Yale University
Department of Computer Science

On the Computational Complexity of Sensor Network Localization

James Aspnes[1] David Goldenberg[2]
Yang Richard Yang[3]

YALEU/DCS/TR-1282
April 7, 2004

[1]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: `aspnes@cs.yale.edu`. Supported in part by NSF.

[2]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: `david.goldenberg@yale.edu`. Supported in part by 2001 NSF Graduate Research Fellowship DGE0202738.

[3]Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: `yry@cs.yale.edu`. Supported in part by NSF grants ANI-0207399 and ANI-0238038.

On the Computational Complexity of Sensor Network Localization

James Aspnes* David Goldenberg† Yang Richard Yang‡

Abstract

Determining the positions of the sensor nodes in a network is essential to many network functionalities such as routing, coverage and tracking, and event detection. The localization problem for sensor networks is to reconstruct the positions of all of the sensors in a network, given the distances between all pairs of sensors that are within some radius r of each other. In the past few years, many algorithms for solving the localization problem were proposed, without knowing the computational complexity of the problem. In this paper, we show that no polynomial-time algorithm can solve this problem in the worst case, even for sets of distance pairs for which a unique solution exists, unless **RP** = **NP**. We also discuss the consequences of our result and present open problems.

1 Introduction

Localization is the process by which the positions of the nodes in an ad-hoc network are determined. Knowing the positions of the network nodes is essential because many other network functionalities such as location-dependent computing (e.g., [11, 32]), geographic routing (e.g., [18]), coverage and tracking (e.g., [19]), and event detection depend on location. Although localization can be achieved through manual configuration or by exploiting the Global Positioning System (GPS) [15], neither methodology scales well and both have physical limitations. For example, GPS receivers are costly both in terms of hardware and power requirements. Furthermore, since GPS reception requires line-of-sight between the receiver and the satellites, it may not work well in buildings or in the presence of obstructions such as dense vegetation, foliage, or mountains blocking the direct view to the GPS satellites.

Recently, novel schemes have been proposed to determine the locations of the nodes in a sensor network, e.g., [1–6, 8, 10, 12–14, 17, 22–26, 28, 30, 31, 33]. In these schemes, network nodes measure the distances to their neighbors and then try to compute their locations. Although the designs of the previous schemes have demonstrated clever engineering ingenuity, and their effectiveness is evaluated through extensive simulations, the focus of these schemes is on algorithmic design, without knowing the fundamental computational complexity of the localization process.

The localization problem is similar to the graph embedding problem. In [27], Saxe shows that testing the embeddability of weighted graphs in k-space is strongly **NP**-hard. However, the graphs

*Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: aspnes@cs.yale.edu. Supported in part by NSF.

†Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: david.goldenberg@yale.edu. Supported in part by 2001 NSF Graduate Research Fellowship DGE0202738.

‡Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: yry@cs.yale.edu. Supported in part by NSF grants ANI-0207399 and ANI-0238038.

he considers are general graphs. In sensor network localization, since only nodes who are within a communication range can measure their relative distances, the graphs formed by connecting each pair of nodes who can measure each other's distance are better modeled as unit disk graphs [7]. Such constraints could have the potential of allowing computationally efficient localization algorithms to be designed. For example, in [3], Biswas and Ye show that network localization in unit disk graphs can be formulated as a semidefinite programming problem and thus can be efficiently solved. A condition of their algorithm, however, is that the graphs are densely connected. More specifically, their algorithm requires that $\Omega(n^2)$ pairs of nodes know their relative distances, where n is the number of sensor nodes in the network. However, for a general network, it is enough for the localization process to have a unique solution when $O(n)$ pairs of nodes know their distances, if certain conditions are satisfied [9].

Researchers are still looking for efficient algorithms that work for sparse networks. Such algorithms are of great importance, because in the limit as a network with bounded communication range and fixed sensor density grows, the number of known distance pairs grows only linearly in the number of nodes.

In this paper, we show that localization in sparse networks is **NP**-hard. The main result and its consequences are presented in Section 2. The proof of the main result, which makes up the bulk of the paper, is presented in Section 3. Finally, a discussion of conclusions and open problems is presented in Section 4.

