Unit Disk Graph Recognition is **NP**-Hard*

Heinz Breu† and David G. Kirkpatrick
Department of Computer Science, University of British Columbia

13 May 1993

Abstract

Unit disk graphs are the intersection graphs of unit diameter closed disks in the plane. This paper reduces SATISFIABILITY to the problem of recognizing unit disk graphs. Equivalently, it shows that determining if a graph has sphericity 2 or less, even if the graph is planar or is known to have sphericity at most 3, is **NP**-hard. We show how this reduction can be extended to 3 dimensions, thereby showing that unit sphere graph recognition, or determining if a graph has sphericity 3 or less, is also **NP**-hard. We conjecture that K-sphericity is **NP**-hard for all fixed K greater than 1.

1 Introduction

A *unit disk graph* is the intersection graph of a set of unit diameter closed disks in the plane. That is, each vertex corresponds to a disk in the plane, and two vertices are adjacent in the graph if the corresponding disks intersect. The set of disks is said to *realize* the graph. Of course, the unit of distance is not critical, since the disks realize the same graph even if the coordinate system is scaled by any convenient amount. Notice that two disks intersect if and only if the distance between their centers is at most the disk diameter. Unit disk graphs can therefore be realized just as well as a set of points in the plane; two vertices are adjacent in the graph exactly when the Euclidean distance between them is at most 1. A *realization* of a unit disk graph is therefore a mapping of the vertices to points which realize the graph in this way. Again, the *unit* of distance is not critical. This paper addresses the recognition problem for unit disk graphs: given a graph, determine if it has a realization.

Unit disk graphs have been used to model several physical problems, for example radio frequency assignment [Hal80] and ship-to-ship communications (attributed to Marc Lipman by [Rob91]). They have also been used as test cases for heuristic algorithms designed for arbitrary graphs [JAMS91]. More applications are described by [CCJ90] and [MHR92].

*Submitted to *Algorithmica*

†On leave from Hewlett-Packard Laboratories

There are several motivations for recognizing unit disk graphs. First, unit disk graphs are a natural generalization of unit interval graphs, also known as indifference graphs [Rob68]. These graphs are interval graphs that do not contain $K_{1,3}$ as an induced subgraph [Rob68]. They can therefore be recognized in polynomial time [FG65, BL76]. Secondly, unit disk graphs can admit more efficient algorithms than arbitrary graphs. For example, the **NP**-complete problem, maximum CLIQUE, can be solved in polynomial time for unit disk graphs [CCJ90]. There are also several familiar **NP**-complete problems whose corresponding optimization problems can be efficiently approximated for unit disk graphs [MHR92]. Although these latter problems remain **NP**-complete for unit interval graphs, their approximating algorithms achieve better guarantees than any known for arbitrary graphs. Some of the algorithms for these problems require a realization of the graph. A third motivation is simply to determine if a graph is a unit disk graph, for example, to test the physical feasibility of a graph modelling molecules [Hav82a]. We show in this paper that the problem of determining if a graph has a realization is itself **NP**-hard. This answers an open question mentioned in [CCJ90] and [MHR92].

This result has a more general context. The *sphericity* of a graph is the smallest dimension for which it is the intersection graph of a set of (hyper) spheres [Hav82a, Fis83]. Unit disk graphs are therefore graphs with sphericity two, and our **NP**-hardness result implies that determining the sphericity of a graph is also **NP**-hard. However, graphs with sphericity 1 are the unit interval graphs, so that the sphericity = 1 question can be answered in polynomial time. This paper reduces SATISFIABILITY to unit disk graph recognition. In the conclusion to this paper, we show that a similar reduction applies to the sphericity = 3 question. We also conjecture that there is an analogous reduction for the sphericity = K question for any $K > 1$.

2 2-SPHERICITY is NP-Hard

A *K-dimensional realization* of a graph $G = (V, E)$ is a function $f : V \to R^K$ such that $(v_i, v_j) \in E$ if and only if $d(f(v_i), f(v_j)) \leq 1$ where d is the Euclidean distance between two points.

K-SPHERICITY

INSTANCE: Graph $G = (V, E)$, positive integer K.

QUESTION: Does G have sphericity K, that is, does it have a K-dimensional realization?

This paper reduces SATISFIABILITY to 2-sphericity. SATISFIABILITY is a common basis for **NP**-completeness proofs [GJ79] and is the first problem to have been proved **NP**-complete [Coo71]. Let $U = \{u_1, u_2, \ldots, u_m\}$ be a set of Boolean *variables*. A clause $c = \{l_1, l_2, \ldots, l_k\}$ is a set of literals, which may be negated, e.g., $\overline{u}_i$, or unnegated, e.g., u_i, variables. A *truth assignment* is a function $t : U \to$

{TRUE, FALSE}. In terms of literals, u_i is TRUE if and only if $t(u_i) =$ TRUE, and $\overline{u}_i$ is TRUE if and only if $t(u_i) =$ FALSE. A clause is *satisfied* by t if at least one $l_i \in c$ is TRUE. Finally, a *satisfying truth assignment* is one which simultaneously satisfies all the clauses.

SATISFIABILITY

INSTANCE A set $U = \{u_1, u_2, \ldots, u_m\}$ of Boolean variables and a set $C = \{c_1, c_2, \ldots, c_n\}$ of clauses over U.

QUESTION Is there a satisfying truth assignment for C?

Theorem 1 *2-SPHERICITY is* **NP***-hard.*

Proof: Given an instance C of SATISFIABILITY, we will construct a graph $G_C = (V_C, E_C)$ such that G_C has a realization[1] if and only if C is satisfiable. We will assume without loss of generality (see comments for SATISFIABILITY in [GJ79]) that each clause in C contains *at most* three literals ($|c_i| \leq 3$) and that each variable appears in *at most* 3 clauses. Note that this is not the same as 3SAT.

This paper builds G_C in several stages. First, Section 3 constructs a graph G_C^{SAT} that corresponds closely to the instance C of SATISFIABILITY. We define a notion of orientability for this graph, and prove that it is orientable if and only if C is satisfiable (Lemma 2). Section 4 considers a canonical drawing of this graph on the grid. We define a notion of orientability for this drawing, and prove that it is orientable if and only if the underlying graph is orientable (Lemma 3). Finally, Section 5 forms G_C by simulating components of the grid drawing. Lemma 8 shows that G_C has a realization if and only if the underlying grid drawing is orientable. We finish by showing that the entire reduction can be executed in polynomial time. □

3 A Graph That Simulates SATISFIABILITY

Begin by constructing a graph G_C^{SAT} from the instance C of SATISFIABILITY. The vertices of the graph correspond to the clauses, variables, and negated variables of the SATISFIABILITY instance C. There is an edge between between a literal vertex and a clause vertex if the literal appears in the clause. More formally, $G_C^{SAT} = (V_C^{SAT}, E_C^{SAT})$, where:

$$\begin{aligned} V_C^{SAT} &= \{\tilde{c} | c \in C\} \cup \{u^+ | u \in U\} \cup \{u^- | u \in U\} \\ E_C^{SAT} &= \{(\tilde{c}, u^+) | c \in C, u \in c\} \cup \{(\tilde{c}, u^-) | c \in C, \overline{u} \in c\} \end{aligned}$$

This graph models the testing of a truth assignment for SATISFIABILITY. Say that the graph is *orientable* if its edges can be directed such that, for each clause vertex, outdegree($\tilde{c}$) $\geq$ 1 and, for each pair of literal vertices, indegree(u^+) $\times$ indegree(u^-) = 0.

[1]For brevity, and unless stated otherwise, the remainder of this paper abbreviates "two-dimensional realization" as "realization'.

