Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 24, no. 4, pp. 657–681 (2020)
DOI: 10.7155/jgaa.00524

Recognizing Stick Graphs with and without Length Constraints

Steven Chaplick[1] *Philipp Kindermann*[1]
Andre Löffler[1] *Florian Thiele*[1] *Alexander Wolff*[1]
Alexander Zaft[1] *Johannes Zink*[1]

[1]Institut für Informatik,
Universität Würzburg, Würzburg, Germany

Abstract

Stick graphs are intersection graphs of horizontal and vertical line segments that all touch a line of slope -1 and lie above this line. De Luca et al. [GD'18] considered the recognition problem of stick graphs when no order is given (STICK), when the order of either one of the two sets is given ($\mathsf{STICK}_\mathsf{A}$), and when the order of both sets is given ($\mathsf{STICK}_\mathsf{AB}$). They showed how to solve $\mathsf{STICK}_\mathsf{AB}$ efficiently.

In this paper, we improve the running time of their algorithm, and we solve $\mathsf{STICK}_\mathsf{A}$ efficiently. Further, we consider variants of these problems where the lengths of the sticks are given as input. We show that these variants of STICK, $\mathsf{STICK}_\mathsf{A}$, and $\mathsf{STICK}_\mathsf{AB}$ are all NP-complete. On the positive side, we give an efficient solution for $\mathsf{STICK}_\mathsf{AB}$ with fixed stick lengths if there are no isolated vertices.

Submitted:	Reviewed:	Revised:	Accepted:
November 2019	January 2020	February 2020	March 2020
	Final:	Published:	
	March 2020	December 2020	
Article type:		Communicated by:	
Regular paper		D. Archambault and C. D. Tóth	

A preliminary version of this paper appeared in Proc. Int. Symp. Graph Drawing & Network Visualization 2019 [9]. S.C. and A.W. acknowledge support from DFG grants WO 758/11-1 and WO 758/9-1, respectively.

E-mail addresses: loeffler@informatik.uni-wuerzburg.de (Andre Löffler) johannes.zink@uni-wuerzburg.de (Johannes Zink)

1 Introduction

For a given collection $\mathcal{S}$ of geometric objects, the *intersection graph of* $\mathcal{S}$ has $\mathcal{S}$ as its vertex set and an edge whenever $S \cap S' \neq \emptyset$, for $S, S' \in \mathcal{S}$. This paper concerns *recognition* problems for classes of intersection graphs of restricted geometric objects, i.e., determining whether a given graph is an intersection graph of a family of restricted sets of geometric objects. An established (general) class of intersection graphs is that of *segment graphs*, the intersection graphs of line segments in the plane[1]. For example, segment graphs are known to include planar graphs [5]. The recognition problem for segment graphs is $\exists\mathbb{R}$-complete [19, 22][2]. On the other hand, one of the simplest natural subclasses of segment graphs is that of the *permutation* graphs, the intersection graphs of line segments where there are two parallel lines such that each line segment has its two end points on these parallel lines[3]. We say that the segments are *grounded* on these two lines. The recognition problem for permutation graphs can be solved in linear time [20]. *Bipartite* permutation graphs have an even simpler intersection representation [25]: they are the intersection graphs of unit-length vertical and horizontal line segments which are again double-grounded (without loss of generality, both lines have a slope of -1). The simplicity of bipartite permutation graphs leads to a simpler linear-time recognition algorithm [27] than that of permutation graphs.

Several recent articles [2, 3, 7, 8] compare and study the geometric intersection graph classes occurring between the simple classes, such as bipartite permutation graphs, and the general classes, such a segment graphs. Cabello and Jejčič [2] mention that studying such classes with constraints on the sizes or lengths of the objects is an interesting direction for future work (and such constraints are the focus of our work). Note that similar length restrictions have been considered for other geometric intersection graphs such as interval graphs [16, 17, 23].

When the segments are not grounded, but still are only horizontal and vertical, the class is referred to as *grid intersection graphs* and it also has a rich history, see, e.g., [7, 8, 14, 18]. In particular, note that the recognition problem is NP-complete for grid intersection graphs [18]. But, if both the permutation of the vertical segments and the permutation of the horizontal segments are given, then the problem becomes a trivial check on the bipartite adjacency matrix [18]. However, for the variant where only one such permutation, e.g., the order of the horizontal segments, is given, the complexity remains open. A few special cases of this problem have been solved efficiently [6, 10, 11], e.g., one such case [6] is equivalent to the problem of *level planarity testing* which can be solved in linear time [15].

[1] We follow the common convention that parallel segments do not intersect and each point in the plane belongs to at most two segments.

[2] Note that $\exists\mathbb{R}$ includes NP, see [22, 24] for background on the complexity class $\exists\mathbb{R}$.

[3] i.e., we think of the sequence of end points on the "bottom" line as one permutation π on the vertices and the sequence on the "top" line as another permutation π', where uv is an edge if and only if the order of u and v differs in π and π'.

In this paper we study recognition problems concerning so-called *stick* graphs, the intersection graphs of grounded vertical and horizontal line segments (i.e., grounded grid intersection graphs). Classes closely related to stick graphs appear in several application contexts, e.g., in *nano PLA-design* [26] and detecting *loss of heterozygosity events in the human genome* [4, 13]. Note that, similar to the general case of segment graphs, it was recently shown that the recognition problem for grounded segments (where arbitrary slopes are allowed) is $\exists\mathbb{R}$-complete [3]. So, it seems likely that the recognition problem for stick graphs is NP-complete (similar to grid intersection graphs), but thus far it remains open. The primary prior work on recognizing stick graphs is due to De Luca et al. [10]. They introduced two constrained cases of the stick graph recognition problem called $\mathsf{STICK}_\mathsf{A}$ and $\mathsf{STICK}_\mathsf{AB}$, which we define next. Similarly to Kratochvíl's approach to grid intersection graphs [18], De Luca et al. characterized stick graphs through their bipartite adjacency matrix and used this result as a basis to develop a polynomial-time algorithm to solve $\mathsf{STICK}_\mathsf{AB}$.

Definition 1.1 (STICK) *Let G be a bipartite graph with vertex set $A \dot\cup B$, and let ℓ be a line with slope -1. Decide whether G has an intersection representation where the vertices in A are vertical line segments whose bottom end-points lie on ℓ and the vertices in B are horizontal line segments whose left end-points lie on ℓ.*[4] *Such a representation is a* stick representation *of G, the line ℓ is the* ground line*, the segments are called* sticks*, and the point where a stick meets ℓ is its* foot point*.*

Definition 1.2 ($\mathsf{STICK}_\mathsf{A}$/$\mathsf{STICK}_\mathsf{AB}$) *In the problem $\mathsf{STICK}_\mathsf{A}$ ($\mathsf{STICK}_\mathsf{AB}$) we are given an instance of the STICK problem and additionally an order σ_A (orders σ_A, σ_B) of the vertices in A (in A and B). The task is to decide whether there is a stick representation that respects σ_A (σ_A and σ_B).*

Our Contribution. We first revisit the problems $\mathsf{STICK}_\mathsf{A}$ and $\mathsf{STICK}_\mathsf{AB}$ defined by De Luca et al. [10]. We provide the first efficient algorithm for $\mathsf{STICK}_\mathsf{A}$ and a faster algorithm for $\mathsf{STICK}_\mathsf{AB}$; see Section 2. For our $\mathsf{STICK}_\mathsf{A}$ algorithm we introduce a new tool, semi-ordered trees (see Section 2.2), as a way to capture all possible permutations of the horizontal sticks which occur in a solution to the given $\mathsf{STICK}_\mathsf{A}$ instance. We feel that this data structure may be of independent interest. Then we investigate the direction suggested by Cabello and Jejčič [2] where specific lengths are given for the segments of each vertex. In particular, this can be thought of as generalizing from unit stick graphs (i.e., bipartite permutation graphs), where every segment has the same length. While bipartite permutation graphs can be recognized in linear time [27], it turns out that all of the new problem variants (which we call $\mathsf{STICK}^\mathsf{fix}$, $\mathsf{STICK}^\mathsf{fix}_\mathsf{A}$, and $\mathsf{STICK}^\mathsf{fix}_\mathsf{AB}$) are NP-complete; see Section 3. Finally, we give an efficient solution for $\mathsf{STICK}^\mathsf{fix}_\mathsf{AB}$ (that is, $\mathsf{STICK}_\mathsf{AB}$ with fixed stick lengths) for the special case that there are no isolated vertices (see Section 3.3). We conclude and state some open problems in Section 4. Our results are summarized in Table 1.

[4] Note that De Luca et al. [10] regarded A as the set of horizontal segments.

Table 1: Previously known and new results for deciding whether a given bipartite graph $G = (A \dot\cup B, E)$ is a stick graph. In $O(\cdot)$, we dropped $|\cdot|$. We abbreviate vertices by vtc., NP-complete by NPC, Theorem by T., and Corollary by C.

given order	variable length		fixed length	
	old	new	isolated vtc.	no isolated vtc.
$\emptyset$	unknown	unknown	NPC [T. 3.1]	NPC [T. 3.1]
A	unknown	$O(AB)$ [T. 2.2]	NPC [T. 3.3]	NPC [T. 3.3]
A,B	$O(AB)$ [10]	$O(E)$ [T. 2.1]	NPC [C. 3.4]	$O((A+B)^2)$ [C. 3.8]

2 Sticks of Variable Lengths

In this section, we provide algorithms that solve the $\mathsf{STICK}_{\mathsf{AB}}$ problem in $O(|A|+|B|+|E|)$ time (see Section 2.1, Theorem 2.1) and the $\mathsf{STICK}_{\mathsf{A}}$ problem in $O(|A| \cdot |B|)$ time (see Section 2.3, Theorem 2.2). Between these subsections, in Section 2.2, we describe *semi-ordered trees*, an essential tool reminiscent of PQ-trees that we will use for the latter algorithm. This tool will allow us to express the different ways one can order the horizontal segments for a given instance of $\mathsf{STICK}_{\mathsf{A}}$.

