Planar Embeddings of Graphs with Specified Edge Lengths*

Sergio Cabello[1], Erik D. Demaine[2], and Günter Rote[3]

[1] Institute of Information and Computing Sciences, Universiteit Utrecht,
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands.
sergio@cs.uu.nl
[2] MIT Laboratory for Computer Science,
200 Technology Square, Cambridge, MA 02139, USA.
edemaine@mit.edu
[3] Institut für Informatik, Freie Universität Berlin,
Takustraße 9, 14195 Berlin, Germany.
rote@inf.fu-berlin.de

Abstract. We consider the problem of finding a planar embedding of a (planar) graph with a prescribed Euclidean length on every edge. There has been substantial previous work on the problem without the planarity restrictions, which has close connections to rigidity theory, and where it is easy to see that the problem is NP-hard. In contrast, we show that the problem is tractable—indeed, solvable in linear time on a real RAM—for planar embeddings of planar 3-connected triangulations, even if the outer face is not a triangle. This result is essentially tight: the problem becomes NP-hard if we consider instead planar embeddings of planar 3-connected infinitesimally rigid graphs, a natural relaxation of triangulations in this context.

1 Introduction

Given a graph and the lengths of the edges from a planar straight-line drawing of a (planar) graph, can we reconstruct the drawing? When is this reconstruction unique? Can we recognize realizable length assignments? These three problems have an extensive history, having been studied in the fields of computational geometry [8, 14, 26, 27], rigidity theory [7, 16, 18], sensor networks [5, 25], and structural analysis of molecules [1, 9, 17]. The reconstruction problem arises frequently when only distance information is known about a given structure, such as the atoms in a protein [1, 9, 17] or the nodes in an ad-hoc wireless network [5, 24, 25]. A reconstruction is always unique and easy-to-compute for a complete graph of (exact) distances, or any graph that can be "shelled" by incrementally locating nodes according to the distances to three noncollinear located neighbors

* The research by the first author has been supported by Cornelis Lely Stichting and by a Marie Curie Fellowship of the European Community programme "Combinatorics, Geometry, and Computation" under contract number HPMT-CT-2001-00282. The third author is partially supported by the National Science Foundation under grant number ITR ANI-0205445.

(Fig. 1). More interesting is that such graphs include visibility graphs [8] and segment visibility graphs [14]. In general, however, the reconstruction problem is NP-hard [27], even in the strong sense [26]. The uniqueness of a reconstruction in the generic case (in 2D) was recently shown to be testable in polynomial time by a simple characterization related to generic rigidity [16, 18], but this result has not yet lead to efficient algorithms for actual reconstruction in the generic case.

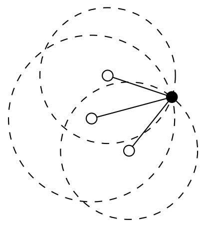

Fig. 1. Locating a vertex from the distances to three located neighbors.

Planar embeddings. We consider a variation on this basic problem of reconstruction from distances: the graph is planar and the planar straight-line drawing must be a planar embedding (edges not incident in the graph should not be incident in the plane). Our problem is then, given a planar graph with prescribed lengths on the edges, to construct a planar embedding of the graph that adheres to the specified edge lengths, and determine whether this embedding is unique, or determine that no such embedding exists.

Applications. The restriction to planar embeddings makes sense in many applications, for example when the underlying structure we want to reconstruct is known to be planar. Another application specifically in the context of graph drawing is the generation of linear cartograms. A *cartogram* is a map in which the size of each entity is proportional to some value associated with the entity [4]. *Area cartograms* are the most common example, in which the area of each region is proportional to some function of the region, e.g., its population. In *linear cartograms*, we want to display a network in such a way that the length of a connection is related to some characteristic of the connection. In common maps, this length is correlated (through a planar projection of the sphere) with the length of the connection in the real world. However, we may be interested in showing, e.g., the traveling time for each connection, or the traffic on each connection. The construction of such a map can be modeled by defining the length of each edge appropriately and trying to realize the graph with these edge lengths. In real-life applications, we would also like to keep some resemblance with the original network, and so we may restrict where the vertices of the graph can be embedded. However, as we will see, the problem is already hard without this restriction.

