**Stick Graphs with Length Constraints**

Steven Chaplick, Philipp Kindermann, Andre Lößler, Florian Thiele, Alexander Wolff, Alexander Zaft, and Johannes Zink

Institut für Informatik, Universität Würzburg, Würzburg, Germany

firstname.lastname@uni-wuerzburg.de

**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 (STICKA), and when the order of both sets is given (STICKAB). They showed how to solve STICKAB efficiently.

In this paper, we improve the running time of their algorithm, and we solve STICKA 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, STICKA, and STICKAB are all NP-complete. On the positive side, we give an efficient solution for STICKAB with fixed stick lengths if there are no isolated vertices.

1 Introduction

For a given collection $S$ of geometric objects, the *intersection graph of $S$* has $S$ as its vertex set and an edge whenever $S \cap S' \neq \emptyset$, for $S, S' \in S$. 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. A classic (general) class of intersection graphs is that of *segment graphs*, the intersection graphs of line segments in the plane.

For example, segment graphs are known to include planar graphs [4]. The recognition problem for segment graphs is $\exists R$-complete [18,22].

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. We say that the segments are *grounded* on these two lines. The recognition problem for permutation graphs can be solved in linear time [19]. *Bipartite* permutation graphs have an even simpler intersection representation [25]; they

---

* S.C. and A.W. acknowledge support from DFG grants WO 758/11-1 and WO 758/9-1.
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 $\exists R$ includes NP, see [22,23] for background on the complexity class $\exists R$.
3 i.e., we think of the sequence of end points on the “bottom” line as one permutation $\pi$ on the vertices and the sequence on the top line as another permutation $\pi'$, where $uv$ is an edge if and only if the order of $u$ and $v$ differs in $\pi$ and $\pi'$. 
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 \[12,14,16\] compare and study the geometric intersection graph classes occurring between the simple classes, such as bipartite permutation graphs, and the general classes, such as a segment graphs. Cabello and Jejič \[10\] 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 \[15,16,23\].

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., \[6,7,13,17\]. In particular, note that the recognition problem is NP-complete for grid intersection graphs \[17\]. 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 \[17\]. 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 \[5,9,10\], e.g., one such case \[5\] is equivalent to the problem of level planarity testing which can be solved in linear time \[14\].

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 \[3,12\]. 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 \(∃R\)-complete \[2\]. 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. \[9\]. Similarly to Kratochvíl’s approach to grid intersection graphs \[17\], De Luca et al. characterized stick graphs through their bipartite adjacency matrix and used this result as a basis to develop polynomial-time algorithms to solve two constrained cases of the stick graph recognition problem called \(\text{STICK}_A\) and \(\text{STICK}_{AB}\), defined next. However, their algorithm for \(\text{STICK}_A\) is incorrect \[21\], leaving \(\text{STICK}_A\) open.

**Definition 1 (STICK).** Let \(G\) be a bipartite graph with vertex set \(A∪B\), 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 \(ℓ\). 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.

\[4\] Note that De Luca et al. \[9\] regarded \(A\) as horizontal segments.
Table 1: Previously known and new results for deciding whether a given bipartite graph $G = (A \cup B, E)$ is a stick graph. In $O(\cdot)$, we dropped $|\cdot|$. NPC means NP-complete.

| given order | variable length | fixed length |
|-------------|----------------|-------------|
| $\emptyset$ | unknown        | unknown     |
| $A$         | $O(AB)$ [Thm. 2] | NPC [Thm. 3] |
| $A,B$       | $O(E)$ [Thm. 4]  | $O((A + B)^2)$ [Cor. 5] |

Definition 2 (STICK$_A$/STICK$_{AB}$). In the problem STICK$_A$ (STICK$_{AB}$) we are given an instance of the STICK problem and additionally an order $\sigma_A$ (orders $\sigma_A, \sigma_B$) of the vertices in $A$ (in $A$ and $B$). The task is to decide whether there is a stick representation that respects $\sigma_A$ ($\sigma_A$ and $\sigma_B$).