2 Localization

The **localization problem** considered in this paper is to reconstruct the positions of a set of sensors given the distances between any pair of sensors that are within some unit disk radius r of each other. Some of the sensors may be **beacons**, sensors with known positions, but our impossibility results are not affected much by whether beacons are available. To avoid precision issues involving irrational distances, we assume that the input to the problem is presented with the distances squared. If we make the further assumption that all sensors have integer coordinates, all distances will be integers as well.

For the main result, we consider a decision version of the localization problem, which we call **UNIT DISK GRAPH RECONSTRUCTION** . This problem essentially asks if a particular graph with given edge lengths can be physically realized as a unit disk graph with a given disk radius in two dimensions.

The input is a graph G where each edge uv of G is labeled with an integer ℓ_{uv}^2, the square of its length, together with an integer r^2 that is the square of the radius of a unit disk. The output is "yes" or "no" depending on whether there exists a set of points in R^2 such that the distance between u and v is ℓ_{uv} whenever uv is an edge in G and exceeds r whenever uv is not an edge in G.

Our main result, is that UNIT DISK GRAPH RECONSTRUCTION is **NP**-hard, based on a reduction from CIRCUIT SATISFIABILITY. The constructed graph for a circuit with m wires has $O(m^2)$ vertices and $O(m^2)$ edges, and the number of solutions to the resulting localization problem is equal to the number of satisfying assignments for the circuit. In each solution to the localization problem, the points can be placed at integer coordinates, and the entire graph fits in an $O(m)$-by-$O(m)$ rectangle, where the constants hidden by the asymptotic notation are small. The construction also permits a constant fraction of the nodes to be placed at known locations.

Formally, we show:

Theorem 1 *There is a polynomial-time reduction from CIRCUIT SATISFIABILITY to UNIT DISK GRAPH RECONSTRUCTION, in which there is a one-to-one correspondence between satisfying assignments to the circuit and solutions to the resulting localization problem.*

The proof of Theorem 1 is given in Section 3. A consequence of this result is:

Corollary 2 *There is no efficient algorithm that solves the localization problem for sparse sensor networks in the worst case unless* $\boldsymbol{P} = \boldsymbol{NP}$.

Proof: Suppose that we have a polynomial-time algorithm that takes as input the distances between sensors from an actual placement in R^2, and recovers the original position of the sensors (relative to each other, or to an appropriate set of beacons). Such an algorithm can be used to solve UNIT DISK GRAPH RECONSTRUCTION by applying it to an instance of the problem (that may or may not have a solution). After reaching its polynomial time bound, the algorithm will either have returned a solution or not. In the first case, we can check if the solution returned is consistent with the distance constraints in the UNIT DISK GRAPH RECONSTRUCTION instance in polynomial time, and accept if and only if the check succeeds. In the second case, we can reject the instance. In both cases we have returned the correct answer for UNIT DISK GRAPH RECONSTRUCTION. ∎

It might appear that this result depends on the possibility of ambiguous reconstructions, where the position of some points is not fully determined by the known distances. However, if we allow randomized reconstruction algorithms, a similar result holds even for graphs that have unique reconstructions

Corollary 3 *There is no efficient algorithm that solves the localization problem for sparse sensor networks that have unique reconstructions unless* $\boldsymbol{RP} = \boldsymbol{NP}$.

Proof: The proof of this claim is by use of the well-known construction of Valiant and Vazirani [29], which gives a randomized Turing reduction from 3SAT to UNIQUE SATISFIABILITY. The essential idea of this reduction is that randomly fixing some of the inputs to the 3SAT problem reduces the number of potential solutions, and repeating the process eventually produces a 3SAT instance with a unique solution with high probability. ∎

Finally, because the graph constructed in the proof of Theorem 1 uses only points with integer coordinates, even an approximate solution that positions each point to within a distance $\epsilon < 1/2$ of its correct location can be used to find the exact locations of all points by rounding each coordinate to the nearest integer. Since the construction uses a fixed value for the unit disk radius r (the natural scale factor for the problem), we have

Corollary 4 *The results of Corollary 2 and Corollary 3 continue to hold even for algorithms that return an approximate location for each point, provided the approximate location is within* $\epsilon \cdot r$ *of the correct location, where* ϵ *is a fixed constant.*

What we do *not* know at present is whether these results continue to hold for solutions that have large positional errors but that give edge lengths close to those in the input. Our suspicion is that edge-length errors accumulate at most polynomially across the graph, but we have not yet carried out the error analysis necessary to prove this. If our suspicion is correct, we would have:

