The Three Dimensional Logic Engine

Matthew Kitching[1][⋆] and Sue Whitesides[2][⋆⋆]

[1] Dept. of Comp. Sci., U. of Toronto, Canada M5S 3G4 kitching@cs.toronto.edu
[2] School of Comp. Sci., McGill U., Montreal, Canada H3A 2A7
sue@cs.mcgill.ca

1 Abstract

We consider the following graph embedding question: given a graph G, is it possible to map its vertices to points in 3D such that G is isomorphic to the mutual nearest neighbor graph of the set P of points to which the vertices are mapped? We show that this problem is NP-hard. We do this by extending the "logic engine" method to three dimensions by using building blocks inpired by the structure of diamond and by constructions of A.G. Bell and B. Fuller.

2 Introduction

Proximity graphs are an important and well studied area of computer science and find applications, for example, in architecture, pattern recognition, and geography. Proximity graphs are defined to capture some kind of spatial relationship between pairs of points on the plane or in space. Two points, regarded as vertices of a graph, are joined by an edge if, and only if, the points satisfy some given proximity criterion. Examples of proximity graphs include mutual nearest neighbour graphs, Gabriel graphs, and the Delauney triangulation. For a review of proximity graphs, see [11, 16].

Given an abstract combinatorial graph whose vertices are labeled, and a proximity criterion, the *recognition problem* is to determine whether there is some set P of points, typically required to lie in 2D or 3D, such that the graph is isomorphic to the proximity graph on P defined by the given proximity criterion. The *realization problem* is to produce such a set P if one exists. This paper proves that the problem of recognizing mutual nearest neighbour graphs in 3D is NP-hard, an open problem in graph drawing (see [4]). Our proof builds a 3D version of a "logic engine". The building blocks we designed are based on the structure of diamond; they may prove useful in extending the logic engine approach to obtain complexity results for other 3D layout and proximity problems studied previously in two dimensions (e.g., [3, 4, 6, 7, 12–14]).

The rest of this section contains background material. Section 3 extends the logic engine approach to 3D, using the *octet truss* of Buckminster Fuller, and gives our NP-hardness result. Section 4 concludes.

[⋆] Research undertaken while in the M.Sc. program in the School of CS at McGill U.
[⋆⋆] Partially supported by NSERC and FCAR.

Preliminaries A *mutual nearest neighbour graph* of a set P of points is a proximity graph for which each pair x,y of vertices arising from points x,y is connected by an edge if, and only if, point x is a nearest neighbor of y, and point y is a nearest neighbor of x. The set P of points typically lies in 2D or 3D. Note that in 2D, points x,y determine an edge if the interior of the union of the two discs centred at x and y, and having radius equal to their separation distance, is empty of points other than x and y.

A simple example of a 3D mutual nearest neighbor graph is given by the combinatorial structure of the vertices and edges of the regular tetrahedron. The four vertices of the regular tetrahedron of unit edge length are positioned at points in space so that each point is unit distance from each of the other three. Thus each pair of points gives rise to an edge in the mutual nearest neighbor graph of the points, which is thus K_4, the complete graph on four vertices.

As we will later see, if we start with a combinatorial graph K_4 whose vertices are labeled and ask whether it can be realized in 3D as the mutual nearest neighbor graph of some set P of four points, we find that there are exactly two realizations, up to translation, rotation, and scaling, and that these two realizations are mirror images of each other.

Similarly, the vertex-edge incidence structure of the regular octahedron can be thought of as a 3D mutual nearest neighbor graph, and the combinatorial graph has exactly two realizations, up to translation, rotation, and scaling.

In the early 1900's, Alexander Graham Bell used rigid tetrahedra and octahedra to construct kites, an unsuccessful flying machine, and a tall tower [1]. The structures that Bell assembled are mutual nearest neighbour graphs, as we will prove. These structures were later rediscovered by Buckminster Fuller [2], who patented the *octet truss* and used it extensively.

Mutual Nearest Neighbour Graph Recognition (MNNGR) Given an undirected graph G, is G realizable as a mutual nearest neighbour graph?

For 2D, MNNGR was proved NP-Hard by Eades and Whitesides [9], by a reduction from Not-All-Equal-3-Satisfiability (NAE3SAT) to MNNGR via a method they called the "logic engine" approach, reviewed below. Recall that NAE3SAT is NP-complete and that an instance consists of m clauses each containing three distinct literals, and that a satisfying assignment must contain at least one true and at least one false literal in each clause ([10]). It may be assumed that no clause contains both a literal and its complement.

