
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 11, no. 1, pp. 259–276 (2007)

Planar Embeddings of Graphs with Specified Edge Lengths

Sergio Cabello

Department of Mathematics, University of Ljubljana,
Jadranska 19, SI-1000 Ljubljana, Slovenia.
sergio.cabello@fmf.uni-lj.si

Erik D. Demaine

MIT Laboratory for Computer Science,
200 Technology Square, Cambridge, MA 02139, USA.
edemaine@mit.edu

Günter Rote

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 straight-line embedding of a 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 straight-line 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 straight-line embeddings of planar 3-connected infinitesimally rigid graphs with unit edge lengths, a natural relaxation of triangulations in this context.

Article Type	Communicated by	Submitted	Revised
Regular paper	J.S.B. Mitchell	June 2005	June 2007

A preliminary version of this paper has appeared in the Proceedings of Graph Drawing 2003, Lecture Notes in Computer Science 2912, Springer Verlag. The research by the first author has been done as PhD student at the Institute of Information and Computing Sciences, Universiteit Utrecht, The Netherlands, 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 second author is partially supported by the National Science Foundation under grant number ITR ANI-0205445.

1 Introduction

Given a graph and a prescribed length for each of its edges, can we make a straight-line drawing of the graph where the length of each edge is the prescribed length? When is this drawing unique? Can we recognize realizable length assignments? These three problems have an extensive history, having been studied in the fields of computational geometry [9, 15, 27, 29], rigidity theory [8, 17, 19], sensor networks [6, 26], and structural analysis of molecules [1, 10, 18]. This reconstruction problem arises frequently when only distance information is known about a given structure, such as the atoms in a protein [1, 10, 18] or the nodes in an ad-hoc wireless network [6, 25, 26]. 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 (Figure 1). More interesting is that visibility graphs [9] and segment visibility graphs [15] can be incrementally "shelled". In general, however, the reconstruction problem is NP-hard [29], even in the strong sense [27]. The uniqueness of a reconstruction in the generic case (in 2D) was shown to be testable in polynomial time by a simple characterization related to generic rigidity [17, 19], but this result has not yet lead to efficient algorithms for actual reconstruction in the generic case.

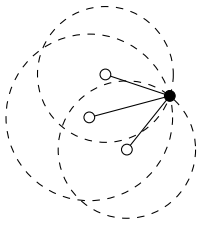


Figure 1: Locating a vertex from the distances to three located neighbors.

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 drawing). Our problem is then, given a planar graph with prescribed lengths on the edges, to construct a planar straight-line 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 [5]. *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 straight-line 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 [28], where weak hardness was shown for (2-connected) planar linkages, and upon [14], where the strong hardness is shown for 2-connected graphs with unit edge lengths and for 3-connected graphs with arbitrary edge lengths. Another (aesthetic) difference with respect to [14] 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 straight-line embedding with specified edge lengths in which only the outer 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 straight-line embeddings in which the triangular faces form a connected family of cells and the nontriangular faces form a forest of cells. This extends the results in [12], where under the assumption that the graph is triangulated and the outer face is convex, the authors can test embeddability in linear time, but without providing an actual embedding. See Section 2.

These results give a fairly precise division between tractable and intractable forms of planar straight-line 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 [8, 17, 19]. Other than visibility graphs [9, 15] and dense graphs [1], our results are the first positive results for efficient embeddings of (special) graphs with specified edge lengths.

[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. [16] for formal definitions.

Model of computation. Even the simple task of describing a straight-line embedding of a triangle with given side lengths 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 [24] which supports constant-time exact arithmetic operations ($+$, $-$, $\times$, $\div$, $\sqrt{}$) on real numbers. The real RAM model is customary in computational geometry. However, this model is unrealistic in practice, since we assume that we can handle arbitrarily long algebraic expressions in constant time. On the other hand, our NP-hardness result is in the standard Turing machine model because we construct a graph with a polynomial number of edges.