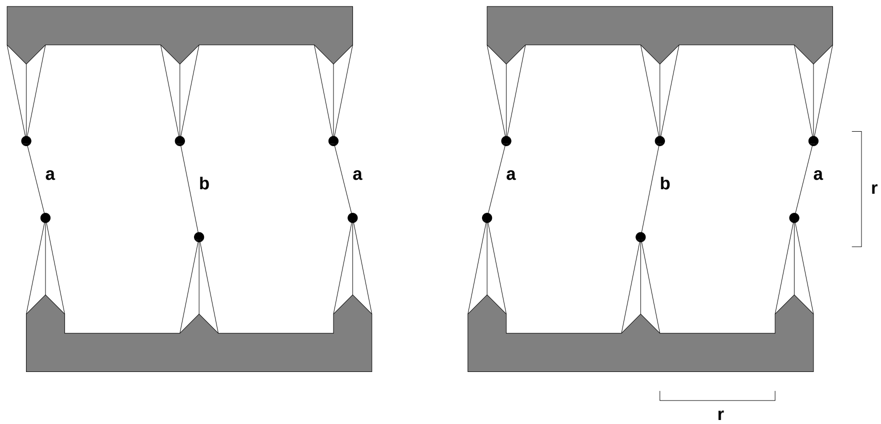

Figure 1: The basic binary hinge. The top and bottom structures, including the prongs supporting the hinges, are rigid; points within the shaded polygons and the edges between them are not depicted explicitly, but are assume to be sufficiently dense to force a unique placement of the hinge endpoints. The three edges between the top and bottom endpoints have lengths a, $b > a$, and a, and enforce that the vertical separation between the two bodies is constant and that the horizontal offset is $\pm h$ for some h. The disk radius r is shown for scale.

Conjecture 5 *The results of Corollary 2 and Corollary 3 continue to hold even for algorithms that return an approximate location for each point, provided the relative error in edge length for each edge is bounded by ϵ/n^c for some fixed constant c.*

3 Proof of Theorem 1

The proof is by constructing a family of graphs that represent arbitrary Boolean circuits, where the relative positions of certain rigid bodies within the graph correspond to signals and NOT and AND gates are built out of additional constraints between these bodies. This gives a reduction to UNIT DISK GRAPH RECONSTRUCTION from CIRCUIT SATISFIABILITY, which shows that determining whether a solution to a particular localization problem in this class exists is **NP**-hard.

The construction requires the development of several preliminary tools. Our main tool will be a two-state hinge structure that permits adjacent parts of the graph to be shifted relative to each other by $\pm h$ for some length h. These hinges will then be used to hook together a two-dimensional grid of rigid bodies whose rows and columns act like wires, transmitting one bit each across the grid. Additional applications of the hinge give junctions between rows and columns (forcing a row and column to carry the same bit, similar to a junction in a circuit), negation, and AND gates (via an implication mechanism that enforces $x \rightarrow y$ for chosen variables x and y).

A binary hinge. We begin by describing a **binary hinge**, which is a connection between two rigid bodies that enforces a fixed offset between them in the direction of the hinge and an offset of $\pm h$ for some h in the perpendicular direction. As depicted in Figure 1, the hinge consists of three edges between fixed points on the surfaces of the bodies, with the inner edge longer than the outer two. Lemma 6 below describes the exact structure of the hinge and its properties.

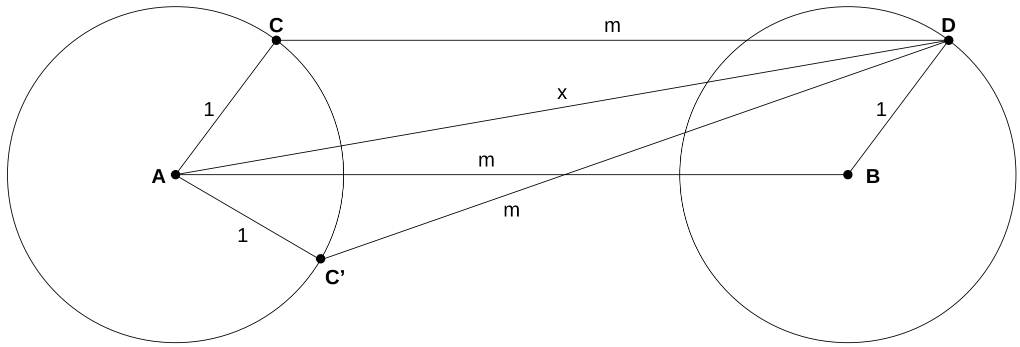

Figure 2: Two configurations of the outer edges of the binary hinge. In the figure, the edges are scaled to have length 1. In the first configuration, the two edges AC and BD are parallel, producing the parallelogram $ACDB$. In the second, the left edge drops to the bottom part of the circle to produce the self-crossing quadrilateral $AC'DB$. The point C' always lies below the AB line provided $m > 1$, which implies that the midpoint of the $C'D$ edge has a y-coordinate no greater than 1/2.

