Available online at www.sciencedirect.com

ScienceDirect

Journal of Discrete Algorithms 6 (2008) 140–161

JOURNAL OF DISCRETE ALGORITHMS

www.elsevier.com/locate/jda

Complexity results for three-dimensional orthogonal graph drawing

Maurizio Patrignani

Dipartimento di Informatica e Automazione, Università di Roma Tre, Rome, Italy

Received 17 November 2005; accepted 23 June 2006

Available online 12 January 2007

Abstract

In this paper we consider the problem of finding three-dimensional orthogonal drawings of maximum degree six graphs from the computational complexity perspective. We introduce a 3SAT reduction framework that can be used to prove the NP-hardness of finding three-dimensional orthogonal drawings with specific constraints. By using the framework we show that, given a three-dimensional orthogonal shape of a graph (a description of the sequence of axis-parallel segments of each edge), finding the coordinates for nodes and bends such that the drawing has no intersection is NP-complete. Conversely, we show that if node coordinates are fixed, finding a shape for the edges that is compatible with a non-intersecting drawing is a feasible problem, which becomes NP-complete if a maximum of two bends per edge is allowed. We comment on the impact of these results on the two open problems of determining whether a graph always admits a drawing with at most two bends per edge and of characterizing orthogonal shapes admitting an orthogonal drawing without intersections.

Keywords: Three-dimensional graph drawing; Orthogonal layout; Computational complexity

1. Introduction

Because of its impact on applications and because of its theoretical appeal three-dimensional orthogonal graph drawing has attracted much research interest through the last decades [3,4,8,14,16,18,22,24–26,31,33,34]. Several algorithms have been conceived to compute orthogonal drawings of graphs of maximum degree six in $O(n^{3/2})$ volume [3,16,18,32], which has been proven to be asymptotically optimal [24] (finding a drawing in the minimum volume is known to be NP-hard [16]). Further, the goal of reducing the number of bends has been pursued, producing algorithms that compute drawings with as few as three bends per edge maximum [18,26,35]. The tradeoff between the volume and the maximum number of bends per edge has also been explored [18,34].

Nevertheless, some basic questions still lack an answer. It is open, for example, whether a graph of maximum degree six always admits a drawing with at most two bends per edge (called a 2-*bend drawing*). This problem, first posed by Eades et al. [18], is mentioned as one of the most interesting problems of the field by several authors ([33, 34], and [10], problem #46). Two bends would be best possible, since any drawing of K_5 uses at least two bends on at least one edge [32]. Hence, drawings with zero bends (0-*bend drawings*) or with a maximum of one bend per edge

E-mail address: patrignani@dia.uniroma3.it.

1570-8667/$ – see front matter
doi:10.1016/j.jda.2006.06.002

(1-*bend drawings*) are not guaranteed to exist for all maximum degree six graphs. In particular, it has been proven that it is NP-hard to recognize graphs admitting 0-bend drawings [16]. On the contrary, the complexity of recognizing graphs that admit 1-bend drawings is unknown, and it is also unknown if all simple graphs with maximum degree at most three admit such drawings [33].

Regarding the question of whether a graph of maximum degree six always admits a 2-bend drawing, in [17] it was conjectured that the answer is false, but the graph that was thought to require three bends, the K_7 graph, was drawn with two bends per edge by Wood [31], along with the other 6-regular complete multi-partite graphs $K_{6,6}$, $K_{3,3,3}$, and $K_{2,2,2,2}$ [32]. The best known upper bound on the number of bends in three-dimensional orthogonal graph drawings is $2 + \frac{2}{7}$ bends per edge on average [34]. Also, it is known that if the graph has maximum degree five, two bends per edge suffice [34]. There are graph families for which a drawing with two bends per edge implies many edges sharing the same axis-perpendicular plane [33].

A further open problem in this field is the characterization of the orthogonal shapes admitting an orthogonal drawing without intersections. Such a characterization is still missing in the general case ([15] and [7], problem 20) and would allow the separation of the task of defining the shape of the drawing from the task of computing its coordinates, extending to three dimensions a well-studied and widely adopted two-dimensional approach [11,28,29]. In 2D, the topology-shape-metrics approach consists of three main steps. In the first step, a planar embedding of the input graph G is defined. In the second step, a two-dimensional orthogonal representation of G is computed. An orthogonal representation is an equivalence class of orthogonal drawings of G all having the same shape and such that no two edges intersect. It can be described by labeling each edge (u, v) of G with a sequence of labels in the set {x+, x–, y+, y–}, representing the directions followed when travelling along edge (u, v) from u to v. In the third step, the coordinates for the nodes and for the bends along the edges are found.

A key component of the 2D topology-shape-metrics approach is a characterization of those labelings that guarantee the existence of an orthogonal drawing without intersections. Such a characterization can be found in the works by Vijayan and Wigderson [30] and by Tamassia [28], and a 3D counterpart of it consists of the solution to the following SHAPE GRAPH REALIZATION problem: Let G be a graph whose edges are directed and labeled with a sequence of labels in the set {x+, x–, y+, y–, z+, z–}. Does a 3D orthogonal drawing of G exist such that each edge has a shape "consistent" with its labeling and no two edges intersect? For example, Fig. 1 shows two graphs, G_1 and G_2, along with a labeling for their edges. For G_1 there exists a 3D orthogonal drawing without intersections and such that every edge has a shape consistent with the sequence of labels associated with it. For G_2, such a drawing does not exist.

Preliminary results toward the recognition of realizable orthogonal shapes are provided in [12,13] where paths (with further additional constraints) and cycles are considered, respectively. In [15] it is shown that the known characterization for cycles does not immediately extend to even seemingly simple graphs such as theta graphs (simple graphs consisting of three cycles).

In this paper we consider three-dimensional orthogonal drawings of a maximum degree six graph from the computational complexity perspective. The main contributions of this paper are the following:

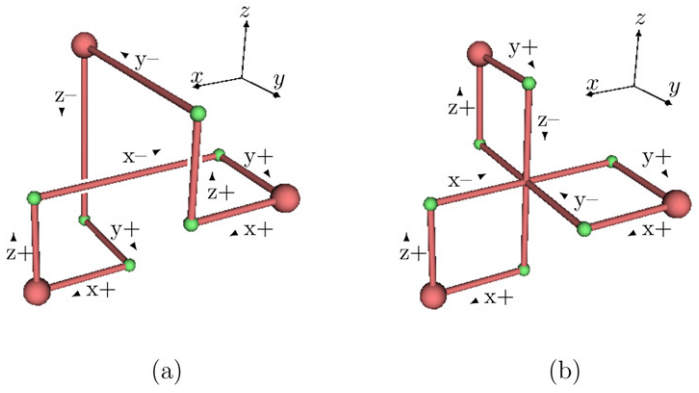

Fig. 1. Graph G_1 (a) admits a non-intersecting 3D orthogonal drawing such that the edges are "consistent" with the directions and labels associated with them. Graph G_2 (b) does not admit such a drawing.

- We introduce a framework that can be used for reducing an NP-hard problem (namely, the 3SAT problem) to a three-dimensional geometric problem. The 3SAT reduction framework can be used to show that it is NP-hard to decide if an orthogonal 3D drawing of a graph satisfying some constraints exists.
- By using such a framework we show that the SHAPE GRAPH REALIZATION problem is NP-complete.
- Conversely, we show that the opposite problem, called N-BEND ROUTING, of producing a non-intersecting N-bend drawing when the position of the nodes is fixed is a feasible problem for $N = 0$, $N = 1$, and for $N \geqslant 6$. In fact, we produce algorithms for deciding if a graph with nodes at fixed positions admits 0- and 1-bend drawings that run in $O(|V|^2)$ time, where $|V|$ is the number of nodes of the input graph. Also, we show that a graph with nodes at fixed positions always admits a 6-bend drawing by describing an algorithm for computing such drawings that runs in $O(|V| \log |V|)$ time.
- By using the 3SAT reduction framework, we show that 2-BEND ROUTING is NP-complete.
- We comment on the impact of these results on the two open problems of determining whether a graph always admits a drawing with at most two bends per edge and of characterizing orthogonal shapes admitting an orthogonal drawing without intersections.

The paper is organized as follows. In Section 2 some preliminary definitions are given. Section 3 introduces the 3SAT reduction framework. Section 4 is devoted to the problem of finding a drawing with a given shape, while Section 5 is devoted to the reverse problem of finding a drawing with nodes at fixed points. Finally, Section 6 contains a discussion of the results and some open problems.

2. Background

We assume familiarity with basic graph drawing, graph theory, and computational geometry terminology (see, e.g. [5,11,27]).

A *3D orthogonal drawing* of a graph $G = (V, E)$ is such that each node $u \in V$ is mapped to a distinct point of the three-dimensional space and each edge $e = (u, v) \in E$ is mapped to a polygonal chain of axis-parallel segments joining u and v. A *bend* is a point shared between two consecutive segments of the same edge. An *intersection* in a three dimensional orthogonal drawing is a pair of edges that overlap in at least one point that does not correspond to a common end-node. A *k-bend* (*3D orthogonal*) *drawing* of a graph, where k is a non-negative integer, is a non-intersecting orthogonal drawing such that each edge has at most k bends.

A *3D orthogonal grid drawing* is a 3D orthogonal drawing such that nodes and bends have integer coordinates. In the remainder of this paper, unless differently specified, 3D orthogonal drawings will be considered (i.e., real-valued coordinates are allowed). Also, for the sake of brevity a 3D orthogonal drawing is called a *drawing*.

An X-plane (Y-plane, Z-plane, respectively) is a plane perpendicular to the X-axis (Y-axis, Z-axis, respectively). Given a drawing Γ of a graph G and two nodes u and v, we write $u >_x v$ if the X-coordinate of u is greater than the X-coordinate of v in Γ, and we write $u =_x v$ if u and v have the same X-coordinate. Also, we write $u >_x >_y v$ if $u >_x v$ and $u >_y v$.

A *direction label* is a label in the set {x+, x–, y+, y–, z+, z–}. Let G be a graph and Γ be a 3D orthogonal drawing of G. Let e be an undirected edge of G whose end-nodes are u and v. Select one of the two possible orientations (u, v) and (v, u) of e and call $p_1, p_2, \ldots, p_m$ the endpoints of the orthogonal segments corresponding to edge e in Γ as they are encountered while moving along e from u to v ($p_1 = u$ and $p_m = v$). The *shape of e in Γ* is the sequence of the direction labels corresponding to the directions of vectors $\overrightarrow{p_i, p_{i+1}}$, $i = 1, \ldots, m - 1$. For example, consider an edge (u, v) drawn with a single bend b and such that $u <_x =_y b =_x <_y v$. The shape of e consists of the orientation from u to v and the sequence of labels x+, y+. We also write $u \xrightarrow{x+} \xrightarrow{y+} v$.

