Approximating the Obstacle Number for a Graph Drawing Efficiently*

Deniz Sarıöz†

Abstract

An obstacle representation for a (straight-line) graph drawing consists of the positions of the graph vertices together with a set of polygonal obstacles such that every line segment between a pair of non-adjacent vertices intersects some obstacle, while the vertices and edges of the drawing avoid all the obstacles. The obstacle number $obs(D)$ for a graph drawing D is the least number of obstacles in an obstacle representation for it. We present an efficient algorithm for computing the obstacle number for a given graph drawing D with approximation ratio $O(\log obs(D))$. This is achieved by showing that the V-C dimension is bounded for the family of hypergraphs of the underlying transversal problem, and using results from epsilon net theory.

1 Introduction

Let D be a straight-line drawing of a (not necessarily planar) graph G on n vertices in the Euclidean plane with no three graph vertices on the same line. We refer to the open line segment between a pair of non-adjacent graph vertices as a *non-edge* of D. We define an *obstacle representation for* D as the set of vertices of G identified with their positions in D together with a collection of polygons (not necessarily convex) called *obstacles*, such that:

1. no edge of D meets any obstacle, and
2. every non-edge of D meets at least one obstacle.

Without loss of generality, the vertices of polygonal obstacles taken together with the graph vertices are in *general position* (no three on a line) and no graph vertices are inside any obstacle. Notice that the positions of the graph vertices and the obstacles in such an obstacle representation are sufficient to determine the abstract structure of the graph, based on whether or not a pair of graph vertices can "see" each other. Graphs obtained in this manner are called *visibility graphs*, and they are extensively studied and used in computational geometry, robot motion planning, wireless sensor networks, and mobile ad-hoc networks; see [5, 23, 18, 17, 11, 9].

We define the *obstacle number for* D as the least number of obstacles over all obstacle representations for D. We denote this parameter by $obs(D)$.

Our main contribution is a polynomial-time approximation algorithm that, given a graph drawing D, computes an obstacle representation for D with $O(obs(D) \log obs(D))$ obstacles.

Related notions of (an) obstacle representation of a graph and (the) obstacle number of a graph were first defined by Alpert, Koch, and Laison [1]. An obstacle representation of a graph G is equivalent to an obstacle representation for some (straight-line) drawing D of it. The obstacle number of a graph G is the minimum value of $obs(D)$ attained over all drawings D of G.

The obstacle numbers of certain graphs have been determined exactly; upper bounds have been established for some graph families, and proofs of unboundedness have been offered for others [1, 21, 16, 20, 10]. However, the question of devising a computational procedure to determine or approximate the obstacle number of a graph has to our knowledge not yet been addressed, even in part. The results presented here can be considered a first step in that direction.

In Section 2, we will explicate the connection to hypergraphs defined by intersection, and present some background about hypergraph transversals including notions of V-C dimension and epsilon net. In Section 3, we prove our main result having to do with bounding the V-C dimensions of various hypergraphs. In Section 4 we present a concrete algorithm and discuss its efficacy.

2 Preliminaries

2.1 Intersection Hypergraphs and their Transversals

A *hypergraph* is a pair $(X, \mathcal{F})$ in which X is a set of ground elements, and $\mathcal{F}$ is a collection of subsets of X. We introduce the following notation and terminology for intersection hypergraphs. Let X and Y be collections of sets. For each $y \in Y$, let $N(y) = \{x \in X \mid x \cap y \neq \emptyset\}$, and say that y *generates* $N(y)$. Let $\mathcal{F} = \{N(y) \mid y \in Y\}$. Then $(X, \mathcal{F})$ is an *intersection hypergraph*, which we shall denote by $H(X, Y)$ whenever convenient.[1] Similarly, for each $x \in X$, let

*Research supported by NSA grant 47149-0001 and PSC-CUNY grant 63427-0041.

†Ph. D. Program in Computer Science, The Graduate Center of The City University of New York (CUNY), sarioz@acm.org

[1] In many well-studied geometric hypergraphs $H(X, Y)$, each set in X is a singleton. The intersection of a member of X with a member of Y thus corresponds to the inclusion of the former in

$N(x) = \{y \in Y \mid x \cap y \neq \emptyset\}$, and say that x *generates* $N(x)$. Let $\mathcal{F}' = \{N(x) \mid x \in X\}$. The hypergraph $(Y, \mathcal{F}')$, which we shall denote by $H(Y, X)$ when convenient, is known as the *dual* of the hypergraph $H(X, Y)$.