Lemma 6 *Fix points A at coordinates $(0, 0)$, B at coordinates $(m, 0)$, and K at coordinates $(m/2, -k)$ for some $k > 0$. Let $h < \ell\sqrt{3}$ and let C and D be points at distance $m > 1$ from each other such that C is at distance $\sqrt{\ell^2 + h^2}$ from A, D is at distance $\sqrt{\ell^2 + h^2}$ from B, and their midpoint M is at distance $\sqrt{(\ell + k)^2 + h^2}$ from K, where $\ell < m$. Then either C is at coordinates $(-h, \ell)$ and D is at coordinates $(m - h, \ell)$, or C is at $(+h, \ell)$ and D is at $(m + h, \ell)$.*

Proof: It is easily verified that $C = (\pm h, \ell)$ is at distance $\sqrt{\ell^2 + h^2}$ from $A = (0, 0)$, that $D = (m \pm h, \ell)$ is at distance $\sqrt{\ell^2 + h^2}$ from $B = (m, 0)$, and that $M = (m/2 \pm h, \ell)$ is at distance $\sqrt{(\ell + k)^2 + h^2}$ from $K = (m/2, -k)$. It remains to show that only these two solutions exist.

First we consider only the constraints of the outer two edges. The distance constraints imply that C and D lie on circles of radius ℓ centered at A and B respectively. Without loss of generality, let $\sqrt{\ell^2 + h^2} = 1$ and consider Figure 2. For a fixed position of D, C lies on a radius-m circle centered at D; this circle intersects the radius-1 circle centered at A in exactly two points C and C'.

In the first case, $ABDC$ is a parallelogram and the midpoint M of C and D lies on a radius-1 circle centered at the midpoint of A and B. This will give rise to our two solutions.

The second case is more complicated. We will first show that the midpoint M' is close to the AB line. Here we have C' at distance m from D, but $ABDC'$ is not a parallelogram but is instead a quadrilateral that crosses itself. To see that the AB and DC' line segments do in fact cross, observe that $\triangle AC'D$ and $\triangle DBA$ both have edges of length 1, m, and x where x is the length of AD and are thus congruent. Within $\triangle AC'D$, the angle $\angle DAC'$ is opposite a longer side than the angle $\angle C'DA$ and is thus larger; it follows that $\angle DAC'$ is larger than $\angle BAD$ and thus C' and D lie on opposite sides of the AB line. The y-coordinate of M' is thus the average of two quantities y_1 and y_2 which lie between -1 and +1 with opposite sign, and so it lies in the closed interval $[-1/2, 1/2]$.

We have shown that any midpoint M of C and D lies either on the circle or within the shaded region depicted in Figure 3. The middle edge of the hinge requires that it also lie on the radius $\sqrt{(\ell + k)^2 + h^2}$ circle with center K. This circle intersects the previous circle in precisely the points $(m/2 \pm h, \ell)$ described in the lemma; because $h < \ell\sqrt{3}$, we have $\sqrt{\ell^2 + h^2} < \sqrt{4\ell^2} = 2\ell$, and so ℓ

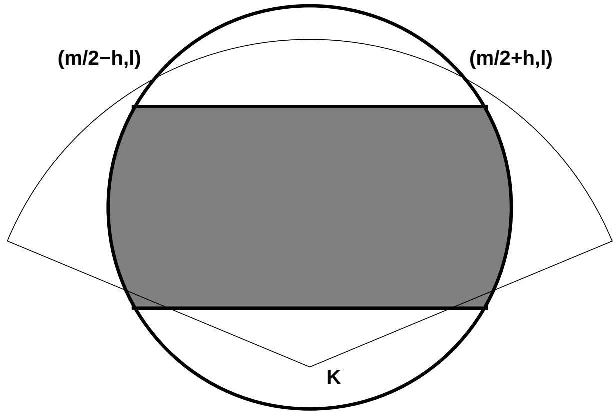

Figure 3: Applying the middle hinge constraint. The middle hinge forces the midpoint M to lie on the circle centered at K (here depicted as a wedge). The proof of Lemma 6 shows that M must either lie on the depicted circle or within the shaded region; if the intersection points of the two circles lie outside the shaded region they are the only solutions.