2.1 Solving $\mathsf{STICK}_{\mathsf{AB}}$ in $O(|A| + |B| + |E|)$ time

De Luca et al. [10] showed how to compute, for a given graph $G = (A \cup B, E)$ and orders σ_A and σ_B, a $\mathsf{STICK}_{\mathsf{AB}}$ representation in $O(|A| \cdot |B|)$ time (if such a representation exists). We improve upon their result in this section. Namely, we prove the following theorem.

Theorem 2.1 *$\mathsf{STICK}_{\mathsf{AB}}$ can be solved in $O(|A| + |B| + |E|)$ time.*

Proof: We apply a sweep-line approach (with a vertical sweep-line moving rightwards) where each vertical stick $a_i \in A$ corresponds to two events: the *enter event of* a_i (abbreviated by i) and the *exit event of* a_i (abbreviated by $i{\rightarrow}$).

Let $\sigma_A = (a_1, \ldots, a_{|A|})$ and $\sigma_B = (b_1, \ldots, b_{|B|})$. Let β_i denote the largest index such that b_{β_i} has a neighbor in $a_1, \ldots, a_i$. Let $\hat{B}^i$ be the subsequence of $(b_1, \ldots, b_{\beta_i})$ of those vertices that have a neighbor in $a_i, \ldots, a_{|A|}$, and let $\hat{B}^{i\rightarrow}$ be the subsequence of $(b_1, \ldots, b_{\beta_i})$ of those vertices that have a neighbor in $a_{i+1}, \ldots, a_{|A|}$. At every event $p \in \{i, i{\rightarrow}\}$, we maintain the following invariants.

(i) We have a valid representation of the subgraph of G induced by the vertices $b_1, \ldots, b_{\beta_i}, a_1, \ldots, a_i$.

(ii) The x-coordinates of the foot points of $\{b_1, \ldots, b_{\beta_i}, a_1, \ldots, a_i\}$ are unique integers in the range from 1 to $\beta_i + i$.

(iii) For the vertices in $\{b_1, \ldots, b_{\beta_i}, a_1, \ldots, a_i\} \setminus \hat{B}^p$, both endpoints are set.

We initialize $\hat{B}^{0\rightarrow} = \hat{B}^0 = \emptyset$ and $\beta_0 = 0$. Here, our invariants trivially hold. Now suppose $i \geq 1$. In the following, we don't create a new variable $\hat{B}^p$ for each event p, but we update a single variable $\hat{B}$, viewing $\hat{B}^p$ as the state of $\hat{B}$ during event p.

Consider the enter event of a_i. We set the x-coordinate of a_i to $\beta_i + i$. We place the foot points of vertices $b_{\beta_{i-1}+1}, \ldots, b_{\beta_i}$ (if they exist) between a_{i-1} and a_i in this order and create $\hat{B}^i$ by appending them to $\hat{B}^{(i-1)\rightarrow}$ in this order. All neighbors of a_i have to start before a_i, and they have to be a suffix of $\hat{B}^i$. If this is not the case, then we simply reject as this is a negative instance of the problem. This is easily checked in $O(\deg(a_i))$ time. The upper endpoint of a_i is placed $1/2$ a unit above the foot point of its first neighbor in this suffix. As such, the invariants (i)–(iii) are maintained.

Consider the exit event of a_i. For each neighbor b_j of a_i, we check whether a_i is the last neighbor of b_j in σ_A. In this case, we finish b_j and set the x-coordinate of its right endpoint to $\beta_i + i + 1/2$. Now $\hat{B}^{i\rightarrow}$ consists of all vertices in $\hat{B}^i$ except those that we just finished. This again maintains invariants (i)–(iii). Note that processing the exit event always succeeds, i.e., negative instances are detected purely in the enter events.

Hence, if we reach and complete the exit event of $a_{|A|}$, we obtain a $\mathsf{STICK}_{\mathsf{AB}}$ representation of G. Otherwise, G has no such representation. Clearly, the whole algorithm runs in $O(|A| + |B| + |E|)$ time.

Note that, even though we have not explicitly discussed isolated vertices, these can be easily realized by sticks of length $1/2$. □

2.2 Data Structure: Semi-Ordered Trees

In the $\mathsf{STICK}_{\mathsf{A}}$ problem, the goal is to find a permutation of the horizontal sticks B that is consistent with the fixed permutation of the vertical sticks A. To this end, we will make use of a data structure that allows us to capture many permutations subject to consecutivity constraints. This might remind the reader of other similar but distinct data structures such as PQ-trees [1].

An *ordered tree* is a rooted tree where the order of the children around each internal is specified. The permutation *expressed* by an ordered tree T is the permutation of its leaves in the pre-order traversal of T.

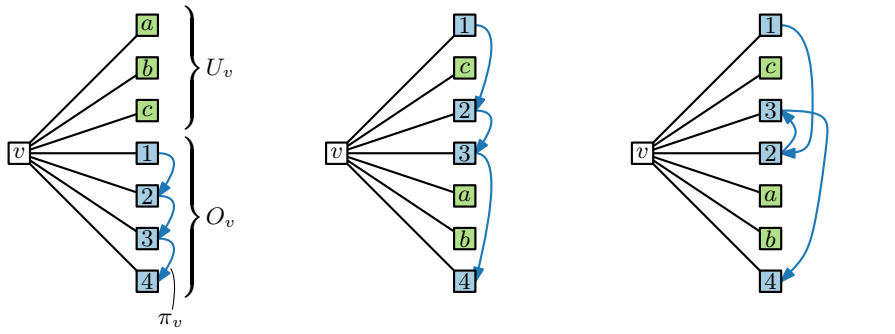
Generalizing this, we define a *semi-ordered tree* where, for each node, there is a fixed permutation for a subset of the children and the remaining children are free. Namely, for each node v, we have

(i) a set U_v of *unordered* children,

(ii) a set O_v of *ordered* children, and

(iii) a fixed permutation π_v of O_v; see Fig. 1.

Hence, every node (except the root) is ordered or unordered depending on its parent. We *obtain* an ordered tree from a semi-ordered tree by fixing, for each node v, a permutation π'_v of $O_v \cup U_v$ that contains π_v as a subsequence. In this



(a) a semi-ordered tree S

(b) an ordered tree obtained from S

(c) an ordered tree that cannot be obtained from S

Figure 1: Definition of a semi-ordered tree.

way, a permutation π is *expressed* by a semi-ordered tree S if there exists an ordered tree T that expresses π and can be obtained from S.

2.3 Solving $\mathsf{STICK}_\mathsf{A}$ in $O(|A| \cdot |B|)$ time

Let $G = (A \dot\cup B, E)$ and $\sigma_A = (a_1, \ldots, a_{|A|})$ be the input. We assume that G is connected and discuss otherwise at the end of this section.

As in the algorithm for $\mathsf{STICK}_\mathsf{AB}$, we apply a sweep-line approach (with a vertical sweep-line moving rightwards) where each vertical stick $a_i \in A$ corresponds to two events: the *enter event of* a_i (abbreviated by i) and the *exit event of* a_i (abbreviated by $i{\rightarrow}$).

Overview. Informally, for each event $p \in \{i, i{\rightarrow}\}$, we will maintain all representations of the subgraph seen so far subject to certain horizontal sticks continuing further (those that intersect the sweep-line and some vertical stick before it). We denote by G^p the induced subgraph of G containing $a_1, \ldots, a_i$ and their neighbors. We distinguish the neighbors as those that are *dead* (that is, have all neighbors before the sweep-line) and those that are *active* (that is, have neighbors before and after the sweep-line). Namely,

- B^p consists of all sticks of B in G^p;
- D^i consists of all (dead) sticks of B^i with no neighbor in $a_i, \ldots, a_{|A|}$; and
- $D^{i\rightarrow}$ consists of all (dead) sticks of $B^{i\rightarrow}$ with no neighbor in $a_{i+1}, \ldots, a_{|A|}$.

To this end, we maintain an ordered forest $\mathcal{T}^p$ of special semi-ordered trees that encodes all realizable permutations (defined below) of the set of horizontal sticks B^p as the permutations expressed by $\mathcal{T}^p$; see Fig. 2. A permutation π of B^p is *realizable* if there is a stick representation of the graph H^p obtained

from G^p by adding a vertical stick to the right of a_i neighboring all horizontal sticks in B^p where B^p is drawn top-to-bottom in order π.

In the enter event of a_i, B^i comprises $B^{i-1\rightarrow}$ and all vertices of B that neighbor a_i and aren't in the data structure yet (we call these *entering vertices*). We constrain the data structure so that all the neighbors of a_i must occur after (below) the non-neighbors of a_i. The set D^p of dead vertices remains unchanged with respect to the last exit event, that is, $D^i = D^{(i-1)\rightarrow}$.

In the exit event of a_i, $D^{i\rightarrow}$ comprises D^i and all sticks of B^i that do not have any neighbor a_j with $j > i$, i.e., those having a_i as their last neighbor (we call these *exiting vertices*). No new horizontal sticks appear in an exit event, hence $B^{i\rightarrow} = B^i$.

