Recognition and Drawing of Stick Graphs

Felice De Luca[1], Md Iqbal Hossain[1], Stephen Kobourov[1], Anna Lubiw[2], and Debajyoti Mondal[3]

[1] Department of Computer Science, University of Arizona, USA
{felicedeluca,hossain}@email.arizona.edu, kobourov@cs.arizona.edu
[2] Cheriton School of Computer Science, University of Waterloo, Canada
alubiw@uwaterloo.ca
[3] Department of Computer Science, University of Saskatchewan, Canada
dmondal@cs.usask.ca

Abstract. A *Stick graph* is an intersection graph of axis-aligned segments such that the left end-points of the horizontal segments and the bottom end-points of the vertical segments lie on a "ground line," a line with slope -1. It is an open question to decide in polynomial time whether a given bipartite graph G with bipartition $A \cup B$ has a Stick representation where the vertices in A and B correspond to horizontal and vertical segments, respectively. We prove that G has a Stick representation if and only if there are orderings of A and B such that G's bipartite adjacency matrix with rows A and columns B excludes three small 'forbidden' submatrices. This is similar to characterizations for other classes of bipartite intersection graphs.

We present an algorithm to test whether given orderings of A and B permit a Stick representation respecting those orderings, and to find such a representation if it exists. The algorithm runs in time linear in the size of the adjacency matrix. For the case when only the ordering of A is given, we present an $O(|A|^3|B|^3)$-time algorithm. When neither ordering is given, we present some partial results about graphs that are, or are not, Stick representable, and we give an exact $O(1.3^{n^2})$-time decision algorithm, where $n = |A| + |B|$.

1 Introduction

Let $\mathcal{O}$ be a set of geometric objects in the Euclidean plane. The *intersection graph* of $\mathcal{O}$ is a graph where each vertex of G corresponds to a distinct object in $\mathcal{O}$, and two vertices are adjacent in G if and only if the corresponding objects intersect. Recognition of intersection graphs that arise from different types of geometric objects such as segments, rectangles, discs, intervals, etc., is a classic problem in combinatorial geometry. Some of these classes, such as interval graphs [2], can be recognized in polynomial-time, whereas many others are NP-hard [4, 26, 28]. There are many beautiful results that characterize intersection classes in terms of a vertex ordering without certain forbidden patterns, and recently, Hell *et al.* [19] unified many previous results by giving a general polynomial time recognition algorithm for all cases of small forbidden patterns.

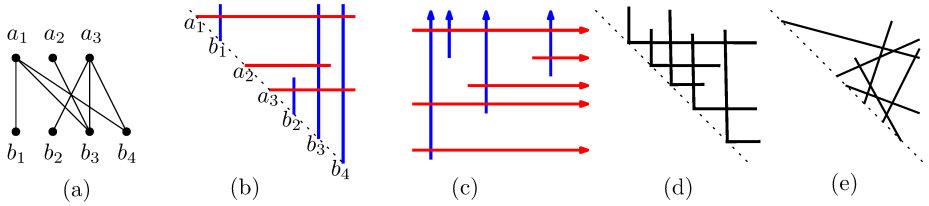


Fig. 1. (a) A bipartite graph $G = (A \cup B, E)$. (b) A Stick representation of G. (c)–(e) Illustration for different types of intersection representations. (c) A *2DOR* representation. (d) A *Hook* representation. (e) A grounded segment representation.

In this paper we study a class of bipartite intersection graphs called *Stick graphs*. A *Stick graph* is an intersection graph of axis-aligned segments with the property that the left end-points of horizontal segments and the bottom end-points of vertical segments all lie on a *ground line*, which we take, without loss of generality, to be a line of slope -1. See Fig. 1(a)–(b). It is an open problem to recognize Stick graphs in polynomial time [8].

Stick graphs lie between two well-studied classes of bipartite intersection graphs. First of all, they are a subset of the *grid intersection graphs* (GIG) [18]—intersection graphs of horizontal and vertical segments in the plane—which are NP-complete to recognize [26]. When all the horizontal segments extend rightward to infinity and the vertical segments extend upward to infinity, we obtain the subclass of *2-directional orthogonal ray* (2DOR) graphs (e.g., see Fig. 1(c)), which can be recognized in polynomial time [32]. It is easy to show that every 2DOR graph is a Stick graph—truncate each ray at a ground line placed above and to the right of every intersection point (and then flip the picture upside-down). Thus the class of Stick graphs lies strictly between these two classes.

What the two classes (GIG and 2DOR) have in common is a nice characterization in terms of vertex orderings. A bipartite graph G with vertex bipartition $A \cup B$ can be represented as a *bipartite adjacency matrix*, $M(G)$ with rows and columns corresponding to A and B, respectively, and a 1 in row i, column j, if (i, j) is an edge. Both GIG graphs and 2DOR graphs can be characterized as graphs G for which $M(G)$ has a row and column ordering without certain 'forbidden' submatrices. (Details below.) Many other bipartite intersection graphs can be similarly characterized in terms of forbidden submatrices, see [25].

One of our main results is a similar characterization of Stick graphs. Specifically, we will prove that a bipartite graph G with vertex bipartition $A \cup B$ has a Stick representation with vertices of A corresponding to horizontal segments and vertices of B corresponding to the vertical segments if and only if there is an ordering of A and an ordering of B such that $M(G)$ has no submatrix of the following form, where $*$ stands for either 0 or 1:

$$\begin{bmatrix} * & 1 & * \\ * & 0 & 1 \\ 1 & * & * \end{bmatrix} \quad \begin{bmatrix} 1 & * \\ 0 & 1 \\ 1 & * \end{bmatrix} \quad \begin{bmatrix} * & 1 & * \\ 1 & 0 & 1 \end{bmatrix}$$

Although this characterization does not (yet) give us a polynomial time algorithm to recognize Stick graphs, it allows us to make some progress. Given a bipartite graph G with vertex bipartition $A \cup B$, we want to know if G has a Stick representation with A and B corresponding to horizontal and vertical segments, respectively. It is easy to show that a solution to this problem is completely determined by a total ordering σ of the vertices of G corresponding to the order (from left to right) in which the segments touch the ground line. A natural way to tackle the recognition of Stick graphs is as a hierarchy of problems, each (possibly) more difficult than the next:

(i) **Fixed As and Bs**: In this case an ordering, σ_a, of the vertices in A and an ordering, σ_B, of the vertices in B are given, and the output ordering σ must respect these given orderings. Because of our forbidden submatrix characterization, this problem can be solved in polynomial time.
(ii) **Fixed As**: In this case only the ordering σ_A is given.
(iii) **General Stick graphs**: In this case, neither σ_A nor σ_B is given, i.e., there is no restriction on the ordering of the vertices.

Our Results: We give an algorithm with run-time $O(|A||B|)$ for problem (i). This is faster than naively looking for the forbidden submatrices. (And in fact, we use our algorithm to prove the forbidden submatrix characterization). Furthermore, the algorithm will find a Stick representation when one exists.

We give an algorithm for problem (ii) with run time $O(|A|^3|B|^3)$ that uses the forbidden submatrix characterization and reduces the problem to 2-Satisfiability. For problem (iii), recognizing Stick graphs, we give some conditions that ensure a graph is a Stick graph, and some conditions that ensure a graph is not a Stick graph. We also give an exact recognition algorithm using 3-SAT with run-time $O(1.3^{n^2})$. This is better than a naive exponential time algorithm, and may be reasonable in practice since SAT solvers can often handle instances with a million variables [23].

