The Wobbly Logic Engine: Proving Hardness of Non-rigid Geometric Graph Representation Problems

(Extended Abstract) *

Sándor P. Fekete[1] **, Michael E. Houle[2] ***, Sue Whitesides[3] **†

[1] Center for Parallel Computing, Universität zu Köln
D–50923 Köln, GERMANY
sandor@zpr.uni-koeln.de
[2] Department of Computer Science, University of Newcastle
Callaghan NSW 2308, AUSTRALIA
mike@cs.newcastle.edu.au
[3] School of Computer Science, McGill University
3480 University St. #318, Montréal, Québec, CANADA
sue@opus.mcgill.ca

Abstract. The *logic engine* technique has been used in the past to establish the NP-hardness of a number of graph representations. The original technique can only be applied in those situations in which subgraphs exist for which the only possible layouts are rigid. In this paper we introduce an extension called the *wobbly logic engine* which can be used to prove the NP-hardness of several graph representations for which no such rigid layouts exist, representations by visibility and intersection in particular. We illustrate the method by using the wobbly technique to show the NP-hardness of deciding whether a graph has a nondegenerate z-axis parallel visibility representation (ZPR) by unit squares.

1 Introduction

The area of automated graph drawing has largely been shaped by its limitations. Many popular aesthetics for graph representation, such as the desire for planar layouts without edge crossings, are realizable only in the most restricted of application settings. Relaxation of the aesthetic considerations, such as allowing a small number of edge crossings, often leads to difficult (that is, NP-hard) computational problems.

With the advent of better architectures and software for graphics, it has become possible to circumvent some of these limitations by visualizing graphs

* A fuller version of this paper can be obtained at http://www.zpr.uni-koeln.de.

** Parts of this work were done while visiting the University of Newcastle, supported by a Visiting Researcher Grant.

*** Parts of this work were done while visiting the Universität zu Köln.

† Supported by NSERC.

in three dimensions instead of just two. For example, whereas edge-crossings are usually unavoidable in two dimensions, in a 3-dimensional setting they may generally be eliminated altogether. Also, by navigating through a 3-dimensional image, the user can often avoid views of the data which are aesthetically unappealing or confusing.

Although the 3-dimensional setting allows for a richer variety of graph representations, it is also true that the increase in dimensionality brings with it new mathematical and computational challenges. In order to open the way for good heuristics, or to justify the relaxation of some of the objectives, it is often a sensible first step to determine whether computation of the representation is an NP-hard problem.

One of the tools for establishing NP-hardness of graph realization problems is the so-called *logic engine*. It was first used by Bhatt and Cosmadakis [6] to show that it is an NP-complete problem to determine whether a tree of maximum degree 4 can be embedded in a planar grid, where tree vertices must be positioned at grid vertices and tree edges must occupy unit length grid edges. The logic engine was later employed to obtain lower bounds for checking whether a graph can be embedded in the plane as a proximity graph [14, 16, 17].

The technique can be summarized as follows (see the above references for more details; an overview will be contained in the upcoming book [13]). The reduction is from the NP-complete problem "Not-All-Equal-3SAT" (NAE-3SAT), whose instances consist of c 3-element clauses on s literals. "Yes" instances are ones with truth assignments such that each clause contains at least one true literal and at least one false literal ([22]).

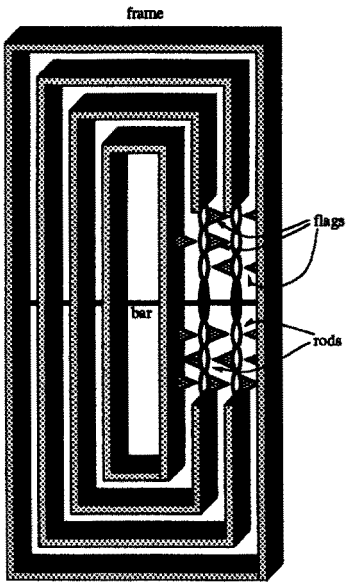

Fig. 1. A logic engine for the NAE-3SAT instance $(x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_3 \vee \overline{x}_4) \wedge (x_1 \vee x_3 \vee x_4)$. The shown position corresponds to the truth setting $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$.