When producing a 3D orthogonal drawing of a graph one can ask if the positions of the vertices and the shapes of the edges can be computed separately. For example, it can be asked if it is always possible to find a drawing of a graph whose vertex positions are fixed. We can define the following family of problems parametric in the integer N.

Problem: N-BEND ROUTING
Instance: A graph $G = (V, E)$ and an injection between nodes and distinct points of the three-dimensional space.
Question: Does a non-intersecting 3D N-bend drawing of G exist such that the nodes have the specified coordinates?

Conversely, it can be asked what is the complexity of deciding if a graph admits a drawing such that its edges have a specified shape. A *shape graph* is a graph where a shape (an orientation and a sequence of direction labels) is specified for each edge. A shape graph H is *realizable* if it admits a non-intersecting drawing Γ such that each edge has the specified shape. Observe that the segments of the polygonal chains of Γ corresponding to the edges are not allowed to have zero length. Formally, the SHAPE GRAPH REALIZATION problem is as follows.

Problem: SHAPE GRAPH REALIZATION
Instance: A shape graph H.
Question: Does a non-intersecting drawing of H exist such that each edge has the specified shape?

Observe that the above defined problems ask for orthogonal drawings where nodes and bends are allowed to have real-valued coordinates. While SHAPE GRAPH REALIZATION clearly has the same complexity in the integer and real-valued coordinate settings, N-BEND ROUTING restricted to integer coordinates yields a different family of problems, possibly with a different time complexity, called N-BEND GRID ROUTING. The complexity of such problems is addressed in Section 6.

3. The 3SAT reduction framework

In [23] a powerful paradigm, called "logic engine", for proving NP-hardness of graph drawing problems was extended to the three-dimensional setting. The logic engine consists of a device that mechanically simulates an instance of an NP-complete problem known as NOT-ALL-EQUAL-3-SAT [2,6,11,16,19,20]. The basic mechanism exploited by the logic engine is the physical collision between its mobile components. This makes the logic engine suitable for proving NP-hardness of problems where metrics have to be taken into account. The list of problems shown to be NP-hard by using the logic engine paradigm confirms this intuition: unit-length grid drawings of trees in 2D [2] and in 3D [16], minimum area grid drawings of trees [6], nearest neighbour graph realization [19], mutual nearest neighbor graph realization in 2D [11] and in 3D [23], Euclidean minimum spanning tree realization [20], unit-length planar straight-line drawings [11], unit disk touching realization [11], etc.

The problems addressed in this paper, instead, seem to have a different nature, more related to the shapes than to the distances involved in the drawings. Hence, we had to develop a novel framework especially targeted at proving NP-hardness of three-dimensional graph drawing problems where shapes, rather than metrics, are taken into account. The 3SAT reduction framework introduced in this section can be used to show that it is NP-hard finding a 3D drawing of a graph within the orthogonal standard that satisfies a generic constraint. By using this framework in Sections 4 and 5, respectively, it is shown the NP-hardness of SHAPE GRAPH REALIZATION and of 2-BEND ROUTING. Also, in Section 6 the NP-hardness of finding 0-bends drawings is reproved.

A constraint is a statement that, given a drawing Γ of a graph G, produces a true or a false value. More formally, a *constraint* γ is a formula obtained by applying the logical operators $\wedge$, $\vee$, and $\neg$ to some atomic sentences. Each atomic sentence is a comparison ($=$, $<$, $\leqslant$, $>$, and $\geqslant$ operators allowed) between two quantities of the following:

- A constant (a real number);
- A function giving the number of bends of an edge in Γ;
- A function giving the X-, Y-, or Z-coordinate of a node or a bend in Γ.

Throughout this section, the target problem is assumed to be as follows:

Problem: TARGET PROBLEM
Instance: A graph $G = (V, E)$ and a constraint γ on the drawing of G.
Question: Does a non-intersecting 3D drawing of G exist such that γ evaluates to true?

The 3SAT problem is as follows:

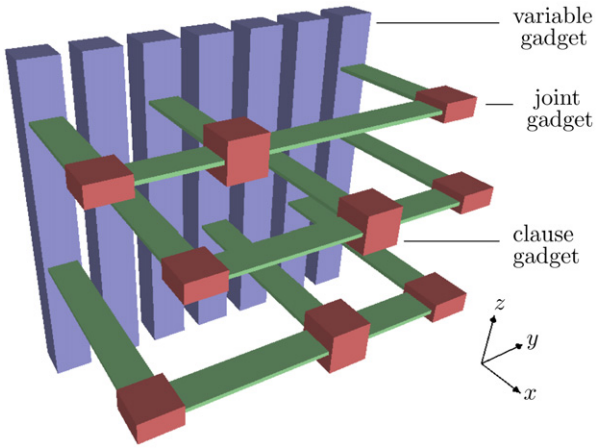

Fig. 2. The basic structure of an instance of the target problem.

Problem: 3-SATISFIABILITY (3SAT)
Instance: A set of clauses $\{c_1, c_2, \ldots, c_m\}$, each containing three literals from a set of Boolean variables $\{v_1, v_2, \ldots, v_n\}$.
Question: Can truth values be assigned to the variables so that each clause contains at least one true literal?

Observe that in the 3SAT definition the three variables contributing to the same clause can be assumed to be different, since multiple occurrences can be trivially eliminated.

Given a 3SAT instance ϕ, the 3SAT reduction framework specifies how to build an instance $I_\phi = (G_\phi, \gamma_\phi)$ of the target problem such that ϕ admits a solution if and only if I_ϕ does. $G_\phi = (V_\phi, E_\phi)$ is composed of different types of gadgets connected together. The bounding boxes of the gadgets are depicted in Fig. 2, while the interior components are not shown and depend on the specific target problem. The three basic gadgets are the following.

Variable gadget. Instance I_ϕ has a variable gadget V_i for each Boolean variable v_i of ϕ. Fig. 2 shows the variable gadgets as tall vertical blocks placed in a row along the Y-axis so that if $i < j$ then variable gadget V_i has lower Y-coordinates than variable gadget V_j.

Clause gadget. Instance I_ϕ has one clause gadget C_i for each clause $c_i = l_h \vee l_j \vee l_k$ of ϕ. Clause gadgets are represented in Fig. 2 as small cubes. Denoted with v_h, v_j, and v_k the variables of literals l_h, l_j, and l_k, respectively, and assumed that $h < j < k$, clause gadget C_i is placed directly in front of the variable gadget V_j.

Joint gadget. For each clause $c_i = l_h \vee l_j \vee l_k$ of ϕ, I_ϕ has two joint gadgets $J_{i,h}$ and $J_{i,k}$, depicted in Fig. 2 as flat blocks placed in front of the variable gadgets V_h and V_k, respectively.

In order to use the 3SAT reduction framework for the NP-hardness proof of a specific target problem a complete specification must be provided, where a *specification* for the 3SAT reduction framework is defined as follows.

- Construction rules describing how, starting from an instance ϕ of the 3SAT problem, variable gadgets, joint gadgets, and clause gadgets are built and connected together and an instance $I_\phi = (G_\phi, \gamma_\phi)$ of the target problem is obtained.
- For each variable gadget V_i a partition of the non-intersecting drawings of G_ϕ satisfying the constraint γ_ϕ into two sets, denoted T_{V_i} and F_{V_i}.
- For each joint gadget $J_{i,k}$ a partition of the non-intersecting drawings of G_ϕ satisfying the constraint γ_ϕ into two sets, denoted $T_{J_{i,k}}$ and $F_{J_{i,k}}$.

Intuitively, drawings in T_{V_i} are those drawings that are meant to correspond to a true value for variable v_i, while drawings in F_{V_i} are meant to correspond to a false value for variable v_i. Analogously, drawings in $T_{J_{i,k}}$ are such that

$J_{i,k}$ is transmitting a true value from the variable gadget V_k to the clause gadget C_i, while drawings in $T_{J_{i,k}}$ are such that $J_{i,k}$ is transmitting a false value to C_i.

A specification is said to be *compliant* if, for any 3SAT instance ϕ, the following four statements hold.

Statement 1. *Given an instance ϕ of* 3SAT *problem, the corresponding instance $I_\phi = (G_\phi, \gamma_\phi)$ of the target problem can be constructed in polynomial time.*

Statement 2. *If a non-intersecting drawing of $G_\phi = (V_\phi, E_\phi)$ satisfying γ_ϕ exists, then it belongs to $T_{J_{i,h}}$ ($T_{J_{i,k}}$) if and only if it belongs to T_{V_h} (T_{V_k}).*

Statement 3. *For each clause $c_i = l_h \vee l_j \vee l_k$, where l_h (l_j, l_k, respectively) is the positive or the negative literal of variable v_h (v_j, v_k, respectively), and for each non-intersecting drawing Γ of $G_\phi = (V_\phi, E_\phi)$ satisfying γ_ϕ at least one of the following conditions holds*:

(1) $\Gamma \in T_{J_{i,h}}$ *and* l_h *is the positive literal of* v_h;
(2) $\Gamma \in F_{J_{i,h}}$ *and* l_h *is the negative literal of* v_h;
(3) $\Gamma \in T_{V_j}$ *and* l_j *is the positive literal of* v_j;
(4) $\Gamma \in F_{V_j}$ *and* l_j *is the negative literal of* v_j;
(5) $\Gamma \in T_{J_{i,k}}$ *and* l_k *is the positive literal of* v_k;
(6) $\Gamma \in F_{J_{i,k}}$ *and* l_k *is the negative literal of* v_k.

Statement 4. *Consider a truth assignment to the variables $v_1, \ldots v_n$ satisfying ϕ. The set $\bigcap_{i=1}^{n} A_i$, where $A_i = T_{V_i}$ if v_i is true and $A_i = F_{V_i}$ if v_i is false, is non-empty.*

Theorem 1. *Given a target problem, whose instance is a graph $G = (V, E)$ and a constraint γ expressed with respect to its nodes and edges, if it admits a compliant specification for the* 3SAT *reduction framework, then finding a non-intersecting 3D orthogonal drawing of G satisfying the constraint γ is NP-hard.*

Proof. First, we show that a 3SAT instance ϕ admits a solution if and only if G_ϕ admits a non-intersecting drawing satisfying γ_ϕ. Consider a non-intersecting drawing Γ of $G_\phi = (V_\phi, E_\phi)$ satisfying γ_ϕ. Determine an assignment of truth values to the Boolean variables that satisfies ϕ by taking $v_i = \mathit{true}$ if $\Gamma \in T_{V_i}$ and $v_i = \mathit{false}$ if $\Gamma \in F_{V_i}$. In fact, because of Statements 2 and 3 we have that each clause $c_i = l_h \vee l_j \vee l_k$ has at least one true literal and thus ϕ is satisfied. Conversely, consider an assignment of truth values to the Boolean variables that satisfies ϕ. Statement 4 guarantees the existence of a drawing of G_ϕ satisfying γ_ϕ. The proof is completed by considering that 3SAT is NP-complete [9] and by showing that instance $I_\phi = (G_\phi, \gamma_\phi)$ can be obtained in polynomial time, which is guaranteed by Statement 1. □

4. Fixing the shape and searching for coordinates

In this section we consider the SHAPE GRAPH REALIZATION problem, that is the problem of finding a non-intersecting drawing for a graph whose orthogonal shape is fixed. We first prove the following lemma.