Data structure. See Fig. 2 for an example. Consider any event p. Observe that G^p may consist of several connected components $G^p_1, \ldots, G^p_{k_p}$. The components of G^p are naturally ordered from left to right by σ_A. Let B^p_j denote the vertices of B^p in G^p_j. In this case, in every realizable permutation of B^p, the vertices of B^p_j must come before the vertices of B^p_{j+1}. Furthermore, the vertices that will be introduced any time later can only be placed at the beginning, end, or between the components. Hence, to compactly encode the realizable permutations, it suffices to do so for each component G^p_j individually via a semi-ordered tree T^p_j. Namely, our data structure will be $\mathcal{T}^p = (T^p_1, \ldots, T^p_{k_p})$. Each data structure T^p_j is a special semi-ordered tree in which the leaves correspond to the vertices of B^p_j, all leaves are unordered, and all internal vertices are ordered.

Correctness and event processing. We argue by induction that this data structure is sufficient to express the realizable permutations of B^p. We maintain the following invariants for each event p during the execution of the algorithm.

(I1) The set of permutations expressed by $\mathcal{T}^p$ contains all permutations of B^p which occur in a stick representation of G.

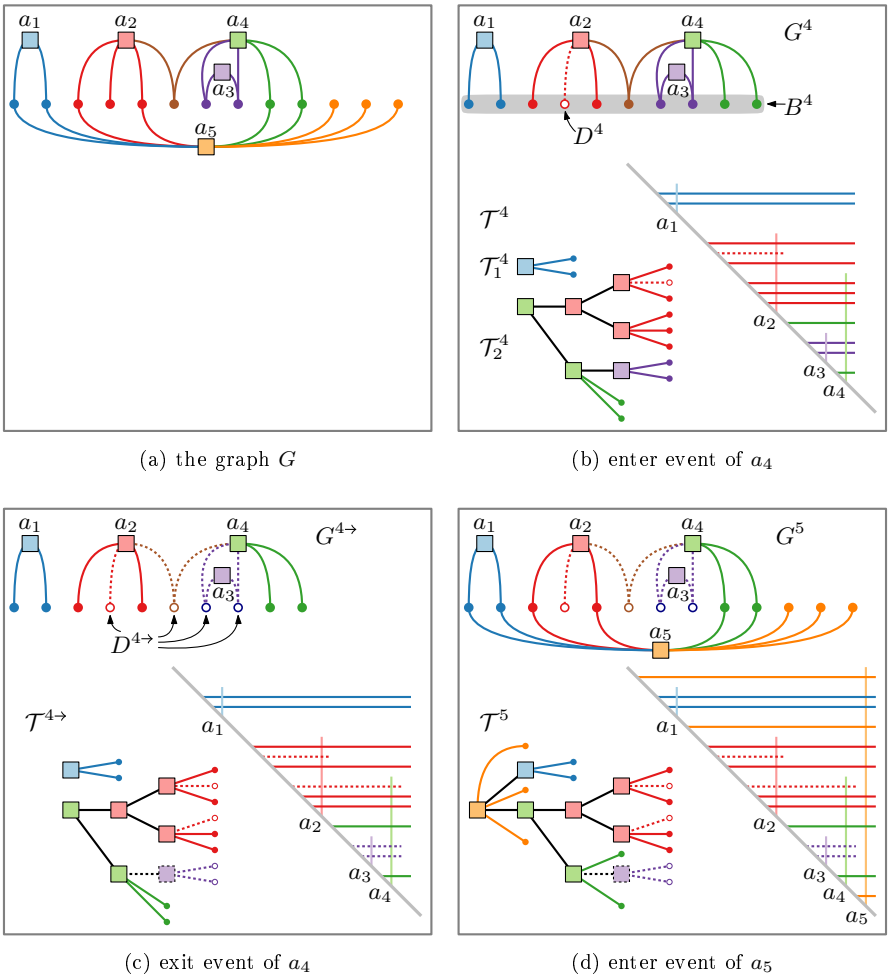
(I2) The set of permutations expressed by $\mathcal{T}^p$ contains only permutations of B^p which occur in a stick representation of G^p.

Since $G^{|A|\rightarrow} = G$ and $B^{|A|\rightarrow} = B$, after the final step these invariants ensure that our data structure expresses exactly those permutations of B which occur in a stick representation of G.

Recall that our data structure consists of an ordered set of semi-ordered trees. Note that these invariants also apply to each semi-ordered tree individually, that is, to its corresponding connected component.

In the base case, consider the enter event of a_1. Our data structure consists of a single component G^1_1 and clearly a single node with a leaf-child for every neighbor of a_1 captures all possible permutations.

In the exit event of a_i, we do not change the shape of $\mathcal{T}^i$, that is, $\mathcal{T}^{i\rightarrow} = \mathcal{T}^i$. Then, in $\mathcal{T}^{i\rightarrow}$, we mark the exiting vertices as dead and add them to $D^{i\rightarrow}$. We



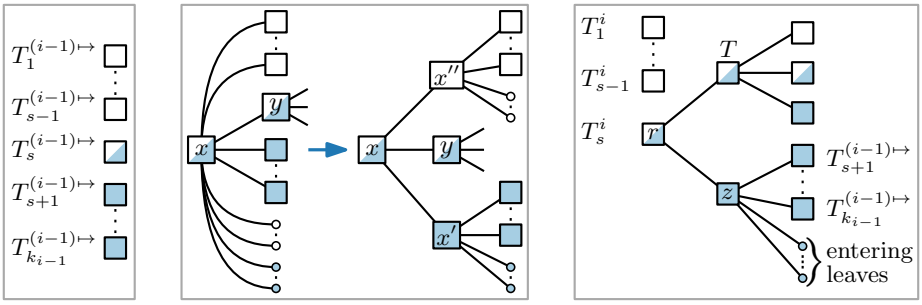
(a) the graph G

(b) enter event of a_4

(c) exit event of a_4

(d) enter event of a_5

Figure 2: An example for the data structures and a drawing corresponding to a realizable permutation of B^p for a graph G with $\pi_a = (a_1, \ldots, a_k)$. Vertices in A are drawn as squares; vertices in B are drawn as circles. The vertices and edges are colored by the entering event in which they appear. Dead vertices are drawn as empty circles; their edges are dotted. In (d), the leaves are permuted in the order in which they are drawn.



(a) $\mathcal{T}^{(i-1)\rightarrow}$ (b) transformation of T (c) $\mathcal{T}^i$

Figure 3: Construction of $\mathcal{T}^i$. Leaves are drawn as small circles. The leaves at the new node z are the entering vertices. Only active vertices are shown.

further mark any internal node in $\mathcal{T}^{i\rightarrow}$ that contains only dead leaves in its subtree as dead as well. Obviously, this procedure maintains all the invariants.

Now consider the enter event of a_i and assume that we have the data structure $\mathcal{T}^{(i-1)\rightarrow} = (T_1^{(i-1)\rightarrow}, \ldots, T_{k_{i-1}}^{(i-1)\rightarrow})$. The essential observation is that the neighbors of a_i must form a suffix of the active vertices in $B^{(i-1)\rightarrow}$ in every realizable permutation after the enter event, which we will enforce in the following. Namely, either

- all active vertices in $B^{(i-1)\rightarrow}$ are adjacent to a_i,
- none of them are adjacent to a_i, or
- there is an $s \in \{1, \ldots, k_{i-1}\}$ such that (i) $B_s^{(i-1)\rightarrow}$ contains at least one neighbor of a_i; (ii) all active vertices in $B_{s+1}^{(i-1)\rightarrow}, \ldots, B_{k_{i-1}}^{(i-1)\rightarrow}$ are neighbors of a_i; and (iii) no active vertices in $B_1^{(i-1)\rightarrow}, \ldots, B_{s-1}^{(i-1)\rightarrow}$ are adjacent to a_i; see Fig. 3a.

Otherwise, there is no realizable permutation for this event and consequently for G. The first two cases can be seen as degenerate cases (with $s = 0$ or $s = k_{i-1} + 1$) of the general case below.

We first show how to process $T_s^{(i-1)\rightarrow}$; see Fig. 3b. Afterwards we will create the data structure $\mathcal{T}^i$. We create a tree T that expresses precisely the subset of the permutations expressed by $T_s^{(i-1)\rightarrow}$ where all leaves that are neighbors of a_i occur as a suffix. We initialize $T = T_s^{(i-1)\rightarrow}$. If all active vertices in $B_s^{(i-1)\rightarrow}$ are neighbors of a_i, then we are already done.

Otherwise, we say that a node of T is *marked* if all active leaves in its subtree are neighbors of a_i; it is *unmarked* if no active leaf in its subtree is a neighbor of a_i; and it is *half-marked* otherwise. Note that the root of T is half-marked.

Since the neighbors of a_i must form a suffix of the active leaves, the marked non-leaf children of a half-marked node form a suffix of the active children, the unmarked non-leaf children form a prefix of the active children, and there is at

most one half-marked child. Hence, the half-marked nodes form a path in T that starts in the root; otherwise, there are no realizable permutations for this event and subsequently for G.

We traverse the path starting from the deepest descendant and ending at the root, i.e., bottom-to-top. Let x be a half-marked node, and let y be its half-marked child (if it exists); see Fig. 3b. We have to enforce that in any ordered tree obtained from T, the unmarked children of x occur before y and the marked children of x occur after y. We create a new marked vertex x'. This vertex receives the following children from x: the marked leaf-children and the suffix of the non-leaf children starting after y. Symmetrically, we create a new unmarked vertex x'', which receives the following children from x: the unmarked leaf-children and the prefix of the non-leaf children ending before y. Then we make x' and x'' children of x such that x'' is before y and y is before x'. If this results in any internal node having no leaf-children and only one child, we merge this node with its parent. (Note that this can only happen to x' or x''.) This ensures that for every permutation expressed by T, the the subsequence of active vertices has the neighbors of a_i as a suffix.