Our results. We prove the following main results:

1. Even for planar 3-connected graphs, deciding planar embeddability with unit edge lengths is strongly NP-hard, even when the embeddings are guaranteed to be infinitesimally rigid. [1] This improves upon results in [13], where the

[1] Infinitesimal rigidity is a strong form of rigidity, stating that no first-order motion of the vertices preserves the lengths of the edges to the first order. See e.g. [15] for formal definitions.

hardness is shown for 2-connected graphs with unit edge lengths and for 3-connected graphs with arbitrary edge lengths. Another (aesthetic) difference is that our reduction is directly from planar 3-SAT, rather than using a synthetic problem as a bridge. See Section 3.

2. For planar 3-connected graphs, we can decide in $O(|V|)$ time whether there is a planar embedding with specified edge lengths in which only the outermost face is not a triangle. Furthermore, such an embedding is always unique up to rigid motions (translations, rotations, and reflections), and can be constructed in $O(|V|)$ time. More generally, we can find planar embeddings in which the triangular faces form a connected family of cells and the nontriangular faces form a forest of cells. See Section 2.

These results give a fairly precise division between tractable and intractable forms of planar embedding with specified edge lengths. To our surprise, triangles seem to play a more fundamental role than other rigid structures, despite the close connections between rigidity and embedding with specified edge lengths [7, 16, 18]. Other than visibility graphs [8, 14] and dense graphs [1], our results are the first positive results for efficient embeddings of (special) graphs with specified edge lengths.

Model of computation. Even the simple task of embedding a triangle with given side lengths in the plane and computing coordinates for the vertices involves square roots. Thus, we do not know whether our embedding problems belong to NP, and for our algorithmic results we have to assume the real RAM model of computation [23] which supports constant-time exact arithmetic operations ($+$, $-$, $\times$, $\div$, $\sqrt{\ }$) on real numbers. This model is customary in computational geometry. On the other hand, our NP-hardness result is in the standard Turing machine model, and therefore must be careful to ensure that the input lengths can be encoded in a number of bits that is polynomial in the input graph size.

2 Triangulated Graphs

Let G be a 3-connected planar graph. By Whitney's Theorem, G has only one topological embedding into the 2-dimensional sphere, or equivalently the faces in any planar embedding of G are always induced by the same cycles [12, Chapter 6]. In particular, all embeddings of G have the same dual graph G^*, and once we have fixed the outer face, the topological embedding into the plane is completely determined. This is the basic ingredient for the following result:

Theorem 1. *If $G = (V, E)$ is a 3-connected graph with specified edge lengths, we can decide in $O(|V|)$ time on a real RAM whether there is a planar embedding such that all faces are triangles, with the possible exception of the outer face.*

Proof. Consider any planar embedding of G, which can be computed in $O(|V|)$ time [11]. If two or more faces are not triangles, then we can decide that the desired realization is not possible because of Whitney's theorem. If exactly one face is not a triangle, that face must be the outer face in the desired realization. If all faces are triangles, all the longest edges have to be part of the outer face, which gives us at most two candidates T and T' for the outer face. If T is the outer face, then T' must fit inside T while sharing the common edge, and vice versa. This test leaves us with at most one candidate for the outer face f_{ext}.

All nodes in $G^* \setminus f_{ext}$ correspond to triangular faces. We pick a node f_0 in this graph, and compute coordinates for the vertices of the corresponding triangle that realize the triangle edge lengths. Now we visit all nodes in $G^* \setminus f_{ext}$ using breadth-first search from f_0. When visiting a node f_i, two options arise:

1. If all vertices of the face f_i have already been assigned coordinates, we check that all the edges in f_i have the specified edge lengths.
2. If some vertex of the face f_i has not been assigned coordinates, we know that the other two vertices u, v of f_i participate in another face f_j that has been already visited, and so they have already been assigned coordinates. We can compute the coordinates of the third vertex using the specified edge lengths and the restriction that f_i and f_j must lie on opposite sides of the line segment uv by Whitney's Theorem.