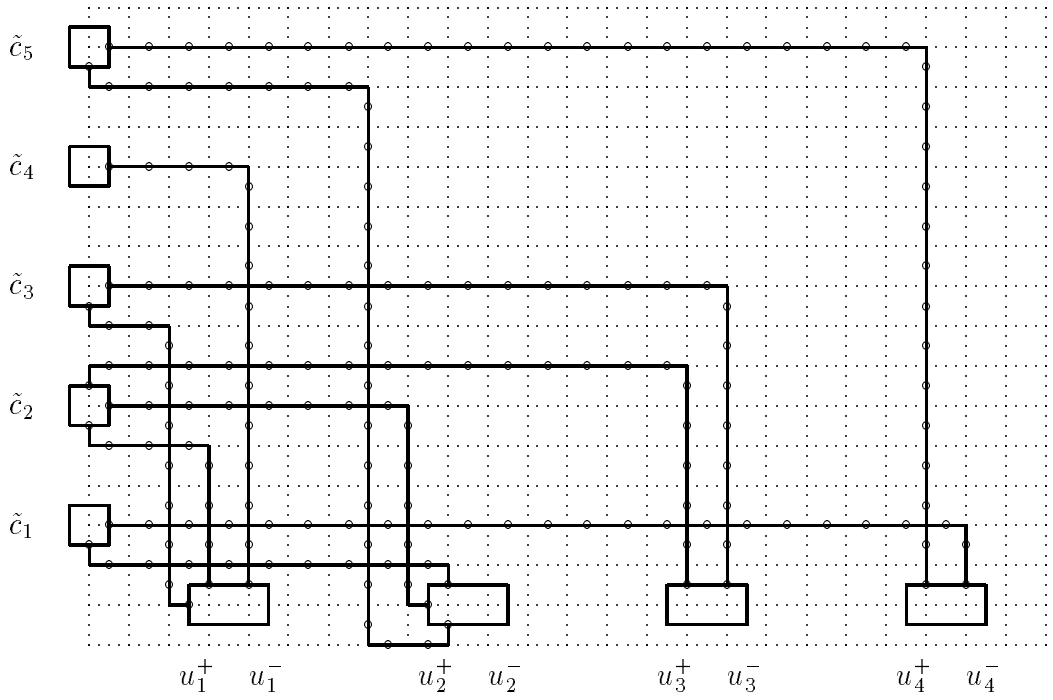


Figure 1: The graph corresponding to the SATISFIABILITY instance $U = \{u_1, u_2, u_3, u_4\}$, $C = \{\{u_2, \overline{u}_4\}, \{u_1, u_2, u_3\}, \{u_1, \overline{u}_3\}, \{\overline{u}_1\}, \{u_2, u_4\}\}$ drawn on the grid. The clauses are drawn as squares. The literals are embedded as adjacent pairs (by variable) and so are drawn as rectangles. A literal component has up to three terminals: the three on the left of the variable box belong to the unnegated literal and the three on the right to the negated literal. Note that the area of the grid is $(6|U|+1) \times (3|C|+2)$.

Intuitively, an edge directed from $\tilde{c}$ to u^+ (resp. u^-) means that clause c has selected literal u (resp. $\overline{u}$) to satisfy it. That is, it requests that $t(u) =$ TRUE (resp. $t(u) =$ FALSE). If C is satisfiable, it must be possible to orient G_C^{SAT} since every clause must be satisfied (outdegree$(\tilde{c}) \geq 1$), and no truth assignment can set both a literal and its complement TRUE (indegree$(u^+) \times$ indegree$(u^-) = 0$). Conversely, if G_C^{SAT} is orientable, there must be a satisfying truth assignment for C since, for every pair of literal vertices indegree$(u^+) \times$ indegree$(u^-) = 0$ (each variable can be set either TRUE or FALSE) and, for every clause vertex $\tilde{c}$, outdegree$(\tilde{c}) \geq 1$ (every clause is satisfied). This proves the following lemma.

Lemma 2 *C is satisfiable if and only if G_C^{SAT} is orientable.*

4 Drawing the Graph on the Grid

Continue the construction by drawing the graph on the grid as shown in Figure 1. Each of the $(6|U|+1) \times (3|C|+2)$ grid vertices in this drawing is either unused, or is associated with a unique *component* of the drawing. Each component is enclosed by a unit square centered on its grid vertex.

The drawing is made up of three groups of components: communication components, literals, and clauses. There are, in turn, three groups of communication components: wires, corners, and cross-overs. A *wire* is a unit length line segment passing through a grid vertex, a *corner* is two half-length line segments meeting at right angles at a grid vertex, and a *cross-over* is two unit length line segments crossing at right angles on a grid vertex. There are therefore two types of wire component (horizontal and vertical), four types of corners, and one type of cross-over.

Each component in the drawing has one to four *terminals*. Each terminal terminates a line segment and is centered on a side of the unit square enclosing the component. The terminal on the top or north side of the unit square is called the T terminal. Similarly, the terminals on the bottom, left, and right are called the B, L, and R terminals respectively. Wires and corners have two terminals each. Cross-overs have four terminals, and literals and clauses have up to three terminals each, equal to their degree in G_C^{SAT}. Two components in the drawing are *adjacent* if they have coincident terminals. We consider a complementary pair of literals to be a single truth setting component, and so do not show terminals between them. Figure 1 depicts the set of all terminals as small circles.

An *orientation* of a terminal is a *direction*, N, S, E, or W. A terminal T (resp. B, L, R) is *directed away* from its component if it is oriented N (resp. S, W, E) and is *directed towards* its component otherwise. Say that a grid drawing is *orientable* if all terminals can be oriented subject to the condition that:

draw1: only terminals adjacent to vertical line segments are directed N or S,

draw2: only terminals adjacent to horizontal line segments are directed E or W,

draw3: every wire, corner, cross-over line segment, and clause has at least one terminal directed away from it,

draw4: every truth setting component has a literal component with *all* terminals directed away from it.

Figure 2 shows an orientation of a portion of Figure 1.

Lemma 3 *A grid drawing is orientable if and only if the underlying graph is orientable.*

Proof: Suppose we have an orientation for G_C^{SAT}. Then orient the terminals along each path in the grid drawing so that they are consistent with the orientation of the corresponding edge in E_C^{SAT}. This ensures that every wire, corner, and cross-over line segment has *precisely* one terminal directed away from it. It also ensures that each clause has one terminal directed away from it, since the clause vertices have outdegree at least one. Finally, every truth setting component has a literal component with all terminals directed away from it since one of the literal vertices has zero indegree in G_C^{SAT}.

Now suppose that grid drawing has been oriented. Condition **draw3** ensures that any path in the drawing, corresponding to an edge in E_C^{SAT}, contains at most

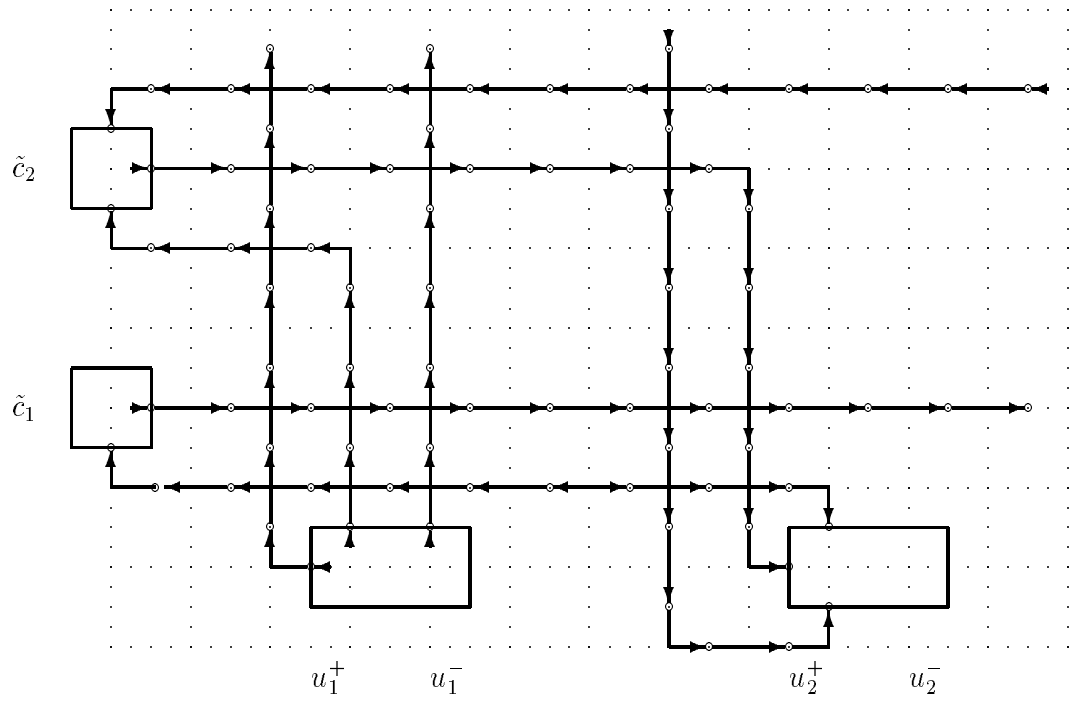


Figure 2: An oriented grid drawing. The directions are drawn as arrows. Note that the path from u_2^+ to $\tilde{c}_1$ has a wire component that has both terminals directed away from it.