Lemma 1. SHAPE GRAPH REALIZATION *is in NP.*

Proof. Given an instance $I = (G, \gamma)$ of the SHAPE GRAPH REALIZATION problem, the search for a 3D orthogonal drawing of $G = (V, E)$ satisfying γ can be restricted to orthogonal grid drawings. Further, we can restrict to those drawings where each axis-orthogonal plane intersecting the drawing contains a node or a bend. Since the number of nodes and bends in any 3D drawing satisfying γ can be easily computed, an upper bound for the sides of the bounding box of such drawings can be obtained. This gives an upper bound for the maximum length λ of any axis-aligned segment of the edges. A non-deterministic Turing machine can be devised that assigns all possible integer lengths between one and λ to the edge segments and then checks in polynomial time if a non-intersecting drawing is produced. □

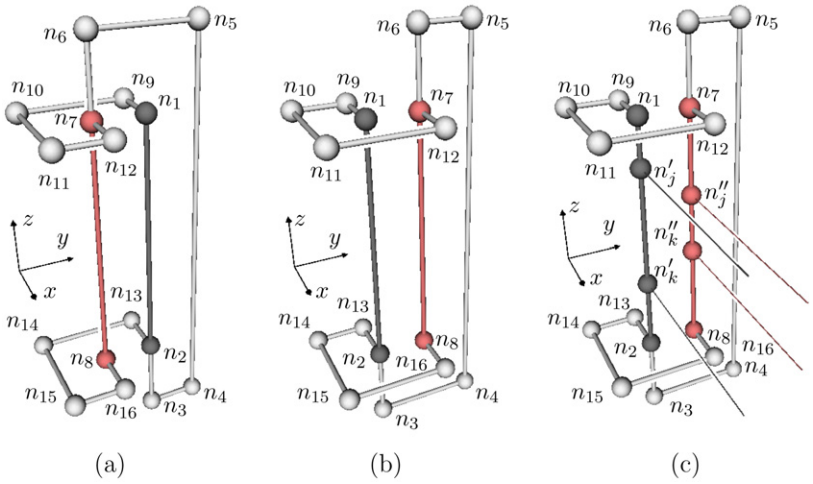

Fig. 3. (a) A drawing of the variable gadget V_i belonging to T_{V_i} and (b) a drawing belonging to F_{V_i}. (c) Nodes n'_j, n''_j, n'_k, and n''_k are inserted in order to transmit the geometric constraints to the clause gadgets of clauses c_j and c_k, respectively.

In the remaining part of this section we show that SHAPE GRAPH REALIZATION is NP-hard. Therefore, the following theorem holds.

Theorem 2. SHAPE GRAPH REALIZATION *is NP-complete.*

We prove that SHAPE GRAPH REALIZATION is NP-hard by using the framework introduced in Section 3 in order to reduce an instance of 3SAT to an instance of SHAPE GRAPH REALIZATION. In the following sections it is shown how the variable gadgets (Section 4.1), joint gadgets (Section 4.2), and clause gadgets (Section 4.3) are built. Section 4.4 contains the hardness proof.

4.1. The variable gadget

The heart of the variable gadget V_i, depicted in Fig. 3, is the path $n_1 \xrightarrow{z-} n_2 \xrightarrow{z-} n_3 \xrightarrow{y+} n_4 \xrightarrow{z+} n_5 \xrightarrow{y} n_6 \xrightarrow{z-} n_7 \xrightarrow{z-} n_8$, whose nodes lie on the same X-plane. Further, the path $n_1 \xrightarrow{x-} n_9 \xrightarrow{y} n_{10} \xrightarrow{x+} n_{11} \xrightarrow{y+} n_{12} \xrightarrow{x-} n_7$ constrains nodes n_1 and n_7 to share the same Z-plane. Analogously, path $n_2 \xrightarrow{x-} n_{13} \xrightarrow{y} n_{14} \xrightarrow{x+} n_{15} \xrightarrow{y+} n_{16} \xrightarrow{x-} n_8$ constrains nodes n_2 and n_8 to share the same Z-plane. We define T_{V_i} as the set of non-intersecting drawings of G_ϕ satisfying the direction constraints and such that $n_1 >_y n_7$ (as in Fig. 3(a)). Analogously, we define F_{V_i} as the set of non-intersecting drawings of G_ϕ satisfying the direction constraints and such that $n_1 <_y n_7$ (as in Fig. 3(b)). The following lemma guarantees that T_{V_i} and F_{V_i} form a bipartition of the non-intersecting drawings of G_ϕ satisfying the direction constraints.

Lemma 2. *For each $i = 1, \ldots, n$, any non-intersecting drawing of $G_\phi = (V_\phi, E_\phi)$ satisfying the direction constraints either belongs to T_{V_i} or to F_{V_i}.*

Proof. Since in any drawing of G_ϕ we have that $n_1 =_x n_7$ and $n_1 =_z n_7$, if the drawing is non-intersecting, then either $n_1 >_y n_7$ or $n_1 <_y n_7$, which is exactly what is requested by the definition of T_{V_i} or F_{V_i}, respectively. □

For each clause c_j of the 3SAT formula in which the variable participates we insert a node n'_j between nodes n_1 and n_2 and a node n''_j between nodes n_7 and n_8. In any drawing Γ of G_ϕ satisfying the direction constraints, nodes n'_j and n''_j have the same relative position with respect to the Y-axis as n_1 and n_7, i.e., $n'_j >_y n''_j$ if $\Gamma \in T_{V_i}$ and $n'_j <_y n''_j$ if $\Gamma \in F_{V_i}$. Suitable edges attached to the nodes n'_j and n''_j along the protruding lines shown in Fig. 3(c) transmit the above constraints from V_i to the clause gadget C_j (possibly via joint gadget $J_{i,j}$). Note that nodes n'_j and n''_j need not lie in the same Z-plane.

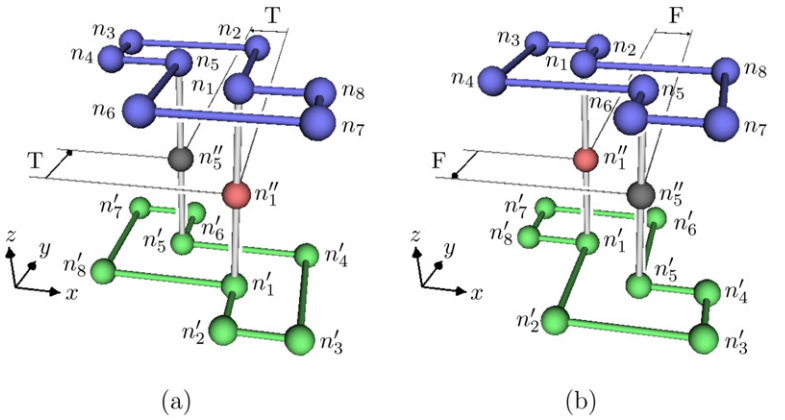

Fig. 4. (a) A drawing of the joint gadget $J_{i,h}$ belonging to $T_{J_{i,h}}$ and (b) a drawing belonging to $F_{J_{i,h}}$.

4.2. The joint gadget

Given a clause $c_i = l_h \vee l_j \vee l_k$, the joint gadget $J_{i,k}$ can be obtained from the reflected image with respect to the Y-axis of the joint gadget $J_{i,h}$. Thus, in the following we will only describe the joint gadget $J_{i,h}$, which is depicted in Fig. 4 and composed of two cycles $\alpha = n_1 \xrightarrow{y+} n_2 \xrightarrow{x-} n_3 \xrightarrow{y} n_4 \xrightarrow{x+} n_5 \xrightarrow{y} n_6 \xrightarrow{x+} n_7 \xrightarrow{y+} n_8 \xrightarrow{x-} n_1$, and $\alpha' = n'_1 \xrightarrow{y} n'_2 \xrightarrow{x+} n'_3 \xrightarrow{y+} n'_4 \xrightarrow{x-} n'_5 \xrightarrow{y+} n'_6 \xrightarrow{x-} n'_7 \xrightarrow{y} n'_8 \xrightarrow{x+} n'_1$. Nodes n_1 and n'_1 are connected by a path $n_1 \xrightarrow{z-} n''_1 \xrightarrow{z-} n'_1$ while nodes n_5 and n'_5 are connected by the path $n_5 \xrightarrow{z-} n''_5 \xrightarrow{z-} n'_5$.

We define $T_{J_{i,h}}$ as the set of non-intersecting drawings of G_ϕ satisfying the direction constraints and such that $n''_5 <_x n''_1$ (as in Fig. 4(a)). Analogously, we define $F_{J_{i,h}}$ as the set of non-intersecting drawings of G_ϕ satisfying the direction constraints and such that $n''_5 >_x n''_1$ (as in Fig. 4(b)).

Lemma 3. *For each $i = 1, \ldots, m$, any non-intersecting drawing of $G_\phi = (V_\phi, E_\phi)$ satisfying the direction constraints either belongs to $T_{J_{i,h}}$ or to $F_{J_{i,h}}$.*

Proof. Suppose for a contradiction that Γ is a non-intersecting drawing of G_ϕ and that Γ does not belong to $T_{J_{i,h}}$ nor to $F_{J_{i,h}}$. In Γ we have that $n''_5 =_x n''_1$, which implies that n_1, n'_1, n_5, and n'_5 lie in the same X-plane. Three are the cases: (i) if $n''_5 >_y n''_1$ then n_5 intersects edge (n_1, n_2); (ii) if $n''_5 <_y n''_1$ then n'_1 intersects edge (n'_5, n'_6); and (iii) if $n''_5 =_y n''_1$ then n_5 overlaps with n_1 and n'_5 overlaps with n'_1. In all three cases we have a contradiction. □

The following lemma shows how a geometric constraint on the relative position of nodes n''_5 and n''_1 with respect to the X-axis has an effect on their relative position with respect to the Y-axis in any non-intersecting drawing of the joint gadget.