Note that every non-leaf of $T_s^{(i-1)\rightarrow}$ is also a non-leaf in T with the same set of leaves in its subtree. In the pre-order traversal of any ordered tree obtained from T, the non-leaves of $T_s^{(i-1)\rightarrow}$ are visited in the same order as in the pre-order traversal of any ordered tree obtained from $T_s^{(i-1)\rightarrow}$. This implies that each permutation expressed by T is also expressed by $T_s^{(i-1)\rightarrow}$. Moreover, invariant (I2) holds locally for T.

The marked leaf-children of any half-marked node x of $T_s^{(i-1)\rightarrow}$ can be placed anywhere before, between, or after its marked children, but not before y (since y has both marked and unmarked children). Symmetrically, the unmarked leaf-children of any half-marked node x of $T_s^{(i-1)\rightarrow}$ can be placed anywhere before, between, or after its unmarked children, but not after y. Hence, each permutation expressed by $T_s^{(i-1)\rightarrow}$ that has the neighbors of a_i as a suffix of the subsequence of its active vertices is also expressed by T. Moreover, invariant (I1) holds locally for T.

Thus, the permutations expressed by T are exactly those expressed by $T_s^{(i-1)\rightarrow}$ that have the neighbors of a_i as a suffix of their active subsequence.

Now, we create the data structure $\mathcal{T}^i$; see Fig. 3c. We set $T_1^i = T_1^{(i-1)\rightarrow}, \ldots, T_{s-1}^i = T_{s-1}^{(i-1)\rightarrow}$. Clearly, both invariants hold locally for $T_1^i, \ldots, T_{s-1}^i$. Next, we create a new semi-ordered tree T_s^i as follows. The tree T_s^i gets a new root r, to which we attach T and a new vertex z, in this order. Then we hang $T_{s+1}^{(i-1)\rightarrow}, \ldots, T_{k_{i-1}}^{(i-1)\rightarrow}$ from z in this order. We further make the entering vertices leaf-children of z. Note that this allows them to mix freely before, after, or between the components $G_{s+1}^{(i-1)\rightarrow}, \ldots, G_{k_{i-1}}^{(i-1)\rightarrow}$. Finally, we set $\mathcal{T}^i = (T_1^i, \ldots, T_s^i)$.

In this way, the order of the components $G_1^{(i-1)\rightarrow}, \ldots, G_{k_{i-1}}^{(i-1)\rightarrow}$ of $G^{(i-1)\rightarrow}$ is maintained in the data structures for G^i. In T_s^i, both invariants clearly hold for the non-leaf children of z and, as argued above, also for T. Furthermore, we

ensure that the entering vertices can be placed exactly before, after, or between the components of $G^{(i-1)\rightarrow}$ that are completely adjacent to a_i. Hence, both invariants hold for $\mathcal{T}^i$.

The decision problem of $\mathsf{STICK}_\mathsf{A}$ can easily be solved by this algorithm. Namely, by our invariants, any permutation σ_B expressed by $\mathcal{T}^{|A|\rightarrow}$ occurs as a permutation of the horizontal sticks in a $\mathsf{STICK}_\mathsf{A}$ representation of G. Thus, executing our algorithm for $\mathsf{STICK}_\mathsf{AB}$ on σ_A and σ_B gives us a stick representation of G. This concludes the proof of correctness for the connected case.

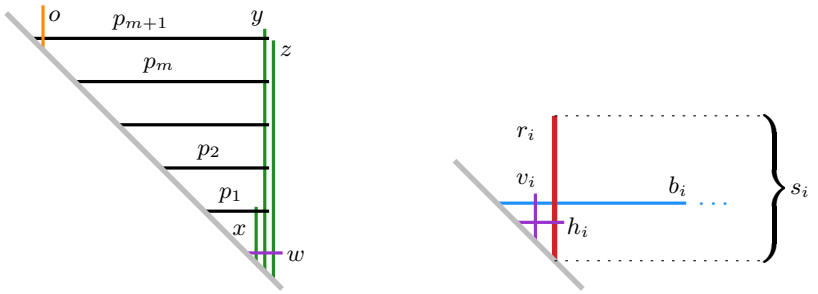
Disconnected graphs. To handle disconnected graphs, we first identify the connected components $H_1, \ldots, H_t$ of G. We label each element of A by the index of the component to which it belongs. Now, observe that if σ_A contains a pattern of indices a and a' that alternate as in $aa'aa'$, then the given $\mathsf{STICK}_\mathsf{A}$ instance does not admit a solution. Otherwise, we can treat each component separately by our algorithm, and then nest the resulting representations. Namely, for each connected component H_r, we run our $\mathsf{STICK}_\mathsf{A}$ algorithm (on σ_A restricted to H_r) to obtain a realizable permutation σ_{B_r} of the horizontal sticks of H_r. Now, since our connected components avoid the pattern $aa'aa'$, there is natural hierarchy of these components which we can use to obtain a total order σ_B on the horizontal sticks of G from the permutations $\sigma_{B_1}, \ldots, \sigma_{B_t}$. Finally, we can use this σ_B, the given σ_A, and G as an input to our $\mathsf{STICK}_\mathsf{AB}$ algorithm to obtain a representation.

Running time. The size of each data structure $\mathcal{T}^p$ is in $O(|B^p|)$ since there are no degree-2 vertices in the trees and each leaf corresponds to a vertex in B^p. In each event, the transformations can clearly be done in time proportional to the size of the data structures. Since $|B^p| \leq |B|$ for each p and there are $2|A|$ events, we get the following running time.

Theorem 2.2 *$\mathsf{STICK}_\mathsf{A}$ can be solved in $O(|A| \cdot |B|)$ time.*

3 Sticks of Fixed Lengths

In this section, we consider the case that, for each vertex of the input graph, its stick length is part of the input and fixed. We denote the variants of this problem by $\mathsf{STICK}^{\mathsf{fix}}$ if the order of the sticks is not restricted, by $\mathsf{STICK}^{\mathsf{fix}}_\mathsf{A}$ if σ_A is given, and by $\mathsf{STICK}^{\mathsf{fix}}_\mathsf{AB}$ if σ_A and σ_B are given. Unlike in the case with variable stick lengths, all three new variants are NP-hard; see Sections 3.1 and 3.2. Surprisingly, $\mathsf{STICK}^{\mathsf{fix}}_\mathsf{AB}$ can be solved efficiently by a simple linear program if the input graph contains no isolated vertices (i.e., vertices of degree 0); see Section 3.3. With our linear program, we can check the feasibility of any instance of $\mathsf{STICK}^{\mathsf{fix}}$ if we are given a total order of the sticks on the ground line. With our NP-hardness results, this implies NP-completeness of $\mathsf{STICK}^{\mathsf{fix}}$, $\mathsf{STICK}^{\mathsf{fix}}_\mathsf{A}$, and $\mathsf{STICK}^{\mathsf{fix}}_\mathsf{AB}$.



(a) frame providing the pockets

(b) number gadget for the number s_i

Figure 4: Gadgets of our reduction from 3-PARTITION to STICK$^{\text{fix}}$.

3.1 STICK$^{\text{fix}}$

We show that STICK$^{\text{fix}}$ is NP-hard by reduction from 3-PARTITION, which is strongly NP-hard [12]. In 3-PARTITION, one is given a multiset S of $3m$ integers $s_1, \ldots, s_{3m}$ such that, for $i \in \{1, \ldots, 3m\}$, $C/4 < s_i < C/2$, where $C = (\sum_{i=1}^{3m} s_i)/m$, and the task is to decide whether S can be split into m sets of three integers, each summing up to C.

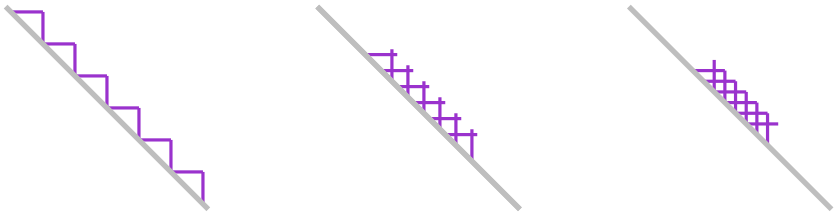
Theorem 3.1 *STICK$^{\text{fix}}$ is NP-complete.*

Proof: As mentioned at the beginning of this section, NP-membership follows from our linear program (see Theorem 3.7 in Section 3.3) to test the feasibility of any instance of STICK$^{\text{fix}}$ when given a total order of the sticks on the ground line.

To show NP-hardness, we describe a polynomial-time reduction from 3-PARTITION to STICK$^{\text{fix}}$. Given a 3-PARTITION instance $I = (S, C, m)$, we construct a fixed cage-like frame structure and introduce a number gadget for each number of S. A sketch of the frame is given in Fig. 4a. The purpose of the frame is to provide pockets, which will host our number gadgets (defined below). We add two long vertical (green) sticks y and z of length $mC + 1 + 2\varepsilon$ and a shorter vertical (green) stick x of length 1 that are all kept together by a short horizontal (violet) stick w of some length $\varepsilon \ll 1$. We use $m+1$ horizontal (black) sticks $p_1, p_2, \ldots, p_{m+1}$ to separate the pockets. Each of them intersects y but not z and has a specific length such that the distance between two of these sticks is $C \pm \varepsilon$.

Additionally, p_1 intersects x and p_{m+1} intersects a vertical (orange) stick o of length $2C$. We use x and o to prevent the number gadgets from being placed below the bottommost and above the topmost pocket, respectively. It does not matter on which side of y the stick x ends up since each b_i of a number gadget intersects y but neither x nor z.