A *transversal* of an intersection hypergraph $H(X, Y)$ is a subset $T \subseteq X$ such that every member of Y—that meets some member of X—meets some member of T. Let τ denote the minimum cardinality of a transversal of $H(X, Y)$. The (optimization version of) the hypergraph transversal problem is to compute τ exactly, and this has an equivalent formulation as the *set cover* problem. The decision version of the hypergraph transversal problem is NP-complete; indeed, the restriction to the case in which every member of Y meets exactly two members of X corresponds to a canonical NP-complete problem, "Vertex Cover."

2.2 Computing the Obstacle Number for a Graph Drawing as a Transversal Problem

For a given graph drawing D, we refer to a connected component of the complement of D as a *face* of the drawing. To rephrase an observation in [1] in our context, in an obstacle representation for D with cardinality $obs(D)$, there can be at most one obstacle per face, for otherwise obstacles in the same face could be merged, contradicting the minimality of $obs(D)$. Hence for any given graph drawing, each polygonal obstacle to be included in a minimal obstacle representation can be considered to be a face of the drawing. If need be, one can compute for each face a representative simple polygon that lies inside the face and meets every non-edge that the face meets. (This is not always a simple matter of perturbing the boundary of a face to lie inside the face, since a face may have holes and so its boundary may be a disconnected set.)

Since an n-vertex graph has less than n^2 edges (with $\Omega(n^2)$ edges attainable), its drawing must have less than n^4 faces (with $\Omega(n^4)$ faces attainable). Computing $obs(D)$ is therefore a matter of computing a transversal for the finite intersection hypergraph $H(X, Y)$ where X is the face set of D and Y is the non-edge set of D. Observe that $|X| < n^4$ and $Y < n^2$, with $|X| = \Omega(n^4)$ attainable simultaneously with $|Y| = \Omega(n^2)$. Using a representation of D with integer coordinates represented as signed integers, an incidence matrix representation of $H(X, Y)$ with fewer than n^8 bits (and possibly $\Omega(n^8)$) can be computed using standard techniques [5] in time polynomial in the number of input bits, and in time poly(n) in the RAM model with unit-cost arithmetic operations. It is important to make this distinction, since coordinates may need to be represented using exponentially many bits in n, see [12]; or more, as discussed in Section 5.

It is well-known that the greedy algorithm for the hypergraph transversal problem, which iteratively adds to an initially empty set T a member $x \in X$ that meets the largest number of sets $y \in Y$ that do not already meet some $x' \in T$, provides a $O(\log |Y|)$-factor approximation [25]. Thus we have a natural $O(\log n)$-factor approximation algorithm for our problem of computing the obstacle number for a given drawing.

2.3 Improving the Approximation Ratio

How about doing better? Not only is the approximation ratio tight for this greedy algorithm, but the general hypergraph transversal problem is known to be $o(\log |Y|)$-inapproximable [25]. But it is also well-known [25, 19] that if every member of X meets at most Δ members of Y, then the greedy algorithm attains an approximation ratio of $O(\log \Delta)$. Unfortunately, this does not make our task easier, since it is seen that a face could meet $\Omega(n^2)$ non-edges by considering any drawing of the null graph on n vertices. Nonetheless, many families of hypergraphs arising in geometric settings lend themselves to algorithms with approximation ratios that do not depend on either $|X|$ or $|Y|$ using the following ideas.

In the context of an intersection hypergraph $H(X, Y)$, a set $S \subseteq X$ is said to be *shattered* if for every $A \subseteq S$ there is some $y \in Y$ such that $S \cap N(y) = A$. The size of a largest shattered set is called the *V-C dimension* of $H(X, Y)$, after Vapnik and Chervonenkis who first defined it in [24]. For some hypergraphs in which X and Y are both infinite, the V-C dimension is undefined and said to be infinite. Furthermore, for a family of hypergraphs of the form $H(X, Y)$, even if each hypergraph has finite V-C dimension, there may exist no absolute constant upper bound for the V-C dimension. If there an integer d such that every hypergraph in that family has V-C dimension at most d, we say that the family has bounded V-C dimension.