one wire, corner, or cross-over segment with both terminals directed away from the component. One terminal of such a component can be redirected by reversing the direction of all terminals in the path from it to the literal. Doing so keeps all conditions satisfied since this operation does not change the orientation of any clause terminals, and it directs any literal terminals away from the literal, in keeping with condition **draw4**. Condition **draw3** then ensures that all terminals along the path are oriented consistently with some orientation for the corresponding edge. Under this orientation, all clause vertices have outdegree at least one, since the corresponding terminal is directed away from the clause. Furthermore, condition **draw4** ensures that $\text{indegree}(u^+) \times \text{indegree}(u^-) = 0$. □

Corollary 4 *A grid drawing is orientable if and only if the underlying instance of SATISFIABILITY is satisfiable.*

5 Reduction from Grid Drawing Orientability

We are now ready to construct G_C. To do so, we create a graph component for each grid drawing component: wires, corners, cross-overs, truth setters, and clauses. Sections 5.2 through 5.6 provide the details. Each of these graph components has two or more *terminals* — labelled T, B, L, or R — that correspond to the grid drawing terminals. Each terminal is an induced subgraph on four vertices, labelled a, b, c,

and d. To construct G_C, connect every pair of adjacent[2] components together by identifying the appropriate terminals, by label. Terminal T (resp. B, L, R) should be identified with an adjacent B (resp. T, R, L) terminal. The reader will find that the example presented in Section 5.7 clarifies this process.

5.1 Properties of Cages

Before giving detailed descriptions of the various graph components, we must describe the building blocks from which they are constructed. These building blocks are cycles ("cages") with additional independent vertices ("beads"). A bead is attached to two vertices (a "hinge") of a cage by a short path through another vertex ("chain"), as illustrated in Figure 3.

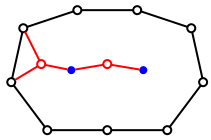


Figure 3: A cage with a double bead on two chain vertices

The *capacity* of a cage is the maximum number of independent beads that can be embedded inside a cage in any realization. The capacity is constant for each cage and depends only on the number of vertices forming the boundary of the cage. A bead embedded inside a cage diminishes its remaining capacity. It thereby displaces other beads, which may otherwise have been embedded inside the cage. The following construction ensures that a bead can only be embedded in one of two adjacent cages. A displaced bead therefore displaces other beads from whatever cage it is embedded in. This is the basic mechanism for propagating information in a realization of G_C.

The corollary to the following lemma shows that the notion of a vertex being embedded inside the realization of a cycle is well defined.

Lemma 5 *Let f be a realization of a unit disk graph $G = (V, E)$. Let (v_a, v_b) and (v_c, v_d) be edges in E with distinct endpoints, and denote $f(v_a) = a$. If the line segments (a, b) and (c, d) cross, then the subgraph induced by $\{v_a, v_b, v_c, v_d\}$ contains K_3 as a subgraph.*

Proof: Suppose that (a, b) and (c, d) cross at e. Let ij denote the distance between point i and point j. Then

$$\begin{aligned} ac + bd &\leq (ae + ec) + (be + ed) \\ &= (ae + be) + (ec + ed) \\ &= ab + cd \leq 1 + 1 = 2 \end{aligned}$$

Therefore $ac \leq 1$ or $bd \leq 1$. Similarly, $ad \leq 1$ or $bc \leq 1$. Any of these four possibilities implies a K_3 subgraph. □

[2]Two graph components are *adjacent* if the associated grid drawing components are adjacent.

Corollary 6 *A realization of a vertex induced cycle is a plane graph.*

Define an *n-cage* as a cage with a capacity for n beads. G_C requires 0-cages, 1-cages, 2-cages, and 3-cages. These are cycles with 3 to 6 vertices, 7 or 8 vertices, 9 vertices, and 10 vertices respectively. The capacities of these cages can be verified by considering the optimal disk packings shown in Figure 4. In these packings, all unit

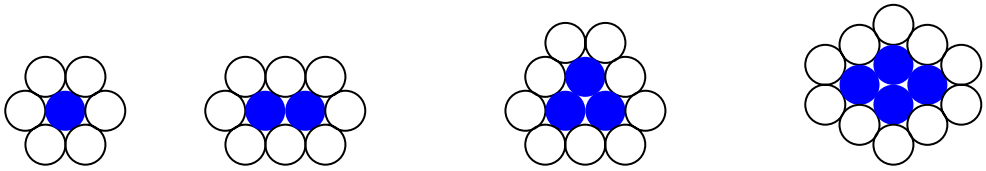


Figure 4: Optimal packings limiting a 0-cage, 1-cage, 2-cage, and 3-cage

disks are adjacent, and the interior beads are themselves optimally packed. These cases show that a 6-vertex cage *cannot* contain a bead, an 8-vertex cage cannot contain two beads, a 9-vertex cage cannot contain three beads, and a 10-vertex cage cannot contain four beads. Recall that all containments must be strict.

The skeleton of the graph under construction will be composed of many cages "hooked together". The construction hooks two cages together by identifying two adjacent hinge vertices from one cage with two adjacent hinge vertices from the other cage. See Figure 5. This connection strategy ensures that, in any realization, the

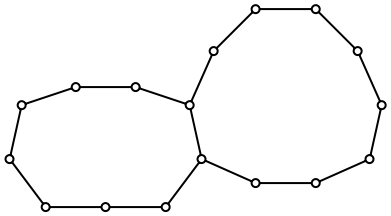


Figure 5: How cages are hooked together

orientation of all cages is determined by giving the orientation of any cage in the connected graph. Lemma 5 also guarantees that cages do not cross or overlap each other in any realization. It is therefore useful to think of the embedding of the skeleton of the graph, that is, the cages without chains or beads, to be invariant under all realizations. Different realizations simply allow beads (and chains) to "flip" from one cage into another. The component definitions which follow will clarify this construction.

5.2 Wires

The graph in Figure 6 implements a wire as a string of five 1-cages (with eight vertices each) connected as described in the previous subsection. The vertices in the wire's two terminals are labelled a_L, b_L, c_L, and d_L, and a_R, b_R, c_R, and d_R in Figure 6. The vertical wire can be obtained from Figure 6 by relabelling terminal R as terminal T, and terminal L as terminal B.

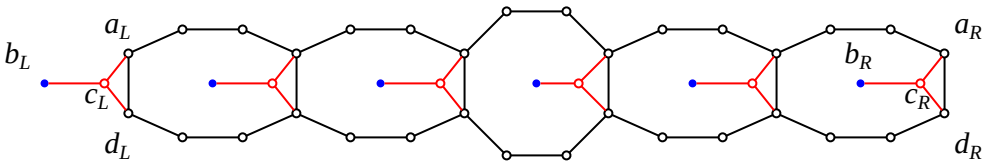


Figure 6: The horizontal wire component: drawn with both terminals oriented W

How do we know that the beads in the wire will be embedded in the cages of the wire in any realization? The answer is, that as things stand, we do not know. The solution adopted in this paper is to add mortar — extra vertices — around the hinge vertices as shown in Figure 7. "Mortar" is a way of constraining a bead to

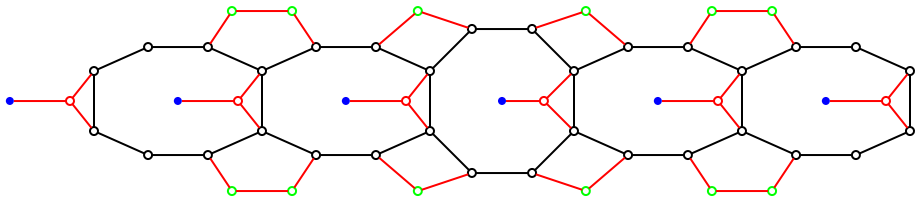


Figure 7: Wire with mortar

certain cages for all realizations. The effect of the mortar is to surround each hinge vertex with small cages. The bead and chain edges cannot cross any of the added cage edges by Lemma 5 since the bead is independent of all cage vertices and the chain is independent of all but the two hinge vertices. Also, neither the bead nor the chain vertex can be embedded inside any of the cages created by the mortar since all such cages are 0-cages. It is hard to see the underlying structure of the components if mortar vertices are shown. The remaining component diagrams, therefore, do not show any mortar vertices, but they should be understood to be present. Mortar is also required when two components are connected.