Lemma 4. *In any non-intersecting drawing of $G_\phi = (V_\phi, E_\phi)$ satisfying the direction constraints either $n''_5 >_y n''_1$ and $n''_5 <_x n''_1$ or $n''_5 <_y n''_1$ and $n''_5 >_x n''_1$.*

Proof. A drawing without intersections of the joint gadget where $n''_5 >_y n''_1$ and $n''_5 <_x n''_1$ is shown in Fig. 5(a). Analogously, a drawing without intersections of the joint gadget where $n''_5 <_y n''_1$ and $n''_5 >_x n''_1$ is shown in Fig. 5(d). From Fig. 5(b) it is easy to see that if $n''_5 >_y n''_1$ and $n''_5 >_x n''_1$ then α necessarily intersects. Analogously, from Fig. 5(c) it is easy to see that if $n''_5 <_y n''_1$ and $n''_5 <_x n''_1$ then α' necessarily intersects. □

4.3. The clause gadget

The clause gadget is depicted in Fig. 6. Its main component is the path $\alpha = n_1 \xrightarrow{y+} n_2 \xrightarrow{x-} n_3 \xrightarrow{y} n_4 \xrightarrow{x+} n_5 \xrightarrow{y+} n_6 \xrightarrow{x+} n_7$, whose nodes lie on the same Z-plane. Attached to α are the paths $n'_1 \xrightarrow{z-} n_1 \xrightarrow{z-} n''_1$, $n'_2 \xrightarrow{z-} n_2 \xrightarrow{z-} n''_2$, $n'_6 \xrightarrow{z-} n_6 \xrightarrow{z-} n''_6$, and $n'''_7 \xrightarrow{x+} n'_7 \xrightarrow{z-} n_7 \xrightarrow{z-} n''_7 \xrightarrow{x+} n_8 \xrightarrow{y+} n_9 \xrightarrow{x-} n''_2$.

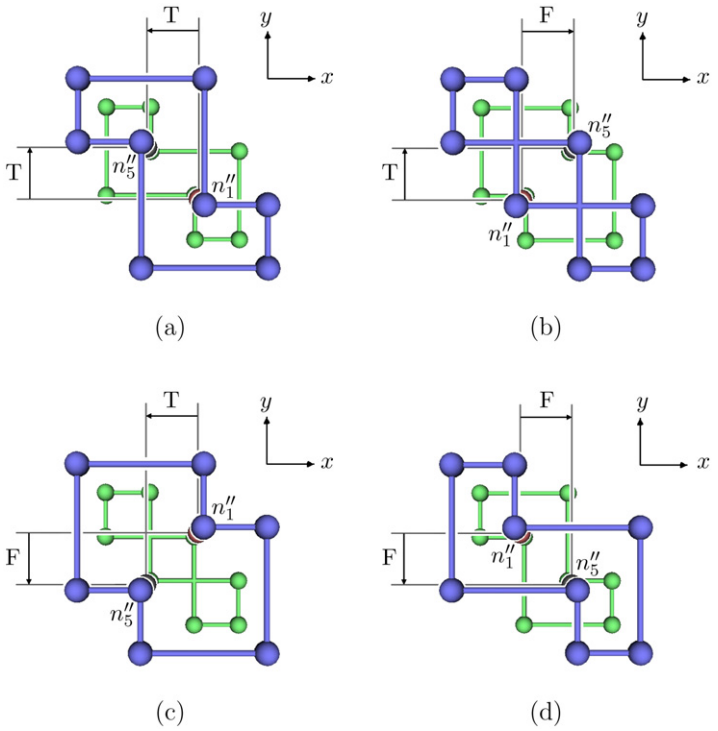

Fig. 5. The joint gadget admits a drawing without intersection if and only if $n''_5 >_y n''_1$ and $n''_5 <_x n''_1$ (a) or $n''_5 <_y n''_1$ and $n''_5 >_x n''_1$ (d).

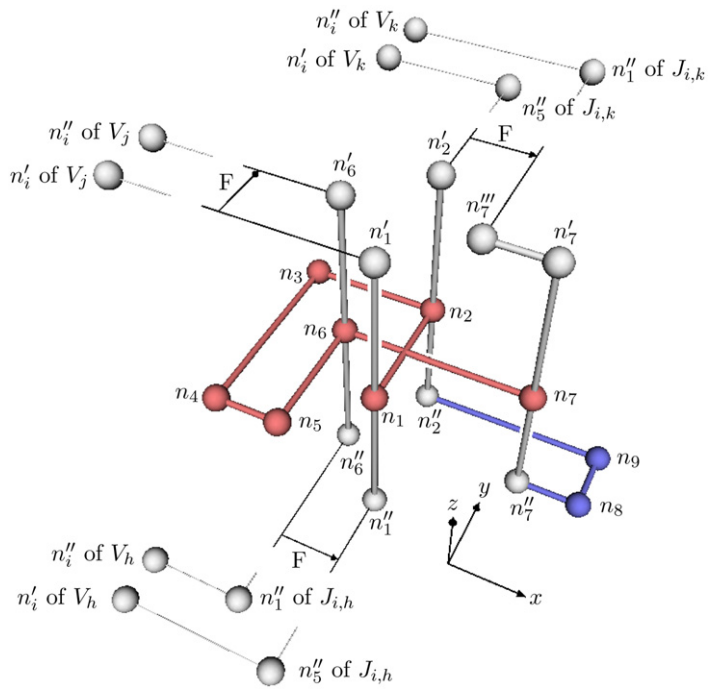

Fig. 6. Clause gadget C_i for clause $c_i = l_h \vee l_j \vee l_k$ when l_h, l_j, and l_k are the positive literals of the variables v_h, v_j, and v_k, respectively.

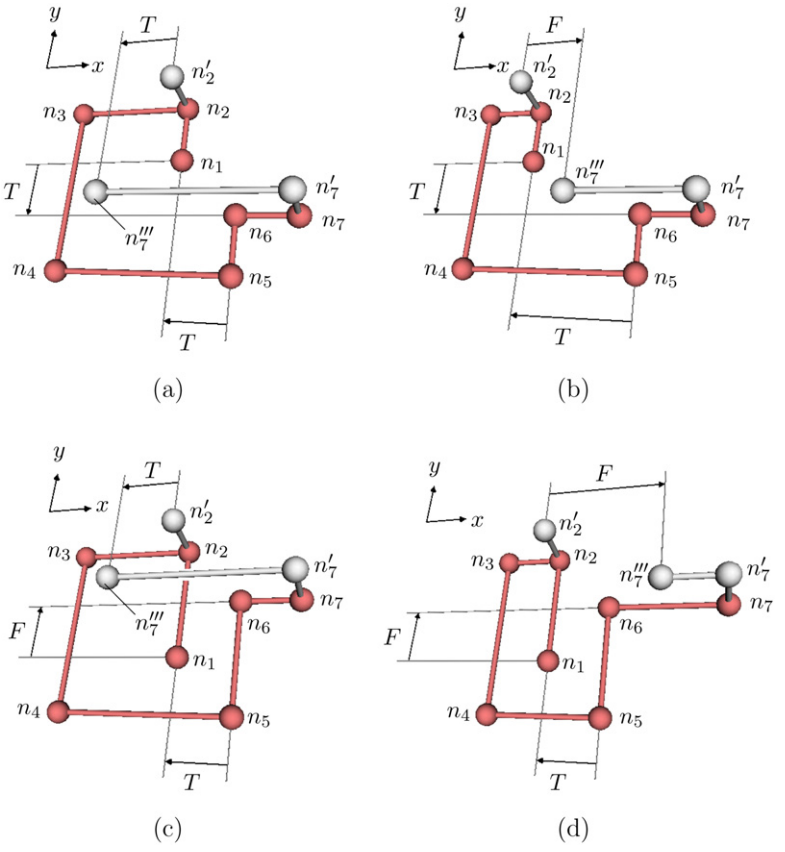

Fig. 7. The first four cases used in the proof of Lemma 5.

The following lemma holds:

Lemma 5. *Clause gadget C_i admits a drawing without intersections if and only if $n_1 <_x n_6$ or $n_1 >_y n_6$ or $n'''_7 <_x n'_2$.*

Proof. Figs. 7 and 8 show that α admits a drawing without intersections if at least one of the three following conditions holds: $n_1 <_x n_6$ (as shown in Figs. 7(a), 7(b), 7(c), and 7(d)), $n_1 >_y n_6$ (as shown in Figs. 7(a), 7(b), 8(a), and 8(b)), or $n'''_7 <_x n'_2$ (as shown in Figs. 7(a), 7(c), 8(a), and 8(c)).

Conversely, suppose that $n_6 \leqslant_x n_1$, $n_1 \leqslant_y n_6$, and $n'''_7 \geqslant_x n'_2$ (as depicted in Figs. 6 and 8(d)). We have that α necessarily intersects. In fact, condition $n'''_7 \geqslant_x n'_2$ implies $n'_7 >_x n'_2$, which in turn implies $n_7 >_x n_2$, or, equivalently, $n_1 <_x n_7$. Also, because of path $n''_7 \xrightarrow{x+} n_8 \xrightarrow{y+} n_9 \xrightarrow{x-} n''_2$, we have $n''_7 <_y n''_2$, which implies $n_6 <_y n_2$. An intersection between edges $n_1 \xrightarrow{y+} n_2$ and $n_6 \xrightarrow{x+} n_7$ follows from $n_6 \leqslant_x n_1 <_x n_7$ and $n_1 \leqslant_y n_6 <_y n_2$ and from the fact that the nodes of α lie on the same Z-plane. □

4.4. The hardness proof

We now describe how the various gadgets are connected together, and we show that the whole construction is a compliant specification for the 3SAT reduction framework.