is at least 1/2 under our scaling assumption. In particular the K-centered circle intersects inner circle above 1/2 and thus avoids the shaded region, yielding only the two solutions $(m/2 \pm h, \ell)$.
$\blacksquare$

The circuit grid. We now use the binary hinge construction to construct a grid of horizontal and vertical wires, represented by rows and columns of rigid bodies in a two-dimensional array. The core of the construction is depicted in Figure 4: each body (which consists of the hinge endpoints and enough internal nodes and internal bracing to assure rigidity) is connected by binary hinges to its four immediate neighbors in the grid. This enforces a fixed vertical spacing ℓ between the centers of the body and its neighbors above and below, and the same fixed horizontal spacing ℓ between the centers of the body and its neighbors to the left and the right. These fixed spacings mean that the horizontal offsets propagate across rows and vertical offsets propagate across columns. At the same time, the hinges allow different bodies in the same row or column to have different vertical or horizontal offsets, respectively.

To organize the grid into wires, we anchor it to an L-shaped external frame as depicted in Figure 5. This frame fixes the horizontal and vertical offsets of every other row and column to some fixed value that we take to be 0. The remaining rows and columns are free to shift by $\pm h$ relative to these fixed rows and columns where h is the offset parameter of the hinges. These rows act as wires transmitting binary signals; we think of a column as having the value 1 when shifted up and 0 when shifted down, and of a row as having the value 1 when shifted right and 0 when shifted left.

The offset h can be chosen somewhat arbitrarily, but should be substantially less than half the disk radius (and thus the length of the hinges) to allow for double-offset hinges to be used later without exceeding the $\ell\sqrt{3}$ limit in Lemma 6. To minimize the size of the graph, we will assume that the offset h, the disk radius r, and the separation between centers of grid elements ℓ are all small constants. Reasonable values that permit the construction to work without its components colliding are $h = 1$, $r = 10$, and $\ell = 150$; with these values, all of the nodes in the graph can be

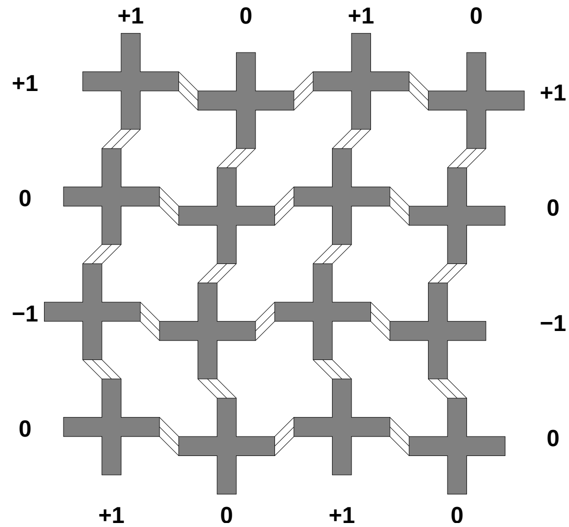

Figure 4: Two-dimensional grid of rigid bodies connected by binary hinges. Because each hinge fixes the separation between the bodies it connects but allows sideways offsets in either direction, row and column offsets propagate from one side of the grid to the other without interference.

placed at integer coordinates and no extraneous edges creep in. In the figures, both the offset and separation of each hinge is greatly exaggerated for visibility.

The fixed rows and columns divide the blocks into four categories, depending on whether the horizontal and/or vertical positions are fixed. Those for which both positions are fixed act as a fixed extension of the frame that will be used later to create circuit connections between horizontal and vertical wires. Those for which neither position is fixed act as crossing points (with no connection) between the wires.

The circuit grid by itself provides only wires but no connections between them. To build a working circuit, we need to build (a) junctions between horizontal and vertical wires, allowing arbitrary routing of signals; (b) negation; and (c) some sort of two-input gate (we will build AND). We start by describing how to build a junction between wires.