At the end, every edge in the graph has been checked whether it satisfies the specified edge length, including the lengths of the edges of the outer face f_{ext}. In the process, we visited each face once, and we spent constant time per face, so, overall, the embedding process takes $O(|V|)$ time.

We need to check that the realization that we constructed is indeed planar, to avoid situations like the ones depicted in Fig. 2. A simple plane sweep would do this in $O(|V| \log |V|)$ time. To get linear time, we first construct a triangulation of the whole plane: We enclose all points in a large triangle T and triangulate the area between T and the boundary of the outer face f_{ext}. To do this, we insert an edge from an extreme vertex of V to a corner of T and triangulate the resulting simple polygon in linear time [6]. Under the assumption that the original embedding was planar, we obtain a graph which is a triangulation of T and is embedded in the plane without crossings. On the other hand, if the original embedding contains crossings, the triangulation algorithm will either (i) terminate in error, or (ii) it will produce a subdivision of T which is topologically consistent but whose embedding contains crossings. Topological consistency means that the two triangle faces incident to an edge are embedded on different sides of the edge, except for the edges of T where the other triangle is embedded inside T. The planarity condition (ii) for a subdivision can be tested in linear time [10]. □

If *all* faces are triangles and we only want to *test* embeddability without constructing coordinates, it suffices to use the following result.

Lemma 1. *Let G be a triangulated planar graph with a designated outer triangle T and an embedding which embeds the two triangles incident to an edge on different sides of the edge, except for the edges of T. Then the realization is planar if and only if, for all vertices $v \notin T$, the sum of the angles that are incident to v is 360 degrees.*

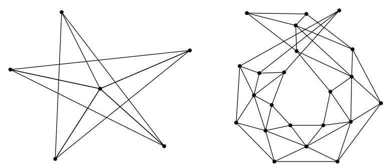

Fig. 2. These examples show that we need to check that the embedding is indeed planar.

It is clearly necessary that the angle sum is 360 degrees for all interior vertices v. To see that this property is sufficient, imagine that the faces (triangles) that we constructed are made of cardboard. The question becomes whether we can arrange these pieces of cardboard, respecting the vertex adjacencies, without overlap in the plane. This question appears in some proofs of the Koebe-Andreyev-Thurston circle packing theorem [22, Chapter 8], and a positive answer is easily shown by induction on the number of vertices: the triangles adjacent to an interior vertex can be packed to form a star-shaped polygon P, which is obviously planar; then you remove that vertex and retriangulate the polygon P.

For graphs of bounded degree, this can be tested in polynomial time in the classical Turing machine model, with rational edge lengths as inputs, using separation bounds for algebraic computations, see [3, 2, 20]. We may also allow square roots of rationals as inputs. (Otherwise, it will be difficult to come up with interesting examples of realizable graphs with rational edge lengths.) For general graphs, this algorithm is singly-exponential in the degree.

On the other hand, observe that, in the proof of the Theorem 1, we have only used that the coordinates of the vertices can be computed by considering the triangular faces in an appropriate order. To get this property, we only need that each vertex of G is incident to a triangular face, and that the set of triangular faces is connected in the dual graph G^*. Therefore, we can weaken the hypothesis on G as follows: the subgraph of G^* induced by vertices of degree 3 is connected, and the subgraph of G^* induced by vertices of degree larger than 3 is a forest. Once we have fixed the outer face, these hypotheses would be enough to prove the result.

3 NP-Hardness

To show the NP-hardness of our problem, we reduce from the P3-SAT (planar 3-satisfiability) problem, which is strongly NP-complete [21]. In an instance of P3-SAT, we are given a planar bipartite graph whose nodes on one side of the bipartition represent the variables $v_1, \ldots, v_n$, and whose nodes on the other side represent the clauses $C_1, \ldots, C_m$, and edges connect each clause to the three variables it contains. Moreover, the variables can be arranged on a horizontal line, and the three-legged clauses be drawn such that all edges lie either above or below this line; and the graph can be drawn on a rectangular grid of polynomial size as shown in Fig. 3, left [19].