Our Contribution. We first revisit the problems STICK$_A$ and STICK$_{AB}$ defined by De Luca et al. [9]. We provide the first efficient algorithm for STICK$_A$ and a faster algorithm for STICK$_{AB}$; see Section 2. Then we investigate the direction suggested by Cabello and Jejčič [1] 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 STICK$^f_A$, STICK$^f_{AB}$, and STICK$^f_{AB}$) are NP-complete; see Section 3. Finally, we give an efficient solution for STICK$^f_{AB}$ (that is, STICK$_{AB}$ 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.

2 Sticks of Variable Lengths

In this section, we provide algorithms for the STICK$_A$ problem in $O(|A| |B|)$ time (Theorem 2) and the STICK$_{AB}$ problem in $O(|A| + |B| + |E|)$ time (Theorem 3). Both algorithms 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 $i^*$).

Theorem 1. STICK$_{AB}$ can be solved in $O(|A| + |B| + |E|)$ time.

Proof. Let $\sigma_A = (a_1, \ldots, a_{|A|})$ and $\sigma_B = (b_1, \ldots, b_{|B|})$. Let $\beta_i$ denote the largest index such that $b_{\beta_i}$ has a neighbor in $a_1, \ldots, a_i$. Let $\tilde{B}^i$ be the elements of $(b_1, \ldots, b_{\beta_i})$ that have a neighbor in $a_1, \ldots, a_{|A|}$ ordered by $\sigma_B$, and let $\tilde{B}^{i*}$ be the elements of $(b_1, \ldots, b_{\beta_i})$ that have a neighbor in $a_{i+1}, \ldots, a_{|A|}$. At every event $p \in \{i, i^*\}$, we maintain the invariants that (i) we have a valid representation of the subgraph of $G$ induced by $b_1, \ldots, b_{\beta_i}, a_1, \ldots, a_i$; (ii) for all these vertices,
their foot points are set as consecutive integers from 1 to $\beta_i + i$; and (iii) for those not in $B^p$, their lengths are set.

Consider the enter event of $a_i$. We place $a_i$ at position $\beta_i + i$. We place the 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 $\hat{B}^{(i-1)*}$ in this order. All neighbors of $a_i$ have to be before $a_i$, and they have to be a suffix of $B^i$. This is easily checked in $\deg(a_i)$ time. The end point of $a_i$ is placed directly above the foot point of its first neighbor in this suffix. As such, the invariants (i)–(iii) are maintained.

Consider the exit event of $a_i$ and each neighbor $b_j$ of $a_i$. If $a_i$ is the last neighbor of $b_j$ in $\sigma_A$, then we end $b_j$ and set its endpoint at $\beta_i + i + 1/2$. We create $\hat{B}^{i*}$ by removing each such $b_j$ from $\hat{B}^i$. This again maintains invariants (i)–(iii). Hence, if we complete the exit event of $a_{i|A|}$, we obtain a STICK$_{AB}$ representation of $G$. Otherwise, $G$ has no such representation. Clearly, the whole algorithm works in $O(|A| + |B| + |E|)$ time. Note that, even though we have not explicitly discussed isolated vertices, these are easily handled with length 0. $\square$

Theorem 2. STICK$_A$ can be solved in $O(|A| \cdot |B|)$ time.

Proof. We assume that $G$ is connected and discuss otherwise in Appendix A.

Overview. For each event $p \in \{i, i^*\}$, we maintain a data structure $T^p$ that compactly encodes all realizable permutations of certain horizontal sticks $B^p \subseteq B$. Namely, each $B^i$ (resp. $B^{i*}$) consists of all sticks of $B$ with a neighbor in $a_1, \ldots, a_i$ and a neighbor in $a_i, \ldots, a_{i|A|}$ (resp. $a_{i+1}, \ldots, a_{i|A|}$). We denote by $G^p$ the induced subgraph of $G$ containing $a_1, \ldots, a_i$ and their neighbors. A permutation $\pi$ of $B^p$ is realizable if there is a stick representation of the graph obtained from $G^p$ by adding a vertical stick to the right of all neighbors of horizontal sticks in $B^p$ where $B^p$ is drawn top-to-bottom in order $\pi$. In the enter event of $a_i$, we add to the data structure all the vertices of $B$ that neighbor $a_i$ and aren’t in the data structure yet (we call these entering vertices), and constrain the data structure so that all the neighbors of $a_i$ must occur after (below) the non-neighbors of $a_i$.