For every instance of the NAE-3SAT problem, a machine can be built such that a non-self-intersecting layout of the machine in the plane exists, if and only if the answer to the corresponding NAE-3SAT instance is "yes". In its most general sense, the machine consists of a rigid frame with vertical left and right sides joined at their midpoints by a horizontal bar. Attached to this bar are vertical rods that can rotate independently about the bar. One half of each rod corresponds to some variable X and the other half to its complement X'. Each half-rod has c positions where flags may be placed. The flags may rotate independently about the vertical rods. Clauses are represented by pairs of rows of flag positions. Clause C_i is represented by the row i units above the bar and the row i units below the bar. The contents of each row depends on how the vertical rods are flipped around the bar. The region above the bar is interpreted as the "true" region, the region below is interpreted as the "false" region.

Once this has been established, it is simple to encode a particular instance: if a literal $Y = X$ or X' fails to appear in a given clause C_i, then a flag is positioned on the vertical rod for (X, X'), in the half corresponding to Y and at distance i from the bar. An unfilled position in the i^{th} row above the bar corresponds to the occurrence of a true literal in C_i; similarly, an unfilled position in the i^{th} row below the bar corresponds to the occurrence of a false literal in C_i. Rods and flags are to be turned and flipped so that flags on the left-most and right-most vertical rods point inward, and so that no flags positioned at the same distance i from the bar on two adjacent half-rods face each other. This can be done if and only if the NAE-3SAT instance is a "yes" instance.

To apply the logic engine to a geometric graph representation problem, one first has to find a special graph whose spatial embedding is "rigid"; that is, one in which relative positionings of points in the embedding is fixed. The frame, rods and flags of the logic engine are then constructed using this graph as a building block. The problem with this approach is that for a large number of representations, there are no graphs for which the relative positioning of any points in the embedding is necessarily fixed. For example, no geometric realization of an intersection graph of open squares is rigid, as any of the squares can be perturbed to some degree relative to the others without changing the intersection graph.

The main contribution of this paper is a variant of the logic engine which can be applied to graph representation problems in which no rigid embeddings are possible. In place of rigid structures, we show how "wobbly" equivalents can be built such that the variation in position of any element of the engine, relative to any other element, is bounded. More precisely, we consider particular elements, called *springs*, having upper and lower bounds on their size. By building an appropriate structure from several springs, we can force some of them to be stretched close to their upper limits, while some are compressed to their lower limits. As a consequence, the overall variation can be made arbitrarily small with respect to the size of the engine, in such a way that the movements of the wobbly logic engine mimic those of a rigid logic engine. These springs can be designed for many problems, allowing NP-hardness proofs by constructing wobbly logic engines.

In this paper, we apply the wobbly logic engine technique to intersection graph intersection problems, and visibility representation problems. In particular, in Sections 2 to 4, we give an NP-hardness proof for the problem of deciding whether a graph has a nondegenerate z-axis parallel visibility representation (ZPR) by unit squares. Section 3 describes properties of representations of a class of graphs called *k-extensors*. These graphs are used as springs in Section 4 for the construction of a wobbly logic engine. The paper concludes in Section 5 with a discussion of other applications of the wobbly logic engine.

2 A 3-dimensional Visibility Representation

The representation for which we establish NP-hardness is the so-called *z-parallel visibility representation* (ZPR) by unit squares, where vertices are represented by axis-aligned, z-orthogonal closed unit squares, and two vertices are adjacent if and only if the corresponding squares have a z-axis-parallel "cylinder of visibility". Also, we allow only *nondegenerate* layouts, in that no two edges of two different squares may have identical x- or y-coordinate.

We formally define what is meant by the term "cylinder of visibility". Given a square s, let $\pi(s)$ be the projection of the interior of s onto the xy-plane, and $\zeta(s)$ be its z-coordinate. Let s_1 and s_2 be squares such that $\zeta(s_1) < \zeta(s_2)$. For a cylinder of visibility to exist between s_1 and s_2, $\pi(s_1) \cap \pi(s_2)$ must contain a disk which does not intersect the projection $\pi(s)$ for any square s such that $\zeta(s_1) < \zeta(s) < \zeta(s_2)$.