For each number s_i in S, we construct a number gadget; see Fig. 4b. We introduce a vertical (red) stick r_i of length s_i. Intersecting r_i, we add a horizontal (blue) stick b_i of length at least $mC + 2$. The stick b_i intersects y and z,



(a) stretched (b) medium (c) compressed

Figure 5: Three stick representations of an ε-path with twelve sticks.

but neither x nor o. Due to these adjacencies, every number gadget can only be placed in one of the m pockets defined by $p_1, \ldots, p_{m+1}$. It cannot span multiple pockets. We require that r_i and b_i intersect each other close to their foot points, so we introduce two short (violet) sticks h_i and v_i—one horizontal, the other vertical—of lengths ε; they intersect each other, h_i intersects r_i, and v_i intersects b_i.

Given a yes-instance $I = (S, C, m)$ and a valid 3-partition P of S, the graph obtained by our reduction is realizable. Construct the frame as described before and place the number gadgets into the pockets according to P. Since the lengths of the three number gadgets' r_i sum up to $C \pm 3\varepsilon$, all three can be placed into one pocket. After distributing all number gadgets, we have a stick representation.

Given a stick representation of a graph G obtained from our reduction, we can obtain a valid solution of the corresponding 3-PARTITION instance $I = (S, C, m)$ as follows. Clearly, the shape of the frame is fixed, creating m pockets. Since the sticks $b_1, \ldots, b_{3m}$ are incident to y and z but neither to x nor to o, they can end up inside any of the pockets. In the y-dimension, each two number gadgets of numbers s_k and s_ℓ overlap at most on a section of length ε; otherwise r_k and b_ℓ or r_ℓ and b_k would intersect. Each pocket hosts precisely three number gadgets: we have $3m$ number gadgets, m pockets, and no pocket can contain four (or more) number gadgets; otherwise, there would be a number gadget of height at most $(C + \varepsilon)/4 + 2\varepsilon$, contradicting the fact that s_i is an integer with $s_i > C/4$. In each pocket, the height of the number gadgets would be too large if the three corresponding numbers of S would sum up to $C + 1$ or more. Thus, the assignment of number gadgets to pockets defines a valid 3-partition of S. □

The sticks of lengths $s_1, \ldots, s_{3m}$ can be simulated by paths of sticks with length ε each. We refer to them as *ε-paths*. Employing them in our reduction, it suffices to use only three distinct stick lengths. Like a spring, an ε-path can be stretched (Fig. 5a) and compressed (Fig. 5c) up to a specific length. We will exploit the following properties regarding the minimum and the maximum size of an ε-path.

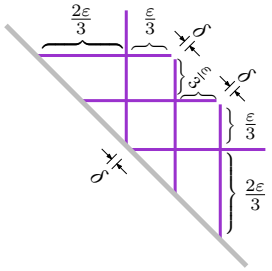


Figure 6: Construction of a compressed stick representation of an ε-path

Lemma 1 *There is a stick representation of a $2n$-vertex ε-path with height and width $n\varepsilon$ and another stick representation with height and width $\frac{n+2}{3}\varepsilon + \delta$ for any $\delta > 0$ and $n \geq 3$. Any stick representation of a $2n$-vertex ε-path has height and width in the range $\left(\frac{n}{3}, n\right]\varepsilon$.*

Proof: We can arrange our sticks such that the foot points or the end points of two adjacent sticks touch each other (see Fig. 5a). This construction has height and width $n\varepsilon$ and, clearly, this is the maximum width and height for a $2n$-vertex ε-path.

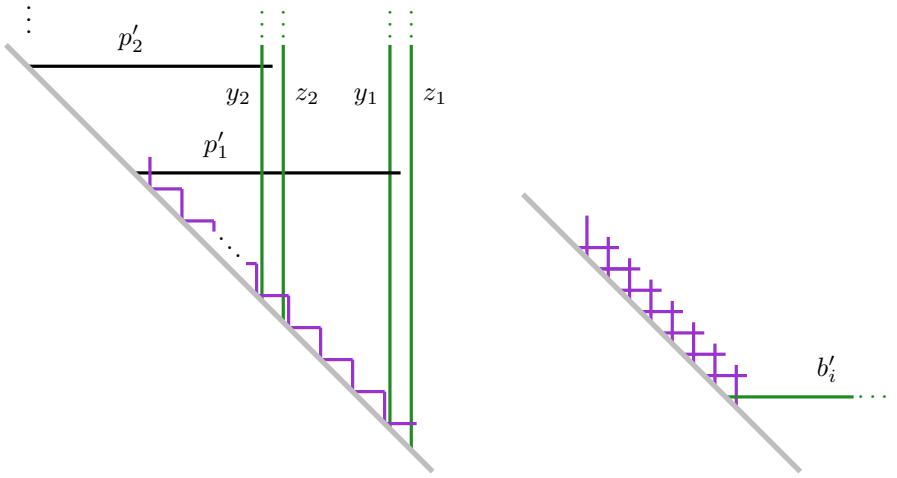
For the compressed ε-paths, we first describe a construction that has the specified width and height and, second, we show the lower bound.

The following construction is depicted in Fig. 6 for $n = 3$. Set the foot point of the first vertical stick in the path to $y = 0$ and the foot point of the third stick, which is also vertical, to $y = \varepsilon/3$. For each $i \in \{2, \ldots, n-1\}$, set the foot point of the $(2i-2)$-th stick (horizontal) to $y = i\varepsilon/3 + (i-2)\delta/(n-2)$ and the foot point of the $(2i+1)$-th stick (vertical) to $y = i\varepsilon/3 + (i-1)\delta/(n-2)$. Set the foot point of the $(n-2)$-th stick to $y = n\varepsilon/3 + \delta$, and the foot point of the last stick to $y = (n+1)\varepsilon/3 + \delta$. Observe that this construction has width and height $\frac{n+2}{3}\varepsilon + \delta$ and is a valid stick representation of a $2n$-vertex ε-path.

Consider the i-th stick of an ε-path. On the one side of the line through this stick, there is the $(i-3)$-th stick, and on the other, there is the $(i+3)$-th stick. E.g., the second stick is to the right of the fifth stick and the eighth stick is to the left of the fifth stick. Since all sticks have length ε and non-adjacent sticks are not allowed to touch each other, the 1st, 4th, 7th, ..., $(6k-2)$-th stick for $k \in \mathbb{N}$ form a zigzag chain of width and height strictly greater than $k\varepsilon$. The same holds for the 2nd, 5th, ... stick and the 3rd, 6th, ... stick. Thus, for an ε-path of $2n$ sticks, we have width and height strictly greater than $\left\lceil \frac{2n}{6} \right\rceil \varepsilon \geq \frac{n}{3}\varepsilon$. □

Corollary 3.2 *STICK$^{\text{fix}}$ with only three different stick lengths is NP-complete.*

Proof: We modify the reduction from 3-PARTITION to STICK$^{\text{fix}}$ described in the proof of Theorem 3.1 such that we use only three distinct stick lengths. We use the three lengths ε, Cm, and $3Cm$ (or longer, e.g. ∞). In Fig. 7, sticks of these lengths are violet, black, and green, respectively.



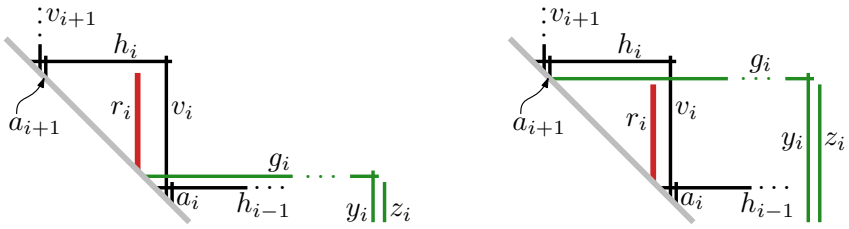
(a) frame providing the pockets (b) number gadget for the number s_i

Figure 7: Gadgets of our reduction from 3-PARTITION to STICK$^{\text{fix}}$ with three stick lengths.

First, we describe the modifications of the frame structure, which are also depicted in Fig. 7a. Instead of the vertical (green) sticks x, y, and z used to fix all pockets, we have two vertical sticks y_j and z_j of length $3Cm$ for $j \in \{1, \dots, m+1\}$. Instead of the sticks $p_1, \dots, p_{m+1}$ of different lengths, we use horizontal (black) sticks $p'_1, \dots, p'_{m+1}$ each with length Cm to separate the pockets. The stick p'_j intersects y_k, z_k for all $k \in \{j+1, \dots, m+1\}$ and y_j but not z_j. All pairs (y_j, z_j) are kept together by a stick of length ε. For each two neighboring pairs (y_j, z_j) and (y_{j+1}, z_{j+1}), these sticks of length ε are connected by an ε-path of $2C/\varepsilon$ sticks. According to Lemma 1, this effects a maximum distance of $(C/\varepsilon) \cdot \varepsilon \pm \varepsilon = C \pm \varepsilon$ between each two pairs of (y_j, z_j) and (y_{j+1}, z_{j+1}). Accordingly, the pockets separated by the sticks $p'_1, \dots, p'_{m+1}$ have height at most $C \pm 2\varepsilon$, similar as in the proof of Theorem 3.1. We keep the vertical (orange) stick o as in Fig. 4a to prevent number gadgets from being placed above the topmost pocket, but now o has length $3Cm$.

Second, we describe the modifications of the number gadgets for each number s_i for $i \in \{1, \dots, 3m\}$, which are also depicted in Fig. 7b. We keep a long stick b'_i similar to b_i—now with length $3Cm$. We replace r_i (together with h_i and v_i) by an ε-path of $6s_i/\varepsilon - 4$ sticks. We make the first stick of the ε-path intersect b'_i. By Lemma 1, this ε-path has a stick representation with height $s_i + \delta$ for any $\delta > 0$, but there is no stick representation with height $s_i - \frac{2}{3}\varepsilon$ or smaller. Clearly, these number gadgets can only be placed into one pocket since none of their sticks intersects a p'_j for $j \in \{1, \dots, m+1\}$.