In the exit event of $a_i$, we remove all sticks of $B$ that do not have any neighbor $a_j$ with $j > i$, i.e., they have $a_i$ as their last neighbor (we call these leaving vertices).

Data structure. See Fig. 1 for an example. Consider any event $p$. Observe that $G^p$ may consist of several connected components $G^p_1, \ldots, G^p_{k_p}$. Since $G$ is connected, the components are naturally ordered from left to right by $\sigma_A$. 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 data structure $T^p_j$. Namely, our data structure will be $T^p = (T^p_1, \ldots, T^p_{k_p})$.

Each data structure $T^p_j$ is a rooted tree. At each node, its children consist of two types: the leaves (which correspond to the vertices of $B^p_j$) and the non-leaves. The non-leaves are ordered, while the leaves are unordered and can be placed anywhere before, after, or between the non-leaves with the same parent. A valid
traversal of $T_2^p$ is a pre-order traversal where, for each node, the non-leaf children are visited in the specified order and the leaves are permuted among the non-leaf children. Each permutation expressed by $T_2^p$ corresponds to a valid traversal. Note that the non-leaves are visited in the same order in every valid traversal.

**Correctness and event processing.** We will argue that this data structure is sufficient to express the realizable permutations of $B_p^i$ by induction. 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.

Consider the exit event of $a_i$ and assume that we have the data structure $T_{i−1}^i = (T_{i−1}^1, \ldots, T_{i−1}^{k_i})$. If there are no leaving vertices, we just keep the data structures and are done. Otherwise, $B_{i−1}^i$ is a strict subset of $B_i^i$. We delete all leaves from $T_{i−1}^i$ corresponding to leaving vertices. If this results in any non-leaf node having only one child and that child is not a leaf, we merge it with its parent. If all children of an internal node get removed, we also remove the node. Obviously, this procedure maintains all realizable permutations of $B_{i−1}^i$ due to $G_{i−1}^i$.

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

- all vertices in $B_{i−1}^{(i−1)\rightarrow}$ are adjacent to $a_i$,
- none of them are adjacent to $a_i$, or
- there is an $s$ such that (i) $B_{1}^{(i−1)\rightarrow}$ contains at least one neighbor of $a_i$; (ii) all vertices in $B_{s+1}^{(i−1)\rightarrow}, \ldots, B_{k_i−1}^{(i−1)\rightarrow}$ are neighbors of $a_i$; and (iii) no vertices in $B_{s+1}^{(i−1)\rightarrow}, \ldots, B_{s}^{(i−1)\rightarrow}$ are adjacent to $a_i$; see Fig. 2a.

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. 2b. After that we will create the data structure $T_i$. We create a tree $T$ whose realizable permutations are
precisely the subset of those of $T_s^{(i-1)\rightarrow}$ where all leaves that are neighbors of $a_i$ occur as a suffix. We initialize $T = T_s^{(i-1)\rightarrow}$. If all vertices in $B_s^{(i-1)\rightarrow}$ are neighbors of $a_i$, then we are already done.

Otherwise, we define a marked node as one where all leaves in its subtree are neighbors of $a_i$; an unmarked node as one where no leaf in its induced subtree is a neighbor of $a_i$; and a node is half-marked otherwise. Note that the root is half-marked. Since the neighbors of $a_i$ must form a suffix, the marked non-leaf children of a half-marked node form a suffix, the unmarked non-leaf children form a prefix, and there is at most one half-mark 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 leaf-to-root. Let $a$ be a half-marked node, and let $b$ be its half-marked child (if it exists). We have to enforce that in any valid traversal of $T$ the unmarked children of $a$ are visited before $b$ and the marked children of $a$ are visited after $b$. We create a new (marked) vertex $a'$ and move all marked children of $a$ to $a'$, preserving the order among the non-leaf children. Then we create a new (half-marked) node $a''$ and we hang $a$, $b$, and $a'$ from $a''$ in this order. Finally, we put $a''$ into the former position of $a$ in $T$. If this results in any internal node $z$ with no leaf-children and only one child, we merge $z$ with its parent. This ensures that all permutations realized by $T$ have the neighbors of $a_i$ as a suffix. Further, observe that the non-leaves of $T_s^{(i-1)\rightarrow}$ are visited in the same order in any valid traversal of $T$ as in a valid traversal of $T_s^{(i-1)\rightarrow}$. The marked (unmarked) leaf-children of any half-marked node $a$ of $T_s^{(i-1)\rightarrow}$ can be placed anywhere before, between, or after its marked (unmarked) children, but not before (after) $b$, since $b$ has both marked and unmarked children. Hence, the permutations realized by $T$ are exactly those realized by $T_s^{(i-1)\rightarrow}$ that have the neighbors of $a_i$ as a suffix.