This representation is a special case of the ZPR for rectangles introduced in [3], variants of which have been considered by a number of authors [1, 4, 5, 8, 19, 21, 23, 25]. Moreover, ZPR for unit squares was considered in [19, 5], where it was shown that K_8 has no layout, whereas K_7 has a nondegenerate layout.

A ZPR for a given graph can be quite useful for an overview of the structure of the graph. In addition, the rectangles representing the nodes can carry extra information by color, shape, or size; furthermore, the orientation of arcs in a directed acyclic graph can be represented by the order in z-direction of the rectangles involved. See [5] for a practical example for a complicated graph representing a software package [2].

From a practical point of view, it would be quite useful to have an efficient (that is, polynomial-time) algorithm that decides whether a given graph can be represented. We will show that the existence of such an algorithm is unlikely. More precisely, we show that the following decision problem is NP-hard:

PROBLEM *ZPR-NDUS*:

Given a graph $G = (V, E)$, does G admit a non-degenerate ZPR layout using unit squares?

In the remainder of the paper, we will often use the same notation for vertices of a graph and the unit squares which represent them. The context in which the notation is used will make it clear whether we are discussing the graph or its layout.

3 Springs for a 3-dimensional Visibility Representation

In this section, we describe a basic structure that is used to build springs for a wobbly logic engine, which will be used in the following section to prove the NP-hardness of Problem ZPR-NDUS.

An *extensor graph of k links* (k-extensor) is a tripartite graph of the form $G_k = (U_k, V_k, W_k, E_k)$, where

$$\begin{aligned} U_k &= \{u_0, u_1, \ldots, u_k\}, \\ V_k &= \{v_0, v_1, \ldots, v_{k-1}\}, \quad \text{and} \\ W_k &= \{w_0, w_1, \ldots, w_{k-1}\}. \end{aligned}$$

Edges of E_k are of the form

$$\begin{aligned} &(v_{i-1}, u_i), (w_{i-1}, u_i) \text{ for } 1 \leq i \leq k, \text{ and} \\ &(u_i, v_i), (u_i, w_i) \qquad \text{for } 0 \leq i \leq k-1. \end{aligned}$$

The vertices u_0 and u_k are called the *tabs* of G_k (see Figure 2.)

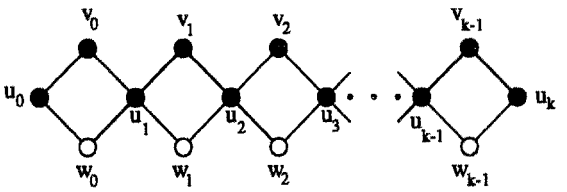

Fig. 2. A k-extensor.

Lemma 3.1 (Helly) *Let R be a collection of n axis-aligned open rectangles in the plane. If every pair of rectangles has a common intersection point, then there is a point common to all rectangles of R.*

Lemma 3.2 *Let G be a graph with the 2-extensor $G_2 = (U_2, V_2, W_2, E_2)$ as an induced subgraph. Furthermore, let the only edges joining vertices of $G \setminus G_2$ with those of G be incident to the tabs u_0 and u_2. Then for any valid non-degenerate layout of G, there exist axis-parallel lines l_0 and l_1 such that*

- *$\pi(u_0)$ is separable from $\pi(u_1)$ by l_0, and $\pi(u_1)$ is separable from $\pi(u_2)$ by l_1,*
- *l_0 and l_1 are parallel,*
- *$\pi(u_1)$ lies between l_0 and l_1, and*
- *all other axis-parallel lines separating $\pi(u_0)$ and $\pi(u_1)$ (or $\pi(u_1)$ and $\pi(u_2)$) are parallel to l_0 and l_1.*

Proof. Contained in the full version of the paper.