Related Work: We now review the research related to the recognition of intersection graphs, in particular those that are bipartite.

Interval graphs, i.e., intersection graph of horizontal intervals on the real line, can be recognized in linear time [2, 13]. Bipartite interval graphs with a fixed bipartition are known as *interval bigraphs* (IBG) [14,29], and can be recognized in polynomial time [29]. In contrast to the interval graphs, no linear-time recognition algorithm is known for IBG.

Many bipartite graph classes have been characterized in terms of forbidden submatrices of the graph's bipartite adjacency matrix, and a rich body of research examines when the rows and columns of a matrix can be permuted to avoid forbidden submatrices [25]. For example, a graph G is chordal bipartite if and only if $M(G)$ can be permuted to avoid the matrix γ_1 in Fig. 2 [25], which led to a polynomial-time algorithm [27]. G is a bipartite permutation graph if and only if $M(G)$ can be permuted to avoid γ_1, γ_2, and γ_3 [11].

A graph is a *two-directional orthogonal ray* (2DOR) graph if it admits an intersection representation of upward and rightward rays [32, 33]. A graph is

$$\gamma_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \gamma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \gamma_3 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \gamma_4 = \begin{bmatrix} 1 & 0 & 1 \\ * & 1 & * \end{bmatrix} \quad \gamma_5 = \begin{bmatrix} * & 1 & * \\ 1 & 0 & 1 \\ * & 1 & * \end{bmatrix}$$

Fig. 2. Forbidden submatrices, where $*$ stands for either 0 or 1.

a 2DOR graph if and only if its incidence matrix admits a permutation of its rows and columns that avoids γ_1 and γ_2 [32]. There is a linear-time algorithm to recognize 2DOR graphs [12,32]. If there are 3 or 4 allowed directions for the rays, then the graphs are called 3DOR or 4DOR graphs, respectively. Felsner *et al.* [15] showed that if the direction (right, left, up, or down) for each vertex is given, then the existence of a 4DOR representation respecting the given directions can be decided in polynomial time. If the horizontal elements are segments and the vertical elements are rays, then the corresponding intersection graphs are called *SegRay* graphs [7–9, 24]. A graph G is a SegRay graph if and only if $M(G)$ can be permuted to avoid γ_4 [10].

The time-complexity questions for 3DOR, 4DOR and SegRay are all open.

The class of *segment graphs* contains the graphs that can be represented as intersections of segments (with arbitrary slopes and intersection angles). Every planar graph has a segment intersection representation [6]. Restricting to axis-aligned segments gives rise to *grid intersection graphs* (GIG) [26]. A bipartite graph is a GIG graph if and only if its incidence matrix admits a permutation of its rows and columns that avoids γ_5 [18]. If all the segments must have the same length, then the graphs are known as *unit grid intersection graphs* (UGIG) [28]. The recognition problem is NP-complete for both GIG [26] and UGIG [28]. We note that 4DOR is a subset of UGIG but Stick is not [8].

Researchers have examined further restrictions on GIG. For example, the graphs that admit a GIG representation with the additional constraint that all the segments must intersect (or be "stabbed by") a ground line form the *stabbable grid intersection* (StabGIG) graph class [8].

Another class of intersection graphs that restricts the objects on a ground line is defined in terms of *hooks*. A *hook* consists of a center point on the ground line together with an incident vertical segment and horizontal segment above the ground line. *Hook* graphs are intersection graphs of hooks [5, 21, 34], e.g., see Fig. 1(d). Hook graphs are also known as *max point-tolerance graphs* [5] and *heterozygosity graphs* [17]. The bipartite graphs that admit a Hook representation are called BipHook [8]. The complexities of recognizing the classes StabGIG, BipHook, and Stick are all open [8]. Chaplick *et al.* [8] examined the containment relations of these graph classes.

Grounded segment representations are a generalization of Stick representations, where the segments can have arbitrary slopes, e.g., see Fig. 1(e). Note that the segments are still restricted to lie on the same side of the ground line. Cardinal *et al.* [4] showed that the problem of deciding whether a graph admits a grounded segment representation is $\exists\mathbb{R}$-complete. We refer to [3, 4] for other

related classes such as outersegment and outerstring graphs, and for the study of their containment relations.

The following table summarizes the time complexities of recognizing different classes of bipartite intersection graphs, where n and m are the sizes of the two vertex sets of the bipartition.

Graph Class	**Time Complexity**	**Ref**
Chordal Bipartite Graphs	$O((n+m)^2)$, or $\|E\| \log(n+m)$	[27, 35]
Bipartite Permutation Graphs	$O(nm)$-time	[36]
2-Directional Ray Graphs (2DOR)	$O(nm)$-time	[12, 32]
3- or 4-Directional Ray Graphs (3DOR, 4DOR)	Open	[12, 32]
4-DOR with given directions for vertices	$f(n,m)$-time[a]	[15]
3-DOR with a given bipartition $(A \cup B)$, and an ordering for As, i.e., vertical rays	$O((n+m)^2)$-time	[15]
Grid Intersection Graphs (GIG)	NP-complete	[26]
Unit Grid Intersection Graphs (UGIG)	NP-complete	[28]
Grounded Segment Intersection Graphs	$\exists\mathbb{R}$-complete	[4]
StabGIG, SegRay, Hook, BipHook and Stick Graphs	Open	[8, 22]

[a] Multiplication time for two $(n+m) \times (n+m)$ matrices

2 Fixed As and Bs

In this section we study Stick representations of graphs with a fixed bipartition of the vertices and fixed vertex orderings for each vertex set. We call this problem Stick_{AB}, defined formally as follows.

Problem: Stick Representation with Fixed As and Bs (Stick_{AB})
Input: A bipartite graph $G = (A \cup B, E)$, an ordering σ_A of the vertices in A, and an ordering σ_B of the vertices in B.
Question: Does G admit a Stick representation such that the ith horizontal segment on the line ℓ corresponds to the ith vertex of σ_A and the jth vertical segment on ℓ corresponds to the jth vertex of σ_B?

We first present an $O(|A||B|)$-time algorithm for Stick_{AB}. A Stick representation is totally determined by the order σ of the segments' intersection with the ground line (details in the proof of Lemma 1). Thus the idea of the algorithm is to impose some ordering constraints between the vertices of A and B based on some submatrices of the adjacency matrix of G. We show that the required Stick representation exists if and only if there exists a total order σ of $(A \cup B)$ that satisfies the constraints and preserves the given orderings σ_A and σ_B. We now describe the details.

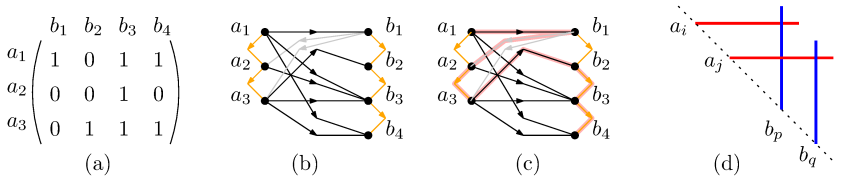


Fig. 3. (a) The incidence matrix M for the graph of Figure 1(a). (b) The directed graph H. (c) A total order $(a_1, b_1, a_2, a_3, b_2, b_3, b_4)$ of the vertices in H. The corresponding Stick drawing is in Figure 1(b). (d) Illustration of a forbidden ordering if $m_{j,p} = 0$.