The Logic Engine Approach The logic engine is a virtual mechanical device that encodes instances of NAE3SAT. The device can be positioned a certain way in the plane if and only if the instance of NAE3SAT that it encodes can be satisfied. The idea for obtaining hardness results for proximity graph recognition problems is to design a graph whose only possible realizations imitate the correctly positioned mechanical device.

The (m,n) logic engine contains a rigid "frame" and a "shaft". To the shaft are attached a series of "armatures" $A_j, 1 \leq j \leq n$, one for each literal x_j in the instance of NAE3SAT. Each armature can rotate about the shaft independently of the others, although the position of each armature along the shaft is fixed.

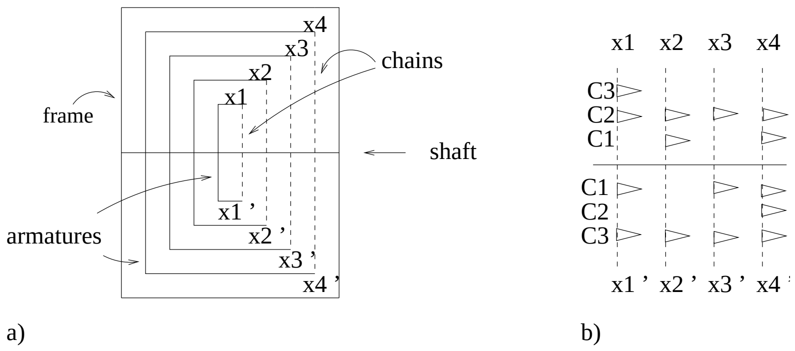

Fig. 1. a) Schematic for a (3,4) logic engine, and b) Encoding for NAE-3SAT instance $c_1 = \{x_1, x_2^{'}, x_3\}$, $c_2 = \{x_1^{'}, x_2^{'}, x_3^{'}\}$, $c_3 = \{x_2, x_3, x_4\}$

Each armature A_j in turn has two "chains" attached to it: a_j from one end of the armature to the shaft, and $a_j^{'}$ from the other end of the armature to the shaft. The length of a chain equals the distance between the shaft and the ends of its armature. Each chain has at least m links, which are numbered 1,2,...,m, outward from the shaft; the first m links correspond to clauses with the same indices, respectively.

The a_j chain of armature A_j represents the uncomplemented literal x_j, and its i^{th} link represents the possible occurrence of x_j in clause c_i; similarly for link i of chain $a_j^{'}$ and the possible occurence of the complemented literal $x_j^{'}$ in c_i. A "flag" attached to link i of armature chain a_j indicates that x_j does NOT occur in clause c_i; similarly, a flag on link i of chain $a_j^{'}$ indicates the non-occurence of $x_j^{'}$ in c_i. See Figure 1 for a (3,4) logic engine encoding of a NAE-3SAT instance.

The entire structure can move in the following ways: each armature can lie in one of two positions, either with a_i above the shaft, or with $a_i^{'}$ above the shaft; and each flag can face to the right, or can be flipped to face the left.

Although each flag is free to rotate, flags that lie on the same row and lie on adjacent armatures must not face one another. If they do face each other, the flags *collide*. Similarly, any flag in armature A_n collides with the frame if it faces outward; any flag in armature A_1 collides with the frame if it faces inward. We later use the following lemma from [9].

Lemma 1. *A given instance of NAE3SAT has a satisfying solution if, and only if, there exists a collision-free configuration for the logic engine.*

3 Mutual Nearest Neighbor Graphs in Three Dimensions

Here we prove the NP-hardness of MNNGR in 3D, settling a problem from [4]. While the result was anticipated in [9], this is the first concrete proof. precisely.

3D Nearest Neighbour Rule: Vertex v_i is a nearest neighbour of v_j if, and only if, the open sphere of radius $d(v_i, v_j)$ around v_j contains only v_j.

3D Mutual Nearest Neighbour Graph: Suppose that P is a set of points in 3D. Then a 3DMNNG G, defined by a set P of points, is an undirected graph with a vertex v_i for every point $p_i \in P$. For every pair of vertices $v_i, v_j \in G$, there is an edge between them if, and only if, v_i is a 3D nearest neighbour of v_j, and v_j a 3D nearest neighbour of v_i.