Let $w : 2^X \to [0, 1]$ be an additive weight function with $w(X) = 1$. For a given $\epsilon > 0$, an ϵ-net (with respect to w) is a set $S \subseteq X$ that is a transversal of $H(X, Y_\epsilon)$, where $Y_\epsilon \subseteq Y$ consists exactly of those members of Y each of which generates a subset of X with weight at least ϵ. Haussler and Welzl have shown [13] that if the V-C dimension of $H(X, Y)$ is some integer d, then for every $\epsilon > 0$ there is an ϵ-net of size at most $cd\epsilon^{-1} \ln \epsilon^{-1}$, where c is a small constant. This is remarkable because the size of an ϵ-net bears no relation to the sizes of X or Y. See also [19, 15].

Based on this observation, various—deterministic as well as randomized—efficient algorithms have been pre-

the latter. The members of Y are commonly referred to as *ranges*, especially in hypergraphs in which Y is a natural feature of the geometric space that the "points" of X belong to, e.g., the set of all half-spaces, all balls, or all axis-parallel boxes. We eschew the use of the term *range* since this is not the case for problems we are interested in, and also because our hypergraphs are defined by intersection not limited to inclusion: A set in X can meet two disjoint sets in Y and vice versa.

sented [3, 7, 8, 6] to compute a transversal of size within a tiny constant factor of $d\tau \ln \tau$. In Section 4, we state and analyze a specific algorithm for computing the obstacle number for a graph drawing. For now, we suffice it to say that bounding the V-C dimension for the corresponding hypergraph implies an efficient algorithm with approximation ratio independent of $|X|$ or $|Y|$ (and hence n).

Before we proceed, we state an important fact that we make immediate use of. It is well-known [15] that if the V-C dimension of $H(Y, X)$ is d, then the V-C dimension of $H(X, Y)$ is at most 2^{d+1}. The V-C dimension of a family of hypergraphs is therefore bounded if and only if the V-C dimension of the family of dual hypergraphs is bounded.

In the next section, we show that the V-C dimension is bounded for the family of hypergraphs of the form $H(Y, X)$ where Y is the set of non-edges of a graph drawing and X is the set of faces of that drawing. This implies that the V-C dimension of $H(X, Y)$ (the hypergraph for the transversal problem at hand) is also bounded.

3 Main Results

Theorem 1 *The V-C dimension is bounded for the family of hypergraphs of the form $H(Y, X)$ in which Y is the set of non-edges in a straight-line drawing D of a graph, and X is the set of faces of D.*

We can replace each face $x \in X$ by a simple path x' inside x that meets every non-edge that x meets and does not meet any non-edges that x does not. This substitution will not alter the hypergraph structure, and the resulting paths x' will be disjoint from one another. Hence Theorem 1 is implied by the following result.

Theorem 2 *The V-C dimension is bounded for the family of hypergraphs of the form $H(Y, X)$ in which Y is a set of line segments (with every pair meeting at a single point or not at all) and X is a set of simple paths disjoint from one another.*

Proof. Assume for contradiction that for every m, there is a hypergraph $H(Y, X)$ such that Y is a set of m line segments (with every pair meeting at a single point or not at all), X is a set of paths disjoint from one another, and Y is shattered.

Given m, and a pair (Y, X) such that $|Y| = m$ and X shatters Y, let $X_3 \subseteq X$ be a minimal set of paths that generate all the $\binom{m}{3}$ *triples* in Y. That is, every path in X_3 meets exactly three segments in Y, and for every three segments $i, j, k \in Y$ exactly one path $\pi_{ijk} \in X_3$ meets all three. To keep the following argument simple, without loss of generality, no $\pi \in X_3$ meets any intersection points among the segments, of which there are at most $\binom{m}{2} = O(m^2)$. If there were such paths in X_3, we could remove them from X_3 and still be left with at least $\binom{m}{3} - \binom{m}{2} = \Omega(m^3)$ paths. We will now charge

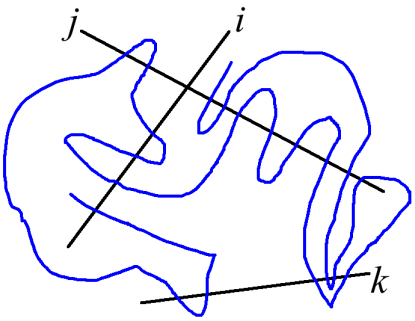

Figure 1: Example of an original path π_{ijk} meeting segments i, j, and k.