Assume that $\sigma_A = (a_1, \ldots, a_n)$ and $\sigma_B = (b_1, \ldots, b_m)$. Let M be the ordered bipartite adjacency matrix of A and B, i.e., M has rows $a_1, \ldots, a_n$ and columns $b_1, \ldots, b_m$, where the entry $m_{i,p}$, i.e., the entry at the ith row and pth column, is 1 or 0 depending on whether a_i and b_p are adjacent or not, as illustrated in Figure 3(a).

We start with the constraints $a_{i-1} \prec a_i$, where $2 \leq i \leq n$, and $b_{p-1} \prec b_p$, where $2 \leq p \leq m$ to enforce the given orderings σ_A and σ_B. We now add some more constraints, as follows.

C_1: If an entry $m_{i,p}$ is 1, then add the constraint $a_i \prec b_p$, e.g., see the black edges in Figure 3(b).

C_2: If M contains an ordered submatrix $\begin{smallmatrix} & b_p & b_q \\ a_i & 1 & * \\ a_j & 0 & 1 \end{smallmatrix}$, then add the constraint $b_p \prec a_j$. For example, see the gray edges in Figure 3(b).

We now test whether the set of constraints is consistent. Consider a directed graph H with vertex set $(A \cup B)$, where each constraint corresponds to a directed edge (Figure 3(b)). Then the set of constraints is consistent if and only if H is acyclic, and the following lemma claims that this occurs if and only if the graph admits a Stick representation.

Lemma 1. *G admits a Stick representation respecting σ_A and σ_B if and only if H is acyclic, i.e., the constraints are consistent.*

Proof. We first show that the constraints are necessary. Every constraint between two vertices of the same set is implied by σ_A or σ_B. For Condition C_1, observe that a horizontal segment a_i can intersect a vertical segment b_p only if a_i precedes b_p, i.e., we must have $a_i \prec b_p$. For Condition C_2, we already have $a_i \prec a_j, b_p \prec b_q, a_i \prec b_p, a_j \prec b_q$. If we assume that $a_j \prec b_p$, then we have $a_i \prec a_j \prec b_p \prec b_q$, and to reach the vertical segment b_q, a_j would intersect b_p. Since $m_{j,p} = 0$, this intersection is forbidden. Figure 3(d) illustrates this scenario. Therefore, we must have the constraint $b_p \prec a_j$. Since all the constraints are necessary, if G admits an intersection representation, then the set of constraints is consistent.

We now prove the converse. Suppose the set of constraints is consistent. Take a total order of $A \cup B$ which is consistent with all the constraints, e.g.,

see Figure 1(c). This is a "topological order" of H. Initiate the drawing of the corresponding orthogonal segments in this order on the ground line ℓ. This determines the y-coordinate of every $a \in A$ and the x-coordinate of every $b \in B$. For each vertex $a \in A$, let $\max_B(a)$ be the neighbor of a in G with the largest index. We extend the horizontal segment corresponding to a to the right until the x-coordinate of $\max_B(a)$. Similarly, for each vertex $b \in B$, let $\min_A(b)$ be the neighbor of b in G with the minimum index. We extend the vertical segment corresponding to b upward until the y-coordinate of $\min_A(b)$.

We must show that the resulting drawing does not contain any forbidden intersection. Suppose by contradiction that the segments of a_j and b_p intersect, but they are not adjacent in G, i.e., $m_{j,p} = 0$. We now have $a_j \prec b_p$, and the entries $b_q = \max_B(a_j)$ and $a_i = \min_A(b_p)$ give the submatrix described in Condition C_2, thus the constraint $b_p \prec a_j$ applies, a contradiction. ■

An algorithm to solve Stick$_{AB}$ follows immediately, and can be implemented in linear time in the size of the adjacency matrix M.

Theorem 1. *There is an $O(|A||B|)$-time algorithm to decide the Stick$_{AB}$ problem, and construct a Stick representation if one exists.*

Proof. The algorithm was given above: We construct the directed graph H from the 0-1 matrix M and test if H is acyclic. This correctly decides Stick$_{AB}$ by Lemma 1. Furthermore, if H is acyclic, then we can construct a Stick representation as specified in the proof of Lemma 1. Pseudocode for the algorithm is given in Appendix A.

The matrix M has size $O(nm)$ where $n = |A|$ and $m = |B|$, and the graph H has $n + m$ vertices and $O(nm)$ edges. We can test acyclicity of a graph and find a topological ordering in linear time. Also, the construction of the Stick representation is clearly doable in linear time.

Thus we only need to give details on constructing H in time $O(nm)$. We can construct the edges of H that correspond to σ_A and σ_B in time $O(n + m)$. The edges arising from constraints C_1 correspond to the 1's in the matrix M, so we can construct them in $O(nm)$ time. The edges arising from constraints C_2 correspond to some of the 0's in the matrix M. Specifically, a 0 in position $m_{j,p}$ gives a C_2 constraint $b_p \prec a_j$ if and only if there is a 1 in row j to the right of the 0 and a 1 in column p above the 0. We can flag the 0's that have a 1 to their right by scanning each row of M from right to left. Similarly, we can flag the 0's that have a 1 above them by scanning each column of M from bottom to top. These scans take time $O(nm)$. Finally, if a 0 in M has both flags, then we add the corresponding edge to H. The total time is $O(nm)$. ■

Lemma 1 also yields a forbidden submatrix characterization for Stick$_{AB}$.

Theorem 2. *An instance of Stick$_{AB}$ with graph $G = (A \cup B, E)$ has a solution if and only if G's ordered adjacency matrix M has no ordered submatrix of the following form:*

$$P_1 = \begin{array}{c} \\ a_i \\ a_j \\ a_k \end{array}\!\!\begin{array}{c} \begin{array}{ccc} b_p & b_q & b_r \end{array} \\ \begin{bmatrix} * & 1 & * \\ * & 0 & 1 \\ 1 & * & * \end{bmatrix} \end{array}, \; P_2 = \begin{array}{c} \\ a_i \\ a_j \\ a_k \end{array}\!\!\begin{array}{c} \begin{array}{cc} b_p & b_q \end{array} \\ \begin{bmatrix} 1 & * \\ 0 & 1 \\ 1 & * \end{bmatrix} \end{array}, \; P_3 = \begin{array}{c} \\ a_i \\ a_j \end{array}\!\!\begin{array}{c} \begin{array}{ccc} b_p & b_q & b_r \end{array} \\ \begin{bmatrix} * & 1 & * \\ 1 & 0 & 1 \end{bmatrix} \end{array}.$$

Observe that P_2 and P_3 are special cases of P_1 with $p{=}q$ and $j{=}k$, respectively.

Proof. We will use the graph H that we constructed above and used in Lemma 1. By Lemma 1, the theorem statement is equivalent to the statement that M has a submatrix P_1, P_2 or P_3 if and only if H has a directed cycle.

We first show that if the matrix M has one of the ordered submatrices P_1, P_2, P_3 then H has a directed cycle. For P_1, the cycle in H is $a_k \prec b_p$ (by C_1), $b_p \prec b_q$ (by σ_B), $b_q \prec a_j$ (by C_2), $a_j \prec a_k$ (by σ_A). For P_2, the cycle is $b_p \prec a_j$ (by C_2), $a_j \prec a_k$ (by σ_A), $a_k \prec b_p$ (by C_1). For P_3, the cycle is $b_q \prec a_j$ (by C_2), $a_j \prec b_p$ (by C_1), $b_p \prec b_q$ (by σ_B).