2 Triangulated Graphs

Let G be a 3-connected planar graph. By Whitney's Theorem, the faces in any planar embedding of G are always induced by the same cycles [2] [13, Chapter 6] [23, Chapter 2]. 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 *Given a planar 3-connected graph $G = (V, E)$ and a prescribed length for each of its edges, we can decide in $O(|V|)$ time on a real RAM whether there is a planar straight-line embedding with the specified edge lengths 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. 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, any longest edge has 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 $G^* \setminus f^*_{ext}$ are dual of triangular faces. For each triangular face we check if the specified lengths for its edges satisfy the triangular inequality, that is, if the length of each edge is smaller than the sum of the lengths of the other two. If the test fails for any face, we can conclude that the desired planar straight-line embedding does not exist. We pick a node f^*_0 in $G^* \setminus f^*_{ext}$, and compute coordinates for the vertices of its dual triangle f_0 that realize the 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 dual face f_i have already been assigned coordinates, we check that all the edges in f_i have the specified edge lengths.

[2] This result holds for arbitrary planar embeddings, where edges do not need to be straight-line segments.

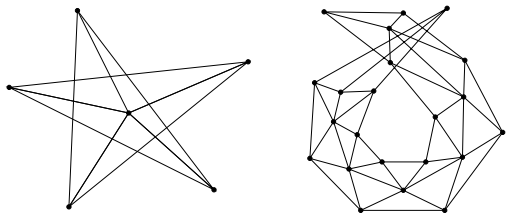


Figure 2: These examples show that we need to check that the drawings are indeed embeddings.

2. If some vertex of the dual 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 whose dual node 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 due to 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 process takes $O(|V|)$ time.

We need to check that the drawing that we constructed is indeed an embedding, to avoid situations like the ones depicted in Figure 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 [7]. 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 drawing 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 existence of crossings (ii) for a convex subdivision can be tested in linear time [11]. □

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 nodes dual to the 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 nodes of degree

3 is connected, and the subgraph of G^* induced by nodes of degree larger than 3 is a forest. Once we have fixed the outer face, these hypotheses would be enough to prove the result.

This result extends the one in [12] in two ways. Firstly, we do not need to assume that the outer face is convex, although for this we have to use linear time triangulation [7]. To avoid using the result in [7], a checker for non-convex subdivisions should be developed, but this problem remains elusive [2, Problem 30]. Secondly, we do not need to assume that the outer face is fixed, which becomes relevant when all the faces of the graph are triangles.

When *all* faces are triangles and we only want to *test* embeddability without constructing coordinates, we can detect the outer face T in linear time like we did in the previous proof, and then use the result by Di Battista and Vismara [12]: the graph G admits a planar realization if and only if, the lengths of the edges of each face satisfy the triangular inequality and, for all vertices $v \notin T$, the sum of the angles that are incident to v (computed from the prescribed edge lengths using the cosine law) is 360 degrees.

This condition can be trivially tested in a real RAM model. However, if we restrict ourselves to a Turing machine model, we can decide this condition as follows. The cosine of an angle incident to v can be described by an algebraic expression on the edge lengths incident to it because of the cosine law. The cosine and sine of a sum of angles can be expressed as a polynomial on the cosines and sines of the angles. Therefore, the condition that the sum of the angles that are incident to v is 360 degrees can be reduced to an evaluation of polynomials: starting from an angle, we consider the sine and cosine of the clockwise partial sums of angles. Studying their signs, we can make sure that the partial sums do not exceed 360 degrees, and then the sum of the angles is 360 degrees if an only if the cosine and the sine of the sum are 1 and 0, respectively.

The maximum degree of the polynomials that we evaluate depends on the degree of the vertices. For graphs of bounded degree, the polynomials have bounded degree, and the condition 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, 4, 21]. 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.

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 [22]. In an instance of P3-SAT, we are given a planar bipartite graph whose nodes on one class of the bipartition represent the variables $v_1, \ldots, v_n$, and whose nodes on the other class 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