A graph G is *realizable* as a **3DMNNG** if for some point set P in 3D, the 3DMNNG on P is isomorphic to G. In that case we often use "G" to denote both the combinatorial graph and its geometric realization in 3D, and we use the same labels for corresponding points and vertices.

3D Mutual Nearest Neighbour Graph Recognition (3DMNNGR): Given an undirected connected graph G, is G realizable as a 3DMNNG?

We will use the logic engine paradigm to transform NAE3SAT to 3DMNNGR in polynomial time, thus showing that 3DMNNGR is NP-hard.

Lemma 2. (straight-forward from Lemma 2 of [9]) *Suppose that G is a connected 3DMNNG. Then all edge segments of G have the same length.*

From now on, we assume all edges in a connected 3DMNNG have *unit* length.

Lemma 3. (by Lemma 2) *Suppose that H is an induced connected subgraph of a combinatorial graph G. Then any 3DMNNG realization of G includes a 3DMNNG realization of H.*

Two realizations of a labeled graph are "the same" if, following possible translation, rotation, and scaling, vertices with the same label coincide. Thus, all mirror images of a given labeled structure are the same in this sense. However, mirror images are not, in general, the same as the initial labeled structure. Taking the mirror image of a vertex-labeled tetrahedron turns it inside out; the mirror image cannot be superimposed on the original, with labels matching, by translation and rotation.

In the next lemma, and throughout the paper, we let h = $\sqrt{6}/3$, which is the distance from the base of a tetrahedron to the top, if all edges are unit length.

Lemma 4. (straight-forward) *The labeled graph K_4 has exactly two realization as a 3DMNNG, both of which are regular tetrahedra.*

Once we prove a labeled combinatorial graph has exactly two realizations as a 3DMNNG, we use the term "graph" to refer to either realization.

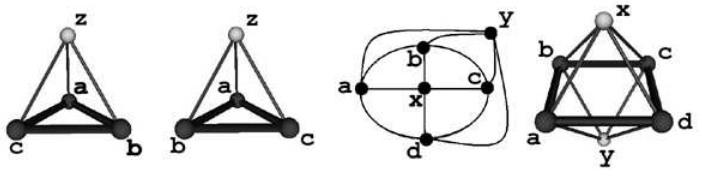

Fig. 2. a) and b) Two realizations of labeled K_4; c) and d) Labeled graph H and one of its realizations as an octahedron

Lemma 5. (straight-forward) *Let H be a labeled graph isomorphic to the combinatorial structure of Figure 2c). Then H has exactly two realizations as a 3DMNNG, namely a regular octahedron and its mirror image.*

Lemma 6. *Let H be the labeled combinatorial graph (called an* ***octet truss****) arising from the geometric structure in Figure 3a). H can be realized as a 3DMNNG in exactly two ways: points a,b,c,d,e,f must lie on a single* ***base plane****, while points u,v,w must lie on a parallel* ***lid plane*** *at a distance h from the base plane.*

Proof. By Lemma 2, all edges must have unit length. By Lemma 5, the points b,c,e,u,v,e construct a regular octahedron. When the plane of b,c,e is viewed from above, then the plane of u,v,w is parallel to it, and may lie on either side of it. (In Figure 3a, the plane of u,v,w is shown closer to the viewer than the plane of b,c,e). Now add the points a,d,f to the graph. These points become members of tetrahedra. Consider each of a,d,f in turn. Although by Lemma 4, a labeled tetrahedron has two realizations, one of these would place the point (a,d or f) inside the octahedron and violate a distance constraint: no vertex in a connected MNNG can lie distance less than one from another vertex. Hence the remaining points lie in the plane of b,c,e as shown. □

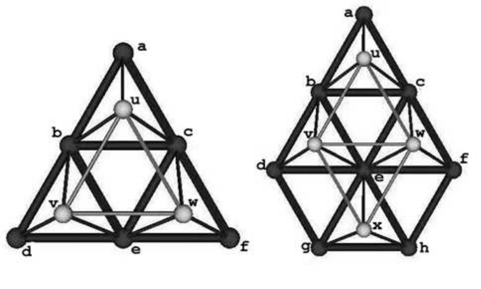

Fig. 3. a) Octet truss and b) Extended octet truss

A **base plane** is defined by the six coplanar vertices of an octet truss (in Figure 3a, the plane of a,b,c,d,e,f). The remaining three vertices of the truss define the **lid plane** (in Figure 3a, the plane of v,u,w). A **base vertex** is any one of the six vertices on the base of an octet truss. A **lid vertex** is any one of the three vertices on the lid of an octet truss. The plane parallel to the base plane, at distance 4h on the same side of the base plane as the lid plane, is the **mid-plane**. The plane parallel to the base plane, at distance 8h on the same side of the base plane as the lid plane, is the **sky plane**.