To prove the other direction, suppose that H has a directed cycle O. We will show that M has one of the submatrices P_1, P_2, P_3. Let b_q be the rightmost vertex of O in σ_B, and let (b_q, z) be the outgoing edge of b_q in O. Since b_q is the rightmost vertex of O in σ_B, z must be a vertex a_j of A. The constraint $b_q \prec a_j$ can only be added by C_2. Therefore, we must have the configuration $\begin{array}{c} \\ a_i \\ a_j \end{array}\!\!\begin{array}{c} \begin{array}{cc} b_q & b_r \end{array} \\ \begin{bmatrix} 1 & * \\ 0 & 1 \end{bmatrix} \end{array}$. The path can now continue from a_j following zero or more A vertices, but to complete the cycle, it eventually needs to reach a vertex b_p of B. Since b_q is the rightmost in σ_B, b_p must appear either before b_q or coincide with b_q. First suppose that $b_p \neq b_q$. If the outgoing edge of a_j is (a_j, b_p), then we obtain the configuration P_3. Otherwise, the path visits several vertices of A and then visits b_p, and we thus obtain the configuration P_1.

Suppose now that $b_p = b_q$. In this case the outgoing edge of a_j cannot be (a_j, b_p), because such an edge can only be added by C_1, which would imply $m_{j,p} = m_{j,q} = 1$, violating the configuration above. If the path visits several vertices of A and then visits $b_p (= b_q)$, then there must be a 1 in the qth column below the jth row. We thus obtain the configuration P_2. ■

Bipartite Graphs Representable for All Orderings: The above forbidden submatrix characterization allows us to characterize the bipartite graphs $G = (A \cup B, E)$ that have a Stick representation for *every* possible ordering of A and B. Observe that the forbidden submatrices P_2, P_3, P_1 correspond, respectively, to the bipartite graphs shown in Figures 4(a)–(c). We can construct $2^2 = 4$ graphs from Figure 4(a) based on whether each of the dotted edges is present or not. Similarly, we can construct $2^2 = 4$ graphs from Figure 4(b), and $2^5 = 32$ graphs from Figure 4(b). Let $\mathcal{H}$ be the set that consists of these 40 graphs. From Theorem 2 we immediately obtain:

Theorem 3. *A bipartite graph $G = (A \cup B, E)$ admits a Stick representation for every possible ordering of A and B if and only if G does not contain any graph of $\mathcal{H}$.*

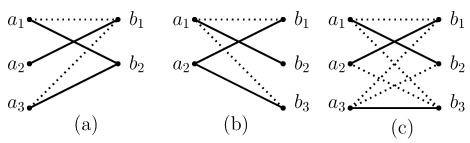


Fig. 4. The forbidden subgraphs for Theorem 3. Dotted edges are optional.

3 Fixed As

In this section we study the Stick representation problem when the ordering of only the vertices in A is given. A formal description of the problem, which we call Stick_A, is as follows.

Problem: Stick Representation with Fixed As (Stick_A)
Input: A bipartite graph $G = (A \cup B, E)$, and a vertex-ordering σ_A of A.
Question: Does G admit a Stick representation such that the ith horizontal segment on the ground line corresponds to the ith vertex of σ_A?

We give a polynomial-time algorithm for Stick_A. The idea is to use the forbidden submatrix characterization for Stick_{AB} (Theorem 2). We need an ordering of the B vertices that, together with the given ordering σ_A, avoids the forbidden submatrices P_1, P_2, P_3. We will express the conditions for the ordering of the B vertices as a 2-SAT formula, i.e., a CNF (conjunctive normal form) formula where each clause contains at most two literals. 2-SAT can be solved in polynomial time [1].

Theorem 4. *There is an algorithm with run-time $O(|A|^3|B|^3)$ to decide the Stick_A problem, and construct a Stick representation if one exists.*

Proof. For each pair of vertices v, w of G, we create variables $p_{v \prec w}$ and $p_{w \prec v}$ (representing the ordering of segments v and w on the ground line). We will enforce $p_{v \prec w} = \neg p_{w \prec v}$ by adding clauses $(\neg p_{v \prec w} \vee \neg p_{w \prec v}) \wedge (p_{v \prec w} \vee p_{w \prec v})$. (One variable would suffice, but it is notationally easier to have both.) We first set the truth values of all the variables involving two vertices of A based on σ_A. We then add a few other clauses based on P_1, P_2, P_3, as follows.

For every b_p, b_q, b_r giving rise to P_1, we add the clauses $(\neg p_{b_q \prec b_r} \vee p_{b_q \prec b_p})$ and $(\neg p_{b_p \prec b_q} \vee p_{b_r \prec b_q})$. The first clause means that if $b_q \prec b_r$, then to avoid P_1, we must have $b_q \prec b_p$. Similarly, the second clause means if $b_p \prec b_q$, then to avoid P_1, we must have $b_r \prec b_q$. These clauses ensure that if the SAT formula has a solution, then no configuration of the form P_1 can arise.

For every b_p, b_q giving rise to P_2, we set $p_{b_q \prec b_p}$ to true. This would avoid any forbidden configuration of the form P_2 in a solution of the 2-SAT formula.

Finally, for every b_p, b_q, b_r giving rise to P_3, we add the clauses $(\neg p_{b_q \prec b_r} \vee p_{b_q \prec b_p})$ and $(\neg p_{b_p \prec b_q} \vee p_{b_r \prec b_q})$. Note that these clauses can be interpreted in the same way as for P_1, i.e., if the 2-SAT formula has a solution, then no configuration of the form P_3 can arise.

Let F be the resulting 2-SAT formula, which can be solved in linear time in the input size [1], i.e., $O((|A| + |B|)^2)$ time. If F does not have a solution, then there does not exist any ordering of the Bs that avoids the forbidden patterns. Thus G does not admit the required Stick representation. If F has a solution, then there exists an ordering σ_B of Bs that together with σ_A avoids all the forbidden patterns. By Theorem 2, G admits the required Stick representation, and it can be constructed from σ_A and σ_B using Theorem 1.

Thus the time complexity of the algorithm is dominated by the time to construct the 2-SAT formula, which is $O(|A|^3|B|^3)$. ■

Bipartite Graphs Representable for All A Orderings: We also considered the class of bipartite graphs $G = (A \cup B, E)$ such that for every ordering of the vertices of A there exists a Stick representation. We will call this the $\text{Stick}_{\forall A}$ class. Although we do not have a characterization of the $\text{Stick}_{\forall A}$ class, we describe some positive and negative instances below in Remark 1 and Remark 2, with proofs in Appendix B.

Remark 1. Any bipartite graph $G = (A \cup B, E)$ with at most three vertices in A belongs to the $\text{Stick}_{\forall A}$ class.

Remark 2. A graph does not belong to $\text{Stick}_{\forall A}$ if its bipartite adjacency matrix contains the submatrix $\begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix}\begin{bmatrix} 1 & * \\ 0 & 1 \\ 1 & 0 \\ * & 1 \end{bmatrix}$. (Here the columns are unordered.)

4 Stick graphs

In this section we examine general Stick representations, i.e., we do not impose any constraints on the ordering of the vertices.

Problem: Stick Representation
Input: A bipartite graph $G = (A \cup B, E)$.
Question: Does G admit a Stick representation such that the vertices in A and B correspond to horizontal and vertical segments, respectively?