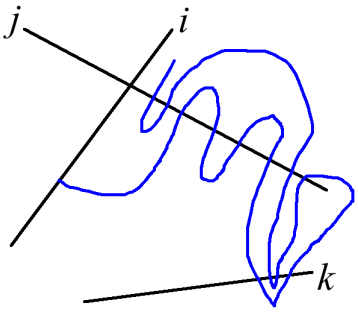

Figure 2: The interim path (after erasing from tail).

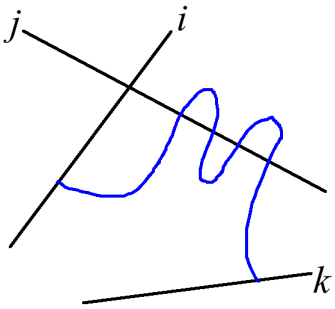

Figure 3: The final path (after erasing from head too), which will be charged to segment j.

each path $\pi \in X_3$ to a line segment in Y, intuitively, "the middle segment" that it meets. Nothing prevents such a path from going back and forth between three segments, so we need to define this more carefully. For a given path $\pi = \pi_{ijk}$ that meets segments $i, j, k \in Y$

(see Fig. 1), arbitrarily label one end of the path as the tail and the other as the head. Starting from the tail, erase π as long as it still meets all three segments, and stop erasing when erasing any longer would cause the remaining path to intersect fewer segments. The tail is now on one of the three segments, without loss of generality, i (see Fig. 2). Note that the path does not intersect i anywhere else but the tail. Now start erasing π from the head in a similar fashion, and stop erasing when erasing any more would cause the remaining path to intersect fewer than three segments. Again, the head must be on some segment when we stop, but it could not be on i by the above observation (see Fig. 3). Without loss of generality, the head is on the segment k. Now notice that the path does not meet k anywhere else but the head. Without changing the shatter property, let us replace π_{ijk} with this shorter version of its former self: starting at i, meeting j but no other segment one or more times, before it ends at k. We charge π_{ijk} to j.

Let $\hat{s}$ be a segment that accumulated the greatest charge at the end of this process. Since at least $\Omega(m^3)$ paths were charged to at most m segments, $\hat{s}$ was charged by at least $\Omega(m^2)$ paths. Let X' denote the set of paths in X_3 that were charged to $\hat{s}$.

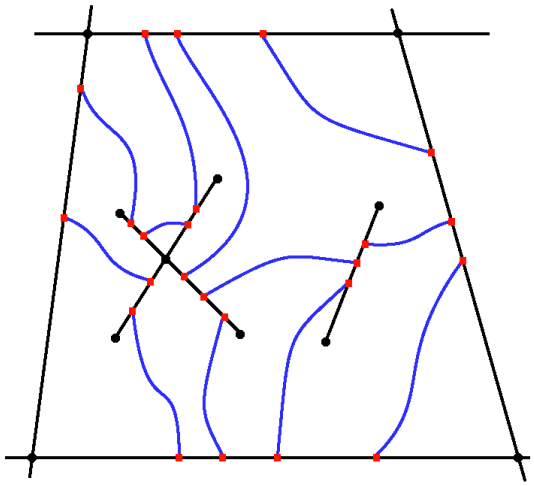

Figure 4: Example of a cell of $\mathcal{A}$ and the segments from the original X' that are inside the cell.

Let $\mathcal{A}$ denote the arrangement of the line segments in $Y \setminus \{\hat{s}\}$. A segment endpoint or an intersection point of two segments is called a *vertex of the arrangement*. An open interval on a segment of the arrangement between two vertices of the arrangement that contains no vertex of the arrangement is called an *edge of the arrangement*. A connected component in the complement of the union of the segments is called a *cell of the arrangement*.