We say that the T (resp. B, L, R) terminal is *oriented* S (resp. N, E, W) if its bead (labelled b) is embedded inside its cage, and that it is oriented N (resp. S, W, E) otherwise. The properties of the cages and the mortar ensure that, if the T (resp. B, L, R) terminal is oriented S (resp. N, E, W), then its B (resp. T, R, L) terminal also is oriented S (resp. N, E, W). Note that it is also possible for the T (resp. L) terminal to be oriented N (resp. W) and the B (resp. R) terminal to be oriented S (resp. E) simultaneously. This ensures that at least one terminal is directed away from the wire.

5.3 Corners

As can be seen from Figure 8, the graph for corners is also a string of five 1-cages. The corner's two terminals are labelled L and B in Figure 8. The other three corners can be obtained from Figure 8 by relabelling B as T (and exchanging label a_L with d_L), or L as R (and exchanging label a_B with d_B), or both. Again, the properties of cages ensures that at least one terminal is directed away from the corner.

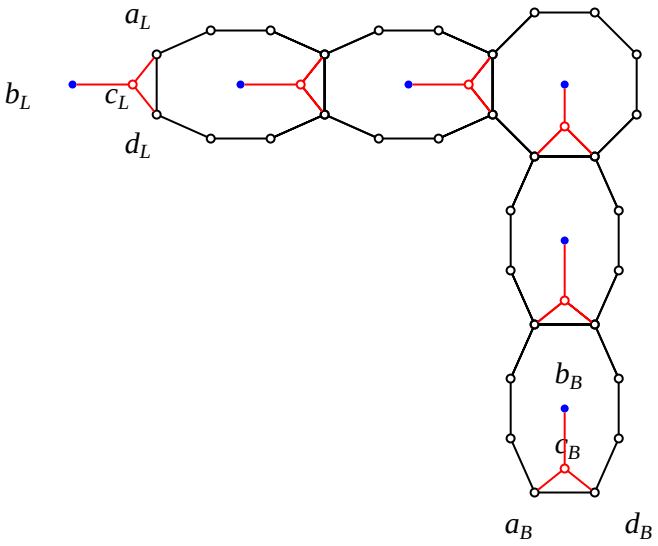


Figure 8: The corner component: drawn with the L terminal oriented E and the B terminal oriented N

5.4 Cross-Overs

As is typical for this type of reduction, the cross-over component is the most difficult to understand. The reader will find it easier to consider the cross-over schematic in Figure 9 before looking at the complete component. In this schematic, cages are depicted as octagons joined at hinges. Each cage is labelled with its capacity of independent beads. Independent beads are depicted as short paths of small circles crossing each hinge. Chain vertices are not shown. Think of the beads as flipping from one cage to another along the hinge.

The heart of the cross-over is the cycle of 3-cages and 2-cages. The following lemma is crucial to the operation of the cross-over component.

Lemma 7 *The 3-cages in the cross-over component each contain precisely one 2-bead cluster in any realization.*

Proof: The 3-cages cannot contain two 2-bead clusters since neither cage can accommodate four independent beads. Assume for the sake of contradiction that the 3-cage E does not contain either 2-bead cluster in some realization. Then the 2-cages D and F must each contain the 2-bead cluster that they share with cage E. Since D and F cannot contain any more than two independent beads, this forces cages A and C to contain the single beads that they share with cages D and F respectively. But this means that A and C cannot contain the 2-bead clusters that they share with cage B. This forces cage B to contain them both. This is the contradiction that proves the lemma since cage B cannot contain 4 independent beads in any realization. A symmetric argument applies if we assume that cage B does not contain either 2-bead cluster. □

Let us now examine the cross-over in action. Assume the bead on the bottom hinge (on cage E) is embedded in cage E. This single bead, together with whichever

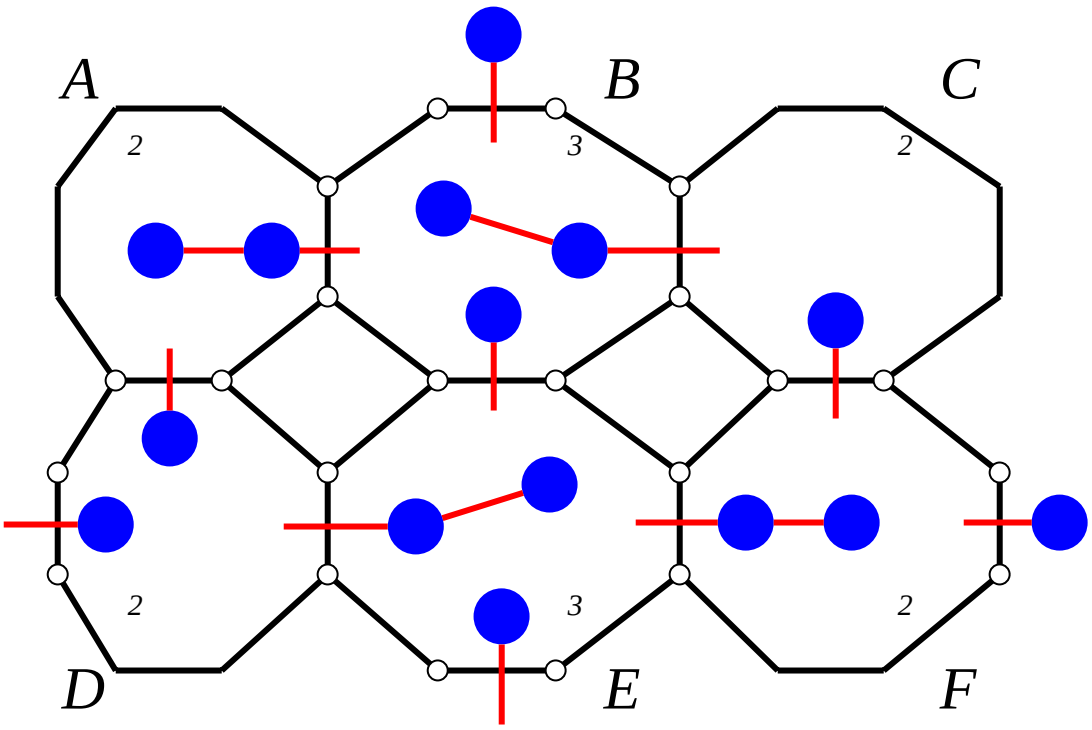


Figure 9: Cross-over schematic

2-bead cluster is in the cage by Lemma 7, fills cage E to capacity. The single bead shared between cage E and B must therefore lie in cage B. This bead, together with whichever 2-bead cluster is in the cage, fills cage B to capacity. This forces the single bead at the top hinge (on cage B) out of the cage. A symmetric argument applies if the bead on the top hinge is embedded in cage B.

Assume now that the bead on the left hinge (on cage D) is embedded inside cage D. It cannot occupy cage D together with the 2-bead cluster shared with cage E. Hence this 2-bead cluster must lie in cage E. The 2-bead cluster shared with cage F must therefore lie in cage F, since cage E cannot accommodate two 2-bead clusters. This 2-bead cluster forces the terminal bead on the right hinge (on cage F) out of the cage, and out of the cross-over. Incidently, it also completes the feed-back cycle exploited in Lemma 7 as follows. It forces a single bead into cage C, which forces a double bead into cage B, which forces a double bead into cage A, which forces a single bead back into cage D. A symmetric argument applies if the bead on the right hinge is embedded inside cage F.