It is an open question to find a polynomial time algorithm for the above problem of recognizing Stick graphs. In this section we give some partial results.

We begin with some positive instances (Remarks 3–4) and some negative instances (Remark 5). The proofs of these remarks are included in Appendix C. We need a few definitions to state the remarks, as follows.

A matrix has the *simultaneous consecutive ones* property if the rows and columns can be permuted so that the 1's in each row and each column appear consecutively [30]. A *one-sided drawing* of a planar bipartite graph $G = (A \cup B, E)$ is a planar straight-line drawing of G, where all vertices in A lie on the x-axis, and the vertices of B lie strictly above the x-axis [16].

Remark 3. Let $G = (A \cup B, E)$ be a bipartite graph and let M be its adjacency matrix, where the rows and columns correspond to As and Bs, respectively. If M has the simultaneous consecutive ones property, then G admits a Stick representation, which can be computed in $O(|A||B|)$ time.

Remark 4. Let $G = (A \cup B, E)$ be an n-vertex bipartite graph that admits a one-sided planar drawing. Then G is a Stick graph, and its Stick representation can be computed in $O(n^2)$ time.

Remark 5. Let H be the graph obtained by deleting a perfect matching from a complete bipartite graph $K_{4,4}$. Any graph $G = (A \cup B, E)$ containing H as an induced subgraph does not admit a Stick representation. Since H is a planar graph, not all planar bipartite graphs are Stick graphs.

Recognition Algorithm: We now give an exact $O(1.3^{n^2})$-time algorithm to decide whether an n-vertex bipartite graph with a fixed bipartition admits a Stick representation. We use 3-SAT, i.e., a CNF (conjunctive normal form) formula such that each of its clauses contains at most three literals. Let $G = (A \cup B, E)$ be the given bipartite graph. We create a 3-SAT formula Φ such that Φ is satisfiable if and only if G admits a Stick representation. We now describe the algorithm in detail.

For each pair of vertices v, w of G, we create variables $p_{v \prec w}$ and $p_{w \prec v}$ (representing the ordering of v and w on the ground line), and add clauses $(\neg p_{v \prec w} \vee \neg p_{w \prec v}) \wedge (p_{v \prec w} \vee p_{w \prec v})$ to enforce $p_{v \prec w} = \neg p_{w \prec v}$. We now express the conditions C1 and C2 from Section 2 as 3-SAT clauses.

- Φ_1**:** (Condition C1.) If $m_{i,p} = 1$, then set $p_{a_i \prec b_p} = 1$.
- Φ_2**:** (Condition C2.) We must express the condition that if the ordered submatrix $\begin{smallmatrix} & b_p & b_q \\ a_i & 1 & * \\ a_j & 0 & 1 \end{smallmatrix}$ exists, then $p_{b_p \prec a_j} = 1$. Thus, if $m_{i,p} = 1, m_{jq} = 1$ and $m_{j,p} = 0$, then we add the clause $(\neg p_{a_i \prec a_j} \vee \neg p_{b_p \prec b_q} \vee \neg p_{a_j \prec b_p})$.
- Φ_3**:** For each triple u, v, w of vertices, add the clause $(\neg p_{u \prec v} \vee \neg p_{v \prec w} \vee p_{u \prec w})$. Intuitively, these are transitivity constraints, which would ensure a total ordering on the ground line.

Let Φ be the resulting 3-SAT formula. We now have the following:

Lemma 2. *The 3-SAT Φ is satisfiable if and only if G admits the required intersection representation.*

Proof. First assume that Φ is satisfiable. We first claim that by the clauses of Φ_3, one can obtain a total order τ of the vertices on the ground line. Suppose for a contradiction that there exists a cycle, e.g., $p_{u_1 \prec u_2}, p_{u_2 \prec u_3}, \ldots, p_{u_{k-1} \prec u_k}, p_{u_k \prec u_1}$ are all set to 1. Consider such a cycle where k is minimal. We first claim that $k > 3$. By construction of Φ_3, k cannot be equal to 3. In addition, if $k = 2$, then we have clauses that ensure that $p_{u_1 \prec u_2}$ and $p_{u_2 \prec u_1}$ cannot have the same truth value. Observe that if $k > 3$, then by the construction of Φ_3, we can find

a smaller cycle $p_{u_1 \prec u_3}, \ldots, p_{u_{k-1} \prec u_k}, p_{u_k \prec u_1}$, which contradicts that the cycle is minimal.

Since Φ_1 and Φ_2 impose the constraints of Lemma 1, one can construct the required Stick representation using τ.

For the other direction, assume that a Stick representation exists. We will construct a satisfying truth assignment for Φ. For each v, w, if v precedes w on the ground line, then we set $p_{v \prec w}$ to 1, and $p_{w \prec v}$ to 0, which is also consistent with Φ_3. If two segments a_i and b_p intersect, then a_i must precede b_p. Therefore, we set $p_{a_i \prec b_p}$ to be 1 and $p_{b_p \prec a_i}$ to be 0, which is consistent with Φ_1. Finally, if $a_i \prec a_j \prec b_p \prec b_q$, and we have the submatrix as described in the construction of Φ_2 then a_j and b_p must intersect. Thus the clauses added in Φ_2 must be satisfied. ■

Algorithm 3 in Appendix C gives pseudocode for our algorithm. An instance of 3-SAT with k variables can be solved in $O(1.3^k)$ time [20]. Note that we have a polynomial number of clauses and the number of variables is $O(n^2)$, where $n = |A| + |B|$. We thus obtain the following theorem.

Theorem 5. *Given an n-vertex bipartite graph $G = (A \cup B, E)$, one can decide whether G admits a Stick representation in $O(1.3^{n^2} f(n))$ time, where $f(\cdot)$ is a polynomial in n.*

Our 3-SAT formulation is in fact a 'mixed Horn formula' [31] with at most three literals per clause. Therefore, an interesting direction of research would be to find faster exponential-time algorithms for such mixed Horn 3-SAT.

5 Open Problems

1. What is the complexity of recognizing Stick graphs? Is the problem NP-complete? By Theorem 2 the problem is equivalent to ordering the rows and columns of a 0-1 matrix to exclude the 3 forbidden submatrices given in the Theorem statement. Note that these forbidden submatrices involve 5 or 6 rows and columns (vertices of the graph) so the results of Hell et al. [19], which apply to patterns of at most 4 vertices in a bipartite graph, do not provide a polynomial time algorithm.