Note that every path in X' starts at an edge of the arrangement $\mathcal{A}$, ends at another edge of $\mathcal{A}$, and its interior is fully contained in a cell of $\mathcal{A}$ (see Fig. 4). Let $G(X')$ be the graph with the endpoints of the paths as the vertex set, and the interiors of the paths as edges. Since the paths are disjoint, clearly, $G(X')$ is planar. Now for each edge of the arrangement, merge the path endpoints on that edge at the midpoint of the edge while making sure that the paths remain interior-disjoint (see Fig. 5). Recall that by Euler's formula a planar graph on n vertices has at most $3n - 6$ edges. It is not clear that we have a contradiction yet, since $\mathcal{A}$ may have up to m^2 edges (attained when every pair among the m segments cross). Hence it seems that $G(X')$ may have $\Theta(m^2)$ vertices, so it is plausible that $G(X')$ has $\Theta(m^2)$ edges.

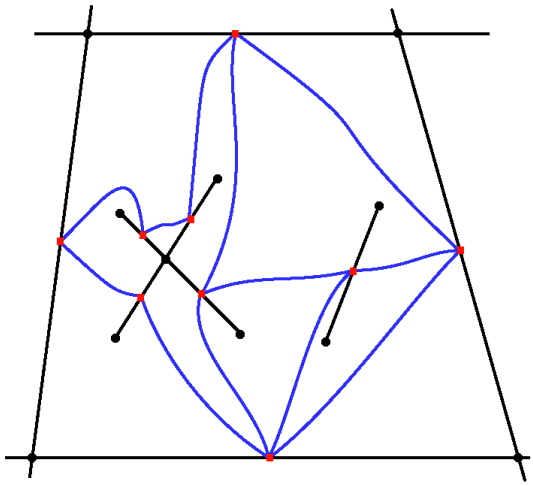

Figure 5: The same cell after merging path endpoints.

However, each edge of $G(X')$ must be contained in a single cell that meets $\hat{s}$. Might this mean that $G(X')$ has $o(m^2)$ vertices? We know that every vertex of $G(X')$ corresponds to a distinct edge of $\mathcal{A}$ in a cell of $\mathcal{A}$ that meets $\hat{s}$. Hence, the complexity of the zone[2] of $\hat{s}$ is an upper bound on $|V(G(X'))|$. We present two lemmas regarding the complexity of the zone of a line segment in an arrangement of line segments, each of which is sufficient by itself.

Lemma 3 *Let $\mathcal{A}$ be an arrangement of n line segments, and let s be another line segment. The zone of s has complexity $O(n^{4/3})$.*

Proof. Even if s meets all n segments of $\mathcal{A}$, its zone will consist of at most n cells including the unbounded cell. The bound then follows from the main result in [2]: That the maximum number of edges bounding m cells in an arrangement of n line segments in the plane is $O(m^{2/3}n^{2/3} + n\alpha(n) + n \log m)$. □

[2]The *complexity* of a cell of an arrangement is the number of edges of the arrangement that are incident to it. The *zone* of a segment is the set of cells that it meets, and the complexity of the zone of a segment is the number of edges incident to all the cells that it meets.

Lemma 4 (B. Aronov, personal communication) *Let $\mathcal{A}$ be an arrangement of n line segments, and let s be another line segment. The zone of s has complexity $O(n\alpha(n))$ where α denotes the very slow growing inverse of Ackerman's function.*

Proof. Let the shape s' be obtained by enlarging s (e.g. taking the Minkowski sum of s with a small enough disk) such that s' meets no vertex of $\mathcal{A}$ that s does not. Obtain a new arrangement $\mathcal{A}'$ of line segments by erasing s' from $\mathcal{A}$. Doing this will possibly disconnect some original line segments that define $\mathcal{A}$ into two, ending up with an arrangement $\mathcal{A}'$ of at most $2n$ line segments. Every point of s' is in the same cell of this new arrangement. Every cell in an arrangement of m line segments has complexity $O(m\alpha(m))$ [22]. Hence, every cell of $\mathcal{A}'$ has complexity at most $O(2n\alpha(2n))$, i.e., $O(n\alpha(n))$, including the cell that s is in. Since every edge bordering a cell of $\mathcal{A}$ that s meets corresponds to one or two edges bordering the cell of $\mathcal{A}'$ that s is in, the complexity of the zone of s in $\mathcal{A}$ is at most $O(n\alpha(n))$. □