Lemma 3.3 *Let G be a graph with the k-extensor $G_k = (U_k, V_k, W_k, E_k)$ as an induced subgraph, for some $k \geq 2$. Furthermore, let the only edges joining vertices of $G \setminus G_k$ with those of G be incident to the tabs u_0 and u_k. Then for any valid non-degenerate layout of G, there must exist a set of axis-parallel lines $L = \{l_0, l_1, \ldots, l_{k-1}\}$ such that for all $0 \leq i < k$,*

- $\pi(u_i)$ *is separable from* $\pi(u_{i+1})$ *by* l_i,
- *the lines of* L *are parallel,*
- $\pi(u_i)$ *lies between* l_{i-1} *and* l_i, *and*
- *all other axis-parallel lines separating* $\pi(u_i)$ *and* $\pi(u_{i+1})$ *are parallel to* l_i.

Proof. Contained in the full version of the paper.

Given a non-degenerate layout of a graph G, we say that an induced k-extensor is *vertical* if the lines separating the projections of squares of U are orthogonal to the y-axis, and *horizontal* otherwise. Throughout the following lemmas, the notation $(a,b)+\{(c,d)\times(e,f)\}$ denotes an open $(d-c)\times(f-e)$ rectangle, with lower left corner at $(a+c,b+d)$.

Lemma 3.4 *Let* G *be a graph with the* k*-extensor* $G_k=(U_k,V_k,W_k,E_k)$ *as an induced subgraph, for some* $k\geq 2$. *Furthermore, let the only edges joining vertices of* $G\setminus G_k$ *with those of* G *be incident to the tabs* u_0 *and* u_k. *Let* (x_i,y_i) *be the* xy*-coordinates of the centres of symmetry of* u_i, *for* $0\leq i\leq k$. *For any valid non-degenerate layout of* G:

- *If* G_k *is vertical, then* $(x_k,y_k)\in(x_0,y_0)+\{(-k,k)\times(k,2k)\}$ or $(x_k,y_k)\in(x_0,y_0)+\{(-k,k)\times(-2k,-k)\}$.
- *If* G_k *is horizontal, then* $(x_k,y_k)\in(x_0,y_0)+\{(k,2k)\times(-k,k)\}$ or $(x_k,y_k)\in\{(-2k,-k)\times(-k,k)\}$.

Proof. Contained in the full version of the paper.

Two cases which come close to the bounds stated in the lemma are shown in Figure 3.

4 A Wobbly Logic Engine for a 3-dimensional Visibility Representation

We can think of a layout of a k-extensor as a flexible *spring* that can be compressed and stretched within the limits described in Lemma 3.4. See Figure 3. In this section, we describe how these springs can be used to construct a wobbly logic engine for the problem ZPR-NDUS.

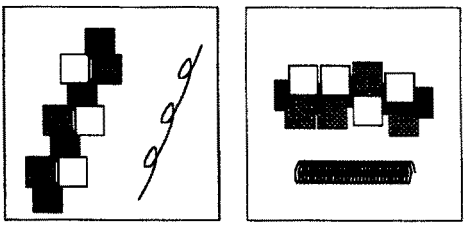

Fig. 3. The layouts "stretched spring" and "compressed spring" for a k-extensor, and their symbols

The idea for the construction is as follows – see Figure 6 for an overview. Using "connector rings", we can combine several springs to "blocks" that have very small relative flexibility; these blocks can be used to build frame, rods, and flags of a logic engine. To allow free rotation of the rods and flags in the logic engine, we use special "pivot connector rings". The overall engine has the same basic properties as a rigid logic engine.

Figure 4 shows the subgraphs C_r (regular connector ring) and C_p (pivot connector ring) and the symbols that are used to represent them in Figure 6. The squares are nodes that may also be tabs of k-extensors of adjacent springs, if necessary in the construction. Figure 5 shows the top view of a feasible layout for both types of connector rings. Note that the top part of a pivot connector ring (containing three tabs) can be transposed with respect to the bottom part (containing five tabs) without violating the layout of the ring. The centre square acts as a pivot for this transposition.