2. Can we improve the time complexity of the recognition algorithm for graphs with fixed As?

References

1. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Information Processing Letters **8**(3), 121–123 (1979)
2. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Sciences **13**(3), 335–379 (1976)
3. Cabello, S., Jejcic, M.: Refining the hierarchies of classes of geometric intersection graphs. Electr. J. Comb. **24**(1), 33 (2017)
4. Cardinal, J., Felsner, S., Miltzow, T., Tompkins, C., Vogtenhuber, B.: Intersection graphs of rays and grounded segments. In: Bodlaender, H.L., Woeginger, G.J. (eds.) Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science. LNCS, vol. 10520, pp. 153–166. Springer (2017)
5. Catanzaro, D., Chaplick, S., Felsner, S., Halldórsson, B.V., Halldórsson, M.M., Hixon, T., Stacho, J.: Max point-tolerance graphs. Discrete Applied Mathematics **216**, 84–97 (2017)
6. Chalopin, J., Gonçalves, D.: Every planar graph is the intersection graph of segments in the plane. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing. pp. 631–638. ACM (2009)
7. Chan, T.M., Grant, E.: Exact algorithms and APX-hardness results for geometric packing and covering problems. Computational Geometry **47**(2, Part A), 112–124 (2014)
8. Chaplick, S., Felsner, S., Hoffmann, U., Wiechert, V.: Grid intersection graphs and order dimension. arXiv preprint arXiv:1512.02482 (2015)
9. Chaplick, S., Cohen, E., Morgenstern, G.: Stabbing polygonal chains with rays is hard to approximate. In: Canadian Conference on Computational Geometry (CCCG) (2013)
10. Chaplick, S., Hell, P., Otachi, Y., Saitoh, T., Uehara, R.: Intersection dimension of bipartite graphs. In: Gopal, T.V., Agrawal, M., Li, A., Cooper, S.B. (eds.) Theory and Applications of Models of Computation. pp. 323–340. Springer (2014)
11. Chen, L., Yesha, Y.: Efficient parallel algorithms for bipartite permutation graphs. Networks **23**(1), 29–39 (1993)
12. Cogis, O.: On the Ferrers dimension of a digraph. Discrete Mathematics **38**(1), 47–52 (1982)
13. Corneil, D.G., Olariu, S., Stewart, L.: The LBFS structure and recognition of interval graphs. SIAM Journal on Discrete Mathematics **23**(4), 1905–1953 (2009)
14. Das, S., Sen, M., Roy, A.B., West, D.B.: Interval digraphs: An analogue of interval graphs. Journal of Graph Theory **13**(2), 189–202 (1989)
15. Felsner, S., Mertzios, G.B., Mustata, I.: On the recognition of four-directional orthogonal ray graphs. In: Proceedings of the 38th Mathematical Foundations of Computer Science (MFCS). LNCS, vol. 8087, pp. 373–384. Springer (2013)
16. Fößmeier, U., Kaufmann, M.: Nice drawings for planar bipartite graphs. In: Bongiovanni, G.C., Bovet, D.P., Battista, G.D. (eds.) Proceedings of the 3rd Italian Conference on Algorithms and Complexity. LNCS, vol. 1203, pp. 122–134. Springer (1997)
17. Halldórsson, B.V., Aguiar, D., Tarpine, R., Istrail, S.: The Clark phase-able sample size problem: Long-range phasing and loss of heterozygosity in GWAS. In: Berger, B. (ed.) Research in Computational Molecular Biology. pp. 158–173. Springer (2010)

18. Hartman, I.B.A., Newman, I., Ziv, R.: On grid intersection graphs. Discrete Mathematics **87**(1), 41–52 (1991)
19. Hell, P., Mohar, B., Rafiey, A.: Ordering without forbidden patterns. In: European Symposium on Algorithms. pp. 554–565. Springer (2014)
20. Hertli, T., Moser, R.A., Scheder, D.: Improving PPSZ for 3-SAT using critical variables. In: Schwentick, T., Dürr, C. (eds.) Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS). LIPIcs, vol. 9, pp. 237–248 (2011)
21. Hixon, T.: Hook graphs and more: Some contributions to geometric graph theory. Master's thesis, Technische Universität Berlin (2013)
22. Hoffmann, U.: Intersection graphs and geometric objects in the plane. Master's thesis, Technische Universität Berlin (2016)
23. Katebi, H., Sakallah, K.A., Silva, J.P.M.: Empirical study of the anatomy of modern SAT solvers. In: Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT). LNCS, vol. 6695. Springer (2011)
24. Katz, M.J., Mitchell, J.S., Nir, Y.: Orthogonal segment stabbing. Computational Geometry **30**(2), 197–205 (2005)
25. Klinz, B., Rudolf, R., Woeginger, G.J.: Permuting matrices to avoid forbidden submatrices. Discrete Applied Mathematics **1**(60), 223–248 (1995)
26. Kratochvíl, J.: A special planar satisfiability problem and a consequence of its NP-completeness. Discrete Applied Mathematics **52**(3), 233–252 (1994)
27. Lubiw, A.: Doubly lexical orderings of matrices. SIAM Journal on Computing **16**(5), 854–879 (1987)
28. Mustaţă, I., Pergel, M.: Unit grid intersection graphs: Recognition and properties. arXiv preprint arXiv:1306.1855 (2013)
29. Müller, H.: Recognizing interval digraphs and interval bigraphs in polynomial time. Discrete Applied Mathematics **78**(1), 189–205 (1997)
30. Oswald, M., Reinelt, G.: The simultaneous consecutive ones problem. Theoretical Computer Science **410**(21-23), 1986–1992 (2009)
31. Porschen, S., Speckenmeyer, E.: Satisfiability of mixed Horn formulas. Discrete Applied Mathematics **155**(11), 1408–1419 (2007)
32. Shrestha, A.M.S., Tayu, S., Ueno, S.: On orthogonal ray graphs. Discrete Applied Mathematics **158**(15), 1650–1659 (2010)
33. Soto, J.A., Telha, C.: Jump number of two-directional orthogonal ray graphs. In: Proceedings of the 15th International Conference on Integer Programming and Combinatoral Optimization. pp. 389–403. IPCO'11, Springer-Verlag (2011)
34. Soto, M., Caro, C.T.: p-box: A new graph model. Discrete Mathematics and Theoretical Computer Science **17**(1), 169 (2015)
35. Spinrad, J.P.: Doubly lexical ordering of dense 0-1 matrices. Information Processing Letters **45**(5), 229–235 (1993)
36. Spinrad, J.P., Brandstädt, A., Stewart, L.: Bipartite permutation graphs. Discrete Applied Mathematics **18**(3), 279–292 (1987)

A Fixed *A*s and *B*s

Algorithm 1 Algorithm for Fixed-A-Fixed-B-Stick Recognition

Input: A bipartite graph $G = (A \cup B, E)$, an ordering σ_A and an ordering σ_B
$M \leftarrow$ A matrix representation of G whose rows follow the ordering σ_A, and columns follow the ordering σ_b
$H \leftarrow$ A graph with vertex set $(A \cup B)$ but without any edge.
for each consecutive vertices a_{i-1}, a_i in σ_A **do** add the edge (a_{i-1}, a_i) to H
for each consecutive vertices b_{j-1}, b_j in σ_B **do** add the edge (b_{j-1}, b_j) to H
for each entry $m_{i,j} = 1$ in M **do** add the edge (a_i, b_j) to H
for each a_i, a_j, b_p, b_q (respecting σ_A and σ_B) of the form $\begin{array}{c} \\ a_i \\ a_j \end{array}\begin{array}{c} \begin{array}{cc} b_p & b_q \end{array} \\ \begin{bmatrix} 1 & * \\ 0 & 1 \end{bmatrix} \end{array}$ **do** add the edge (b_p, a_j) to H
if H contains a cycle **then return** false //No solution exists
$\sigma \leftarrow$ A topological ordering of the vertices of H
return σ

B Fixed *A*s

Proof (Remark 1).

The proof is by construction. For any ordering σ_A, we can categorize the columns into at most eight categories (assuming A has exactly three vertices, the other cases are straightforward). We then organize those categories in the following order: $\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array}\begin{array}{c} \begin{array}{cccccccc} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$. It is now straightforward to see that the resulting ordering corresponds to the required Stick representation, e.g., see Fig. 5(a). ■

Proof (Remark 2).