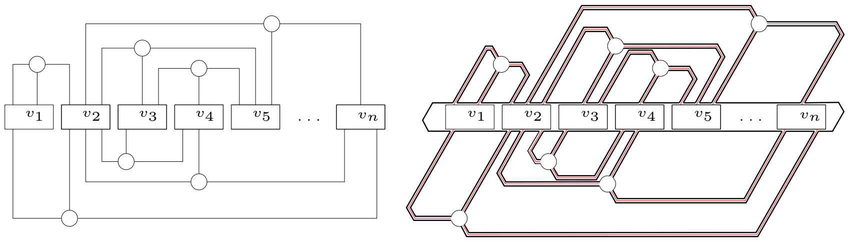

Fig. 3. Left: example of a planar 3-satisfiability instance. The variables can be arranged on a straight line, and the clauses are represented as a vertex with three orthogonal edges leaving from it and one bend in each edge. Right: High-level sketch of NP-hardness reduction. We will make a rigid 3-connected structure that resembles the bold lines.

The high-level workings of the reduction are as follows. We slant the grid into a hexagonal grid to get angles that are multiples of 60 degrees. This slant will allow us to make all lengths one. Furthermore, we modify the drawing so that all the corners have angles of 120 degrees, and the three edges arriving at a clause form angles of 120 degrees; see Fig. 3, right. We make a rigid structure that will leave a tunnel for each edge connecting a variable with a clause. A variable will be represented by a rigid structure that has two different realizations, representing the truth assignment of the variable. The value of the literal will be transmitted to the clause through the tunnel corresponding to the edge, and we will represent the clause by a structure that can be realized if and only if at least one of the literals is true. Furthermore, each of the lines in the figure will be represented by a rigid 3-connected bar, like a "thick" line. This will be the basic trick to make the whole graph 3-connected as well.

The construction relies on three basic rigid structures that are depicted on Fig. 4, and that we explain in the following. In all cases, the grey regions represent 3-connected, rigid structures which are fixed. Firstly, in Fig. 4A, the edges p_1q_1 and p_2q_2 have the same length, and so do p_1p_2 and q_1q_2. Under these conditions, in any realization of this structure, the edges p_1p_2 and q_1q_2 have to be parallel. Secondly, in Fig. 4B, if the vertices p_1 and p_2, marked with squares, are fixed, then the vertex marked with a circle has two possible positions, q and q'. This is so because the distance between this vertex and p_1 and p_2 is fixed, and therefore it has to be placed at the intersection of two circles centered at p_1 and p_2. Finally, in Fig. 4C, there is a 3-connected structure that allows q to rotate around p.

Theorem 2. *Deciding planar embeddability of a planar 3-connected graph with specified edge lengths is NP-hard.*

Proof. We have already described the general idea, so it only remains to describe the gadgets that are used. For the tunnels, we need a structure that allows us

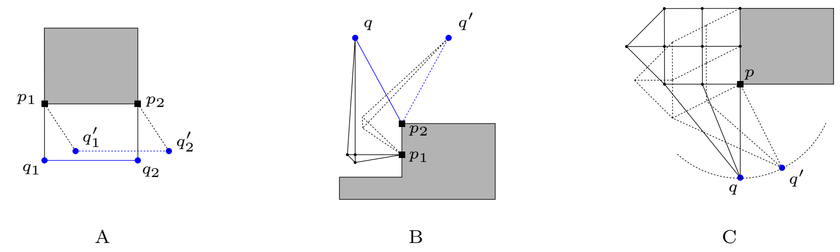

Fig. 4. Assume that the grey regions are rigid and fixed. A. The segments p_1p_2 and q_1q_2 are parallel in any realization. B. The vertex q can only be realized in two positions. C. How to make rotations while keeping 3-connectedness.

to fix the relative positions of both sides of the tunnel, while transmitting the value of the literal through the tunnel. The value will be either true or false, so we need a structure that allows two realizations.

In Fig. 5A the *holder* gadget is shown. Consider the upper half of it. Observe that the two points that are marked with big dots, p_1, p_2, and the two points that are marked with squares, q_1, q_2, represent a situation like shown in Fig. 4A. Therefore, the bar that supports q_1, q_2 is always parallel to the one that supports p_1, p_2, and the point q_2 is always vertically above point q. The points q, q_2 and p_2, implement the idea shown in Fig. 4B, and so p_2 has only two possible placements with respect to q, q_2. Overall, this implies that the upper half of Fig. 5A can be realized in two ways.