Lemma 7. *Let H be the labeled combinatorial graph arising from the geometric structure in Figure 3b) (called the* ***hexagonal octet truss****). H can be realized in exactly two ways. Furthermore, u,v,w and x are coplanar, and the remaining vertices are coplanar, and these planes are parallel.*

Proof. By Lemmas 3 and 6, the subgraph induced by vertices a,b,c,d,e,f,u,v,w must be realized as one of two octet trusses, with the lid plane on either side of the base plane. Point x lies unit distance from each of v,e,w, with which it forms a regular tetrahedron. Since it cannot lie inside the octahedron b,c,e,u,v,w, it must lie as shown in Figure 3b), in the lid plane of the octet truss.

Point g must lie on a circle perpendicular to d,e, centred at the mid-point of d,e. Likewise, g must lie on a circle centred at the mid-point of x,e. The circles intersect in two points, so g must lie either as shown in 3b), or at the position occupied by vertex v, which is clearly not allowed. Similarly for point f. □

The **link graph**, the **flagged link graph** and the **k-tower** are the labeled graphs having the combinatorial structures shown in Figures 4 a), b), and c), respectively. All three graphs can be realized as 3DMNNG's. The next lemma proves that each graph has exactly two realizations, namely, the realizations shown in the figure, together with their mirror images. We also define the *direction* of a link graph realization, or flagged link graph realization, to be the vector of the directed line segment from the s to the t vertices of the realization. The line defined by the points s and t is called the **s-t axis**.

Lemma 8. *The link graph, flagged link graph, and k-tower each have only two realizations.*

Proof. The combinatorial structure of the k-tower has the form $H_1 \cup ... \cup H_k$, where H_i is an octet truss, and for i odd, $H_i \cap H_{i+1}$ consists of three shared lid vertices, and for i even, $H_i \cap H_{i+1}$ consists of six shared base vertices. By Lemma 6, each H_i has two realizations. However, once a realization is chosen for an octet truss, say H_1, the remaining realizations are determined. Thus a k-tower has exactly two realizations, which are mirror images of each other.

The link graph and flagged link graph also have two realizations. By Lemma 7, the position of each vertex, with the exception of u, v, and t, is determined once a realization of the octet truss containing s is determined, since all vertices are connected by extended octet trusses. To place u,v,t, note that points b,w,c,u,v,t form a regular octahedron for which there are exactly two realizations by Lemma 5. However, one of these realizations violates a distance constraint with respect to the rest of the graph, so the octahedron is as shown in Figure 5. □

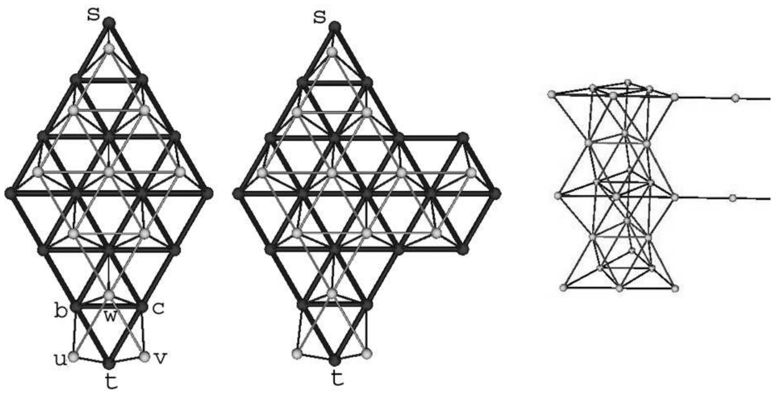

Fig. 4. a) Link graph b) Flagged link graph c) Tower with mid-wire and sky-wire

The **k-flagged link graph** is a sequence of k link graphs or flagged link graphs joined together as shown in Figure 5. Each link graph in the k-flagged link graph is called a **block**. Since the 3DMNNG realization of a flagged link graph has exactly two realizations, which are mirror images of each other, the Euclidean distance from vertex s to vertex t is a constant, which we denote by $d_{flagged\_link}$.