Joint gadget $J_{i,h}$ is connected to both variable gadget V_h and clause gadget C_i. In particular, n'_i of V_h is connected to n''_5 of $J_{i,h}$ with the edge $n'_i \xrightarrow{x+} n''_5$ and n''_i of V_h is connected to n''_1 of $J_{i,h}$ with the edge $n''_i \xrightarrow{x+} n''_1$ (see Fig. 6). Due to the above described connections the following lemma holds:

Lemma 6. *Statement 2 holds; that is, for any* 3SAT *instance ϕ, if a non-intersecting drawing of $G_\phi = (V_\phi, E_\phi)$ exists such that the edges have the prescribed shape, then it belongs to $T_{J_{i,h}}$ if and only if it belongs to T_{V_h}.*

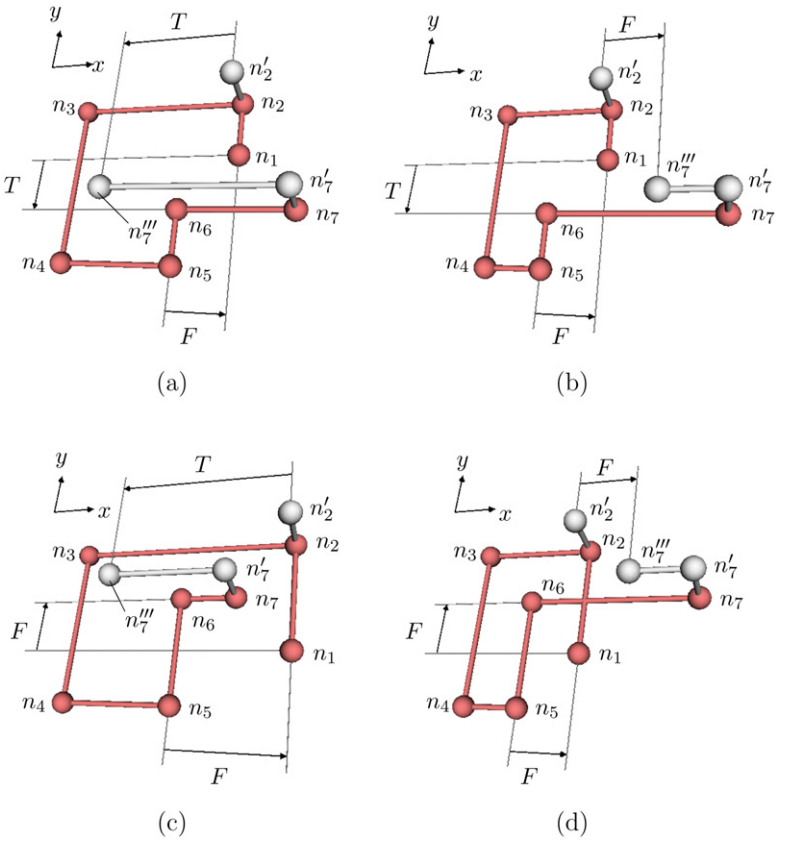

Fig. 8. Four cases used in the proof of Lemma 5.

Proof. The statement follows by considering that n''_1 and n''_5 of $J_{i,h}$ share their Y-coordinates with n''_i and n'_i of V_h, respectively, and by applying Lemma 4. □

Each clause gadget C_i, corresponding to clause $c_i = l_h \vee l_j \vee l_k$, is connected to joint gadget $J_{i,h}$, variable gadget V_j, and joint gadget $J_{i,k}$ as follows (see also Fig. 6):

- If l_h is the positive (negative) literal of variable v_h, we attach nodes n''_1 and n''_5 of the joint gadget $J_{i,h}$ to nodes n''_6 and n''_1 (n''_1 and n''_6), respectively.
- If l_j is the positive (negative) literal of variable v_j, we attach nodes n'_i and n''_i of the variable gadget V_j to n'_1 and n'_6 (n'_6 and n'_1), respectively.
- If l_k is the positive (negative) literal of variable v_k, we attach nodes n''_1 and n''_5 of the joint gadget $J_{i,k}$ to nodes n'_2 and n'''_7 (n'''_7 and n'_2), respectively.

Lemma 7. *Statement 3 holds; that is, for each clause $c_i = l_h \vee l_j \vee l_k$ and for each non-intersecting drawing Γ of $G_\phi = (V_\phi, E_\phi)$ such that the edges have the prescribed shape, at least one of the following conditions holds*:

(1) $\Gamma \in T_{J_{i,h}}$ *and* l_h *is the positive literal of* v_h;
(2) $\Gamma \in F_{J_{i,h}}$ *and* l_h *is the negative literal of* v_h;
(3) $\Gamma \in T_{V_j}$ *and* l_j *is the positive literal of* v_j;
(4) $\Gamma \in F_{V_j}$ *and* l_j *is the negative literal of* v_j;
(5) $\Gamma \in T_{J_{i,k}}$ *and* l_k *is the positive literal of* v_k;
(6) $\Gamma \in F_{J_{i,k}}$ *and* l_k *is the negative literal of* v_k.

Proof. Due to Lemma 5 each clause gadget C_i admits a drawing without intersections if and only if: (i) $n_1 <_x n_6$, or (ii) $n_1 >_y n_6$, or (iii) $n_7''' <_x n_2'$. Considering the connection rules described above these three conditions can be rewritten as:

(i) In $J_{i,h}$

$$\begin{cases} n_5'' <_x n_1'' & \text{if } l_h \text{ is the positive literal of } v_h, \\ n_1'' <_x n_5'' & \text{otherwise.} \end{cases}$$

(ii) In V_j

$$\begin{cases} n_i' >_y n_i'' & \text{if } l_j \text{ is the positive literal of } v_j, \\ n_i'' >_y n_i' & \text{otherwise.} \end{cases}$$

(iii) In $J_{i,k}$

$$\begin{cases} n_1'' <_x n_5'' & \text{if } l_k \text{is the positive literal of } v_k, \\ n_5'' <_x n_1'' & \text{otherwise.} \end{cases}$$

Recalling the definitions of true and false variable gadget and true and false joint gadget, the six conditions of the statement follow. □

Lemma 8. *Statement 4 holds; that is, if ϕ admits a solution, then $G_\phi = (V_\phi, E_\phi)$ admits a non-intersecting drawing whose edges have the prescribed shape.*

Proof. Consider a truth assignment satisfying ϕ. If variable v_i is true (false) we can use the true (false) drawing of variable gadget V_i depicted in Fig. 3(a) (Fig. 3(b)). Also, given a clause $c_i = l_h \vee l_j \vee l_k$, if variable v_h is true (false) we can use the true (false) drawing of variable gadget $J_{i,h}$ depicted in Fig. 4(a) (Fig. 4(b)). If variable v_k is true (false) the case is analogous to the one of variable v_h. In order to draw the clause gadget C_i without intersection a suitable drawing can be selected between the ones depicted in Figs. 7(a)–(d) and 8(a)–(c). □

Now we can prove the following lemma.

Lemma 9. SHAPE GRAPH REALIZATION *is NP-hard.*

Proof. The proof is based on the fact that a compliant specification can be found for the 3SAT reduction framework introduced in Section 3. Lemmas 6, 7, and 8 prove that Statements 2, 3, and 4 hold, respectively. Also, given a 3SAT instance ϕ, the corresponding SHAPE GRAPH REALIZATION instance I_ϕ can be built in polynomial time (Statement 1 holds). Therefore, the construction rules described in Sections 4.1, 4.2, and 4.3 correspond to a compliant specification for the 3SAT reduction framework, and Theorem 1 applies. □

5. Fixing the coordinates and searching for a shape

In this section we tackle the reverse problem with respect to the one addressed in Section 4, that is, the problem of finding a routing for the edges when the position of the nodes is fixed. We first show in Section 5.1 that 0-, 1-, and 6-BEND ROUTING are feasible and then in Section 5.2 that 2-BEND ROUTING is NP-hard. It is easy to show the following lemma.

Lemma 10. *N*-BEND ROUTING *is in NP.*

Proof. First note that a graph admits a 3D orthogonal drawing with nodes at prescribed positions if and only if it admits an orthogonal grid drawing such that each pair of nodes have the same relative position with respect to the X-, Y-, and Z-axis as the one prescribed. It follows that in order to search for a solution of N-BEND ROUTING we may restrict to consider orthogonal drawings on the three-dimensional grid, searching for one that has the nodes in

the same relative position. Analogously to the proof of Lemma 1, we restrict to those drawings where each axis-orthogonal plane intersects at least a node or a bend. An upper bound for the bounding box of such drawings can be computed by considering that an N-bend drawing of a graph of maximum degree six has at most $3n \times N$ bends, where n is the number of vertices of the graph. Therefore, if the graph admits an N-bend drawing, then it admits one in a bounding box whose sides span $n(3N + 1)$ grid planes. A non-deterministic Turing machine could generate all grid drawings of the graph within the computed bounding box and then check in polynomial time whether each pair of nodes has the same relative position as the one specified by the 2-BEND ROUTING instance. □

Due to Lemma 10 and due to the NP-hardness of 2-BEND ROUTING which is proved in Section 5.2 (Lemma 19 of Section 5.2.5) the following theorem holds.

Theorem 3. 2-BEND ROUTING *is NP-complete.*

5.1. *Feasibility of* 0-, 1-, *and* 6-BEND ROUTING

First, we consider the problem of routing edges when zero bends are allowed.

Lemma 11. 0-BEND ROUTING *can be answered in* $O(|V|^2)$ *time.*

Proof. A 0-bend drawing exists if and only if: (i) each pair of adjacent nodes share two coordinates and (ii) no two edges intersect. The first condition can be checked in linear time. Since the number of edges in bounded-degree graph is $O(|V|)$, the second condition can be checked in $O(|V|^2)$ time. □

Second, we tackle the analogous problem where one bend per edge is allowed.

Lemma 12. 1-BEND ROUTING *can be answered in* $O(|V|^2)$ *time.*

Proof. Suppose all adjacent nodes share at least a coordinate. This is a necessary condition for the existence of a 1-bend drawing and can be checked in linear time. We reduce 1-BEND ROUTING to 2SAT, the version of the SAT problem in which each clause has exactly two literals. An instance of 2SAT can be solved in linear time in the size of the formula [1] providing a solution for the corresponding instance of 1-BEND ROUTING.