Hence, we can represent a yes-instance of 3-PARTITION as such a stick graph if and only if the ε-paths of the number gadgets are (almost) as much compressed as possible (to make the number gadgets small enough) and the



(a) variable gadget set to false (b) variable gadget set to true

Figure 8: Variable gadget in our reduction from MONOTONE-3-SAT to $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$.

ε-paths between the (y_j, z_j)-sticks are (almost) as much stretched as possible (to make the pockets tall enough). Using this, the proof is the same as in Theorem 3.1. □

3.2 $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$ and $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$

We show that $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$ and $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ are NP-hard by reducing from MONOTONE-3-SAT, which is NP-hard [21]. In MONOTONE-3-SAT, one is given a Boolean formula Φ in CNF where each clause contains three distinct literals that are all positive or all negative. The task is to decide whether Φ is satisfiable.

Theorem 3.3 *$\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$ is NP-complete.*

Proof: Recall that, as mentioned before, NP-membership follows from our linear program (see Theorem 3.7 in Section 3.3) to test the feasibility of any instance of $\mathsf{STICK}^{\mathsf{fix}}$ when given a total order of the sticks on the ground line. To show NP-hardness, we describe a polynomial-time reduction from MONOTONE-3-SAT to $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$. A schematization of our reduction is depicted in Figs. 8, 9, and 10.

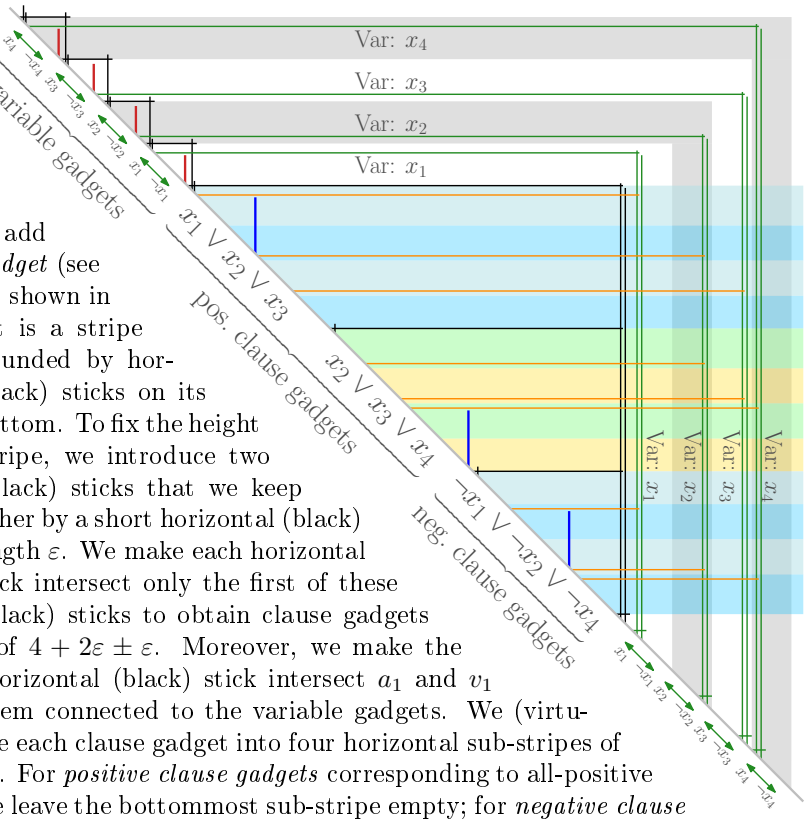
Given a MONOTONE-3-SAT instance Φ over variables $x_1, \ldots, x_n$, we construct a *variable gadget* for each variable as depicted in Fig. 8. Inside a (black) *cage*, there is a vertical (red) stick r_i with length 1 and from the inside a long horizontal (green) stick g_i leaves this cage. We can enforce the structure to be as in Fig. 8 as follows. We prescribe the order σ_A of the vertical sticks as in Fig. 8. Since a_{i+1} has length $\varepsilon \ll 1$, the horizontal (black) stick h_i intersects the two vertical (black) sticks v_{i+1} and a_{i+1} close to its foot point. Since we have prescribed $\sigma_A(a_{i+1}) < \sigma_A(r_i) < \sigma_A(v_i)$, we see that r_i is inside the cage bounded by h_i and v_i and fixes its height—as it does not intersect h_i—making the sticks h_i and v_i intersect close to their end points (both have length $1 + 2\varepsilon$). Moreover, r_i cannot be below h_{i-1} because a_i is shorter than r_i and intersects h_{i-1} to the right of r_i. This leaves the freedom of placing g_i above or below r_i (as g_i does not intersect r_i) but still with its foot point inside the cage formed by h_i and v_i because it intersects v_i, but neither v_{i-1} nor v_{i+1}.

We say that the variable x_i is set to false if the foot point of g_i is below the foot point of r_i, and true otherwise. For each x_i, we add two long vertical (green)

sticks y_i and z_i that we keep close together by using a short horizontal (violet) stick of length ε (see Fig. 9 on the bottom right). We make g_i intersect y_i but not z_i. The three sticks g_i, y_i, and z_i get the same length ℓ_i. Hence, y_i and g_i intersect each other close to their end points as otherwise g_i would intersect z_i. We choose ℓ_1 sufficiently large such that the foot point of y_1 is to the right of the clause gadgets (see Fig. 9) and for each ℓ_i with $i \geq 2$, we set $\ell_i = \ell_{i-1} + 1 + 3\varepsilon$.

Now compare the end points of g_i when x_i is set to false and when x_i is set to true relative to the (black) cage structure. When x_i is set to true, the end point of g_i is $1 \pm 2\varepsilon$ above and $1 \pm 2\varepsilon$ to the left compared to the case when x_i is set to false. Observe that, since g_i and y_i intersect each other close to their end points, this offset is also pushed to y_i and z_i and their foot points. Consequently, the position of the foot point of y_i (and z_i) differs by $1 \pm 2\varepsilon$ relative to the (black) frame structure depending on whether x_i is set to true or false. Our choice of ℓ_i allows this movement. In other words, no matter which truth value we assign to each x_i, there is a stick representation of the variable gadgets respecting σ_A.

Figure 9: Illustration of our reduction from MONOTONE-3-SAT to $\mathsf{STICK}_{\mathsf{A}}^{\mathsf{fix}}$



For each clause, we add a *clause gadget* (see Fig. 10) as shown in Fig. 9. It is a stripe that is bounded by horizontal (black) sticks on its top and bottom. To fix the height of each stripe, we introduce two vertical (black) sticks that we keep close together by a short horizontal (black) stick of length ε. We make each horizontal (black) stick intersect only the first of these vertical (black) sticks to obtain clause gadgets of height of $4 + 2\varepsilon \pm \varepsilon$. Moreover, we make the topmost horizontal (black) stick intersect a_1 and v_1 to keep them connected to the variable gadgets. We (virtually) divide each clause gadget into four horizontal sub-stripes of height ≥ 1. For *positive clause gadgets* corresponding to all-positive clauses, we leave the bottommost sub-stripe empty; for *negative clause gadgets* corresponding to all-negative clauses, we leave the topmost sub-stripe empty. We add three horizontal (orange) sticks—one per remaining horizontal sub-stripe—and assign them bijectively to the variables of the clause. We make

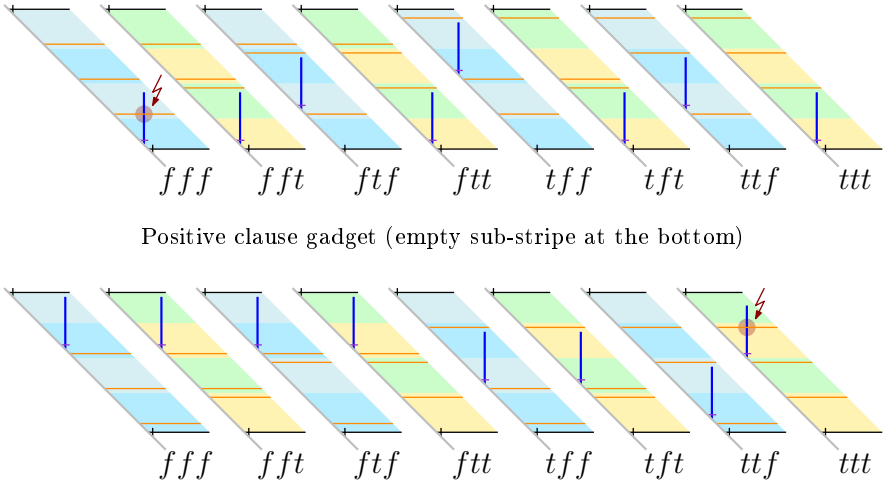


Figure 10: Clause gadget in our reduction from MONOTONE-3-SAT to $\mathsf{STICK}_{\mathsf{A}}^{\mathsf{fix}}$. Here, a clause gadget for each of the eight possible truth assignments of a MONOTONE-3-SAT clause is depicted. In particular, it shows that the (blue) isolated vertical stick fits inside the gadget if and only if the corresponding clause is satisfied. E.g., tft means that the first variable is set to true, the second to false, and the third to true.

each horizontal (orange) stick o that is assigned to x_i intersect y_i and all y_j and z_j for $j < i$, but not z_i or y_k or z_k for any $k > i$. Thus, o intersects y_i close to o's end point. We choose the length of each such o so that its foot point is at the bottom of its sub-stripe if x_i is set to false or is at the top of its sub-stripe if x_i is set to true. Within the positive and the negative clause gadgets, this gives us two times eight possible configurations of the orange sticks depending on the truth assignment of the three variables of the clause (see Fig. 10). Within each clause gadget, we have a vertical (blue) stick b of length 2, which does not intersect any other stick. Each horizontal (black) stick that bounds a clause gadget intersects a short vertical (black) stick of length ε to force b into its designated clause gadget.