Lemma 9. (from the definition of $d_{flagged\_link}$) *If the 3DMNNG realization of a k-flagged link graph spans a distance of $k \cdot d_{flagged\_link}$, then the s-t axis of all link graphs and flagged link graphs must coincide.*

Such a realization is a **taut realization** of a k-flagged link graph.

Lemma 10. *In a taut realization of a k-flagged link graph, the base planes of all the flagged link graph realizations must be the same, and similarly, the lid planes must be the same.*

Proof. To show that all the base planes form a common plane, consider two joining link graphs X and Y (see Figure 5). Note that t_{x1},t_{x2},t_{y0},t form a regular tetrahedron. Fix block X in space, thereby determining the position of points t,t_{x1},t_{x2}, and therefore the position of t_{y0}. This prevents the rotation of block Y about the s-t axis. The midpoint of t_{y1} and t_{y2} must lie on the s-t axis at a known position. Points t_{y1} and t_{y2} must lie on a circle centred at the midpoint of t_{y1},t_{y2}, and also, on a circle centred at midpoint of t,t_{y0}, which is also a known position. These two distinct circles intersect in two points, namely the positions occupied by t_{y1} and t_{y2} in Figure 5. These positions lie in the base plane of block X; hence blocks X and Y have the same base plane. Since the position of t_{y0} is determined by block X, the lid plane of block Y is thus the same as the lid plane of block X. □

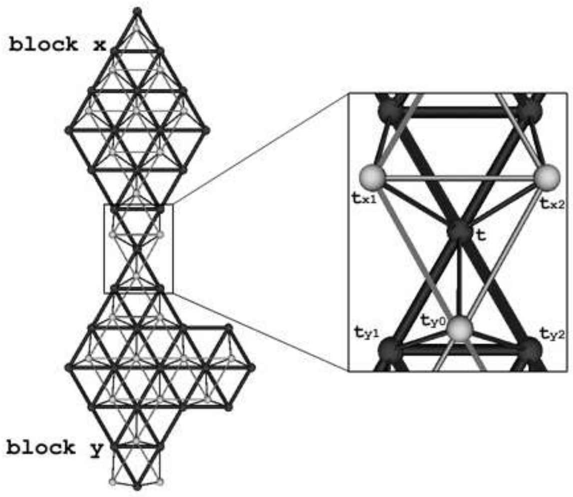

Fig. 5. 2-flagged link graph

Lemma 11. *Any taut realization of a k-flagged link graph has exactly* 2^k *realizations as a 3DMNNG.*

Proof. As seen in Lemma 10, a taut realization of a k-flagged link graph must have a common base plane and a common lid plane. However, any block can have two realizations, which are mirror images of each other. To see this, consider a realization of a block, and take its mirror image with respect to the plane containing the s-t axis and perpendicular to the base plane. This second realization has the same s-t axis, base plane, and lid plane as the first. □

Note that for blocks that are flagged link graphs, the two mirror images point in opposite directions. We will use the taking of mirror images to imitate rotations in the virtual logic engine.

Now, we build an (m,n) 3D logic graph, to imitate a logic engine, by constructing the following components, as seen in Figure 6. A **frame** consists of a series of octet trusses, together with two 8-towers (the locations of which are shown in gray in Figure 6). Attached to one side of the tower is a **mid-wire** at height 4h, and a **high wire** at height 8h (see Figure 4c). Each wire consists of a path of vertices and runs between both frame towers. By a similar argument to that in the proof of Lemma 9, we can ensure that all vertices of the mid-wire are colinear by forcing them to span a set distance to the other tower. The same is true for the sky wire.

Each of these wires intersects the towers of each of the armatures in a path of three vertices. As we will later prove, this ensures that the frame and all the armatures share a common base plane.

We define the π plane to be the plane perpendicular to the base plane, containing the mid-wire and the high wire.

An armature is built with a series of octet truss components forming three sides of a rectangle (see Figure 6). The last side consists of two m-flagged link graphs: a_i from one end of the armature to the π plane, and $a_i^{'}$ from the other end of the armature to the π plane. The armature also has a tower of height 8h intersecting the mid-wire and the high-wire in paths of three vertices lying on the π plane.