Junctions. For a junction, we want to enforce that a horizontal wire is in its rightward position (representing a 1 bit) if and only if the joined vertical wire is in its upward position (also representing a 1 bit). We do so by placing a hinge with offset $\pm h\sqrt{2}$ and separation $\ell\sqrt{2}$ at a 45° angle between the crossing body for the two wires and an adjacent fixed body, as shown in Figure 6. Note that these irrational lengths are a result of the 45° rotation; the nodes that make up the hinge are still at integer coordinates.

NOT gates. To force two adjacent horizontal wires to have opposite values, we place a double-width hinge with offset $\pm 2h$ between them at the right edge of the grid as shown in Figure 7. This permits one to be offset by $+h$ and the other by $-h$, but prevents both from being offset by the same amount. We will assume hereafter that the horizontal wires are paired off in this way with the two members of each pair representing some variable and its negation. To extract these variables for use we create a junction that sends a copy up one of the vertical wires.

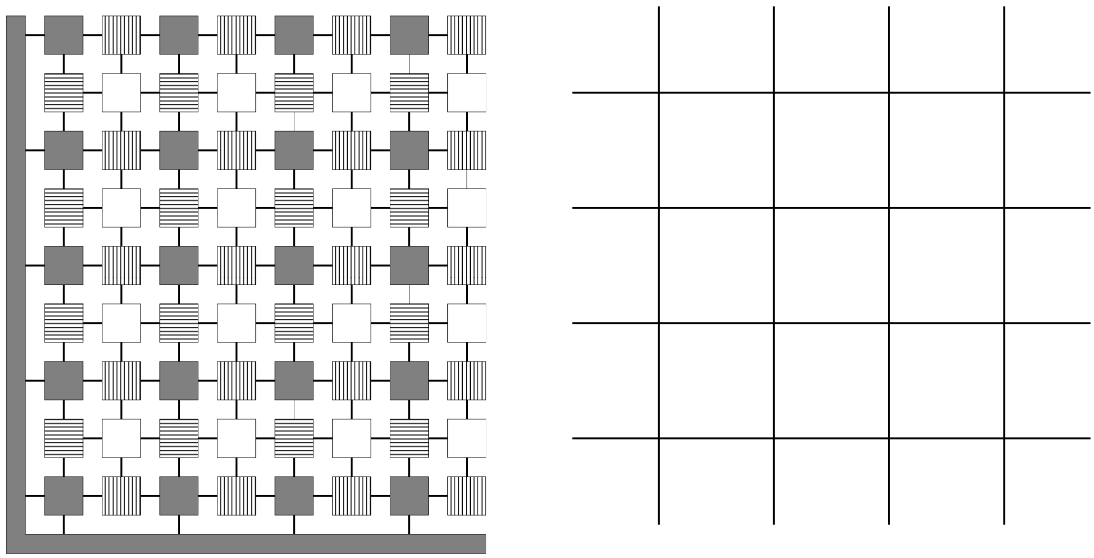

Figure 5: Two-dimensional grid of rigid bodies (squares) joined by binary hinges (dark lines) and with every other row and column anchored by binary hinges to an outer frame (L-shaped body) is used to simulate a circuit consisting of horizontal and vertical wires with no junctions (right-hand figure). The horizontally cross-hatched bodies are offset by $\pm h$ horizontally and correspond to horizontal wires. The vertically cross-hatched bodies are offset by $\pm h$ vertically and correspond to vertical wires. Bodies without cross-hatching are wire crossings that can move both horizontally and vertically; shaded bodies are fixed.

