). Because \( C \) is a complete cut set, there will be a cut point \( p \in C \) placed on some rod \( t \) participating in \( \tilde{\Psi} \) that breaks the cycle. Let \( s \) and \( u \) be the rods just before and after \( t \) in \( \tilde{\Psi} \). Hence \( s < t < u \). Let \( \tilde{v}_i(t) \) correspond to \( \tilde{t} \cap \pi \) and \( \tilde{v}_j(t) \) correspond to \( \tilde{t} \cap \pi \). Assume \( i < j \); the case \( i > j \) is similar. Then \( p \) is placed at some point \( v_i(t) \) for \( i < \ell < j \). Hence \( v_i^+(t) \in V^* \) or \( v_i^-(t) \in V^* \) depending on whether \( v_i(t) \) is up or down, which implies the cycle \( \Psi \) is hit by \( V^* \).

**Lemma 3.4.** Let \( V^* \) be a set of vertices in \( G \) that hits every cycle in \( \tilde{G} \). From \( V^* \) we can construct a complete cut set for \( S \) of size at most \(|V^*|\).

**Proof.** Consider an arbitrary vertex \( v \in V^* \). Then there exists a rod \( s \in S \) such that \( v \in \tilde{V}(s) \cup V^+(s) \cup V^-(s) \). Each vertex in \( \tilde{V}(s) \cup V^+(s) \cup V^-(s) \) corresponds to an intersection point of \( \pi \) with some other projected rod \( \tilde{t} \). We cut \( s \) at this intersection point. Let \( C \) be the collection of cut points obtained in this manner. The size of \( C \) is at most \(|V^*|\). Since \( V^* \) covers all of the cycles in \( G \), and every cycle in \( G \) for \( S \) corresponds to a cycle in \( G \), \( C \) is a complete cut set for \( S \).

**Theorem 3.1.** Let \( S \) be a set of \( n \) disjoint rods in \( \mathbb{R}^3 \). Then we can compute a complete cut set for \( S \) of size \( O(\tau \log \tau \log \log \tau) \), where \( \tau \) is the minimum size of a complete cut set for \( S \). The running time of our algorithm is \( O(n^3M(n^2)\log^2 n) \), where \( M(n) \) denotes the complexity of multiplying two \( n \times n \) matrices.

**Proof.** We run an approximation algorithm for Feedback Vertex Set [7] on \( G \). The cited algorithm produces a feedback vertex set with an approximation factor of \( O(\log \tau \log \log \tau) \). Its running time is dominated by \( O(|V|^2M(|V|)\log^2 |V|) \) [8], where \(|V| = O(n^2)\) in our case, since we add a vertex for every intersection between rods. The size of the optimal Feedback Vertex Set for \( G \) is at most \( 2\tau \) by Lemma 3.3. Hence using Lemma 3.4 we can produce a complete cut set for \( S \) of size \( O(\tau \log \tau \log \log \tau) \).

**4 Conclusions**

We have shown that cutting rods to remove cycles from their depth order is an NP-complete problem. We have also given an approximation algorithm that finds a set of cuts that eliminates all cycles in the depth order of a set of rods. Our algorithm has an approximation factor that is at most twice the best approximation factor known for the Feedback Vertex Set problem.

Many open problems in this area remain. Foremost is whether we can cut cycles of three-dimensional objects that are not rods. Our NP-hardness proof crucially uses the fact that all interactions between two objects can be removed by making one cut. With objects that are not rods, this is not the case—the object that is not cut will interact with at least one of the fragments of the cut object. This requirement is not as important to
our approximation algorithm, but here we use the idea that intersections can be ordered along rods. Therefore, extending our approximation algorithm to more general objects does not seem easy either.

References