For each edge e_i we introduce the Boolean variable v_i. If e_i joins a pair of nodes sharing two coordinates, then e_i admits a single route, labeled with the direct literal v_i of variable v_i. Otherwise, if e_i joins two nodes that share a single coordinate, then e_i can be routed in two different ways, that we arbitrarily label v_i and $\bar{v}_i$. Intuitively, when $v_i = \mathit{true}$, the route labeled v_i is used by the 1-bend drawing, while when $v_i = \mathit{false}$ the route labeled $\bar{v}_i$ is used instead. We construct the formula of the 2SAT problem as follows. For any route labeled v_i for which $\bar{v}_i$ does not exist, we introduce clause $(v_i \vee v_i)$, whose purpose is to make sure that such a route is chosen for edge e_i. For any pair of intersecting routes (v_h, v_k) $((\bar{v}_h, v_k)$, $(\bar{v}_h, \bar{v}_k)$, respectively) we introduce clause $(\bar{v}_h \vee \bar{v}_k)$ $((v_h \vee \bar{v}_k)$, $(v_h \vee v_k)$, respectively), whose purpose is to make sure that the two intersecting routes are not simultaneously chosen for e_h and e_k. We impose that all the constraints are simultaneously satisfied by considering the Boolean "and" of all the clauses. Since each clause has two literals, we have mapped the 1-BEND ROUTING instance to a 2SAT formula. The graph admits a non-intersecting drawing with one bend per edge maximum and with the nodes at the specified positions if and only if the 2SAT formula admits a solution. Since the number of clauses introduced is $O(|V|^2)$, the statement follows. □

We show that 6-BEND ROUTING can always be answered in the affirmative, by producing an algorithm that computes a non-intersecting 6-bend drawing of a graph $G = (V, E)$ with nodes at prescribed positions. The drawing algorithm, that runs in $O(|V| \log |V|)$ time, takes advantage of the relative coordinates scenario [26], where it is possible to insert an axis-perpendicular plane in the drawing in constant time. X-, Y-, and Z-planes are kept into three double linked lists called X-, Y-, and Z-list, respectively. Nodes and bends are not explicitly given coordinates, but are linked to the X-, Y-, and Z-plane that contains them. An axis-perpendicular plane can be inserted in the drawing

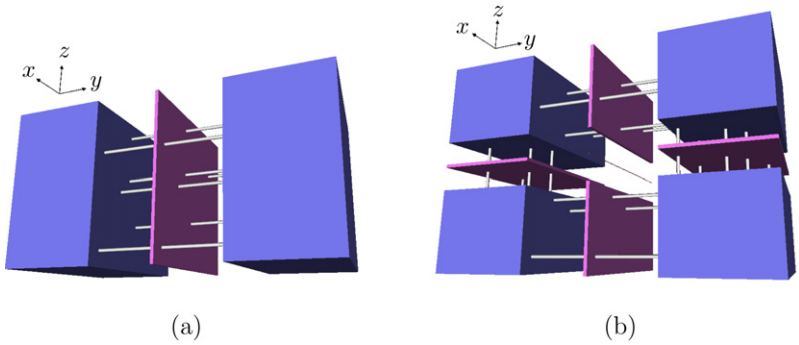

Fig. 9. (a) Inserting an orthogonal plane in a drawing in the relative coordinates scenario. (b) If two perpendicular planes are inserted the grid line common to the two planes does not intersect any node or edge of the drawing.

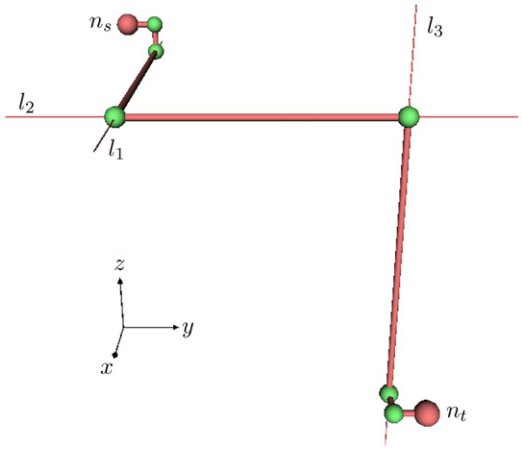

Fig. 10. The insertion of an edge from n_s to n_t with six bends.

in constant time by locally changing the corresponding list. Hence, the X-, Y-, and Z-list contains the X-, Y-, and Z-planes sorted based on their coordinates. Therefore, any algorithm based on this approach runs in $\Omega(|V| \log |V|)$ time. Our algorithm starts by sorting the nodes in ascending order according to their X-, Y-, and Z-coordinates. Second, a drawing where only the nodes are present is built. Finally, edges are added one at a time, routing them without introducing intersections by using the same construction rule described in the proof of Lemma 19 of [33].

In order to insert an edge in the drawing without introducing intersections, consider the operation of inserting a plane perpendicular to the Y-axis (see Fig. 9(a)). After the insertion, no node or edge will lay on the newly inserted plane, although some edges may perpendicularly intersect the plane. Now consider a second insertion of a plane perpendicular to the Z-axis (see Fig. 9(b)). After this second insertion, the X-parallel grid line that is common to both the two inserted planes is not intersected by any edge or node of the drawing. The algorithm uses this strategy of plane insertion in order to create the space to route each edge from the source node to the target node.

Suppose that edges $e_1, e_2, \ldots, e_{i-1}$ have been inserted and that edge e_i needs to be added to the drawing, connecting node n_s to node n_t. Consider the most unfavorable case in which n_s and n_t lay on different X-, Y-, and Z-planes and suppose e_i enters n_s along the A_s-axis and enters n_t along the A_t-axis. The route from n_s to n_t uses three grid lines l_1, l_2, and l_3, introduced with the technique described above. Line l_1 is orthogonal to the A_s-axis, l_3 is orthogonal to both the A_t-axis and l_1, and l_2 is orthogonal to both l_1 and l_3. Also, l_1 intersects l_2 and l_2 intersects l_3. Further, l_1 (l_3) is such that n_s (n_t) can be attached to it with a path of two segments of length one that are orthogonal with respect to each other and with respect to l_1 (l_3). Fig. 10 shows how edge e_i could be drawn in the case in which $n_t >_x >_y <_z n_s$, node n_s has no edge leaving in direction $y+$ and node n_t has no edge leaving in direction $y-$. Observe that, at most four planes need to be inserted in the drawing in order to create the three non-intersected grid lines needed for the new edge to be routed.

Since all other cases can be analogously handled by inserting at most six bends for each edge, the following lemma holds.

Lemma 13. *There exists an algorithm that computes in* $O(|V|\log|V|)$ *time an orthogonal drawing with nodes at prescribed positions and with a maximum of* 6 *bends per edge.*

By virtue of Lemma 13, 6-BEND ROUTING can always be answered in the affirmative in constant time.

5.2. 2-BEND ROUTING *is NP-hard*

In this section we use the 3SAT reduction framework in order to show that 2-BEND ROUTING is NP-hard. Namely, Sections 5.2.1, 5.2.2, 5.2.3, and 5.2.4 contain the construction rules for the 2-BEND ROUTING instance I_ϕ corresponding to a given 3SAT instance ϕ, while Section 5.2.5 shows that the described rules correspond to a compliant specification for the 3SAT reduction framework and provides the NP-hardness proof.

5.2.1. *The basic gadget*

The basic gadget (see Fig. 11) is used as a building block of several parts of the 2-BEND ROUTING instance. Fig. 11(a) shows its nodes and how they are connected, while Fig. 11(b) shows nodes prescribed positions. The basic gadget is composed of ten nodes. Node n_1 is connected to the three nodes n_2, n_3 and n_4. Analogously, node n_5 is connected to the three nodes n_6, n_7 and n_8. Nodes n_1 and n_5 are connected both with the single edge (n_1, n_5) and with the path of three edges $(n_1, n_{1,5})$, $(n_{1,5}, n_{5,1})$ and $(n_{5,1}, n_5)$.

As for nodes prescribed positions, they are such that:

- $n_1 <_x <_y <_z n_2 =_x =_y <_z n_3 =_x =_y <_z n_4$,
- $n_1 =_x >_y >_z n_{1,5} =_x >_y >_z n_{5,1} =_x >_y >_z n_5$, and
- $n_5 <_x >_y >_z n_6 =_x =_y >_z n_7 =_x =_y >_z n_8$.

Fig. 11(b) shows also some lines and points to help understanding the node prescribed positions and their mutual relationships. Points $p_{t,1}$, $p_{t,2}$ and $p_{t,3}$ are defined as follows. Point $p_{t,1}$ has the same coordinates of $n_{1,5}$ with the

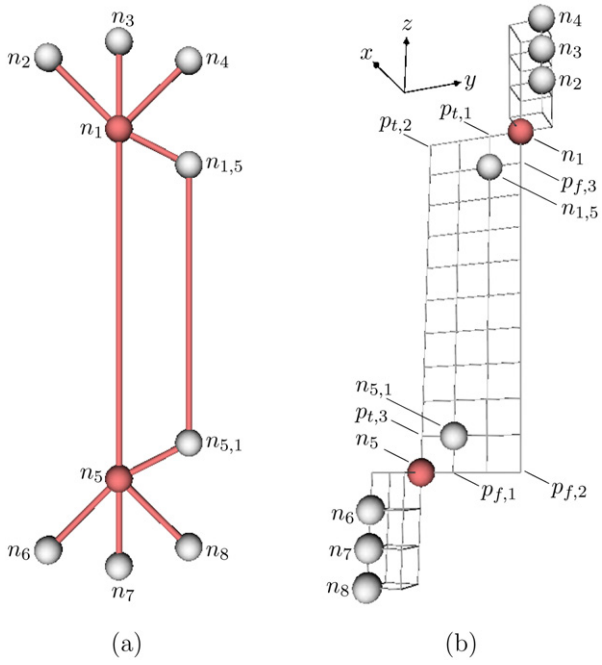

Fig. 11. (a) The basic gadget is composed of ten nodes, joined by ten edges. In (b) the prescribed node positions are represented.

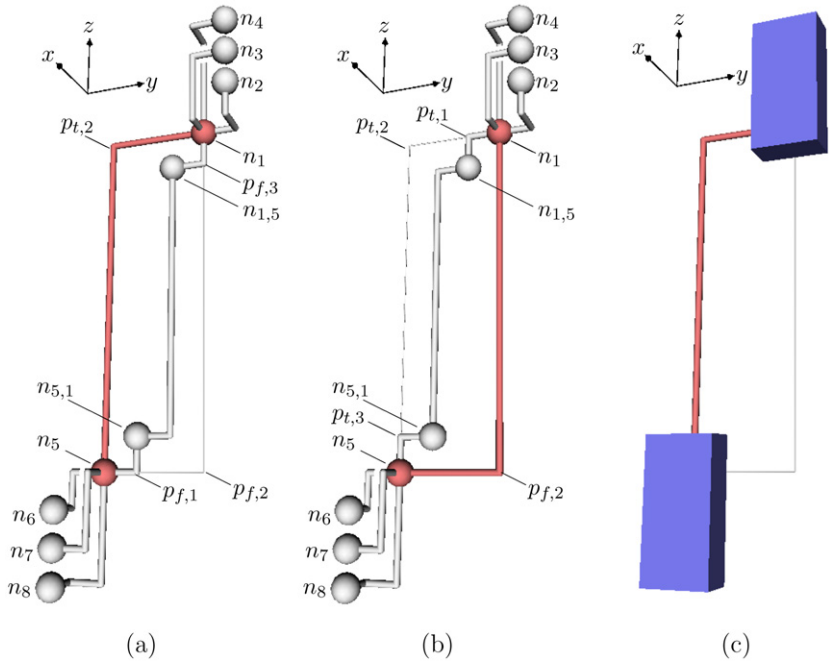

Fig. 12. (a) A true drawing and (b) a false drawing of the basic gadget. In (c) it is shown the schematic representation of the basic gadget that is used in the remaining part of the paper.

exception of the Z-coordinate which is shared with n_1. Point $p_{t,2}$ has the same coordinates of n_1 with the exception of the Y-coordinate which is shared with n_5. Point $p_{t,3}$ has the same coordinates of $n_{5,1}$ with the exception of the y coordinate which is shared with n_5. Analogously, nodes $p_{f,1}$, $p_{f,2}$ and $p_{f,3}$ can be defined by replacing n_1 with n_5 and $n_{1,5}$ with $n_{5,1}$.