Now, we create the data structure $T^i$; see Fig. 2c. We set $T_i^i = T_i^{(i-1)\rightarrow}, \ldots, T_{s-1}^i = T_{s-1}^{(i-1)\rightarrow}$. We additionally create $T_i^u$ as follows. We hang $T_{s+1}^{(i-1)\rightarrow}, \ldots, T_{k_i-1}^{(i-1)\rightarrow}$ from a new node $x$ in this order. We further insert the entering vertices as leaf-
children of $x$ (note that this allows them to mix freely before, after, or between
the components $G_{s_{i+1}}^{(i-1)}, \ldots, G_{k_{i-1}}^{(i-1)}$. Then, we hang $T$ followed by $x$ off another
node $r$, and make $r$ the root of $T^*$. Finally, we set $T^* = (T_1^*, \ldots, T_s^*)$. This way,
the order of the components $G_1^{(i-1)}, \ldots, G_{k_{i-1}}^{(i-1)}$ of $G^{(i-1)}$ is maintained
in the data structures for $G^i$. Furthermore, we ensure that the entering vertices
can be placed exactly before, after, or between the components of $G^{(i-1)}$ that
are completely adjacent to $a_i$. Hence, this data structure captures all realizable
permutations of $B^i$ due to $G^i$.

The decision problem of $\text{STICK}_A$ can easily be solved by this algorithm. To find
a stick representation, however, one has to backtrack through the data structures
to find a valid permutation for the input problem. In Appendix A, we show how
to do the backtracking and that the whole algorithm takes $O(|A||B|)$ time. \(\square\)

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 $\text{STICK}^{\text{fix}}$, by $\text{STICK}^{\text{fix}}_A$ if additionally $\sigma_A$ is given, and by $\text{STICK}^{\text{fix}}_{AB}$ if
$\sigma_A$ and $\sigma_B$ given. Unlike the case with variable stick length, all three variants are
NP-hard; see Sections 3.1 and 3.2. Surprisingly, $\text{STICK}^{\text{fix}}_{AB}$ 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 $\text{STICK}^{\text{fix}}$ if we are given a total order of the sticks
on the ground line. With our NP-hardness results, this implies NP-completeness.

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

We show that $\text{STICK}^{\text{fix}}$ is NP-hard by reduction from 3-PARTITION, which is
strongly NP-complete. 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.** $\text{STICK}^{\text{fix}}$ is NP-complete.
Proof. We describe a polynomial-time reduction from 3-PARTITION. 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. 3a. 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. 3b. 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 \( z \), 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 \( \varepsilon \); 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 \( \varepsilon \); 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 \( S \). \( \square \)

The sticks of lengths \( s_1, \ldots, s_{3m} \) can be simulated by paths of sticks with length \( \varepsilon \) each. Exploiting this, we can modify our reduction to use only three distinct stick lengths. We prove the following corollary in Appendix B.

**Corollary 1.** \( \text{STICK}^{1x} \) with only three different stick lengths is NP-complete.
Theorem 4. \textit{STICK}^{\text{fix}}_A is NP-complete.