Complete the cross-over component by adding a 1-cage "lead" to the left, top, and right hinges. Also add a lead, consisting of a string of two 1-cages, to the bottom hinge. In the schematic, the 2-cages are shown directly connected to one another. In the actual component, each pair is connected with a 1-cage. The completed cross-over component is shown in Figure 10. Although no mortar *vertices* are added, some mortaring by edges between cages is evident.

The cross-over's four terminals are labelled T, B, L, and R in Figure 10. The cross-over construction ensures that, if the T (resp. B, L, R) terminal is oriented

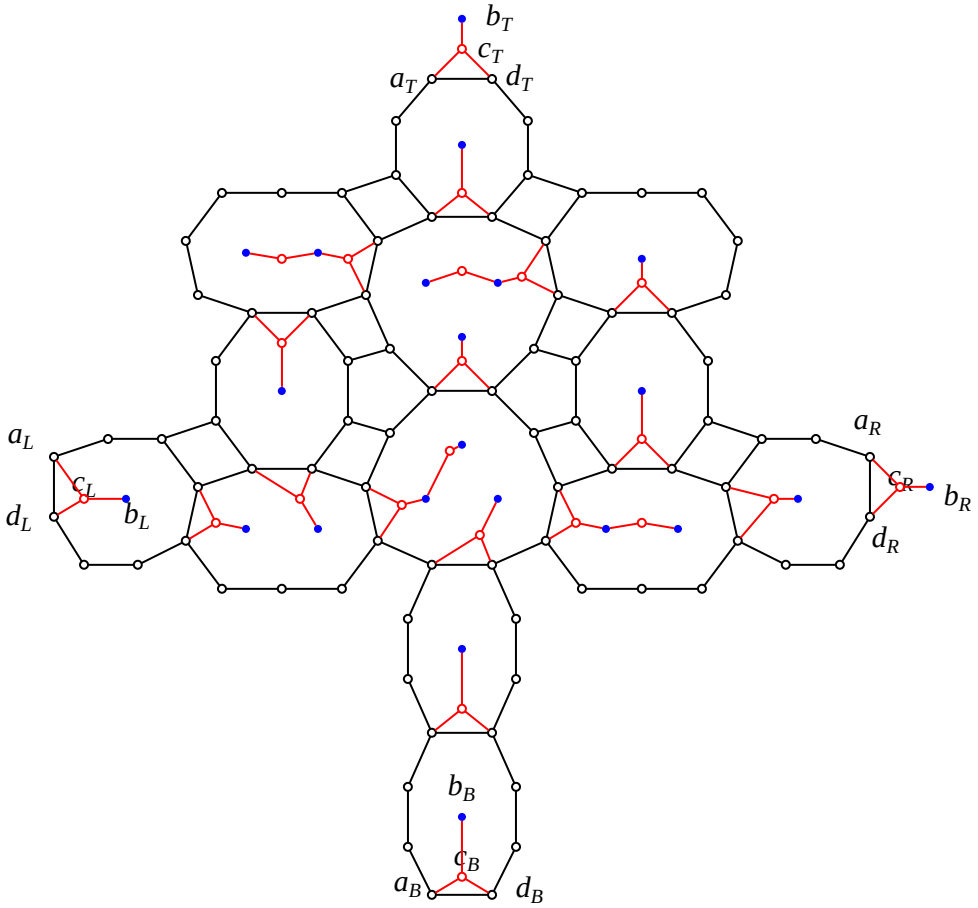


Figure 10: The cross-over component: drawn with the T and B terminals oriented N, and the L and R terminals oriented E.

S (resp. N, E, W), then the B (resp. T, R, L) terminal also is oriented S (resp. N, E, W). However, it is possible for the T and B terminals simultaneously to be oriented N and S respectively. Similarly, it is possible for the L and R terminals simultaneously to be oriented W and E respectively.

5.5 Truth Setters

The truth setting component is shown in Figure 11. Its heart is a pair of connected

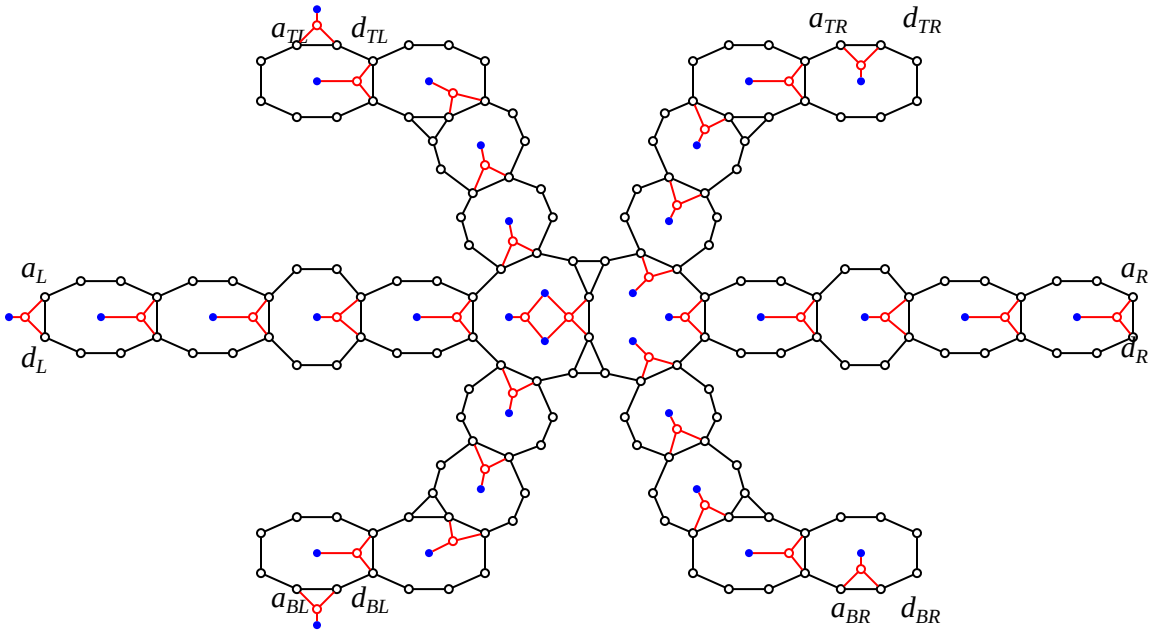


Figure 11: The truth setting component: drawn with the 3-cluster in the 3-cage corresponding to the unnegated literal.

3-cages. The cage on the left corresponds to the unnegated literal and the cage on the right to the negated literal. A cluster of three independent beads is connected with two chain beads to the common hinge of the 3-cages. Since this cluster is in one of the two 3-cages in any realization, it displaces all incident single beads, ensuring that all associated terminals are oriented away from the 3-cage.

5.6 Clauses

The clause testing component is shown in Figure 12. Its heart is a single 2-cage. Since this cage can contain at most two independent beads, it must be that at least one terminal is oriented away from it.

If a terminal is not used in the grid drawing, the corresponding graph component terminal is "capped" by appending a 0-cage to it. Figure 13 shows a clause that has one terminal with cap and mortar (couldn't resist). This ensures that the terminal is oriented *towards* the clause, thereby forcing one of the remaining terminals to be oriented away by reducing the capacity of the 2-cage by one.

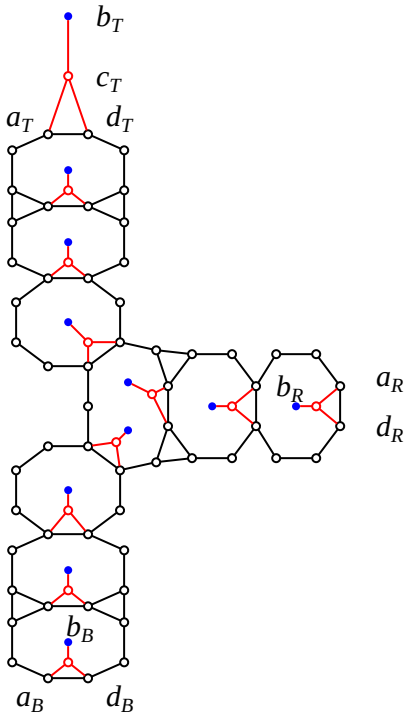


Figure 12: Clause component: drawn with T and B terminal oriented N, and R terminal oriented W.