Clearly, if Φ is satisfiable, there is a stick representation of the $\mathsf{STICK}_{\mathsf{A}}^{\mathsf{fix}}$ instance obtained from Φ by our reduction by placing the sticks as described before (see also Fig. 9). In particular, each blue stick can be placed in one of the ways depicted in Fig. 10.

On the other hand, if there is a stick representation of the $\mathsf{STICK}_{\mathsf{A}}^{\mathsf{fix}}$ instance obtained by our reduction, Φ is satisfiable. As argued before, the shape of the (black) frame structure of all gadgets is fixed by the choice of the adjacencies and lengths in the graph and σ_A. The only flexibility is, for each $i \in \{1, \dots n\}$, whether g_i has its foot point above or below r_i. This enforces one of eight distinct configurations per clause gadget. As depicted in Fig. 10, precisely the configurations that correspond to satisfying truth assignments are realizable.

Thus, we can read a satisfying truth assignment of Φ from the variable gadgets.

Our reduction can obviously be implemented in polynomial time. □

In our reduction, we enforce an order of the horizontal sticks. So, prescribing σ_B makes it even easier to enforce this structure. Hence, we can use exactly the same reduction for $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$.

Corollary 3.4 *$\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ (with isolated vertices in A or B) is NP-complete.*

Proof: Given a MONOTONE-3-SAT instance Φ, consider the construction described in the proof of Theorem 3.3. We use the same graph, the same stick lengths and the same ordering σ_A of the vertical sticks. Now, we additionally prescribe the order of the remaining horizontal sticks as depicted in Fig. 9 via σ_B. This defines an instance of $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$.

Clearly, the ordering of the horizontal sticks σ_B neither affects the placement of the vertical isolated (red) sticks inside a variable gadget nor does it affect the placement of the vertical isolated (blue) sticks inside a clause gadget. Moreover, there was only one possible ordering of the horizontal sticks in the construction described in the proof of Theorem 3.3. Thus, its correctness proof applies here as well. □

The reduction we described before uses isolated vertices inside the variable and the clause gadgets. In the case of $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$, this is indeed necessary to show NP-hardness. This is not true for $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$, which remains NP-hard (and hence is NP-complete due to our linear program) even without isolated sticks.

Corollary 3.5 *$\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$ without isolated vertices is NP-complete.*

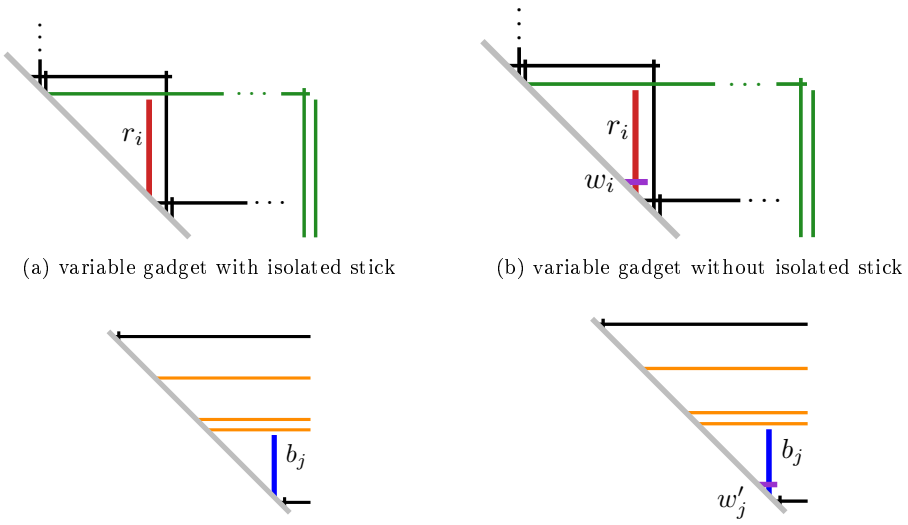
Proof: We use the same reduction as in the proof of Theorem 3.3, but we additionally add, for each isolated stick, a short stick of length $\varepsilon \ll 1$ that only intersects the isolated stick; see Fig. 11. In each variable gadget, for the isolated vertical (red) stick r_i, we add a short horizontal (violet) stick w_i of length ε. Similarly, in each clause gadget, for the isolated vertical (blue) stick b_j, we add a short horizontal (violet) stick w'_j of length ε. After these additions, no isolated sticks remain.

Observe that, for any placement of the isolated (red and blue) sticks inside their gadgets in the proof of Theorem 3.3, we can add the new horizontal (violet) stick since it has length only ε. Moreover, since these new (violet) sticks are horizontal, we do not get any new ordering constraints in the version $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{A}}$.

We therefore conclude that the rest of the proof still holds. □

3.3 $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ without isolated vertices

In this section, we constructively show that $\mathsf{STICK}^{\mathsf{fix}}$ is efficiently solvable if we are given a total order of the vertices in $A \cup B$ on the ground line. Note that if there is a stick representation for an instance of $\mathsf{STICK}_{\mathsf{AB}}$ (and consequently also $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$), the combinatorial order of the sticks on the ground line is always the same except for isolated vertices, which we formalize in the following lemma. The proof also follows implicitly from the proof of Theorem 2.1.



(a) variable gadget with isolated stick

(b) variable gadget without isolated stick

(c) clause gadget with isolated stick

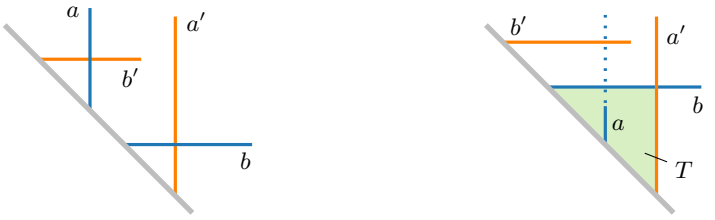
(d) clause gadget without isolated stick

Figure 11: We add a short horizontal (violet) stick w_i (or w'_j) that intersects r_i (or b_j) to avoid isolated sticks in a variable (or clause) gadget for variable x_i (or clause C_j).

Lemma 3.6 *In all stick representations of an instance of* $\mathsf{STICK}_{\mathsf{AB}}$*, the order of the vertices* $A \cup B$ *on the ground line is the same after removing all isolated vertices. This order can be found in time* $O(|E|)$.

Proof: Assume there are stick representations Γ_1 and Γ_2 of the same instance of $\mathsf{STICK}_{\mathsf{AB}}$ without isolated vertices that have different combinatorial arrangements on the ground line. Without loss of generality, there is a pair of sticks $(a, b) \in A \times B$ such that in Γ_1, a comes before b, while in Γ_2, a comes after b (see Fig. 12). Clearly, a and b cannot be adjacent. Since a is not isolated, there is a b' that is adjacent to a and comes before b. Analogously, there is an a' that is adjacent to b and comes after a. In Γ_2, stick b, stick a', and the ground line define a triangular region T (see Fig. 12b), which completely contains a since a occurs between b and a', but is adjacent to neither of them. However, b' is outside of T as it comes before b. This contradicts b and a' being adjacent. The unique order can be determined in $O(|E|)$ time as described in Section 2. □

We are given an instance of $\mathsf{STICK}^{\mathsf{fix}}$ and a total order $v_1, \ldots, v_n$ of the vertices ($n = |A| + |B|$) with stick lengths $\ell_1, \ldots, \ell_n$. We create a system of difference constraints, that is, a linear program $Ax \leq b$ where each constraint is a simple linear inequality of the form $x_j - x_i \leq b_k$, with n variables and $m \leq 3n - 1$ constraints. Such a system can be modeled as a weighted graph with a vertex per variable x_i and a directed edge (x_i, x_j) with weight b_k per constraint. The system has a solution if and only if there is no directed cycle of negative weight. In this case, a solution can be found in $O(nm)$ time using the Bellman–Ford algorithm.



(a) a comes before b (valid representation) (b) a comes after b (no representation)

Figure 12: Trying to represent a subgraph of the edges ab' and $a'b$ while respecting σ_A and σ_B.

In the following, we describe how to construct such a linear program for $\mathsf{STICK}^{\mathsf{fix}}$. For each stick v_i, we create a variable x_i that corresponds to the x-coordinate of v_i's foot point on the ground line, with $x_1 = 0$. To ensure the unique order, we add $n-1$ constraints $x_{i+1} - x_i \leq -\varepsilon$ for some suitably small ε and $i = 1, \ldots, n-1$.

Let $v_i \in A$ and $v_j \in B$. If $(v_i, v_j) \in E$, then the corresponding sticks have to intersect, which they do if and only if $x_j - x_i \leq \min\{\ell_i, \ell_j\}$. If $i < j$ and $(v_i, v_j) \notin E$, then the corresponding sticks must not intersect, so we require $x_j - x_i > \min\{\ell_i, \ell_j\} \geq \min\{\ell_i, \ell_j\} + \varepsilon$. This easily gives a system of difference constraints with $O(n^2)$ constraints. We argue that a linear number suffices.