Lemma 12. *The frame component has two realizations, both of which have a single base, lid, mid, and sky plane. The armature component has a single base, lid, mid, and sky plane. The base planes of all armatures and the frame are coplanar. The same is true for the lid, mid, and sky planes.*

Proof. The frame is built with overlapping hexagonal octet trusses in such a way that all trusses must share a common base plane and lid plane. The towers of the frame intersect the rest of the frame in octet trusses, and this also determines the orientation of the towers. Similarly, within each armature the base, lid, mid, and sky planes are the same for all but the k-flagged link graph. Since the distance between the extreme vertices of the k-flagged link graph is determined by the rest of the armature to be $k \cdot d_{flagged\_link}$, Lemma 9 applies to the k-flagged link graph. The k-flagged link graph is connected to the rest of the armature with the same connection seen between flagged link graphs.

The intersections of each armature tower with the mid-wire and sky wire force the base plane and lid plane of the armature to be the same as the base plane and lid plane of the frame. □

An (m,n) 3D logic engine graph can now be constructed for any given instance of NAE3SAT.

Lemma 13. *In a 3DMNNG realization, two neighbouring flagged link graphs may not have flags facing one another.*

Proof. From Lemma 12, all the base planes of all armatures must be the same. If a flagged linked graph and its neighbouring flagged link graph point in opposite directions, then the two vertices at the tips of the flags must be unit distance apart in the logic engine. This would imply the two vertices would be connected in the combinatorial version of the graph, which is not the case. Thus, the flags must not point towards one another. Similarly, the flags on armatures A_1 and A_n must point away from the frame. □

Lemma 14. (by Lemma 13 and Lemma 1) *The (m,n) 3D logic graph can be customized to encode instances of NAE3SAT so that the logic graph is realizable if, and only if, the NAE3SAT instance is satisfiable.*

Lemma 15. (straight-forward) *There is a polynomial time transformation from NAE3SAT to 3DMNNG.*

Theorem 1 *(by Lemmas 14, 15 and the NP-completeness of NAE3SAT)* The 3DMNNGR problem is NP-hard.

4 Conclusion

The result that 3DMNNGR is NP-hard is not an immediate consequence of the fact that MNNGR is NP-hard in 2D, in part because of the difficulty pointed out by Fuller: regular tetrahedra do not fill space. We believe that the 3D building blocks seen here suggest that the logic engine approach indeed is applicable to three dimensional problems in graph drawing.

References

1. A.G. Bell, "Tetrahedral principle in kite structure", *National Geographic Magazine*, 14(6), June 1903, pp. 219-251.
2. R. Buckminster Fuller. Inventions, the Patented Works of R. Buckminster Fuller. St. Martin's Press, 1983.
3. P. Bose, W. Lenhart, and G. Liotta, "Characterizing proximity trees", *Algorithmica*, 16, 1996, pp. 83-110, 1996.
4. F.J. Brandenburg, D. Eppstein, M. T. Goodrich, S. G. Kobourov, G. Liotta, and P. Mutzel, "Selected open problems in graph drawing", *Proc. 11th Int. Symp. Graph Drawing, 2003*, Springer-Verlag LNCS vol. 2912, pp. 515-539.
5. G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis. Graph Drawing. Prentice Hall, 1999, Chapter 11.2.
6. M.B. Dillencourt, "Realizability of Delaunay triangulations", *Informa. Process. Lett.*, 33(6), Feb. 1990, pp. 283-287.
7. M.B. Dillencourt and W.D. Smith, "Graph-theoretical conditions for inscribability and Delaunay realizability", *Proc. 6th Canad. Conf. Comput. Geom.*, 1994, pp. 287-292.
8. P. Eades and S. Whitesides, "The logic engine and the realization problem for nearest neighbour graphs", *Theoretical Computer Science*, 169, 1996, pp. 23-37.
9. P. Eades and S. Whitesides, "The realization problem for Euclidean minimum spanning trees is NP-hard", *Algorithmica*, 16, 1996, pp. 60-82.
10. M. Garey and D. Johnson. Computers and Intractability: a Guide to the Theory of NP-Completeness. W.H. Freeman, 1979.
11. J.W. Jaromczyk and G.T. Toussaint, "Relative neighborhood graphs and their relatives", *Proc. IEEE*, 80(9), 1992, pp. 1502-1517.
12. W. Lenhart and G. Liotta, "The drawability problem for minimum weight triangulations", *Theoret. Comp. Sci.* vol. 27, 2002, pp. 261-286.
13. G. Liotta and G. Di Battista, "Computing proximity drawings of trees in the 3-dimensional space", *Proc. 4th Workshop Algorithms and Data Structures WADS 1995*, Springer-Verlag LNCS vol. 955, pp. 239-250.
14. G. Liotta, A. Lubiw, H. Meijer, and S.H. Whitesides, "The rectangle of influence drawability problem", *Comput. Geom. Theory Appl.*, 10(1):1-22, 1998.
15. G. Liotta and H. Meijer, "Drawing of trees", *Computational Geometry: Theory and Applications*, 24(3), 2003, pp. 147-178.
16. G. Toussaint, "A graph-theoretical primal sketch", in Computational Morphology. North-Holland, 1988, pp. 229-260.