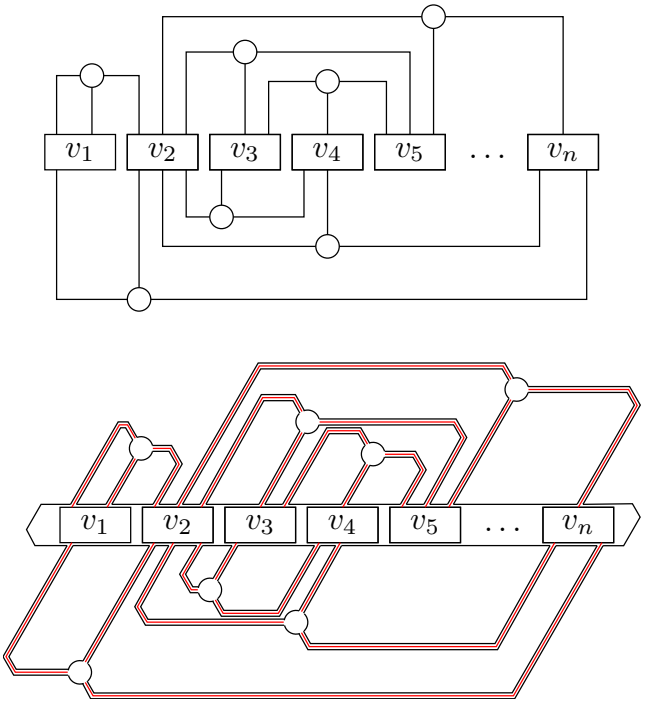
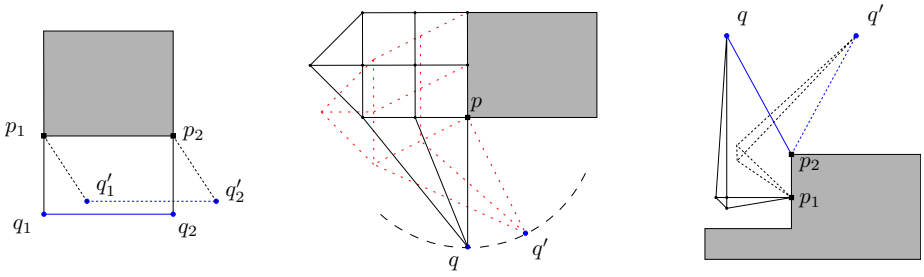


Figure 3: Top: example of a planar 3-satisfiability instance. The variables can be arranged on a straight line, and the clauses are represented as a vertex with three orthogonal edges leaving from it and at most one bend in each edge. Bottom: High-level sketch of NP-hardness reduction. Each line will be replaced by a rigid 3-connected structure.

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 Figure 3, top [20].

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 Figure 3, bottom. We make a rigid structure that will leave a tunnel for each edge connecting a variable with a clause. This rigid structure is realized in an hexagonal grid. 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



A B C

Figure 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. How to make rotations while keeping 3-connectedness. C. The vertex q can only be realized in two positions.

make the whole graph 3-connected as well.

The construction relies on three basic rigid structures that are depicted in Figure 4. In all cases, the grey regions represent 3-connected, rigid structures which are fixed. Firstly, in Figure 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 Figure 4B, there is a 3-connected structure that allows q to rotate around p. Finally, in Figure 4C, 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 C_1 and C_2 centered at p_1 and p_2, respectively. The circles C_1 and C_2 intersect at point q by construction, and by symmetry with respect to the vertical line through their centers p_1, p_2, they must also intersect at the symmetric point q'.

Theorem 2 *Deciding planar straight-line embeddability of planar 3-connected graphs with unit 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 *holder* gadget that allows us 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 gadget that allows two realizations.

In Figure 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 Figure 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 Figure 4C, and so p_2 has only two possible placements with respect to q, q_2. Overall, this implies that

the upper half of Figure 5A can be realized in two ways. The lower half of the holder is a mirrored copy of upper half, and so it also has two realizations.

The holder gadget can be realized in four different ways: two of them keep the relative position of both sides of the tunnel (Figure 5A and 5B), while two of them would move them (Figure 6A and 6B). We can concatenate two of these gadgets with one *bend*, as shown in Figure 6C, in such a way that the realizations in Figure 6A and 6B 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. Observe that in one of the possible realizations of the holder, the transmitter is shifted four units with respect to the other possible realization. Below we will discuss the meaning of 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 Figure 3, bottom, 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 Figure 6C. 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.) Note that in any realization of this rigid structure, the vertices and edges lie on a hexagonal grid because we are joining equilateral triangles.

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 Figure 7. Consider the 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 Figure 7. We have to make the variable-bar large enough that tunnels for all occurrences of each variable can be accommodated on its sides. (In Figure 7, 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 Figure 6C, so that it can be used at the clause.