Let $v_i \in A$. Let j be the largest j such that $(v_i, v_j) \in E$ and $\ell_j \geq \ell_i$. We add a constraint $x_j - x_i \leq \ell_i$. Further, let k be the smallest k such that $(v_i, v_k) \notin E$ and $\ell_k \geq \ell_i$. We add a constraint $x_k - x_i > \ell_i \Leftrightarrow x_i - x_k \leq -\ell_i - \varepsilon$. Symmetrically, let $v_i \in B$. Let j be the smallest j such that $(v_j, v_i) \in E$ and $\ell_j > \ell_i$. We add a constraint $x_i - x_j \leq \ell_i$. Further, let k be the largest k such that $(v_k, v_i) \notin E$ and $\ell_k > \ell_i$. We add a constraint $x_i - x_k > \ell_i \Leftrightarrow x_k - x_i \leq -\ell_i - \varepsilon$.

We now argue that these constraints are sufficient to ensure that G is represented by a solution of the system. Let $v_i \in A$ and $v_j \in B$. If $i > j$, then the corresponding sticks cannot intersect, which is ensured by the fixed order. So assume that $i < j$. If $\ell_j \geq \ell_i$ and $(v_i, v_j) \in E$, then we either have the constraint $x_j - x_i \leq \ell_i$, or we have a constraint $x_k - x_i \leq \ell_i$ with $i < j < k$; together with the order constraints, this ensure that $x_j - x_i \leq x_k - x_i \leq \ell_i$. If $\ell_j \geq \ell_i$ and $(v_i, v_j) \notin E$, then we either have the constraint $x_i - x_j \leq -\ell_i - \varepsilon$, or we have a constraint $x_i - x_k \leq -\ell_i - \varepsilon$ with $i < k < j$; together with the order constraints, this ensure that $x_i - x_j \leq x_i - x_k \leq -\ell_i - \varepsilon$. Symmetrically, the constraints are also sufficient for $\ell_j < \ell_i$. We obtain a system of difference constraints with n variables and at most $3n - 1$ constraints proving Theorem 3.7. By Lemma 3.6, there is at most one realizable order of vertices for a $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ instance without isolated vertices, which can be found in linear time and proves Corollary 3.8.

Theorem 3.7 *$\mathsf{STICK}^{\mathsf{fix}}$ can be solved in $O((|A| + |B|)^2)$ time if we are given a total order of the vertices.*

Corollary 3.8 *$\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ can be solved in $O((|A| + |B|)^2)$ time if there are no isolated vertices.*

4 Open Problems

We have shown that $\mathsf{STICK}^{\mathsf{fix}}$ is NP-complete even if the sticks have only three different lengths, while $\mathsf{STICK}^{\mathsf{fix}}$ for unit-length sticks is solvable in linear time. But what is the computational complexity of $\mathsf{STICK}^{\mathsf{fix}}$ for sticks with one of *two* lengths? Also, the three different lengths in our proof depend on the number of sticks. Is $\mathsf{STICK}^{\mathsf{fix}}$ still NP-complete if the fixed lengths are bounded?

We have shown that $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ is NP-complete if there are isolated vertices (in at least one of the bipartitions). In our NP-hardness reduction we use a linear number of isolated vertices. Clearly, $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ is in XP in the number n_{isolated} of isolated vertices. An XP-algorithm could first compute the unique ordering of the non-isolated vertices and then try to insert each isolated vertex at each possible position in the permutation brute-force. However, the question remains open whether $\mathsf{STICK}^{\mathsf{fix}}_{\mathsf{AB}}$ is fixed-parameter tractable (FPT) in n_{isolated} [5].

Additionally, the complexity of the original problem STICK, i.e., recognizing a stick graph without vertex order or stick lengths, is still open.

Acknowledgments

We thank Anna Lubiw for detailed information about our most relevant reference [10] and for her constructive remarks regarding an earlier version of this paper.

[5] This question has been asked by Paweł Rzążewski at the 27th International Symposium on Graph Drawing and Network Visualization 2019 (GD'19) in Prague.

References

[1] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

[2] S. Cabello and M. Jejčič. Refining the hierarchies of classes of geometric intersection graphs. *Electr. J. Comb.*, 24(1):P1.33, 2017. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/v24i1p33.

[3] J. Cardinal, S. Felsner, T. Miltzow, C. Tompkins, and B. Vogtenhuber. Intersection graphs of rays and grounded segments. *J. Graph Algorithms Appl.*, 22(2):273–295, 2018. doi:10.7155/jgaa.00470.

[4] D. Catanzaro, S. Chaplick, S. Felsner, B. V. Halldórsson, M. M. Halldórsson, T. Hixon, and J. Stacho. Max point-tolerance graphs. *Discrete Appl. Math.*, 216:84–97, 2017. doi:10.1016/j.dam.2015.08.019.

[5] J. Chalopin and D. Gonçalves. Every planar graph is the intersection graph of segments in the plane: Extended abstract. In *STOC*, pages 631–638. ACM, 2009. doi:10.1145/1536414.1536500.

[6] S. Chaplick, P. Dorbec, J. Kratochvíl, M. Montassier, and J. Stacho. Contact representations of planar graphs: Extending a partial representation is hard. In D. Kratsch and I. Todinca, editors, *WG*, volume 8747 of *LNCS*, pages 139–151. Springer, 2014. doi:10.1007/978-3-319-12340-0_12.

[7] S. Chaplick, S. Felsner, U. Hoffmann, and V. Wiechert. Grid intersection graphs and order dimension. *Order*, 35(2):363–391, 2018. doi:10.1007/s11083-017-9437-0.

[8] S. Chaplick, P. Hell, Y. Otachi, T. Saitoh, and R. Uehara. Ferrers dimension of grid intersection graphs. *Discrete Appl. Math.*, 216:130–135, 2017. doi:10.1016/j.dam.2015.05.035.

[9] S. Chaplick, P. Kindermann, A. Löffler, F. Thiele, A. Wolff, A. Zaft, and J. Zink. Stick graphs with length constraints. In D. Archambault and C. D. Tóth, editors, *GD*, volume 11904 of *LNCS*, pages 3–17. Springer, 2019. URL: http://arxiv.org/abs/1907.05257, doi:10.1007/978-3-030-35802-0_1.

[10] F. De Luca, M. I. Hossain, S. G. Kobourov, A. Lubiw, and D. Mondal. Recognition and drawing of stick graphs. *Theor. Comput. Sci.*, 796:22–33, 2019. doi:10.1016/j.tcs.2019.08.018.

[11] S. Felsner, G. B. Mertzios, and I. Mustata. On the recognition of four-directional orthogonal ray graphs. In K. Chatterjee and J. Sgall, editors, *MFCS*, volume 8087 of *LNCS*, pages 373–384. Springer, 2013. Some results herein are incomplete, see the warning in the full version: http://

page.math.tu-berlin.de/~felsner/Paper/dorgs.pdf. doi:10.1007/978-3-642-40313-2_34.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[13] B. V. Halldórsson, D. Aguiar, R. Tarpine, and S. Istrail. The Clark phaseable sample size problem: Long-range phasing and loss of heterozygosity in GWAS. *J. Comput. Biol.*, 18(3):323–333, 2011. doi:10.1089/cmb.2010.0288.

[14] I. B. Hartman, I. Newman, and R. Ziv. On grid intersection graphs. *Discrete Math.*, 87(1):41–52, 1991. doi:10.1016/0012-365X(91)90069-E.

[15] M. Jünger, S. Leipert, and P. Mutzel. Level planarity testing in linear time. In S. H. Whitesides, editor, *GD*, volume 1547 of *LNCS*, pages 224–237. Springer, 1998. doi:10.1007/3-540-37623-2_17.

[16] P. Klavík, Y. Otachi, and J. Sejnoha. On the classes of interval graphs of limited nesting and count of lengths. *Algorithmica*, 81(4):1490–1511, 2019. doi:10.1007/s00453-018-0481-y.

[17] J. Köbler, S. Kuhnert, and O. Watanabe. Interval graph representation with given interval and intersection lengths. *J. Discrete Algorithms*, 34:108–117, 2015. doi:10.1016/j.jda.2015.05.011.

[18] J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Appl. Math.*, 52(3):233–252, 1994. doi:10.1016/0166-218X(94)90143-0.

[19] J. Kratochvíl and J. Matoušek. Intersection graphs of segments. *J. Comb. Theory, Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

[20] D. Kratsch, R. M. McConnell, K. Mehlhorn, and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM J. Comput.*, 36(2):326–353, 2006. doi:10.1137/S0097539703437855.

[21] W. N. Li. Two-segmented channel routing is strong NP-complete. *Discrete Appl. Math.*, 78(1-3):291–298, 1997. doi:10.1016/S0166-218X(97)00020-6.

[22] J. Matoušek. Intersection graphs of segments and $\exists\mathbb{R}$. ArXiv, https://arxiv.org/abs/1406.2636, 2014.

[23] I. Pe'er and R. Shamir. Realizing interval graphs with size and distance constraints. *SIAM J. Discrete Math.*, 10(4):662–687, 1997. doi:10.1137/S0895480196306373.

[24] M. Schaefer. Complexity of some geometric and topological problems. In *GD*, volume 5849 of *LNCS*, pages 334–344. Springer, 2009. doi:10.1007/978-3-642-11805-0_32.

[25] M. K. Sen and B. K. Sanyal. Indifference digraphs: A generalization of indifference graphs and semiorders. *SIAM J. Discrete Math.*, 7(2):157–165, 1994. doi:10.1137/S0895480190177145.

[26] A. M. S. Shrestha, A. Takaoka, S. Tayu, and S. Ueno. On two problems of nano-PLA design. *IEICE Transactions*, 94-D(1):35–41, 2011. doi:10.1587/transinf.E94.D.35.

[27] J. Spinrad, A. Brandstädt, and L. Stewart. Bipartite permutation graphs. *Discrete Appl. Math.*, 18(3):279–292, 1987. doi:10.1016/S0166-218X(87)80003-3.