Lemma 14. *In any non-intersecting* 2*-bend drawing of the basic gadget, edge* (n_1, n_5) *has exactly one bend placed either in* $p_{t,2}$ *or in* $p_{f,2}$.

Proof. Since n_1 and n_5 share the same X-plane, but do not share any axis-parallel line, edge (n_1, n_5) must lie on the X-plane in order to be drawn with a maximum of two bends. Also, due to the prescribed positions of nodes n_2, n_3, and n_4 with respect to node n_1, the three directions $x+$, $y+$, and $z+$ of n_1 are used by edges (n_1, n_2), (n_1, n_3), and (n_1, n_4) (not necessarily in this order, see Figs. 12(a) and 12(b)). Analogously, the three directions $x+$, $z-$, and $y-$ of n_5 are used by edges (n_5, n_6), (n_5, n_7), and (n_5, n_8). It follows that edge (n_1, n_5) must use the y– or z– direction of node n_1 and the y+ or z+ direction of node n_5. Due to the path of three edges $(n_1, n_{1,5})$, $(n_{1,5}, n_{5,1})$ and $(n_{5,1}, n_5)$, edge (n_1, n_5) cannot be routed with two bends, but must have a single bend placed in such a way to share its z coordinate with n_1 and its Y-coordinate with n_5 (point $p_{t,2}$), or to share its z coordinate with n_5 and its Y-coordinate with n_1 (point $p_{f,2}$). □

Given a 2-bend drawing of the basic gadget, we call *true* the basic gadget when it is drawn with the bend of edge (n_1, n_5) placed in $p_{t,2}$ (see Fig. 11(a)) and *false* the basic gadget when it is drawn with the bend of edge (n_1, n_5) placed in $p_{f,2}$ (see Fig. 11(b)). Also, in what follows we use the graphic representation of the basic gadget shown in Fig. 11(c), where the nodes n_1, n_2, n_3, n_4, and $n_{1,5}$ are replaced by their bounding box, and analogously for the nodes n_5, n_6, n_7, n_8, and $n_{5,1}$. In this representation only edge (n_1, n_5) is shown, and it is assumed to have its bend in $p_{t,2}$.

Lemma 15. *In any non-intersecting* 2*-bend drawing of the basic gadget such that the internal points of the segments* $\overline{p_{t,1}p_{t,2}}$ *and* $\overline{p_{t,2}p_{t,3}}$ *are not used by any edge of the gadget, segments* $\overline{p_{f,1}p_{f,2}}$ *and* $\overline{p_{f,2}p_{f,3}}$ *are used by edge* (n_1, n_5).

5.2.2. The variable gadget

The variable gadget V_i is composed of a single basic gadget. Given a variable gadget V_i, we define as T_{V_i} (F_{V_i}) the set of non-intersecting 2-bend drawings of $G_\phi = (V_\phi, E_\phi)$ such that the basic gadget is true (false). Lemma 14 guarantees that sets T_{V_i} and F_{V_i} correspond to a bipartition of the non-intersecting 2-bend drawings of G_ϕ.

5.2.3. The joint gadget

Fig. 13 shows how basic gadgets can be interleaved together. In fact, a basic gadget can be suitably rotated with respect to another basic gadget, and node positions can be chosen in such a way that if one basic gadget is true the other also need to be true. In particular, a variable gadget V_i can be intersected by a suitable number of basic gadgets, one for each clause in which the variable v_i participates, in order to transfer the geometric constraints determined by the drawing of V_i to the clause gadgets.

Given a clause $c_i = l_h \vee l_j \vee l_k$, where l_h (l_j, l_k, respectively) is a literal of the variable v_h (v_j, v_k, respectively) and $h < j < k$, the joint gadget $J_{i,k}$ is the reflected image with respect to the Y-axis of the joint gadget $J_{i,h}$. Thus, in the following we only describe the joint gadget $J_{i,h}$, which is depicted in Fig. 14 and built by interleaving four basic gadgets B_1, B_2, B_3, and B_4 as follows. B_1 intersects the variable gadget (not shown in Fig. 14(a)). B_2 is placed on an orthogonal plane as shown in Fig. 14(a). B_3 intersects only B_2 and is placed on a plane orthogonal to the first two (see

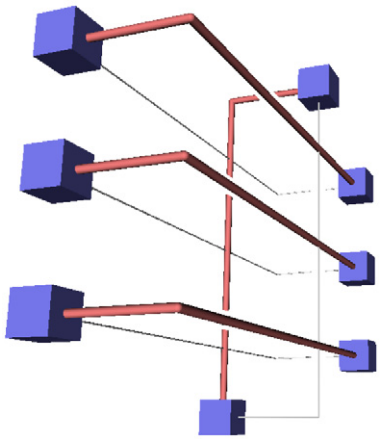

Fig. 13. Four basic gadgets interleaved in such a way that in any 2-bend drawing of them they are all true or all false.

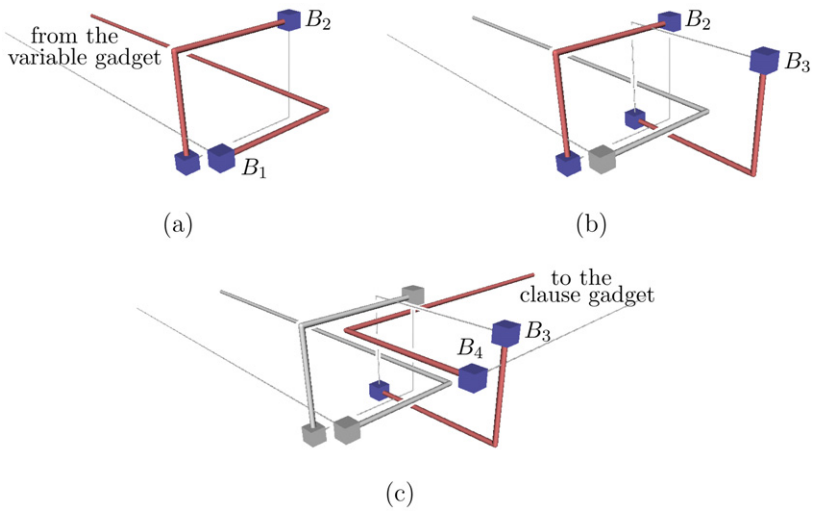

Fig. 14. Joint gadget $J_{i,h}$ is composed of four interleaved basic gadgets.

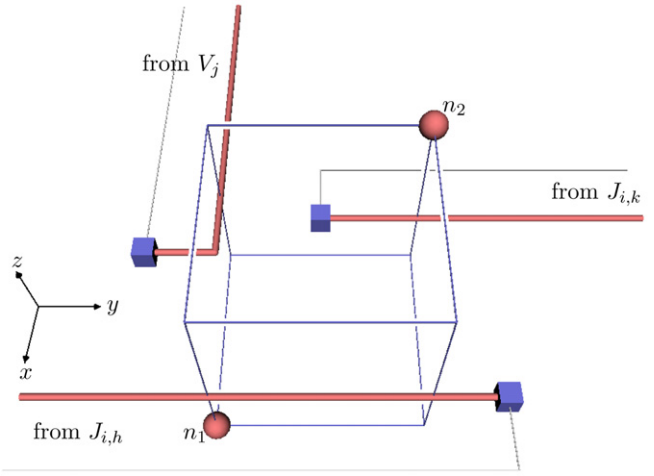

Fig. 15. The clause C_i for clause $c_i = l_h \vee l_j \vee l_k$ in the case in which l_h, l_j and l_k are negative literals: if the three variables v_h, v_h and v_k are true, there is no way of adding edge (n_1, n_2) with at most two bend.

Fig. 14(b)). Finally, B_4 is placed on a plane parallel to the first one and intersects B_3 only as shown in Fig. 14(c). We define $T_{J_{i,h}}$ ($F_{J_{i,h}}$) as the set of non-intersecting 2-bend drawings of G_ϕ satisfying γ_ϕ such that B_4 is true (false).

5.2.4. The clause gadget

The clause C_i for clause $c_i = l_h \vee l_j \vee l_k$ is shown in Fig. 15. It is composed of two nodes n_1 and n_2 placed at the opposite vertices of a cube. The two nodes are joined by edge (n_1, n_2) (not shown in Fig. 15). In any 2-bend drawing of the clause gadget edge (n_1, n_2) uses one of the four vertical edges of the cube. The basic gadget B_4 of joint gadgets $J_{i,h}$ and $J_{i,k}$ and the basic gadget coming from V_j suitably intersect the vertical edges of the cube such that only if one literal is true the clause gadget admits a non-intersecting drawing.

5.2.5. The hardness proof

By using Lemmas 14 and 15 it is easy to show the following lemma.

Lemma 16. *Statement 2 holds; that is, if a non-intersecting 2-bend drawing of G_ϕ satisfying exists with nodes at the prescribed positions, then it belongs to $T_{J_{i,h}}$ if and only if it belongs to T_{V_h}.*

Lemma 17. *Statement 3 holds; that is, for each clause $c_i = l_h \vee l_j \vee l_k$ and for each non-intersecting drawing Γ of $G_\phi = (V_\phi, E_\phi)$ with the nodes at the prescribed positions, at least one of the following conditions holds:*

(1) $\Gamma \in T_{J_{i,h}}$ *and* l_h *is the positive literal of* v_h;
(2) $\Gamma \in F_{J_{i,h}}$ *and* l_h *is the negative literal of* v_h;
(3) $\Gamma \in T_{V_j}$ *and* l_j *is the positive literal of* v_j;
(4) $\Gamma \in F_{V_j}$ *and* l_j *is the negative literal of* v_j;
(5) $\Gamma \in T_{J_{i,k}}$ *and* l_k *is the positive literal of* v_k;
(6) $\Gamma \in F_{J_{i,k}}$ *and* l_k *is the negative literal of* v_k.