The holder gadget can be realized in four different ways: two of them keep the relative position of both sides of the tunnel (A and B), while two of them would move them (C and D). We can concatenate two of these gadgets with one *bend*, as shown in Fig. 5E, in such a way that the realizations C and D are not possible. Thus, the two sides of the tunnel are connected in a (globally) rigid way. We define the *transmitter* to be the bar that is inside the tunnel, because it will transmit the truth value of the literal from the variable to the clause. Below we will discuss the possible realizations of the transmitter.

The structure that we have described is 3-connected, and so we can construct a rigid 3-connected structure, as shown in Fig. 3, right, where the distance between the upper and the lower part will be defined later on by the height of the variables. The sides of the tunnels taken together form a rigid structure in which the transmitters and the variables can move: If a tunnel contains a bend, its two sides can be connected rigidly by two holders, as in Fig. 5E. One can check that the sides of a tunnel without a bend are always connected to a tunnel with a bend, and therefore are also immobile. (Or we could introduce two bends in a zigzag way to make an otherwise straight tunnel rigid in its own right.)

We still have to discuss how the variables, the transmitter, and the clauses work.

For each variable we repeat the structure of the upper half of the holder gadget, but with a thicker bar (*variable-bar*) inside; see Fig. 6. Consider the

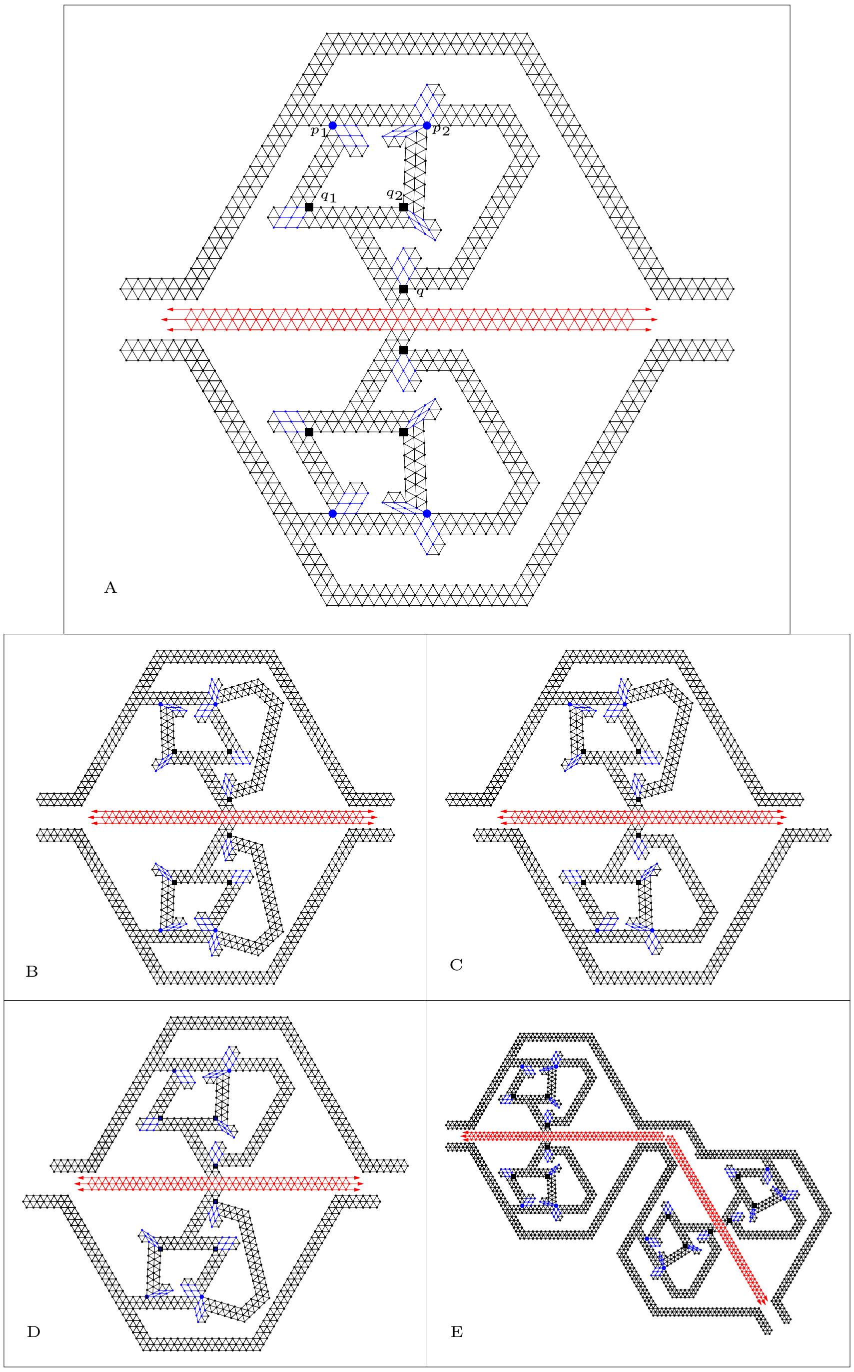