Either lemma implies that the zone of $\hat{s}$ has complexity $o(m^2)$, hence the number of vertices of the planar graph $G(X')$ is $o(m^2)$. But since $G(X')$ has $\Omega(m^2)$ edges, we have a contradiction due to Euler's Formula. Therefore, the V-C dimension is bounded for the family of hypergraphs $H(Y, X)$ in which Y is a set of line segments and X is a set of paths disjoint from one another. □

4 Algorithm

We continue using the terminology of having been given a drawing D of an n-vertex graph, with X as the set of faces in D and Y as the set of non-edges in D. For the rest of this section, we assume that D has been processed into the incidence matrix of the hypergraph $H(X, Y)$ with $O(n^8)$ entries in memory, as discussed in Section 2.

First, we present a randomized algorithm modeled around an algorithm presented by Efrat and Har-Peled [6] and relies on the analysis in [4], for a totally unrelated family of hypergraphs, in which an upper bound on the V-C dimension was in fact known. Here we use dovetailing to not only vary k (essentially a guess for τ) but also vary d (essentially a guess for the V-C dimension).

According to the analysis therein, for a given graph drawing D that admits an obstacle representation with τ obstacles, our algorithm will return a traversal of size at most $O(\tau \log \tau)$, and it is clear that its running time is polynomial in n even in the worst case.

In the following C-style pseudocode, $a = b$ means a takes on the value of b, the macro $sampleSize(d, k)$ expands to $\lceil 2dk \lg k \rceil$, and the macro $numRounds(k, |X|)$ expands to $\lceil 8k \lg |X| \rceil$.

ComputeObstacleRepresentation(set_of_faces X, set_of_nonedges Y) {
  bestSoln = X; bestSize = $|X|$;
  for(deekay = 2; 2 deekay < bestSize;
                deekay = 2 deekay) {
    for($k = 2; k \leq$ deekay; $k = 2k$) {
      $d =$ deekay / k;
      if ($sampleSize(d, k) \geq$ bestSize) break;
      Assign weight 1 to each face in X;
      for ($i = 1; i \leq numRounds(k, |X|); i = i + 1$) {

- Pick randomly a set S of $sampleSize(d, k)$ obstacles, choosing each obstacle randomly and independently from the face set X according to their weights.
- Check if the obstacles in S together meet all of the non-edges in Y; if so, set bestSoln = S, set bestSize = $|S|$, and break the innermost loop.
- Else, find a non-edge y that does not meet any obstacle in S, and let $N(y)$ be the set of faces in X that the non-edge y meets.
- Compute ω, the sum of weights of faces in $N(y)$. If $2k\omega \leq$ the sum of weights of all faces in X, double the weight of every face in $N(y)$.

      } // end for i
    } // end for k
  } // end for deekay
  return bestSoln;
} // end ComputeObstacleRepresentation

5 Remarks

The problem of computing the obstacle number for a graph drawing D exactly is in NP, since it can be transformed into a hypergraph transversal problem in polynomial time. Hence, a naive deterministic algorithm can compute $obs(D)$ in time $2^{O(n)}$, or if we allow ourselves to be output-sensitive, in time merely $n^{O(obs(D))}$, by trying every k-face combination for every value of k from 0 up to $obs(D)$. Since the submission of the accepted version of this paper, an NP-completeness proof has been scheduled to appear on arXiv [14].

What about the original problem of determining the obstacle number of a given abstract graph on n vertices? If all drawings of a graph could be enumerated up to the incidence matrix of faces versus non-edges, then by using our approximation algorithm in the "inner loop," we could obtain a $O(\log OPT)$-approximation to the original problem. While this may be viable for small instances (perhaps in conjunction with a distributed approach), we conjecture that this problem lies outside of NP and believe it to be intractable in a centralized model of computation. Our rationale follows.

For some simple order types of n-point configurations on the plane, a coordinate representation on an integer lattice needs exponentially many bits in n in order to allow the order type to be inferred [12]. Further, we know that a particular labeled graph has two drawings with different obstacle numbers but vertex sets of the same simple labeled order type. (The dual labeled order type of the $\binom{n}{2}$ connecting lines of n vertices in a drawing appears to be sufficient to determine the obstacle number for the drawing.) Hence, coordinate representations of some drawings for the present purpose seem to require at least exponential storage in n. Some drawing-based certificates will in turn have sizes super-polynomial in the number of bits that represent the abstract graph. It may be tempting to think that a certificate could instead be based on the poly(n) sized incidence matrix of faces versus non-edges, but it seems unlikely that one can decide in polynomial time whether or not the given graph has *some* drawing corresponding to a given incidence matrix.