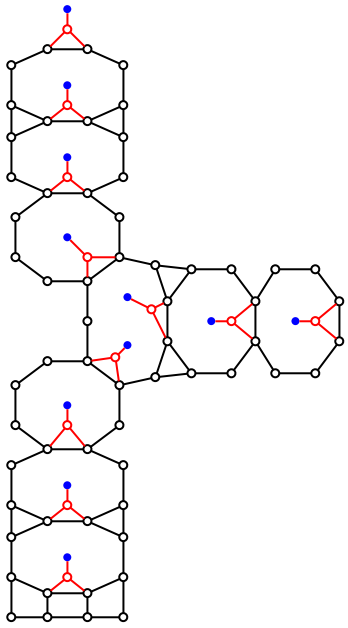


Figure 13: Clause testing component with bottom terminal capped

5.7 Building and Realizing G_C

Now that all components have been described, we can present an example showing how components are connected. Note that terminals always have the same structure; a bead vertex connected to two hinge vertices by a chain vertex. Figure 14 shows how two components are connected, two corners in this case.

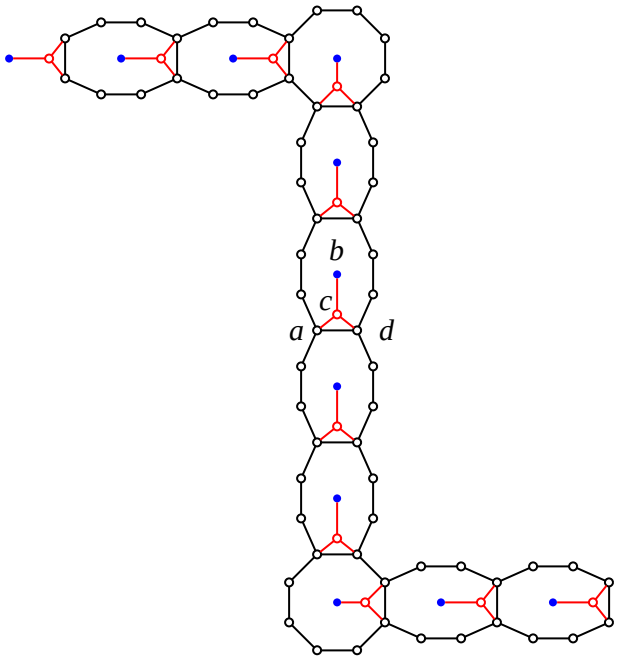


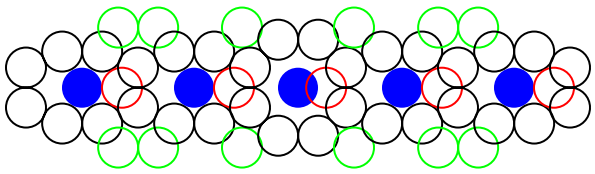
Figure 14: Two connected corners. The T terminal of the lower corner has been identified with the B terminal of the upper corner.

Recall that a literal component in the grid drawing has *at most* 3 terminals, whereas each literal of the truth setting component above has *exactly* 3 terminals. The unused truth setting terminals are not connected to the cages of any other component in G_C. In these cases, the incident terminal bead may simply be "absorbed by the ether", if the adjacent literal 3-cage is occupied in a realization.

Lemma 8 *The graph G_C has a realization if and only if the underlying grid drawing is orientable.*

Proof: Given a realization, orient the grid drawing terminals exactly as the realization terminals. The definition of orientation for realization terminals ensures that conditions **draw1** and **draw2** are met. The nature of the wire, corner, cross-over, and clause components ensures that condition **draw3** is met. Finally, the nature of the truth setting component ensures condition **draw4** is met. That is, the grid drawing is orientable if G_C has a realization.

Now assume that the terminals of the grid drawing have been oriented. From this we will construct a realization for G_C. This realization has a unit of distance equal to 10. Equivalently, divide all coordinates by 10 for a unit of distance equal to 1. Let the lower left grid corner have x-y coordinates (0, 0).



-68	-5		-68	5		-59	-9		-59	9		-54	0	bead
-49	-9		-49	9		-45	-15	mortar	-45	15	mortar	-44	0	chain
-40	-5		-40	5		-35	-15	mortar	-35	15	mortar	-31	-9	
-31	9		-26	0	bead	-21	-9		-21	9		-16	0	chain
-14	-15	mortar	-14	15	mortar	-12	-5		-12	5		-5	-12	
-5	12		0	0	bead	5	-12		5	12		7	0	chain
12	-5		12	5		14	-15	mortar	14	15	mortar	21	-9	
21	9		26	0	bead	31	-9		31	9		35	-15	mortar
35	15	mortar	36	0	chain	40	-5		40	5		45	-15	mortar
45	15	mortar	49	-9		49	9		54	0	bead	59	-9	
59	9		64	0	chain	68	-5		68	5				

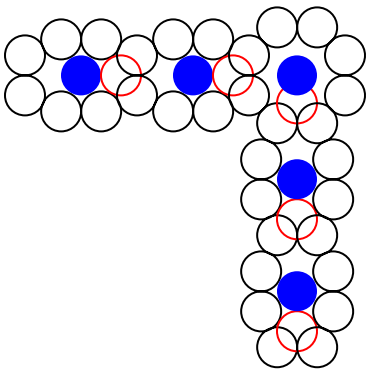
Table 1: Wire (with mortar) coordinates. This wire is horizontal and has both terminals oriented W. Rotate the coordinates by 180 degrees to orient the terminals E and by 90 degrees or -90 degrees to get the vertical wire with terminals oriented S or N respectively.

The coordinates for each component realization are given in Tables 1 through 5, allowing the reader to verify all claims of adjacency and non-interference. To make this task somewhat easier, the points for each component are plotted as circles of radius 5. Note that two circles intersect exactly when the Euclidean distance between the corresponding points is less than or equal to 10. For each component in the grid drawing, construct the corresponding component realization, depending on its orientation. Then translate the component realization to $(136x, 136y)$ where (x, y) are the grid coordinates of the component. The grid coordinates of a component are the coordinates of the corresponding grid vertex in all cases but one. Truth setters are the exception since they are really two literal components (which do have integer coordinates); variable u_i has coordinates $(3.5 + 6i, 1)$.

This procedure will result in duplicated points corresponding to terminal hinge vertices. Remove one set; they were only duplicated for clarity of exposition of each component. Note that beads and chains on terminals are *not* duplicated in this procedure. □

5.8 The Reduction can be Implemented in Polynomial Time

The grid drawing has area $(6|U| + 1) \times (3|C| + 2)$, and therefore at most this many components since each component is enclosed by a unit square. Each component of G_C has a constant number of vertices and edges belonging to cages, chains, beads, and mortar. Finally, each connection of components requires only a constant number of additional mortar vertices. Since the size of the SATISFIABILITY instance is a polynomial function of $|U|$ and $|C|$, the entire recognition instance can be built in polynomial time.



-68	-5		-68	5		-59	-9		-59	9		-54	0	bead
-49	-9		-49	9		-44	0	chain	-40	-5		-40	5	
-31	-9		-31	9		-26	0	bead	-21	-9		-21	9	
-16	0	chain	-12	-5		-12	5		-9	-59		-9	-49	
-9	-31		-9	-21		-5	-68		-5	-40		-5	-12	
-5	12		0	-64	chain	0	-54	bead	0	-36	chain	0	-26	bead
0	-7	chain	0	0	bead	5	-68		5	-40		5	-12	
5	12		9	-59		9	-49		9	-31		9	-21	
12	-5		12	5										

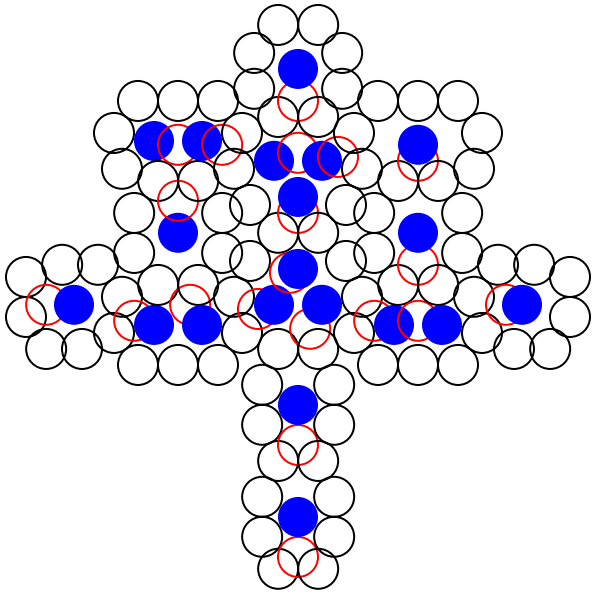
Table 2: Corner coordinates. This corner has the (0,0) grid vertex in the northeast corner. The L terminal is oriented W and the B terminal is oriented N. The other orientations and three corners can be obtained by rotating these coordinates by 90 degrees and by reflecting about the coordinate axes.