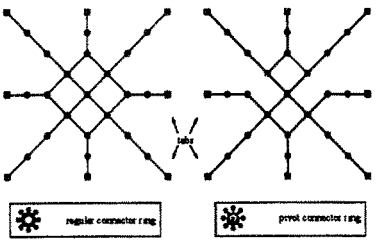

Fig. 4. A standard connector ring (left) and a pivot connector ring (right)

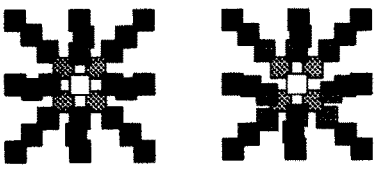

Fig. 5. Feasible representations for the connector rings

The next lemma shows that the tabs of a connector ring cannot be very far from its center; the proof is immediate.

Lemma 4.1 *Let $G = (V, E)$ be a graph having a connector ring C (C_r or C_p) as an induced subgraph. In a feasible layout of G, let $(0, 0)$ be the location of the central node of C. If (x_i, y_i) are the xy-coordinates of one of the tabs, then we have $(x_i, y_i) \in \{(-4, 4) \times (-4, 4)\}$.*

In the following, we will refer to specific connector rings in Figure 6 by their reference numbers; we will also use the location of the canonical center of a connector ring to describe its position.

Next we describe the structure of a *block*, which is the arrangement induced by the connector rings 1-9 in Figure 6. A block consists of nine connector rings,

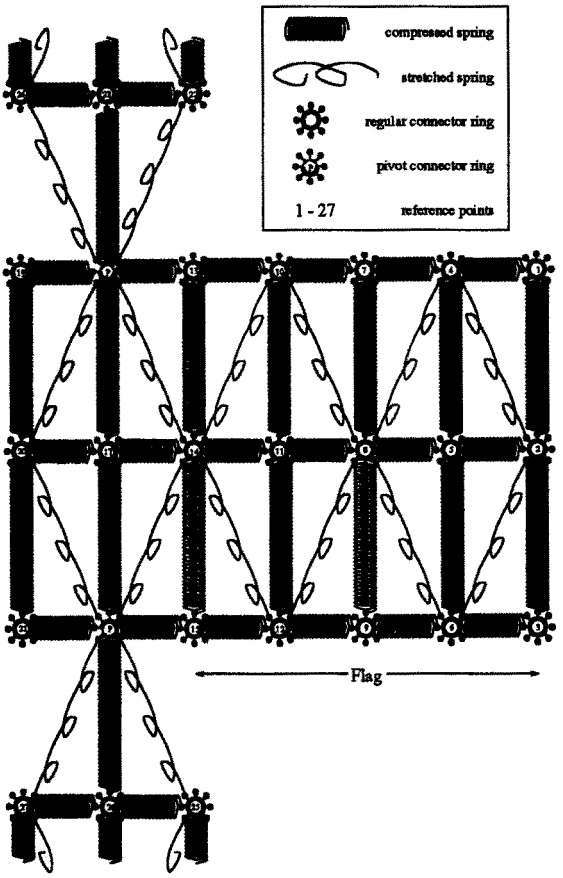

Fig. 6. A flag in a wobbly logic engine

forming a 3×3 lattice, in one direction connected by six $2m$-extensors (shown as the six vertical compressed springs $(1,2)$, $(2,3)$, $(4,5)$, $(5,6)$, $(7,8)$, $(8,9)$), in the other direction connected by six m-extensors (the horizontal compressed springs $(1,4)$, $(2,5)$, $(3,6)$, $(4,7)$, $(5,8)$, $(6,9)$), and diagonally connected by four $(m+1)$-extensors (the stretched springs $(2,4)$, $(2,6)$, $(4,8)$, $(6,8)$). Note that the four diagonal extensors form a diamond shape around the canonical center of the block (location 5 in the figure).

Finally, blocks are used for building the higher-order structures, by joining them as shown in the figure. As the following lemma shows, the relative location of two adjacent block centers is dominated by the size of the parameter m, within error bounds of constant size.