Fig. 5. A. The holder gadget. A–D. Four possible positions of the holder. E. Two holders connected by a bend. This rigidly connects the sides of the tunnel.

realization of the structure assigned to true. On the sides of the variable-bar that are facing the tunnels, for each literal that is not negated, we place an indentation on it that prolongates the tunnel of the literal. For the literals that are negated, we place such an indentation on the part of the variable bar that faces the tunnel in the “false” realization of the structure; see Fig. 6. We have to make the variable-bar large enough that tunnels for all occurrences of each variable can be accommodated on its sides. (In Fig. 6, there are three tunnels on each side.)

The graph that we have constructed so far is 3-connected and rigid. Furthermore, whenever a literal is true, the transmitter bar inside the tunnel can be pushed towards the variable-bar. Furthermore, we can transmit this “pushing information”, or pressure, through the tunnel, and also through the corners using Fig. 5E, so that it can be used at the clause.

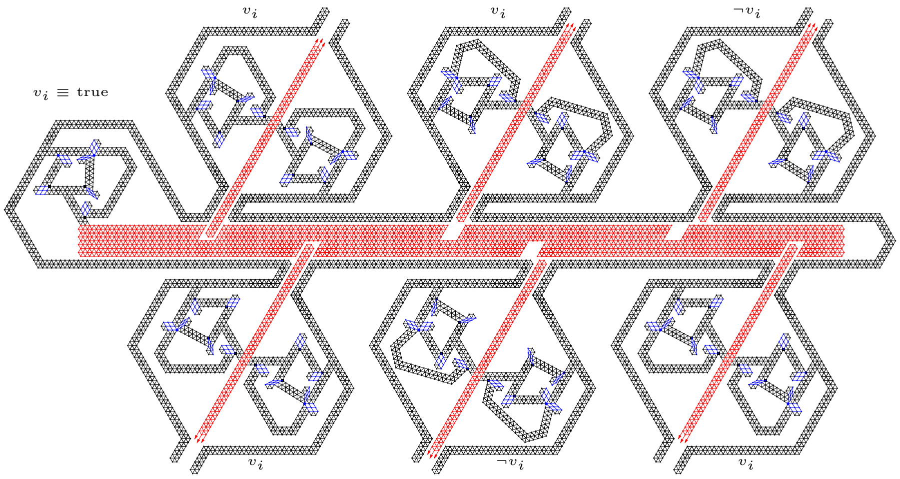

Fig. 6. A variable assigned to true.

Our next goal is to design a *clause checker* that is realizable if and only if one of the three transmitters can be pushed towards its variable. It turns out to be easier to solve the reverse problem: a clause that is realizable if and only if one of the transmitters is pushed towards the clause. Therefore, we design a pushing-inverter which we place on each tunnel just before the clause. It is described in Fig. 7, where its two possible realizations are displayed. The inverter is 3-connected, and inverts pressure towards the clause into pressure towards the variable, and vice versa.