6 Concluding Remarks

This paper shows that unit disk graph recognition, or K-SPHERICITY for $K = 2$, is **NP**-hard by reducing SATISFIABILITY to it. What can be said about the complexity of K-SPHERICITY for other values of K? We mention in the Introduction that graphs with sphericity 1 are called unit interval graphs and can be recognized in polynomial time. That is, K-SPHERICITY is solvable in polynomial time for $K = 1$.

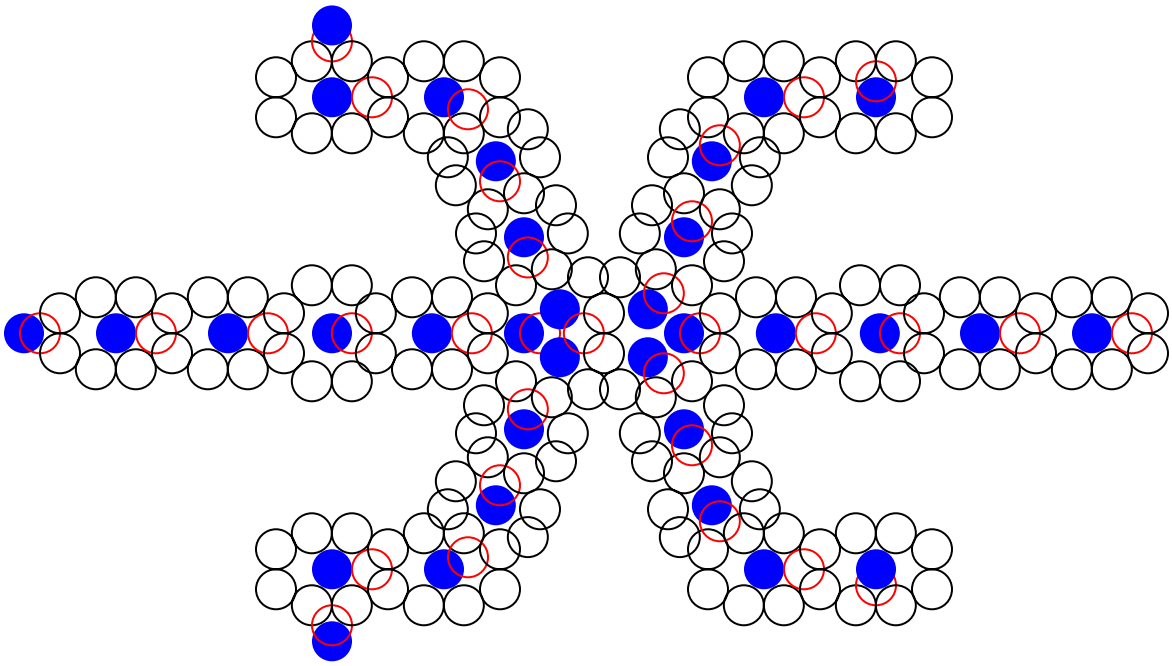
The complexity of 3-SPHERICITY is an open question due to [Hav82a, Hav82b, HKC83] arising from studies of molecular conformation. This problem, too, is **NP**-hard since our reduction for 2-SPHERICITY can be modified for 3-SPHERICITY. The basic building blocks are again beads in cages, but this time the cages are three-dimensional. Their capacities again would be deduced by optimal sphere packing arguments, but in three dimensions. Optimal packing is admittedly more complicated in higher dimensions, but we simply require that some such packings, and therefore constrained cages, exist. Note that the optimal packings need not be unique.

Embed the SATISFIABILITY graph in the three-dimensional grid as follows. Begin by embedding the clauses and variables in a two-dimensional grid, as before. Think of this as a horizontal layer, with $z = 1$. Now, add more horizontal layers to the grid for a total of $3|C|$ layers, so that each occurrence of a literal in a clause has its own layer. Route a literal to a clause by first routing from the literal to the corresponding layer, using the third dimension. Then route over to the clause's (x, y)



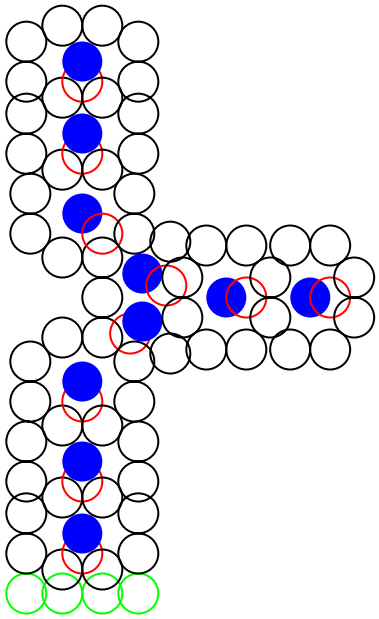
-68	-5		-68	5		-63	-13		-63	-2	chain	-59	8	
-56	-2	bead	-54	-13		-50	8		-46	-9		-46	41	
-44	0		-44	32		-41	-6	chain	-41	11		-41	21	
-40	-17		-40	49		-36	-7	bead	-36	39	bead	-35	3	
-35	29		-30	-17		-30	16	bead	-30	24	chain	-30	38	chain
-30	49		-27	-2	chain	-25	3		-25	29		-24	-7	bead
-24	39	bead	-20	-17		-20	49		-19	11		-19	21	
-19	38	chain	-16	0		-16	32		-14	-9		-14	41	
-12	9		-12	23		-11	52		-11	61		-10	-3	chain
-9	-60		-9	-50		-9	-32		-9	-22		-6	-2	bead
-6	34	bead	-5	-68		-5	-41		-5	-13		-5	16	
-5	45		-5	68		-2	6	chain	0	-65	chain	0	-55	bead
0	-37	chain	0	-27	bead	0	7	bead	0	21	chain	0	25	bead
0	36	chain	0	49	chain	0	57	bead	3	-8	chain	5	-68	
5	-41		5	-13		5	16		5	45		5	68	
6	-2	bead	6	34	bead	9	-60		9	-50		9	-32	
9	-22		10	35	chain	11	52		11	61		12	9	
12	23		14	-9		14	41		16	0		16	32	
19	-6	chain	19	11		19	21		20	-17		20	49	
24	-7	bead	25	3		25	29		30	-17		30	-6	chain
30	8	chain	30	16	bead	30	34	chain	30	38	bead	30	49	
35	3		35	29		36	-7	bead	40	-17		40	49	
41	11		41	21		44	0		44	32		46	-9	
46	41		50	8		52	-2	chain	54	-13		56	-2	bead
59	8		63	-13		68	-5		68	5				

Table 3: Crossover coordinates. The T and B terminals are oriented N, and the L and R terminals are oriented E. The other orientations can be obtained by keeping all cage and bead locations the same, but rearranging the chain vertices as needed by symmetry.