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 checker that is realizable if and only if at least one of the three transmitters can be pushed towards the clause. Therefore, we design a pushing-inverter which we place on each tunnel just

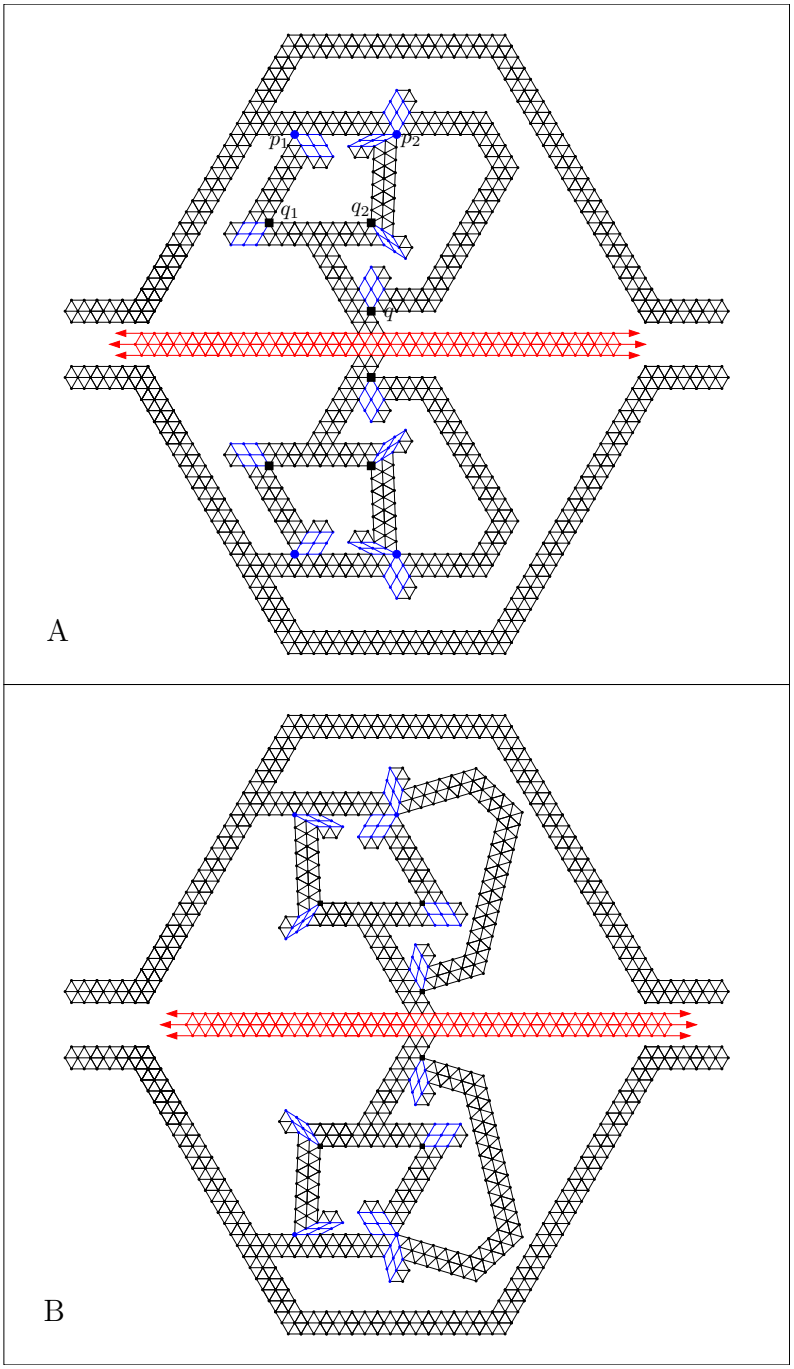


Figure 5: A. The holder gadget. A–B. Two possible realizations of the holder. In B, the transmitter is four units to the right with respect to its position in A.