Finally, a clause is described in Fig. 8, with its relevant realizations. The big dots in each literal indicate the two possible positions for the end of the transmitter, and the one that is closer to the center indicates that the literal is true (pushing towards the clause). In all cases, the position of the big dot in

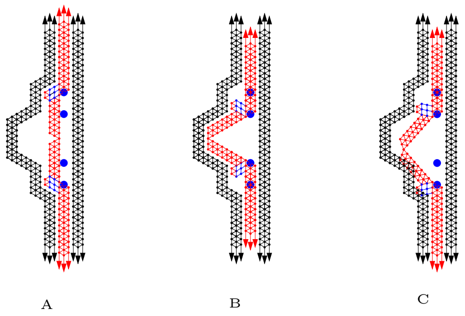

Fig. 7. An inverter. A–B are realizable, but C is not.

the center is completely determined by the values of l_i and l_j. When all l_i, l_j, l_k are false, then, the big dot in the center is too far from l_k to be realizable; see Fig. 8A. In the other cases, it is always realizable; see Fig. 8B–D for some cases.

To conclude, we summarize the argument why a realization of the graph corresponds to a satisfying truth assignment. The clause checker can be realized if and only if at least one transmitter is at the position closer to the clause checker. This can only be the case if, at the variable side of the corresponding inverter, the transmitter is pushed away from the clause checker. This pushing is transmitted through all bends and holders to the variable wheels. It follows that the literal must be true.

One can check by inspection that the clause-gadget is 3-connected, and therefore the whole construction is 3-connected. Furthermore, the lengths of the constructed graph are one because the vertices and edges lie on a hexagonal grid. The grid has polynomial size. Therefore, we are using a polynomial number of edges and bits, and so the reduction can be done in polynomial time. □

Observe that, when the graph is realizable, the realization is infinitesimally rigid. In other words, its vertices cannot be infinitesimally perturbed in a way that preserves the edge lengths to the first order. This condition is stronger than rigidity, and implies that the underlying graph is generically rigid [15]. Therefore, the problem remains NP-hard even when we know that the graph is generically rigid.

The 3-SAT problem is NP-hard even if each variable occurs at most 6 times, and this property is maintained in the reduction from 3-SAT to P3-SAT. Therefore, the faces that participate in the variable gadget have bounded degree. By filling the free space between the tunnels and on the outside by a triangulation, we can make sure that all the faces have bounded degree. Therefore, the problem remains NP-hard even if we assume bounded face degree.

Acknowledgements

We would like to thank the anonymous referee who brought [10, 13] to our attention.