Proof. We describe a polynomial-time reduction from \textit{MONOTONE}-3-SAT. A schematization of our reduction is depicted in Figs. 4 to 6. Given a \textit{MONOTONE}-3-SAT instance $\Phi$ over variables $x_1, \ldots, x_n$, we construct for each variable $x_i$ (with $i \in \{1, \ldots, n\}$) a variable gadget as depicted in Fig. 4. Inside a (black) cage, there is a vertical (red) stick $r_i$ with length 1 and from inside, a long horizontal (green) stick $g_i$ leaves this cage. We can enforce the structure to look like in Fig. 4 as follows. We prescribe the order $\sigma_A$ of the vertical sticks as in Fig. 4. Since $a_{i+1}$ has length $\varepsilon < 1$, the horizontal (black) stick $h_i$ intersects the two vertical (black) sticks $v_i+1$ and $a_{i+1}$ close to its foot point. We have $\sigma_A(a_{i+1}) < \sigma_A(r_i) < \sigma_A(v_i)$, so $r_i$ is inside the cage bounded by $h_i$ and $v_i$ and fixed its height—as it does not intersect $h_i$—making 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 $r_i$. The stick $w_i$ intersects $r_i$ close to $r_i$’s foot point because $w_i$ has length $\varepsilon$. 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 $v_{i+1}$.

Fig. 4: Variable gadget in our reduction from \textit{MONOTONE}-3-SAT to \textit{STICK}^{\text{fix}}_A.

Fig. 5: Positive clause gadget (empty sub-stripe at the bottom). Here, a clause gadget for each of the eight possible truth assignments of a \textit{MONOTONE}-3-SAT clause is depicted. E.g., $tft$ means that the first variable is set to true, the second to false, and the third to true. Similarly, a negative clause gadget has an empty sub-stripe at the top.

3.2 \textit{STICK}^{\text{fix}}_A and \textit{STICK}^{\text{fix}}_{A_B}

We show that \textit{STICK}^{\text{fix}}_A and \textit{STICK}^{\text{fix}}_{A_B} are NP-hard by reduction from \textit{MONOTONE}-3-SAT, which is NP-complete [20]. In \textit{MONOTONE}-3-SAT, one is given a Boolean formula $\Phi$ in conjunctive normal form where each clause contains three distinct literals that are all positive or all negative. The task is to decide if $\Phi$ is satisfiable.
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 a short horizontal (violet) stick of length $\varepsilon$ (see Fig. 6 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 $\ell_i$. Hence, $y_i$ and $g_i$ intersect each other close to their end points as otherwise $g_i$ would intersect $z_i$. We choose $\ell_1$ sufficiently large such that the foot point of $y_1$ is to the right of the clause gadgets (see Fig. 6) and for each $\ell_i$ 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 $y_i$ and $z_i$ 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 $\ell_i$ 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 $\sigma_A$.

**Fig. 6:** Illustration of our reduction from MONOTONE-3-SAT to STICK$^A_k$.

For each clause, we add a clause gadget (see Fig. 6) as shown in Fig. 6. 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 $\varepsilon$. 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 $\geq 1$. 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 each horizontal (orange) stick $o$ that is assigned to $x_i$
intercept $y_i$ 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 $o_i$’s end points. 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. 5). Within each clause gadget, we have a vertical (blue) stick $b$ of length 2. Each horizontal (black) stick that bounds a clause gadget intersects a short vertical (black) stick of length $\epsilon$ to force $b$ into its designated clause gadget. Moreover, $b$ is not isolated because it intersects a short (violet) stick of length $\epsilon$.

Clearly, if $\Phi$ is satisfiable, there is a stick representation of the STICK$_A^f$ instance obtained from $\Phi$ by our reduction by placing the sticks as described before (see also Fig. 6). In particular, the blue sticks can be placed as depicted in Fig. 5.

On the other hand, if there is a stick representation of the STICK$_A^f$ instance obtained by our reduction, $\Phi$ 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 $\sigma_A$. The only flexibility is, for each $i \in \{1, \ldots, n\}$, 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. 5 precisely the configurations that correspond to satisfying truth assignments are realizable. Thus, we can read a satisfying truth assignment of $\Phi$ from the variable gadgets.

We enforce an order of the horizontal sticks except for a set $W$ of sticks, which are the short (violet) sticks of length $\epsilon$ that are adjacent to the red and the blue sticks in the variable and clause gadgets. For STICK$_{AB}$ we can prescribe $\sigma_B$ if we remove the sticks $W$ and use the same reduction to obtain Corollary 2. Observe that we now have isolated vertices (the red and blue vertical sticks).

Corollary 2. STICK$_{AB}^f$ with isolated vertices in $A$ or $B$ is NP-complete.