Acknowledgments

The author is indebted to Boris Aronov, Alon Efrat, Nabil Mustafa, and János Pach for fruitful discussions and feedback, and the anonymous CCCG reviewers for keen comments. Alon Efrat's encouragement in the early stages of this work and pointers to recent publications were especially helpful. The author assumes full responsibility for all errors and omissions.

References

[1] H. Alpert, C. Koch, and J. Laison. Obstacle numbers of graphs. *Discrete Comput. Geom.*, 44:223–244, 2010.

[2] B. Aronov, H. Edelsbrunner, L. J. Guibas, and M. Sharir. The number of edges of many faces in a line segment arrangement. *Combinatorica*, 12:261–274, 1992.

[3] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14(4):463–479, 1995.

[4] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. Algorithms and Data Structures, 3rd Workshop, WADS '93, Montréal, Canada, August 11–13, 1993*, LNCS 709, pages 246–252. Springer, 1993.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry. Algorithms and Applications (2nd ed.)*. Springer-Verlag, 2000.

[6] A. Efrat and S. Har-Peled. Guarding galleries and terrains. *Info. Proc. Letters*, 100:238–245, December 2006.

[7] A. Efrat, F. Hoffmann, C. Knauer, K. Kriegel, G. Rote, and C. Wenk. Covering with ellipses. *Algorithmica*, 38:145–160, 2003.

[8] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Info. Proc. Letters*, 95(2):358–362, 2005.

[9] A. Frieze, J. Kleinberg, R. Ravi, and W. Debany. Line-of-sight networks. *Combinatorics, Probability and Computing*, 18:145–163, 2009.

[10] R. Fulek, N. Saeedi, and D. Sarıöz. Convex obstacle numbers of outerplanar graphs and bipartite permutation graphs, 2011. arXiv:1104.4656v2 [cs.DM].

[11] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, Cambridge, 2007.

[12] J. E. Goodman, R. Pollack, and B. Sturmfels. Coordinate representation of order types requires exponential storage. In *Proc. 21st annual ACM Symposium on Theory of Computing*, STOC '89, pages 405–410, New York, NY, USA, 1989. ACM.

[13] D. Haussler and E. Welzl. ϵ-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[14] M. P. Johnson and D. Sarıöz. Computing the obstacle number of a plane graph, July 2011. http://arXiv.org.

[15] J. Matoušek. *Lectures on Discrete Geometry*. Graduate Texts in Mathematics. Springer, 2002.

[16] P. Mukkamala, J. Pach, and D. Sarıöz. Graphs with large obstacle numbers. In *Graph Theoretic Concepts in Computer Science*, LNCS 6410, pages 292–303. Springer, 2010.

[17] J. O'Rourke. Visibility. In *Handbook of Discrete and Computational Geometry*, CRC Press Ser. Discrete Math. Appl., pages 467–479. CRC, 1997.

[18] J. O'Rourke. Open problems in the combinatorics of visibility and illumination. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemp. Math.*, pages 237–243. AMS, Providence, RI, 1999.

[19] J. Pach and P. K. Agarwal. *Combinatorial Geometry*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., 1995.

[20] J. Pach and D. Sarıöz. Small $(2, s)$-colorable graphs without 1-obstacle representations, 2010. arXiv:1012.5907v2 [cs.DM].

[21] J. Pach and D. Sarıöz. On the structure of graphs with low obstacle number. *Graphs and Combinatorics*, 27:465–473, 2011.

[22] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete Comput. Geom.*, 3:123–136, January 1988.

[23] J. Urrutia. Art gallery and illumination problems. In *Handbook of Computational Geometry*, pages 973–1027. North-Holland, Amsterdam, 2000.

[24] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies to their probabilities. *Theory Probab. Appl.*, 16(2):264–280, 1971.

[25] V. V. Vazirani. *Approximation Algorithms*. Springer, 2004.