We refer the reader to Fig. 5(b). Any σ_A consistent with the given matrix would serve as a negative instance, as follows. The adjacency matrix $\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array}\begin{array}{c} \begin{array}{cc} b_1 & b_2 \end{array} \\ \begin{bmatrix} 1 & * \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array}$

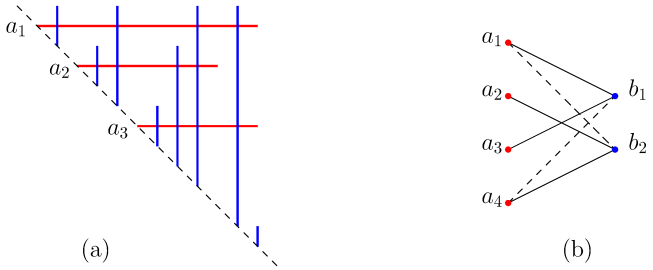


Fig. 5. (a) Illustration for Remark 1. (b) Illustration for Remark 2.

would require b_2 to come after b_1, while the matrix $\begin{array}{c} \\ a_2 \\ a_3 \\ a_4 \end{array}\begin{array}{c} \begin{array}{cc} b_1 & b_2 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ * & 1 \end{bmatrix} \end{array}$ would require b_1 to come after b_2. Thus there is no consistent ordering for the vertices of B, and hence G does not have a Stick representation respecting σ_A. ■

Algorithm 2 Algorithm for Fixed-A-Stick Recognition

1: **Input:** A bipartite graph $G = (A \cup B, E)$, an ordering σ_A

2: $P_1 \leftarrow \begin{array}{c} \\ a_i \\ a_j \\ a_k \end{array}\begin{array}{c} \begin{array}{ccc} b_p & b_q & b_r \end{array} \\ \begin{bmatrix} * & 1 & * \\ * & 0 & 1 \\ 1 & * & * \end{bmatrix} \end{array}$, $P_2 \leftarrow \begin{array}{c} \\ a_i \\ a_j \\ a_k \end{array}\begin{array}{c} \begin{array}{cc} b_p & b_q \end{array} \\ \begin{bmatrix} 1 & * \\ 0 & 1 \\ 1 & * \end{bmatrix} \end{array}$, $P_3 \leftarrow \begin{array}{c} \\ a_i \\ a_j \end{array}\begin{array}{c} \begin{array}{ccc} b_p & b_q & b_r \end{array} \\ \begin{bmatrix} * & 1 & * \\ 1 & 0 & 1 \end{bmatrix} \end{array}$.

3: $\Phi \leftarrow$ A 2-SAT with variables of the form $p_{u \prec w}$ and $p_{w \prec u}$, for every pair of vertices u, w of G.

4: **for each** a_i, a_j (respecting σ_A) **do** $p_{a_i \prec a_j} \leftarrow true$, $p_{a_j \prec a_i} \leftarrow false$

5: **for each** P_1 or P_3 **do** add the clauses $(\neg p_{b_q \prec b_r} \vee p_{b_q \prec b_p})$ and $(\neg p_{b_p \prec b_q} \vee p_{b_r \prec b_q})$

6: **for each** P_2 **do** $p_{b_q \prec b_p} \leftarrow true$, $p_{b_p \prec b_q} \leftarrow false$

7: **for each** P_3 **do** add the clauses $(\neg p_{b_q \prec b_r} \vee p_{b_q \prec b_p})$ and $(\neg p_{b_p \prec b_q} \vee p_{b_r \prec b_q})$

8: **if** Φ does not admit an affirmative answer **then return** false //No solution exists

9: $\sigma \leftarrow$ A total ordering of vertices determined by any solution of Φ

10: **return** σ

C Free *A*s and Free *B*s

Algorithm 3 Algorithm for Stick Recognition

Input: A bipartite graph $G = (A \cup B, E)$
$\Phi \leftarrow$ A 3-SAT with variables of the form $p_{u \prec w}$ and $p_{w \prec u}$, for every pair of vertices u, w of G.
for each pair of vertices u, w of G **do** add clauses $(\neg p_{v \prec w} \vee \neg p_{w \prec v}) \wedge (p_{v \prec w} \vee p_{w \prec v})$.
for each edge (a_i, b_j) in G **do** set $p_{a_i \prec b_k} \leftarrow true$, and $p_{b_k \prec a_i} \leftarrow false$.
for each a_i, a_j, b_k, b_l **do**
    if $m_{i,p} = 1, m_{jq} = 1$ and $m_{j,p} = 0$ **then** add $(\neg p_{a_i \prec a_j} \vee \neg p_{b_p \prec b_q} \vee \neg p_{a_j \prec b_p})$
for each triple u, v, w of vertices of G **do** add the clause $(\neg p_{u \prec v} \vee \neg p_{v \prec w} \vee p_{u \prec w})$.
if Φ does not admit an affirmative answer **then return** false //No solution exists
$\sigma \leftarrow$ A total ordering of vertices determined by any solution of Φ
return σ

Proof (Remark 3).

One can determine whether M has the simultaneous consecutive one's property in $O(|A||B|)$ time [30], and if so, then one can construct such a matrix M' within the same time complexity.

We now show how to construct the Stick representation from M'. For each row (resp., column), we draw a horizontal (resp., vertical) segment starting from the rightmost (resp., topmost) 1 entry. We extend the horizontal segments to the left and vertical segments downward such that they touch a ground line ℓ, e.g., see Fig. 6(a)–(b).

Let the resulting drawing be D, which may contain many unnecessary crossings. However, for each unnecessary crossing, we can follow the segments involved in the crossings upward and rightward to find two distinct 1 entries. Since the matrix has the simultaneous ones property, the violated entries in each row (column) must lie consecutively at the left end of the row (bottom end of the column). Therefore, one can find a $(+x, -y)$-monotone path P that separates the violated entries from the rest of the matrix, e.g., see the shaded region in Fig. 6(b).

Let $b_1, b_2, \ldots, b_k$ be the bend points creating 90° angles towards ℓ. To compute the required Stick representation, we remove these bends one after another, as follows. Consider the topmost bend point b_i, e.g., see b_1 in Fig. 6(c). Imagine a Cartesian coordinate system with origin at b_i. Move the rows above b_i and columns to the right of b_i towards the upward and rightward directions, respectively, as illustrated in Fig. 6(d). It is straightforward to observe that one now can construct a ground line ℓ' through b_i such that the violated entries lie in the region below the path determined by $b_{i+1}, \ldots, b_k$. ■