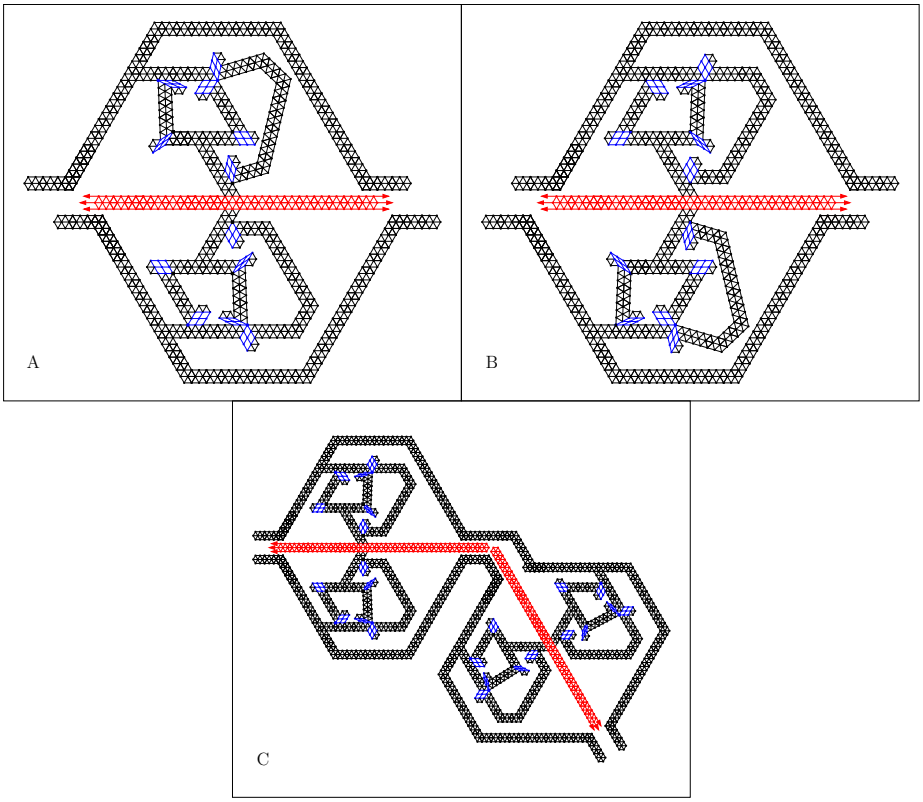


Figure 6: A–B. Two other possible realizations of the holder. C. We avoid these realizations connecting by a bend two consecutive holders. This rigidly connects the sides of the tunnel.

before the clause. It is described in Figure 8, where its two possible realizations are displayed. In particular, the top two thick dots correspond to the possible positions of a vertex depending on whether the pressure is towards the clause or the variable. The *inverter* gadget changes pressure towards the clause into pressure towards the variable, and vice versa. We can make it 3-connected by putting a holder gadget just before it, and another holder gadget immediately after it.

Finally, a clause is described in Figure 9, with its relevant realizations. The big dots in each literal are at four units apart, and they indicate the two possible positions for the end of the transmitter. 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 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 Figure 9A. In the other cases, it is always realizable; see Figure 9B–9D for some cases.