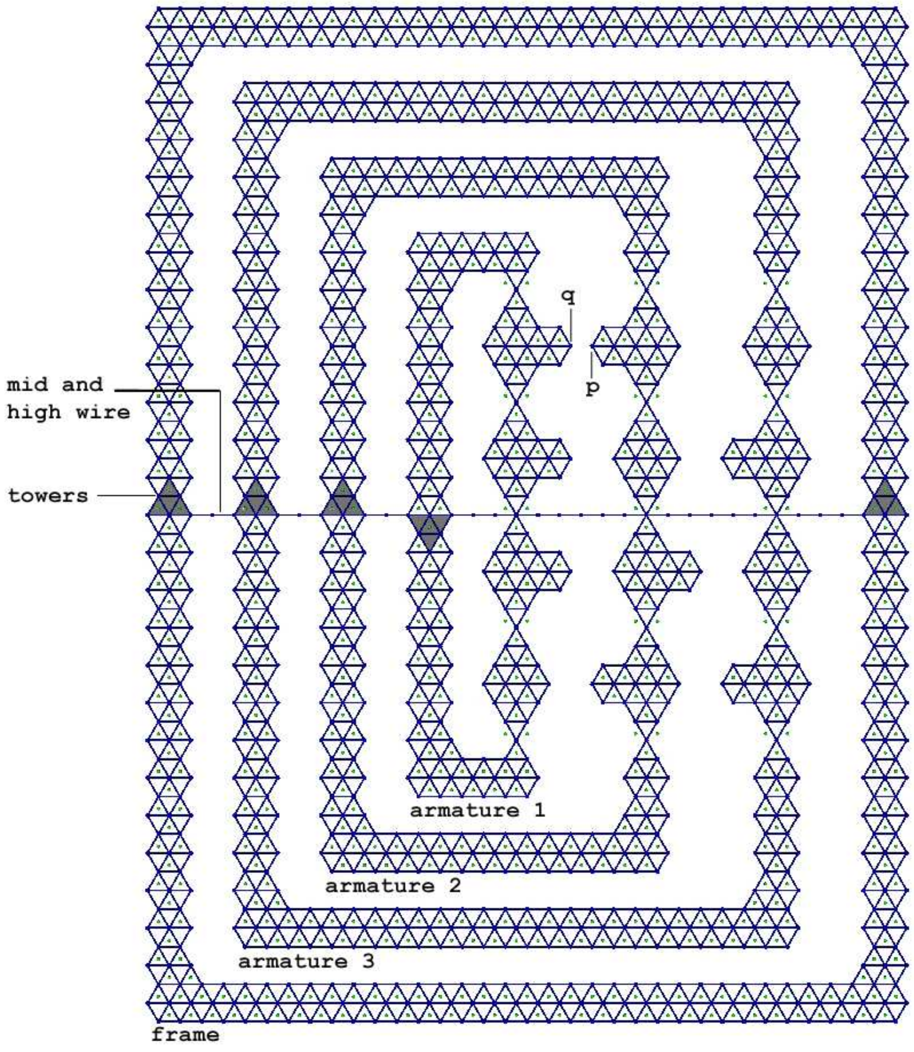

Fig. 6. A 2D projection of the entire logic engine. Towers project to the gray regions, and the mid-wire and sky-wire project to a common line. Edges between lid vertices are not shown, and edges between lid vertices and base vertices are not shown. There is a distance violation between vertices p and q; however a valid realization results when flag p is flipped to the right.