### 3.3 STICK$_{AB}^f$ without isolated vertices

In this section, we constructively show that STICK$_{AB}^f$ 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 STICK$_{AB}$ (and consequently also STICK$_{AB}^f$), 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 follows implicitly from the proof of Theorem 1. An explicit proof is given in the full version [8].

**Lemma 1.** In all stick representations of an instance of STICK$_{AB}$, 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 $O(|E|)$.

We are given an instance of STICK$_{AB}^f$ 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 is solvable if and only if there is no directed cycle of negative weights, and a solution can be found in $O(nm)$ time with the Bellman–Ford algorithm.

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 $\varepsilon$ 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 $x_k - x_i > \ell_i \Rightarrow x_i - x_k \leq -\ell_i - \varepsilon$. Symmetrically, let $v_i \in B$. Let $j$ be the smallest $j$ such that $(v_j, v_i) \in E$ and $\ell_j \geq \ell_i$. 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 $x_i - x_k > \ell_i \Rightarrow x_k - x_i \leq -\ell_i - \varepsilon$.

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 4. By Lemma 1, there is at most one realizable order of vertices for a STICKfix instance without isolated vertices, which can be found in linear time and proves Corollary 8.

**Theorem 5.** STICKfix can be solved in $O((|A| + |B|)^2)$ time if we are given a total order of the vertices.

**Corollary 3.** STICKfix without isolated vertices is solvable in $O((|A| + |B|)^2)$ time.

### 4 Open Problems

We have shown that STICKfix is NP-complete even if the sticks have only three different lengths, while STICKfix for unit-length sticks is solvable in linear time. But what is the computational complexity of STICKfix for sticks with one of two lengths? Also, the three different lengths in our proof depend on the number of sticks. Is STICKfix still NP-complete if the fixed lengths are bounded? Beside this, the complexity of the original problem STICK is still open.
References

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

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

3. Catanzaro, D., Chaplick, S., Felsner, S., Halldórsson, B.V., Halldórsson, M.M., Hixon, T., Stacho, J.: Max point-tolerance graphs. Discrete Appl. Math. 216, 84–97 (2017). https://doi.org/10.1016/j.dam.2015.08.019

4. Chalopin, J., Gonçalves, D.: Every planar graph is the intersection graph of segments in the plane: Extended abstract. In: STOC. pp. 631–638. ACM (2009). https://doi.org/10.1145/1536414.1536500

5. Chaplick, S., Dorbec, P., Kratochvíl, J., Montassier, M., Stacho, J.: Contact representations of planar graphs: Extending a partial representation is hard. In: Kratsch, D., Todinca, I. (eds.) WG. LNCS, vol. 8747, pp. 139–151. Springer (2014). https://doi.org/10.1007/978-3-319-12340-0_12

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

7. Chaplick, S., Hell, P., Otachi, Y., Saitoh, T., Uehara, R.: Ferrers dimension of grid intersection graphs. Discrete Appl. Math. 216, 130–135 (2017). https://doi.org/10.1016/j.dam.2015.05.035

8. Chaplick, S., Kindermann, P., Löffler, A., Thiele, F., Wolff, A., Zaft, A., Zink, J.: Stick graphs with length constraints. Arxiv report (2019), http://arxiv.org/abs/1907.05257

9. De Luca, F., Hossain, M.I., Kobourov, S.G., Lubiw, A., Mondal, D.: Recognition and drawing of stick graphs. In: Biedl, T.C., Kerren, A. (eds.) GD. LNCS, vol. 11282, pp. 303–316. Springer (2018). https://doi.org/10.1007/978-3-030-04414-5_21

10. Felsner, S., Knauer, K.B., Mertzios, G.B., Ueckerdt, T.: Intersection graphs of L-shapes and segments in the plane. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS. LNCS, vol. 8635, pp. 299–310. Springer (2014), some results herein are incomplete, see the warning on the full version: http://page.math.tu-berlin.de/~felsner/Paper/dorgs.pdf

11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)

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

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

14. Jünger, M., Leipert, S., Mutzel, P.: Level planarity testing in linear time. In: Whitesides, S.H. (ed.) GD. LNCS, vol. 1547, pp. 224–237. Springer (1998). https://doi.org/10.1007/3-540-37623-2_17