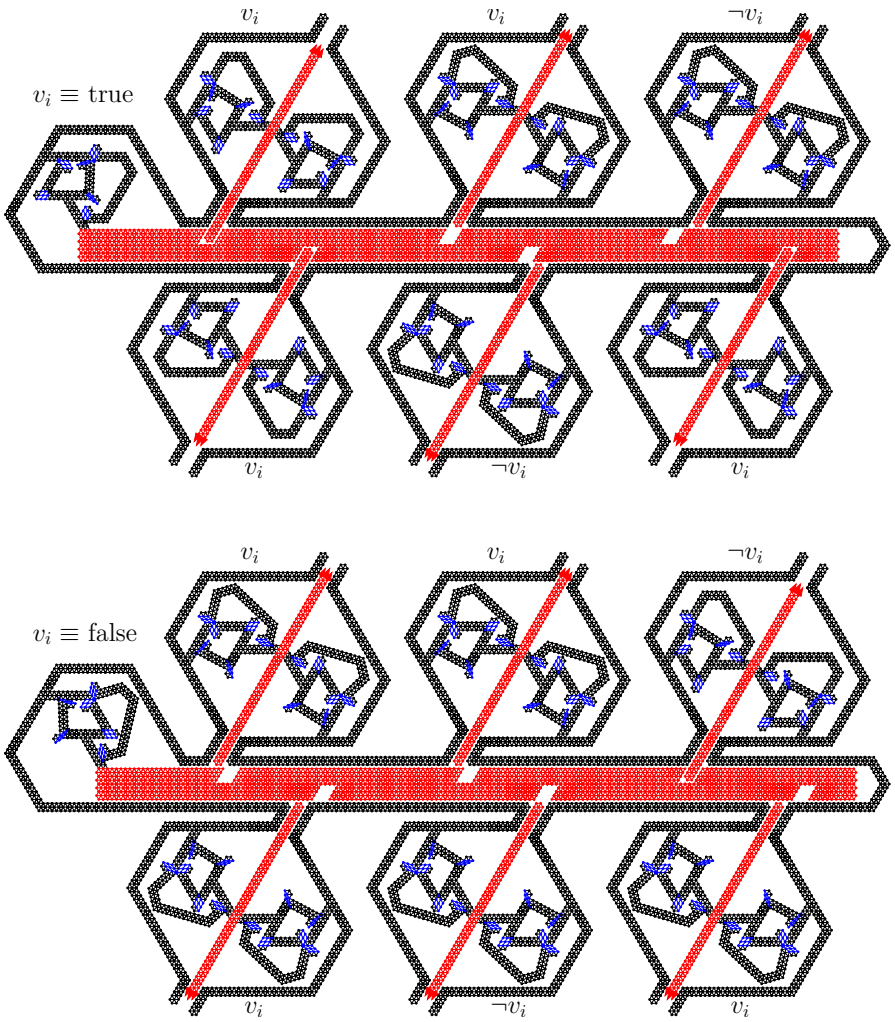


Figure 7: A variable assigned to true (top) and false (bottom). The transmitter can be pushed towards the variable only when the literal is true.

A small example showing all the gadgets at work is in Figure 10. 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.

Placing holders at each transmitter immediately before the clause-gadget and

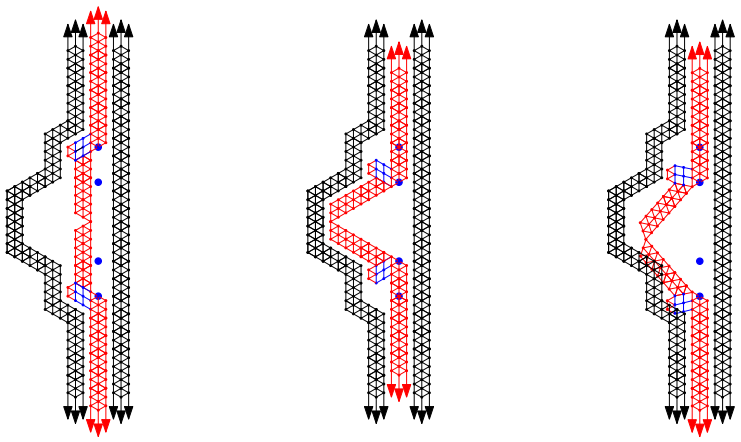


Figure 8: An inverter. A–B are realizable, but C is not.

connecting the rigid structures from different transmitters, as done in Figure 10, we ensure that the construction is 3-connected. Our gadgets only use edges of length one: for any edge e in any gadget, there is at least one realization of the gadget such that e lies on a hexagonal grid. The graph that we construct has a polynomial number of edges because the graph representing the P3-SAT instance was in a grid of polynomial size, and hence 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 [16]. 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 [22]. If a variable needs to accommodate at most six tunnels, then a variable is formed by a bounded number of edges and the faces that participate in the variable gadget have bounded degree. By filling the free space between the tunnels, we can make sure that all the faces have bounded degree. Therefore, the problem remains NP-hard for unit lengths even if we assume bounded face degree, except for the outer face. If we do not require unit lengths, then we can also triangulate the outer face, and we obtain an NP-hardness proof for bounded face degree.