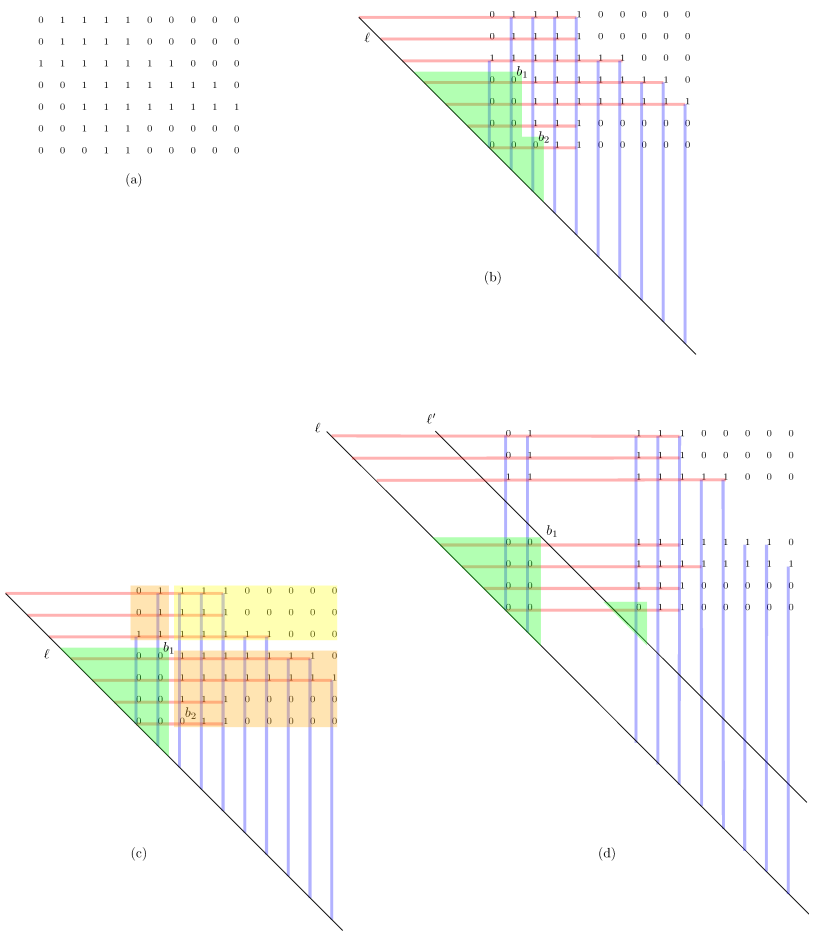


Fig. 6. Illustration for Remark 3.

Proof (Remark 4).

Given an n-vertex bipartite graph $G = (A \cup B, E)$, one can decide whether G admits a one-sided planar drawing in $O(n^2)$ time [16], and if so, then one can construct such a drawing within the same time complexity.

It now suffices to show how to construct a Stick representation from the one-sided drawing. Let Γ be a one-sided drawing of G. Imagine that we connected the A-vertices in a path from left to right in order they appear on the x-axis such that all the edges of the path are on the outerface. We define the *upper set* of Γ to be the B vertices that lie on the outerface, including their neighbors.

We start by drawing the upper set. It is straightforward to draw a Stick representation by first drawing the A-vertices in the order they appear on the upper set from top to bottom on the ground line, and then the B-vertices to

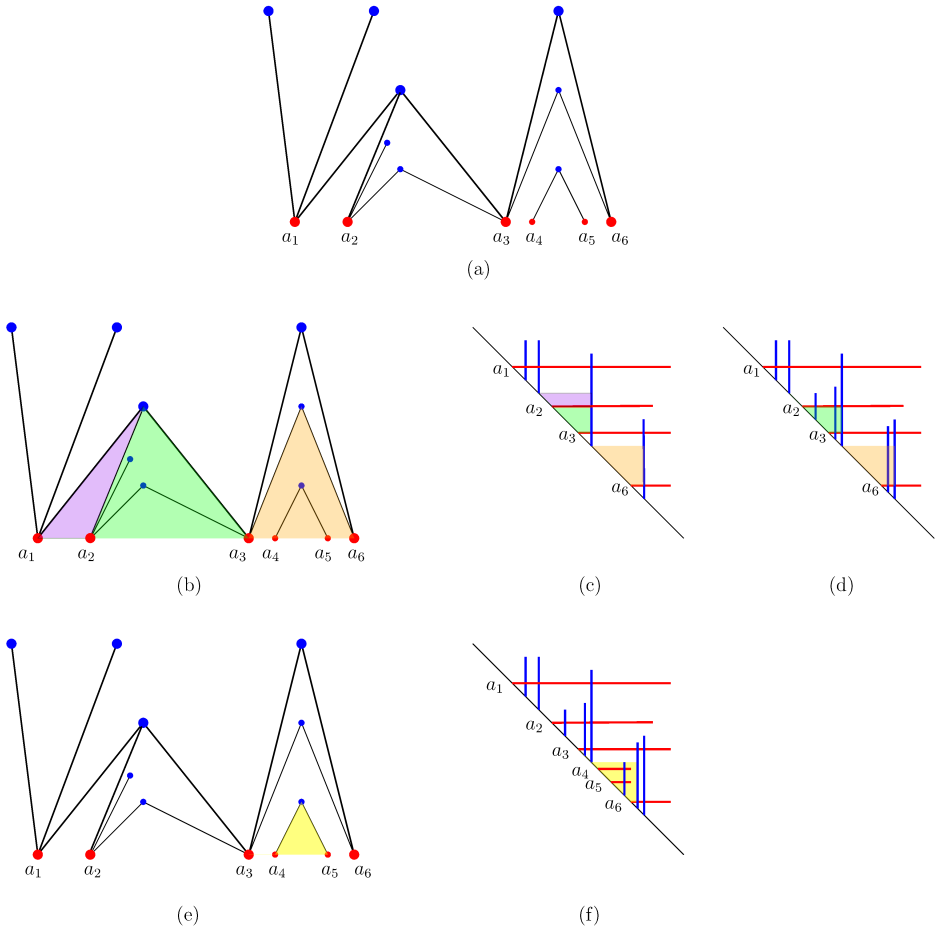
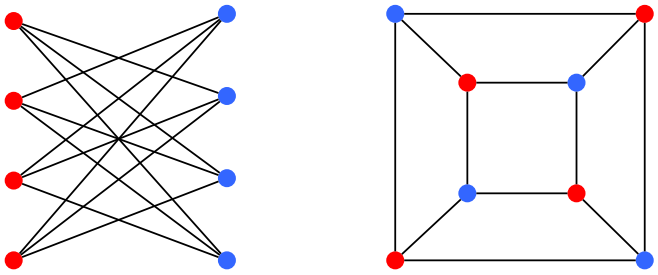


Fig. 7. Illustration for Remark 4.

complete the Stick representation of the set. Fig. 7(a) illustrates a one-sided drawing, and Fig. 7(c) illustrates a drawing of its upper set.

Let Γ' be the one-sided drawing by deleting the B-vertices of degree one from the upper set of Γ. Note that every B-vertex w of degree two or more on the upper set, 'covers' smaller one-sided drawings below the edges that connect w to a pair of consecutive A-neighbors, e.g., see the shaded regions of Fig. 7(b). We continue drawing each of these smaller one-sided drawings recursively between their corresponding leftmost and rightmost A-vertices, which have already been drawn in the previous level. Fig. 7(d) illustrates the drawing in the second recursion level, and Fig. 7(e)–(f) illustrates the third recursion level. ■

Proof (Remark 5).



(a) (b)

Fig. 8. Graph that does not admit a Stick representation. Its matrix representation can have all zeros placed on a diagonal.

It suffices to show that H (Fig. 8(a)) does not admit a Stick representation. It is straightforward to permute the rows and columns of the matrix representation of H such that all zeros lie on the main diagonal, e.g., $M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$. Therefore, for every permutation of rows and columns, we can pick the two columns that have zeros in the middle two rows to obtain the configuration $\begin{bmatrix} 1 & * \\ 0 & 1 \\ 1 & 0 \\ * & 1 \end{bmatrix}$. As we discussed in Sec. 3, there cannot be any Stick representation with this row ordering. Consequently, the graph H, as well as no graph containing H as an induced subgraph, can have a Stick representation.

Since H is a planar graph (see Fig. 8(b)), not all planar bipartite graphs are Stick graphs. ■