Lemma 4.2 *Let $G = (V, E)$ be a graph consisting of blocks as described above. Let $(0,0)$ be the canonical center of a block and let (x_i, y_i) be the canonical center of an adjacent block.*

- *If (x_i, y_i) is horizontally adjacent, then $(x_i, y_i) \in (2m, 0) + \{(-16, 52) \times (-36, 36)\}$ or $(x_i, y_i) \in (-2m, 0) + \{(-52, 16) \times (-36, 36)\}$.*

- *If (x_i, y_i) is vertically adjacent, then we have $(x_i, y_i) \in (0, 4m) + \{(-34, 34) \times (-16, 56)\}$ or $(x_i, y_i) \in (0, -4m) + \{(-34, 34) \times (-56, 16)\}$.*

Sketch: Representatively, we consider the horizontal case. In Figure 6, let the position 11 be $(0, 0)$ and let 5 be the adjacent block. Without loss of generality assume that 5 lies to the right of 11. By considering the path from connector 11 via 8 to connector 5, we conclude that

$$x_i \geq -4 + m - 4 - 4 + m - 4 = 2m - 16.$$

By considering the path from connector 11 via 14, 10, 8, 6, 2 to connector 5, we conclude that

$$\begin{aligned} x_i &\leq 4 - m + 4 + 4 + (m+1) + 4 + 4 + (m+1) + 4 + 4 + (m+1) + 4 \\ &\quad + 4 + (m+1) + 4 + 4 - m + 4 \\ &= 2m + 52. \end{aligned}$$

For the y-coordinate, consider the path from 11 via 10, 8, 6 to 5, implying

$$x_i \geq -4 + 2m - 4 - 4 - 2(m+1) - 4 - 4 - 2(m+1) - 4 - 4 + 2m - 4 = -36.$$

Similarly, the path from 11 via 12, 8, 4 to 5 forces

$$x_i \leq 4 - 2m + 4 + 4 + 2(m+1) + 4 + 4 + 2(m+1) + 4 + 4 - 2m + 4 = 36.$$

The vertical case can be shown in a similar way. Note that this is also true if the blocks are connected by a pivot connector ring: consider the paths $[17, 18, 19]$, $[17, 16, 14, 18, 27, 29, 26]$, $[17, 14, 18, 27, 26]$, $[17, 20, 18, 25, 26]$ for the distance between 17 and 26, where 18 is the pivot connector ring in between, and 29 is the connector ring below 26. □

Lemma 4.3 *Suppose we have an instance of NAE-3SAT that cannot be satisfied. Then the corresponding wobbly logic engine has no feasible layout.*

Sketch: Suppose we have an NAE-3SAT instance which cannot be satisfied. This means that for any layout of a rigid logic engine, we have a collision between flags. By choosing an appropriate representation of the flags by blocks, there will be an overlap of at least $O(m)$ if we keep all the blocks rigid. As we showed above, allowing for wobble will produce variations of $O(dist(p, q))$ for the relative position of two block centers p and q, where $dist(p, q)$ is the rectilinear distance of p and q in the canonical grid graph of block centers. Since this variation is independent of m and polynomial in the input size, choosing the free parameter m to be of sufficiently large value that is still polynomial in the input size forces an overlap of two flags in any layout of the wobbly logic engine. □

Lemma 4.4 *Suppose we have an instance of NAE-3SAT that can be satisfied. Then there exists a feasible layout of the corresponding wobbly logic engine.*

Proof. Contained in the full version of the paper. See Figure 6 for an example.

Summarizing, we state:

Theorem 4.5 *The problem ZPR-NDUS is NP-hard.*

5 Other Applications

The original logic engine has been used for a variety of "rigid" graph layout problems. One of them is the representation of a graph as the intersection graph of closed objects such as unit squares or unit disks. It is not hard to see how degenerate arrangements can be forced that can be used as rigid building blocks for a logic engine. On the other hand, these degenerate arrangements are necessarily unstable if perturbed even slightly. This makes it more attractive, when considering which graph representations to use, to require objects to intersect in a "fat" manner. This is equivalent to considering open objects only.

The wobbly logic engine can be used to prove NP-hardness of a number of these problems. In this section, we sketch how we can prove NP-hardness of the problem of deciding whether a given graph is an intersection graph of open unit squares or open unit cubes.