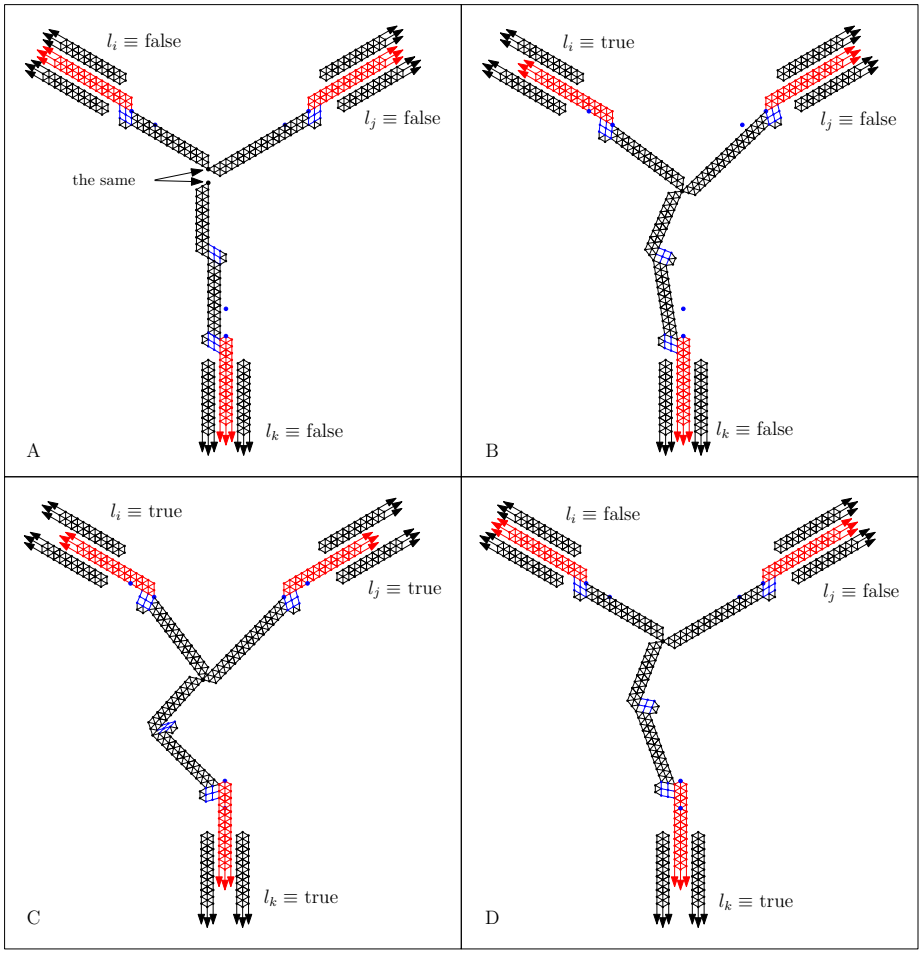


Figure 9: A clause checker. The situation in A is not realizable, but the ones in B–D are realizable.

Acknowledgements

We would like to thank an anonymous referee for several useful recommendations, the anonymous referee of GD'03 who brought [11, 14] to our attention, and Luca Vismara for pointing out the work [12].