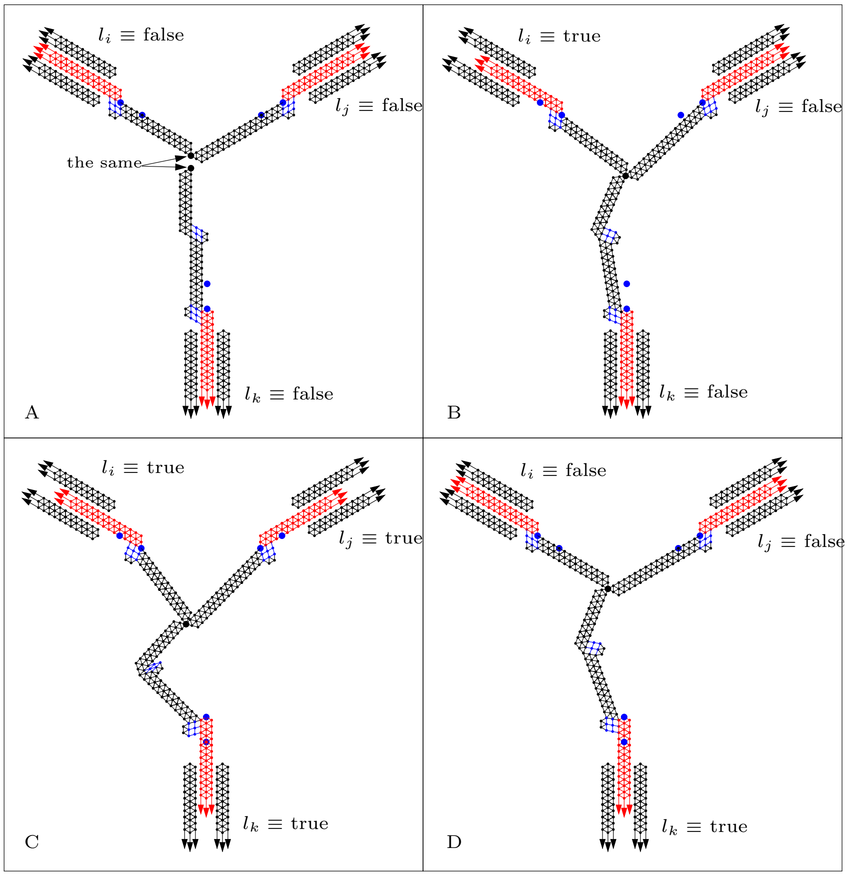

Fig. 8. A clause checker. The situation in A is not realizable, but the ones in B–D are realizable.

References

1. B. Berger, J. Kleinberg, and T. Leighton. Reconstructing a three-dimensional model with arbitrary errors. In *Proc. 28th Annu. ACM Sympos. Theory Comput.*, pages 449–458, May 1996.
2. C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. A strong and easily computable separation bound for arithmetic expressions involving radicals. *Algorithmica*, 27(1):87–99, 2000.
3. C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In *Algorithms — ESA 2001, 9th Annual European Symposium, Proceedings*, volume 2161 of *Lecture Notes in Computer Science*, pages 254–265. Springer-Verlag, 2001.
4. J. Campbell. *Map Use and Analysis*. McGraw-Hill, Boston, 4th edition, 2001.
5. S. Čapkun, M. Hamdi, and J. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, pages 3481–3490, January 2001.

6. B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.
7. R. Connelly. On generic global rigidity. In P. Gritzman and B. Sturmfels, editors, *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, volume 4 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 147–155. AMS Press, 1991.
8. C. Coullard and A. Lubiw. Distance visibility graphs. *Internat. J. Comput. Geom. Appl.*, 2(4):349–362, 1992.
9. G. M. Crippen and T. F. Havel. *Distance Geometry and Molecular Conformation.* John Wiley & Sons, 1988.
10. O. Devillers, G. Liotta, F. P. Preparata, and R. Tamassia. Checking the convexity of polytopes and the planarity of subdivisions. *Comput. Geom. Theory Appl.*, 11:187–208, 1998.
11. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.
12. R. Diestel. *Graph Theory.* Springer-Verlag, New York, 2nd edition, 2000.
13. P. Eades and N. Wormald. Fixed edge length graph drawing is NP-hard. *Discrete Appl. Math.*, 28:111–134, 1990.
14. H. Everett, C. T. Hoàng, K. Kilakos, and M. Noy. Distance segment visibility graphs. Manuscript, 1999. `http://www.loria.fr/~everett/publications/distance.html`.
15. J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity.* American Mathematical Society, 1993.
16. B. Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65–84, February 1992.
17. B. Hendrickson. The molecule problem: Exploiting structure in global optimization. *SIAM J. on Optimization*, 5:835–857, 1995.
18. B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs. Manuscript, March 2003.
19. D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. on Discrete Mathematics*, 5(3):422–427, Aug. 1992.
20. C. Li and C. Yap. A new constructive root bound for algebraic expressions. In *Proc. 12th Annu. ACM–SIAM Sympos. Discrete Algorithms*, pages 496–505, 2001.
21. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
22. J. Pach and P. Agarwal. *Combinatorial Geometry.* John Wiley & Sons, New York, NY, 1995.
23. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 3rd edition, Oct. 1990.
24. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *Proceedings of 6th Annual International Conference on Mobile Computing and Networking*, pages 32–43, Boston, MA, August 2000.
25. C. Savarese, J. Rabaey, and J. Beutel. Locationing in distributed ad-hoc wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2037–2040, Salt Lake City, UT, May 2001.
26. J. B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proc. 17th Allerton Conf. Commun. Control Comput.*, pages 480–489, 1979.
27. Y. Yemini. Some theoretical aspects of position-location problems. In *Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 1–8, 1979.