The first result is an immediate corollary of Theorem 4.5; note that all constructions of the two preceding sections can be used without any changes:

PROBLEM *Open Unit Square Intersection Graph*:
Given a graph $G = (V, E)$, is G the intersection graph of axis-aligned open unit squares in the plane?

Corollary 5.1 *The problem* Open Unit Square Intersection Graph *is NP-hard.*

With some modifications, the approach may be extended to a 3-dimensional variant of this problem.

PROBLEM *Open Unit Cube Intersection Graph*:
Given a graph $G = (V, E)$, is G the intersection graph of axis-aligned open unit cubes in 3-space?

Theorem 5.2 *The problem* Open Unit Cube Intersection Graph *is NP-hard.*

Consider the modified k-extensor shown in Figure 7. For this type of k-extensor, it is straightforward to prove lemmas analogous to those in Sections 3 and 4. For constructing a wobbly logic engine, we use a 3-dimensional $3 \times 3 \times 3$ lattice as a block, with appropriate diagonal springs forming a diamond shape around the center. Most of the engine can be built from these blocks analogously to the 2-dimensional case. However, since in the 3-dimensional setting flags can otherwise be oriented in four instead of two directions, we exclude two undesired directions by building appropriate protrusions near the pivots.

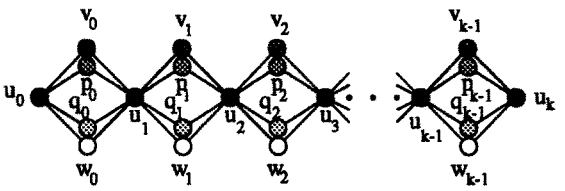

Fig. 7. A k-extensor for the problem *3-dimensional unit box intersection graph*

In the same way, we can generalize even further:

PROBLEM *Open Unit Hypercube Intersection Graph*:
Given a graph $G = (V, E)$, is G the intersection graph of axis-aligned open unit hypercubes in d-space?

Theorem 5.3 *The problem* Open Unit Hyperube Intersection Graph *is NP-hard.*

The wobbly logic engine technique can be applied to classes of objects other than squares or cubes. In particular, the following can be shown:

Theorem 5.4 *The problem* Open Unit Disk Intersection Graph *is NP-hard.*

References

1. H. Alt, M. Godau, and S. Whitesides. Universal 3-dimensional visibility representations for graphs. *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science Vol. 1027, Springer-Verlag, 1996, pp. 8–19.
2. A. Bachem, S. P. Fekete, B. Knab, R. Schrader, I. Vannahme, I. Weber, R. Wegener, K. Weinbrecht, and B. Wichern. Analyse großer Datenmengen und Clusteralgorithmen im Bausparwesen. In *Geld, Finanzwirtschaft, Banken und Versicherungen*. Editors C. Hipp et al., VVW Karlsruhe, 1997.
3. P. Bose, H. Everett, S. P. Fekete, A. Lubiw, H. Meijer, K. Romanik, T. Shermer, and S. Whitesides. On a visibility representation for graphs in three dimensions. *Proc. Graph Drawing '93*, Paris (Sèvres), 1993, pp. 38–39.
4. P. Bose, H. Everett, S. P. Fekete, A. Lubiw, H. Meijer, K. Romanik, T. Shermer, and S. Whitesides. On a visibility representation for graphs in three dimensions. Snapshots of Computational and Discrete Geometry, **3**, *eds.* D. Avis and P. Bose, McGill University School of Computer Science Technical Report SOCS-94.50, July 1994, pp. 2–25.
5. P. Bose, H. Everett, S. P. Fekete, M. E. Houle A. Lubiw, H. Meijer, K. Romanik, G. Rote, T. Shermer, S. Whitesides, and C. Zelle. A visibility representation for graphs in three dimensions. Technical Report ZPR 97-253, Center for Parallel Computing, 1997. Available at `ftp://ftp.zpr.uni-koeln.de/pub/papers/zpr97-253.ps.gz`.
6. S. Bhatt and S. Cosmadakis. The complexity of minimizing wire lengths in VLSI layouts. *Information Processing Letters*, v. 25, 1987, pp. 263-267.
7. F. J. Brandenburg. Nice drawings of graphs and trees are computationally hard. Technical Report MIP-8820, Fakultät für Mathematik und Informatik, Univ. Passau, 1988.
8. F.J. Cobos, J.C. Dana, F. Hurtado, A. Marquez, and F. Mateos On a visibility representation of graphs. *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science Vol. 1072, Springer-Verlag, 1996, pp. 152–161.
9. R. Cohen, P. Eades, T. Lin, and F. Ruskey. Three-dimensional graph drawing. *Proc. Graph Drawing '94*, Princeton NJ, 1994. Lecture Notes in Computer Science LNCS Vol. 894, Springer-Verlag, 1995, pp. 1–11.
10. L. Danzer, B. Grünbaum, and V. Klee. Helly's theorem and its relatives. *Convexity, Proc. 7^{th} ACM Symp. Pure Math.*, 1963, pp. 101–181.