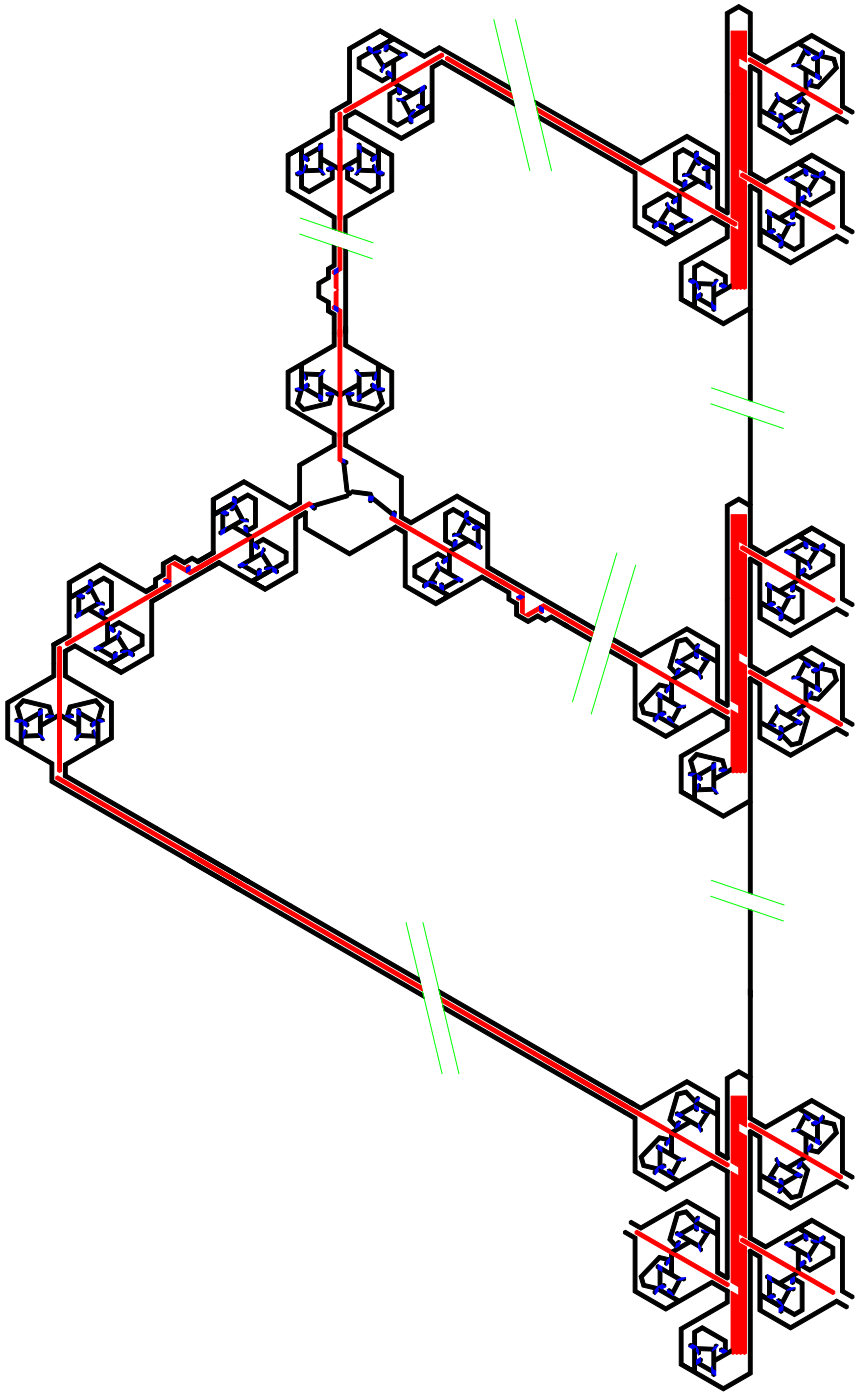


Figure 10: Small example showing all the gadgets together.

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] F. Brandenburg, D. Eppstein, M. Goodrich, S. Kobourov, G. Liotta, and P. Mutzel. Selected open problems in graph drawing. In G. Liotta, editor, *Proc. 11th Int. Symp. Graph Drawing (GD 2003)*, number 2912 in Lecture Notes in Computer Science, pages 515–539. Springer-Verlag, September 2003.

[3] 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.

[4] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In F. Meyer auf der Heide, editor, *Algorithms — ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28–31, 2001, Proceedings*, volume 2161 of *Lecture Notes in Computer Science*, pages 254–265. Springer-Verlag, 2001.

[5] J. Campbell. *Map Use and Analysis.* McGraw-Hill, Boston, 4th edition, 2001.

[6] 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.

[7] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[8] 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.

[9] C. Coullard and A. Lubiw. Distance visibility graphs. *Internat. J. Comput. Geom. Appl.*, 2(4):349–362, 1992.

[10] G. M. Crippen and T. F. Havel. *Distance Geometry and Molecular Conformation.* John Wiley & Sons, 1988.

[11] 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.

[12] G. Di Battista and L. Vismara. Angles of planar triangular graphs. *SIAM Journal on Discrete Mathematics*, 9(3):349–359, 1996. A preliminary version appeared in *STOC'93*.

[13] R. Diestel. *Graph Theory*. Springer-Verlag, New York, 2nd edition, 2000.

[14] P. Eades and N. Wormald. Fixed edge length graph drawing is NP-hard. *Discrete Appl. Math.*, 28:111–134, 1990.

[15] H. Everett, C. T. Hoàng, K. Kilakos, and M. Noy. Distance segment visibility graphs. Manuscript, 1999. `http://www.loria.fr/~everett/publications/distance.html`.

[16] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*. American Mathematical Society, 1993.

[17] B. Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65–84, February 1992.

[18] B. Hendrickson. The molecule problem: Exploiting structure in global optimization. *SIAM J. on Optimization*, 5:835–857, 1995.

[19] B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs, 2005.

[20] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. on Discrete Mathematics*, 5(3):422–427, Aug. 1992.

[21] 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.

[22] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[23] B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, 2001.

[24] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, Oct. 1990.

[25] 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.

[26] 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.

[27] 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.

[28] S. Whitesides. Algorithmic issues in the geometry of planar linkage movement. *Australian Computer Journal*, 24(2):42–50, May 1992.

[29] Y. Yemini. Some theoretical aspects of position-location problems. In *Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 1–8, 1979.