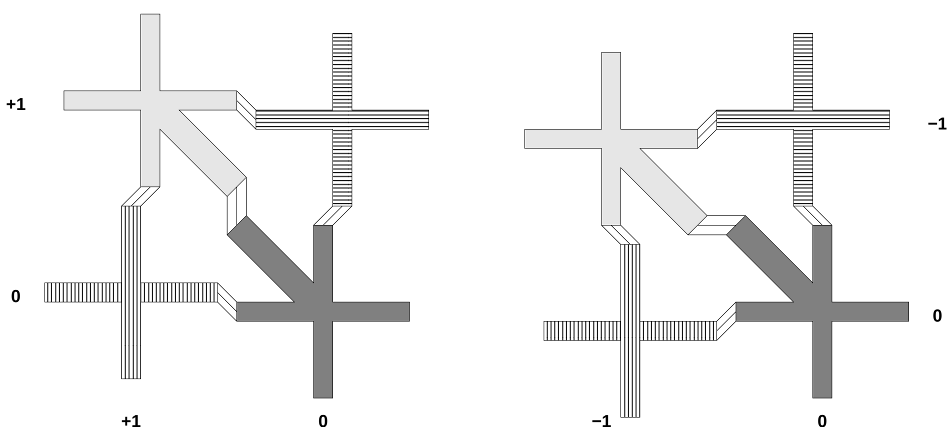

Figure 6: Junction between two wires. The hinge between the fixed body (at lower right) and the crossing point (at upper left) forces the crossing point into either its $(+1, +1)$ position or its $(-1, -1)$ position, which guarantees that the horizontal and vertical wires carry the same signal.

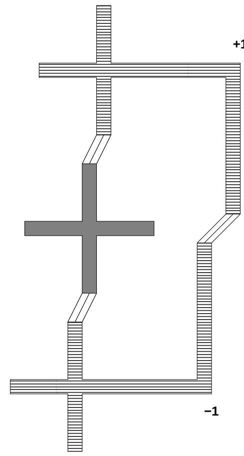

Figure 7: Negation. Double-width hinge between top and bottom rows allows only $+h$ and $-h$ or $-h$ and $+h$ as offsets.

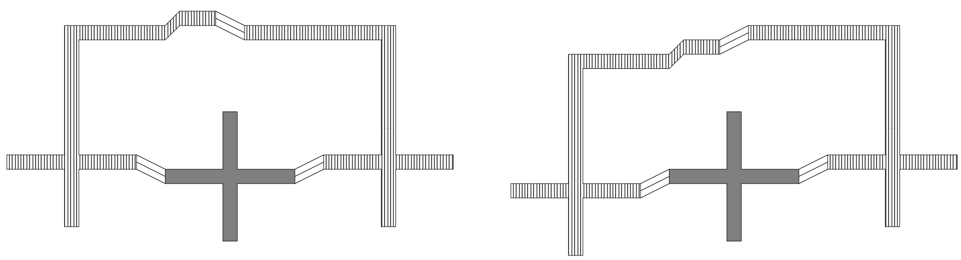

Figure 8: Implication. If left-hand column is up, right-hand column is also forced up. If left-hand column is down, right-hand column may be up or down.

Implications. At the top of the grid, we place implications. These are structures built out of single-width hinges (with offset $\pm h$) that enforce $x \to y$ by permitting only the offset combinations $(-1,-1)$, $(-1,+1)$, and $(+1,+1)$. This restriction is enforced by setting the x side of the hinge h units higher than the right side; the sides of the hinge are then at vertical offsets $(0,-h)$, $(0,h)$, and $(2h,h)$ in the three permissible configurations. The configuration where x is one and y is zero would have offsets $(2h,-h)$, which is not permitted by the hinge. By placing junctions appropriately we can create an implication between any two variables represented by horizontal wires. See Figure 8.

Implication is a rather weak primitive, but it is enough to build AND gates.

AND gates. An AND gate is a structure that forces some variable z to equal $x \wedge y$. We can build an AND gate out of four implications: $z \to x$, $z \to y$, $\overline{x} \to \overline{z}$, and $\overline{y} \to \overline{z}$, which between them allow precisely those settings of x, y, and z for which $z = x \wedge y$.

Since we have both AND gates and negation, we can build OR gates as well using DeMorgan's rule $x \vee y = \overline{\neg x \wedge \neg y}$. This is enough to build arbitrary Boolean circuits. In particular, we have a:

Reduction from CIRCUIT SATISFIABILITY. Given a Boolean circuit C, construct a graph G using the mechanisms described previously, where each wire (including input and output wires) and its negation is represented by a row of rigid bodies, and each AND gate is implemented using eight columns paired together with four implication structures. Any solution to the localization problem for the distances in this graph corresponds to an assignment of bits to the wires in C that is consistent with the behavior of the gates in C; to enforce that the output of the circuit is 1, anchor the appropriate row to the L-shaped frame to force it into the 1 position. Now there is a solution to the localization problem if and only if there is a satisfying assignment for the circuit.

We thus have a polynomial-time reduction from CIRCUIT SATISFIABILITY to localization, and Theorem 1 is proved.