11. A. Dean and J. Hutchinson. Rectangle visibility representations of bipartite graphs. *Proc. Graph Drawing '94*, Princeton NJ, 1994. Lecture Notes in Computer Science LNCS Vol. 894, Springer-Verlag, 1995, pp. 159–166.
12. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, Algorithms for automatic graph drawing: an annotated bibliography. *Comput. Geometry: Theory and Applications*, **4**, 1994, pp. 235–282. Also available from wilma.cs.brown.edu by ftp.
13. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing. Algorithms for geometric representation of graphs.* Prentice Hall, to appear.
14. G. Di Battista, G. Liotta and S. Whitesides. The strength of weak proximity. *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science Vol. 1072, Springer-Verlag 1996.
15. G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, **61**, 1988, pp. 175–198.
16. P. Eades and S. Whitesides. The realization problem for Euclidean minimum spanning trees is NP-hard. *in* Proc. 10^{th} ACM Symp. on Computational Geometry, 1994, pp. 49-56.
17. P. Eades and S. Whitesides. The logic engine and the realization problem for nearest neighbor graphs. *Theoretical Computer Science*, 169 (1), 1996, pp. 23-37.
18. H. ElGindy, G. Liotta, A. Lubiw, H. Meijer and S. Whitesides. Recognizing rectangle of influence drawable graphs. *Proc. Graph Drawing '94*, Princeton NJ, 1994, Lecture Notes in Computer Science LNCS #894, Springer-Verlag, 1995 pp. 352-363.
19. S. P. Fekete, M. E. Houle, and S. Whitesides. New results on a visibility representation of graphs in 3D. *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science Vol. 1072, Springer-Verlag 1996, pp. 234–241.
20. S. P. Fekete, M. E. Houle, and S. Whitesides. The wobbly logic engine: proving hardness of non-rigid graph representation problems. (Full version). Technical Report ZPR 97-273, Center for Parallel Computing, 1997. Available at `ftp://ftp.zpr.uni-koeln.de/pub/papers/zpr97-273.ps.gz`.
21. S. P. Fekete and H. Meijer. Rectangle and box visibility graphs in 3D. To appear in *International Journal of Computational Geometry and Applications*. Available at `ftp://ftp.zpr.uni-koeln.de/pub/papers/ zpr96-224.ps.gz`.
22. M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco, 1979.
23. J. Hutchinson, T. Shermer, and A. Vince. On representations of some thickness-two graphs. *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science Vol. 1072, Springer-Verlag 1996, pp. 324–332.
24. P. J. Idicula. Drawing trees in grids. Master's thesis, Department of Computer Science, University of Auckland, 1990.
25. K. Romanik. Directed VR-representable graphs have unbounded dimension. *Proc. Graph Drawing '94*, Princeton, NJ, 1994, Lecture Notes in Computer Science LNCS Vol. 894, Springer-Verlag, 1995, pp. 177–181.
26. R. Tamassia and I. G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete Comput. Geom.* **1**, 1986, pp. 321–341.