15. Klavík, P., Otachi, Y., Sejnoha, J.: On the classes of interval graphs of limited nesting and count of lengths. Algorithmica 81(4), 1490–1511 (2019). https://doi.org/10.1007/s00454-018-0481-y

16. Kobler, J., Kuhnert, S., Watanabe, O.: Interval graph representation with given interval and intersection lengths. J. Discrete Algorithms 34, 108–117 (2015). https://doi.org/10.1016/j.jda.2015.05.011
17. Kratochvíl, J.: A special planar satisfiability problem and a consequence of its NP-completeness. Discrete Appl. Math. 52(3), 233–252 (1994). https://doi.org/10.1016/0166-218X(94)90143-0

18. Kratochvíl, J., Matoušek, J.: Intersection graphs of segments. J. Comb. Theory, Series B 62(2), 289–315 (1994). https://doi.org/10.1006/jctb.1994.1071

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

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

21. Lubiw, A.: Private communication (2019)

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

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

24. Schaefer, M.: Complexity of some geometric and topological problems. In: GD. LNCS, vol. 5849, pp. 334–344. Springer (2009). https://doi.org/10.1007/978-3-642-11803-0_32

25. Sen, M.K., Sanyal, B.K.: Indifference digraphs: A generalization of indifference graphs and semiorders. SIAM J. Discrete Math. 7(2), 157–165 (1994). https://doi.org/10.1137/S0895480190177143

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

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

A Omitted proofs of Section 2

Theorem 2. \( \text{STICK}_A \) can be solved in \( O(|A| \cdot |B|) \) time.

Proof (Continued). We continue the proof by first noting how one can handle disconnected graphs, and then discussing the backtracking and representation construction.

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 \( \sigma_A \) contains a pattern of indices that alternate \( abab \), then there can be no solution to \( \text{STICK}_A \). Otherwise, we can treat each component separately by our algorithm, and then nest the resulting representations (whose construction we describe next via the backtracking) according to how the components nest in \( \sigma_A \).

It remains to show how to do the backtracking and how to obtain the running time. The size of each data structure \( T_p \) is in \( O(|B|^p) \), since there are no degree-2 vertices in the trees and each leaf corresponds to a vertex in \( B \). 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, the whole construction works in \( O(|A||B|) \) time.

In the main text, we said that whenever a node \( x \) has exactly one child \( y \) and that child is an internal node, we merge \( x \) with its parent \( z \). Instead of doing this, we will create a “shortcut” from \( y \) to \( z \) (and associate this shortcut with the last operation which caused \( x \) to be in this state). This way, we can traverse the tree without having to look at internal degree-2 nodes, but we keep them in the data structure for future reference. Also, we do not remove any leaves from the tree; we just mark them as dead and do not consider them anymore.

Assume that the algorithm processes all events without stopping. This means that, in every step, there was some realizable permutation. We now consider the data structure \( T^{|A|\rightarrow} \). Since we never actually removed anything but just marked leaves as dead and introduced shortcuts, removing the shortcuts and marking leaves as alive can be done in \( O(|A||B|) \) time. This gives us a data structure \( T \) that contains all vertices of \( B \) as leaves. In particular, \( T \) gives us a permutation \( \sigma_B \) of \( B \). Moreover, for every event \( p \), \( \sigma_B \) restricted to \( B^p \) is a realizable permutation of \( B^p \) due to \( G^p \). Thus, executing our algorithm for \( \text{STICK}_{AB} \) on \( \sigma_A \) and \( \sigma_B \) gives us a stick representation of \( G \).

\[ \square \]

B Omitted proofs of Section 3

We modify the reduction used in the proof of Theorem 3 so that we use only three lengths. To this end, we use paths of sticks of length \( \varepsilon \). We refer to them as \( \varepsilon \)-paths. Like a spring, an \( \varepsilon \)-path can be stretched (Fig. 7a) and compressed (Fig. 7c) up to a specific length. We will exploit the following properties regarding the minimum and the maximum size of an \( \varepsilon \)-path.
Lemma 2. There is a stick representation of a $2n$-vertex $\varepsilon$-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 $\varepsilon$-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. 7a). This construction has height and width $n\varepsilon$ and, clearly, this is the maximum width and height for a $2n$-vertex $\varepsilon$-path.