4 Conclusions and open problems

We have shown that the localization problem is hard in the worst case for sparse graphs unless $\mathbf{P} = \mathbf{NP}$ or $\mathbf{RP} = \mathbf{NP}$, if certain mild forms of approximation are permitted. This worst-case result for sparse graphs stands in contrast to results that show that localization is possible for dense graphs [3] or with high probability for random geometric graphs [9]. The open questions that remain are where the boundary lies between our negative result and these positive results. In particular:

- Is there an efficient algorithm for *approximate* localization in sparse graphs, either by permitting moderate errors on distances or by permitting the algorithm to misplace some small fraction of the sensors?
- Given that the difficulty of the problem appears to be strongly affected by the density of nodes (and the resulting number of known distance pairs), what minimum density is necessary to allow localization in the worst case?
- How are these results affected by more natural assumptions about communications ranges, allowing different maximum distances between adjacent nodes or the possibility of placing small numbers of high-range beacons?

Answers to any of these questions would be an important step toward producing practical localization algorithms.

References

[1] Joe Albowicz, Alvin Chen, and Lixia Zhang. Recursive position estimation in sensor networks. In *Proceedings of the 9th International Conference on Network Protocols '01*, pages 35–41, Riverside, CA, November 2001.

[2] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE INFOCOM '00*, pages 775–784, Tel Aviv, Israel, March 2000.

[3] Pratik Biswas and Yinyu Ye. Semidefinite programming for ad hoc wireless sensor network localization. In Feng Zhao and Leonidas Guibas, editors, *Proceedings of Third International Workshop on Information Processing in Sensor Networks*, Berkeley, CA, April 2004.

[4] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[5] Srdan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad-hoc networks. In *HICSS*, 2001.

[6] Krishna Chintalapudi, Ramesh Govindan, Gaurav Sukhatme, and Amit Dhariwal. Ad-hoc localization using ranging and sectoring. In Infocom 2004 [16].

[7] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[8] L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of IEEE INFOCOM '01*, pages 1655–1633, Anchorage, AK, April 2001.

[9] T. Eren, D. Goldenberg, W. Whitley, Y.R. Yang, S. Morse, B.D.O. Anderson, and P.N. Belhumeur. Rigidity, computation, and randomization of network localization. In Infocom 2004 [16].

[10] Deborah Estrin, Ramesh Govindan, John S. Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of The Fifth International Conference on Mobile Computing and Networking (Mobicom)*, pages 263–270, Seattle, WA, November 1999.

[11] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, Apr. 1994.

[12] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSI Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[13] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes in large scale sensor networks. In *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, pages 81–95, San Diego, CA, Sep 2003.

[14] Jeffrey Hightower and Gaetano Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[15] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer-Verlag, 1997.

[16] *Proceedings of IEEE INFOCOM '04*, Hong Kong, China, April 2004.

[17] Xiang Ji. Sensor positioning in wireless ad-hoc sensor networks with multidimensional scaling. In Infocom 2004 [16].

[18] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In Mobicom 2000 [20].

[19] Seapahn Meguerdichian, Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Exposure in wireless ad-hoc sensor networks. In Mobicom 2001 [21].

[20] *Proceedings of The Sixth International Conference on Mobile Computing and Networking (Mobicom)*, Boston, MA, August 2000.

[21] *Proceedings of The Seventh International Conference on Mobile Computing and Networking (Mobicom)*, Rome, Italy, July 2001.

[22] D. Niculescu and B. Nath. Ad-hoc positioning system. In *Proceedings of IEEE Globecom 2001*, November 2001.

[23] Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS) using AOA. In *Proceedings of IEEE INFOCOM '03*, San Francisco, CA, April 2003.

[24] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In Mobicom 2000 [20], pages 32–43.

[25] C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conference*, Monterey, CA, June 2002.

[26] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In Mobicom 2001 [21], pages 166–179.

[27] J.B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.

[28] Yi Shang and Wheeler Ruml. Improved MDS-based localization. In Infocom 2004 [16].

[29] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.

[30] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. Technical Report 92.1, Olivetti Research Ltd. (ORL), 24a Trumpington Street, Cambridge CB2 1QA, 1992.

[31] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[32] M. Weiser. Some computer science problems in ubiquitous computing. *Communications of ACM*, July 1993.

[33] J. Werb and C. Lanzl. Designing a positioning system for finding things and people indoors. *IEEE Spectrum*, 35(9):71–78, October 1998.