-145	0	bead	-141	0	chain	-136	-5		-136	5		-127	-9				
-127	9		-122	0	bead	-117	-9		-117	9		-112	0	chain			
-108	-5		-108	5		-99	-9		-99	9		-94	0	bead			
-89	-9		-89	9		-84	0	chain	-82	-64		-82	-54				
-82	54		-82	64		-80	-5		-80	5		-73	-68				
-73	-50		-73	-12		-73	12		-73	50		-73	68				
-68	-77	bead	-68	-73	chain	-68	-59	bead	-68	0	bead	-68	59	bead			
-68	73	chain	-68	77	bead	-63	-68		-63	-50		-63	-12				
-63	0	chain	-63	12		-63	50		-63	68		-58	-59	chain			
-58	59	chain	-57	-5		-57	5		-54	-64		-54	-54				
-54	54		-54	64		-48	-9		-48	9		-45	-68				
-45	-50		-45	50		-45	68		-43	0	bead	-40	-59	bead			
-40	59	bead	-39	-44		-39	44		-38	-9		-38	9				
-37	-37		-37	37		-35	-68		-35	-50		-35	50				
-35	68		-34	-56	chain	-34	56	chain	-33	0	chain	-32	-25				
-32	25		-30	-18		-30	18		-29	-31		-29	-5				
-29	5		-29	31		-27	-43	bead	-27	43	bead	-26	-64				
-26	-54		-26	-38	chain	-26	38	chain	-26	54		-26	64				
-22	-12		-22	12		-20	-35		-20	-24	bead	-20	0	bead			
-20	24	bead	-20	35		-19	-51		-19	-19	chain	-19	19	chain			
-19	51		-16	-44		-16	0	chain	-16	44		-13	-16				
-13	16		-12	-32		-12	32		-11	-6	bead	-11	6	bead			
-9	-25		-9	25		-5	0	chain	-4	-14		-4	14				
0	-5		0	5		4	-14		4	14		9	-25				
9	25		11	-6	bead	11	6	bead	12	-32		12	32				
13	-16		13	16		15	-10	chain	15	10	chain	16	-44				
16	44		19	-51		19	51		20	-35		20	-24	bead			
20	0	bead	20	24	bead	20	35		22	-28	chain	22	-12				
22	12		22	28	chain	24	0	chain	26	-64		26	-54				
26	54		26	64		27	-43	bead	27	43	bead	29	-47	chain			
29	-31		29	-5		29	5		29	31		29	47	chain			
30	-18		30	18		32	-25		32	25		35	-68				
35	-50		35	50		35	68		37	-37		37	37				
38	-9		38	9		39	-44		39	44		40	-59	bead			
40	59	bead	43	0	bead	45	-68		45	-50		45	50				
45	68		48	-9		48	9		50	-59	chain	50	59	chain			
53	0	chain	54	-64		54	-54		54	54		54	64				
57	-5		57	5		63	-68		63	-50		63	50				
63	68		64	-12		64	12		68	-63	chain	68	-59	bead			
68	59	bead	68	63	chain	69	0	bead	73	-68		73	-50				
73	50		73	68		74	-12		74	0	chain	74	12				
80	-5		80	5		82	-64		82	-54		82	54				
82	64		89	-9		89	9		94	0	bead	99	-9				
99	9		104	0	chain	108	-5		108	5		117	-9				
117	9		122	0	bead	127	-9		127	9		132	0	chain			
136	-5		136	5													

Table 4: Truth setter coordinates. This truth setting component realization has the 3-bead cluster inside the left 3-cage. Therefore, the three leftmost T, L, and B terminals are oriented N, W, and S respectively. The other possible configurations can be obtained by symmetry: retain the coordinates of the cages and beads, but alter the chain coordinates accordingly. This realization shows the leftmost T, L, and B terminal beads being absorbed by the ether. In a complete realization, delete those external chain and bead vertices on terminals connected to wires or corners.



-14	-74	mortar	-14	-64		-14	-54		-14	-46		-14	-36	
-14	36		-14	46		-14	54		-14	64		-13	-26	
-13	-16		-13	16		-13	26		-5	-74	cap	-5	-68	
-5	-50		-5	-32		-5	-10		-5	10		-5	32	
-5	50		-5	68		0	-64	chain	0	-59	bead	0	-46	chain
0	-41	bead	0	-26	chain	0	-21	bead	0	21	bead	0	36	chain
0	41	bead	0	54	chain	0	59	bead	5	-74	cap	5	-68	
5	-50		5	-32		5	-10		5	0		5	10	
5	16	chain	5	32		5	50		5	68		12	-9	chain
13	-26		13	-16		13	16		13	26		14	-74	mortar
14	-64		14	-54		14	-46		14	-36		14	36	
14	46		14	54		14	64		15	-6	bead	15	6	bead
21	3	chain	22	-14		22	14		25	-5		25	5	
31	-13		31	13		36	0	bead	41	-13		41	0	chain
41	13		47	-5		47	5		52	-13		52	13	
57	0	bead	62	-13		62	0	chain	62	13		68	-5	
68	5													

Table 5: Clause (with cap and mortar) coordinates. The Bottom terminal is capped in this realization. Both the Top and Bottom terminals are oriented N. The Right terminal is oriented W. Other caps and orientations can be obtained by symmetry. All cage and bead vertex locations are the same; only the chain vertices need be rearranged.

coordinates on the dedicated grid layer. Conclude by routing to the clause using the third dimension. In this way, no wires interfere, and the construction does not even require cross-over components. We believe that an analogous construction is possible also in higher dimensions.

Conjecture 9 *K-SPHERICITY is* **NP***-hard for all fixed K greater than 1.*

It is apparent from the nature of cages and beads that the assumption of at most 3 literals per clause, and that each variable appears in at most 3 clauses, is not really necessary to our reduction. Fan out and fan in components can be manufactured from the basic building blocks used in the construction. They are avoided here in order to clarify the exposition.

Note that an instance of 2-SPHERICITY constructed by the reduction in this paper is planar. This is true whether or not it has a realization, since it is always possible to embed all beads and chains inside their incident cage without crossing edges. That is, 2-SPHERICITY remains **NP**-hard even if the graph is planar.

Finally, an instance of 2-SPHERICITY constructed by our reduction has a 3-dimensional realization. To see this, embed the cages in the (horizontal) plane, as usual. Then embed the chains and beads directly above their incident hinges, using the third dimension (imagine helium filled balloons on strings). Since the hinges are independent of one another, so are the chains and beads. This implies that 2-SPHERICITY remains **NP**-hard even if the graph has sphericity at most 3.

References

[BL76] K.S. Booth and G.S. Luecker. Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. *J. Comput. System Sci.*, 13:335–379, 1976.

[CCJ90] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 165–177, 1990.

[Coo71] Steven A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual Symposium on Theory of Computing*, pages 151–158, Association for Computing Machinery, New York, 1971. Cited by [GJ79].

[FG65] D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–355, 1965.

[Fis83] Peter C. Fishburn. On the sphericity and cubicity of graphs. *Journal of Combinatorial Theory, Series B*, 35:309–318, 1983.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of* **NP***-completeness.* W.H. Freeman and Company, 1979.

[Hal80] William K. Hale. Frequency assignment: theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[Hav82a] Timothy Franklin Havel. *The Combinatorial Distance Geometry Approach to the Calculation of Molecular Conformation.* PhD thesis, University of California, Berkeley, 1982. Cited by [Fis83].

[Hav82b] Timothy Franklin Havel. The combinatorial distance geometry approach to the calculation of molecular conformation. *Congressus Numerantium*, 35(4):361–371, 1982.

[HKC83] Timothy Franklin Havel, Irwin D. Kuntz, and Gordon M. Crippen. The combinatorial distance geometry method for the calculation of molecular conformation. I. a new approach to an old problem. *Journal of Theoretical Biology*, 35(4):359–381, 1983.

[JAMS91] David S. Johnson, Cecilia A. Aragon, Lyle.A. McGeogh, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partioning. *Operations Research*, 39(3):378–406, May-June 1991.

[MHR92] M.V. Marathe, H.B. Hunt III, and S.S. Ravi. Geometry based approximations for intersection graphs. In *Fourth Canadian Conference on Computational Geometry*, pages 244–249, 10–14 August 1992.

[Rob68] Fred S. Roberts. Indifference graphs. In *Proof Techniques in Graph Theory*, pages 139–146, Academic Press, New York and London, February 1968. Proceedings of the Second Ann Arbor Graph Theory Conference.

[Rob91] Fred S. Roberts. *Quo Vadis, Graph Theory.* Technical Report 91–33, DIMACS, May 1991.