For the compressed $\varepsilon$-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. 7d 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 $\varepsilon$-path.

Consider the $i$-th stick of an $\varepsilon$-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 $\varepsilon$ 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 $\varepsilon$-path of $2n$ sticks, we have width and height strictly greater than $\left\lceil \frac{2n}{6} \right\rceil \varepsilon \geq \frac{n}{3} \varepsilon$. $\square$

Corollary 1. $\text{STICK}_{\text{fix}}$ with only three different stick lengths is NP-complete.

Proof. We modify the reduction from 3-PARTITION to $\text{STICK}_{\text{fix}}$ described in the proof of Theorem 3 such that we use only three distinct stick lengths. We use the three lengths $\varepsilon$, $Cm$, and $3Cm$ (or longer, e.g. $\infty$). In Fig. 8 sticks of these lengths are violet, black, and green, respectively.

![Fig. 7: An $\varepsilon$-path of 12 sticks in (a)–(c) and 6 sticks in (d).](image-url)
First, we describe the modifications of the frame structure, which are also depicted in Fig. 8a. 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 $j \in \{1, \ldots, m+1\}$. Instead of the sticks $p_1, \ldots, p_{m+1}$ of different lengths, we use horizontal (black) sticks $p'_1, \ldots, p'_{m+1}$ each with length $Cm$ to separate the pockets. The stick $p'_j$ intersects $y_k, z_k$ for all $k \in \{j+1, \ldots, m+1\}$ and $y_j$ but not $z_j$. All pairs $y_j-z_j$ are kept together by a stick of length $\varepsilon$. For each two neighboring pairs $y_j-z_j$ and $y_{j+1}-z_{j+1}$, these sticks of length $\varepsilon$ are connected by an $\varepsilon$-path of $2C/\varepsilon$ sticks. According to Lemma 2, this affects 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 $p'_1, \ldots, p'_{m+1}$ have height at most $C \pm 2\varepsilon$, similar as in the proof of Theorem 3. We keep the vertical (orange) stick $o$ as in Figure 3a 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 $i \in \{1, \ldots, 3m\}$, which are also depicted in Fig. 8b. We keep a long stick $b'_i$ similar to $b_i$—now with length $3Cm$. We replace $r_i$ (and $h_i$ and $v_i$) by an $\varepsilon$-path of $6s_i/\varepsilon - 4$ sticks. We make the first stick of the $\varepsilon$-path intersect $b'_i$. By Lemma 2 this $\varepsilon$-path has a stick representation with height $s_i + \delta$ for any $\delta > 0$, but there is no stick representation with height $s_i - \frac{2\varepsilon}{\varepsilon}$ or smaller. Clearly, these number gadgets can only be placed into one pocket since none of their sticks intersects a $p'_j$ for $j \in \{1, \ldots, m+1\}$.

Hence, we can represent a yes-instance of 3-PARTITION as such a stick graph if and only if the $\varepsilon$-paths of the number gadgets are (almost) as much compressed.
as possible (to make the number gadgets small enough) and the $\varepsilon$-paths between the $y_j^i z_j^i$ 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.

Lemma 1. In all stick representations of an instance of $\text{STICK}_{AB}$, 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 $\Gamma_1$ and $\Gamma_2$ of the same instance of $\text{STICK}_{AB}$ without isolated vertices that have different combinatorial arrangements on the ground line. Without loss of generality, there is an $a \in A$ and a $b \in B$, such that in $\Gamma_1$, $a$ comes after $b$, while in $\Gamma_2$, $a$ comes before $b$ (see Fig. 9). Clearly, $a$ and $b$ cannot be adjacent. Since $a$ is not isolated, there is a $b'$ that is adjacent to $a$ and comes after $b$. Analogously, there is an $a'$ that is adjacent to $b$ and comes before $a$. In $\Gamma_2$, $a$ and $b'$ define a triangle $T$ (see Fig. 9b), which completely contains $b$ since $b$ occurs between $a$ and $b'$, but is adjacent to neither of them. However, $a'$ is outside of $T$ as it comes before $a$. This contradicts $b$ and $a'$ being adjacent. The unique order can be determined in $O(|E|)$ time as described in Section 2. \qed