Proof. There is a way to route edge (n_1, n_2) with only two bends only if one of the four vertical edges of the cube of clause gadget C_i is not intersected by a basic gadget. If a drawing Γ of G_ϕ satisfying γ_ϕ exists, and edge (n_1, n_2) is routed with two bends, one of the edges is not blocked, and one of the six conditions in the statement is verified. □

Lemma 18. *Statement 4 holds; that is, if ϕ admits a solution, then $G_\phi = (V_\phi, E_\phi)$ admits a non-intersecting drawing with nodes at the prescribed positions.*

Problem	$N=0$	$N=1$	$N=2$	$N=3$	$N=4$	$N=5$	$N\geqslant 6$
N-BEND ROUTING	$O(n^2)$	$O(n^2)$	NP-hard		Unknown		Trivial
N-BEND GRID ROUTING	$O(n^2)$	$O(n^2)$	NP-hard		Conjectured NP-hard		

Fig. 16. Complexity of three-dimensional routing problems.

Proof. Consider a truth assignment satisfying ϕ. If variable v_i is true (false) we can use the true (false) drawing of variable gadget V_i depicted in Fig. 12(a) (Fig. 12(b)). Also, for each clause $c_i = l_h \vee l_j \vee l_k$, at least one of its literals is true. This implies that one of the vertical edges of the clause gadget C_i is not blocked, and edge (n_1, n_2) can be routed with two bends without intersection. □

Lemma 19. 2-BEND ROUTING *is NP-hard.*

Proof. The proof is based on the fact that a compliant specification can be found for the 3SAT reduction framework introduced in Section 3. Lemmas 16, 17, and 18 prove that Statements 2, 3, and 4 hold, respectively. Since the 2-BEND ROUTING instance I_ϕ corresponding to a 3SAT instance ϕ can be built in polynomial time, Statement 1 also holds. Therefore, the construction rules described in Sections 5.2.1, 5.2.2, 5.2.3, and 5.2.4 correspond to a compliant specification for the 3SAT reduction framework, and Theorem 1 applies. □

6. Discussion and open problems

This paper shows that SHAPE GRAPH REALIZATION is NP-hard, while the reverse problem, N-BEND ROUTING, is polynomial or even trivial for some values of N. This asymmetry may explain why most three-dimensional drawing algorithms in the literature determine edge shapes as a consequence of node relative positions and not vice-versa.

Fig. 16 summarizes the complexity of N-BEND ROUTING problems. We signal as open the problems determining the complexity of 3-, 4-, and 5-BEND ROUTING.

Fig. 16 also compares N-BEND ROUTING with N-BEND GRID ROUTING, the analogous problem where nodes and bends are restricted to have integer coordinates. Observe that the proof that 2-BEND ROUTING is NP-complete, provided in Section 5, implies that 2-BEND GRID ROUTING is NP-complete too. Further, the $O(|V|^2)$ algorithms described in Section 5.1 to compute 0- and 1-bend drawings can be used also to compute 0- and 1-bend grid drawings. On the contrary, the proof that 6-BEND ROUTING is feasible heavily relies on the relative coordinates scenario of [26] and cannot be modified to show the feasibility of 6-BEND GRID ROUTING in the integer-coordinates setting. Hence, an interesting open problem is the following: What is the complexity of N-BEND GRID ROUTING? We conjecture that this problem is NP-hard for any N greater than 2.

The 3SAT reduction framework offers a way to reprove NP-hardness for problems already known to be hard. For example, consider the problem of deciding whether a graph admits a 0-bend drawing. Such a problem was shown to be NP-complete in [16] by extending to the three-dimensions the original reduction used in [21] for the analogous bidimensional problem.

A proof based on the 3SAT reduction framework uses gadgets analogous to those used to prove the NP-hardness of SHAPE GRAPH REALIZATION in Section 4 and may be constructed as follows. Consider all 0-bend drawings of a *cube graph*, i.e., a graph of eight nodes and twelve edges that can be drawn as a cube. All such drawings correspond, up to rotations and mirrorings, to the same shape graph. It is not difficult to prove that a graph composed by cube graphs attached together side-by-side has the same property. Further, given a shape graph, for example the one depicted in Fig. 17(a), it is possible to construct a graph composed by cube graphs attached side-by-side, as the one depicted in Fig. 17(b), which admits a 0-drawing if and only if the starting shape graph is realizable. Based on these considerations, a compliant specification for the 3SAT reduction framework can be defined by using gadgets and construction rules analogous to those described in Section 4. Fig. 17(c), for example, shows the construction of a variable gadget analogous to that depicted in Fig. 3(a). These same constructions could be used to show that SHAPE GRAPH REALIZATION is NP-hard even when restricted to triconnected graphs.

Finally, observe that the 3SAT reduction framework could be extended to different versions of the SAT problem known to be NP-complete. For example, some problems may be more easier to map to a NOT-ALL-EQUAL-3SAT

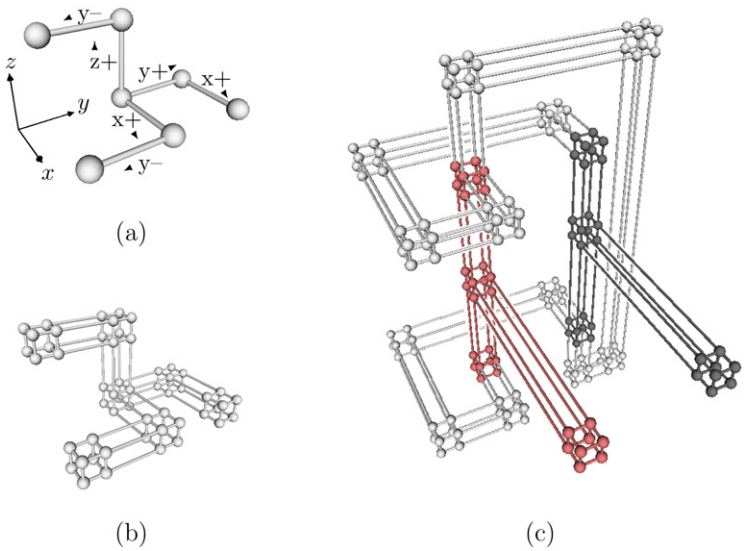

Fig. 17. (a) A shape graph. (b) A construction that admits a 0-drawing if and only if the shape graph represented in (a) is realizable. (c) The variable gadget that can be used to show that finding a drawing without bends is NP-hard (compare with Fig. 3(a)).

instance rather than to a 3SAT instance. As another example, since PLANAR-3SAT is NP-complete, the 3SAT reduction framework could assume the 3SAT instance to be planar. This may offer a way to prove NP-hardness of problems restricted to planar graphs, which are excluded by the current constructions (it suffice having three variables occurring together in three clauses of the 3SAT formula to construct an instance of the target problem that contains a subdivision of $K_{3,3}$ and is, hence, non-planar).

In the remaining part of this section, we discuss the impact of the results described in this paper on the two problem that we consider the most intriguing of the field, namely, the characterization of realizable orthogonal shapes and the existence of 2-bend drawings for every graph of maximum degree six.

6.1. Characterization of realizable orthogonal shapes

With respect to the problem of characterizing realizable orthogonal shapes, deciding whether a shape graph is realizable is shown here to be NP-complete. Of course, the problem of characterizing realizable orthogonal shapes remains open, although we now know that in the general case it implies a heavy computation.

As a consequence of the complexity of the SHAPE GRAPH REALIZATION problem in the general case, in any hypothetical 3D drawing process in which the definition of the shape of the drawing is followed by the actual computation of its coordinates, the first step should be very carefully conceived in order for the second step to be efficiently computable. In fact, focusing on peculiar classes of shape graphs seems to be an obliged strategy for practical applications. Are there non-trivial families of shape graph for which the SHAPE GRAPH REALIZATION problem is feasible? In particular, is there a "universal" set of shape graphs such that any graph is represented and such that the SHAPE GRAPH REALIZATION problem is guaranteed to be polynomial and to have a positive answer?

6.2. Existence of 2-bend drawings

With respect to the problem of determining if a graph of degree six always admits a 2-bend drawing, this paper shows the NP-completeness of two problems related with finding such drawings. Namely, it is NP-complete when node positions are fixed (Section 5) and it is NP-complete when edge shapes are fixed (Section 4). The two analogous problems restricted to integer coordinates retain the same complexity. Some other 3D drawing problems involving the number of the bends are known to be NP-hard, as, for example, deciding whether a graph admits a 2-bend drawing when vertices are placed on the diagonal of a cube [35].

The number of NP-hard problems related with the computation of a 2-bend drawing raises the following question: What is the complexity of finding a 2-bend drawing of a graph? If finding such a drawing was also NP-hard, then any attempt to prove that such a drawing always exists should produce an algorithm for an intractable problem, which is hard to conceive without resorting to an enumerative approach (which, in turn, assumes the existence of a solution). However both the conception of such an algorithm and the description of a graph not admitting a 2-bend drawing appear to be elusive goals.

Acknowledgements

We would like to thank Giuseppe Di Battista and Giuseppe Liotta for constant encouragement and interesting conversations. Also, we would like to thank Andrea Orlandini for his help focusing on the propositional logic needed to define what a constraint is. Finally, we thank the anonymous referees for numerous helpful comments and suggestions, that greatly contributed to the readability, accuracy, and breadth of the paper.

Work partially supported by European Commission—Fet Open project COSIN—COevolution and Self-organisation In dynamical Networks—IST-2001-33555, by European Commission—Fet Open project DELIS—Dynamically Evolving Large Scale Information Systems—Contract no. 001907, and by MIUR under Project ALGO-NEXT (Algorithms for the Next Generation Internet and Web: Methodologies, Design, and Experiments).

References

[1] B. Aspvall, M.F. Plass, R.E. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, Inform. Process. Lett. 8 (3) (1979) 121–123.
[2] S. Bhatt, S. Cosmadakis, The complexity of minimizing wire lengths in VLSI layouts, Inform. Process. Lett. 25 (1987) 263–267.
[3] T.C. Biedl, Heuristics for 3D-orthogonal graph drawings, in: Proc. 4th Twente Workshop on Graphs and Combinatorial Optimization, 1995, pp. 41–44.
[4] T.C. Biedl, T.M. Chan, A note on 3D orthogonal graph drawing, Discrete Appl. Math. 148 (2) (2005) 189–193.
[5] J.A. Bondy, U.S.R. Murty, Graph Theory with Applications, Macmillan, London, 1976.
[6] F. Brandenburg, Nice drawings of graphs are computationally hard, in: Proc. of Informatics and Psychology Workshop, 1988, pp. 1–15.
[7] F. Brandenburg, D. Eppstein, M.T. Goodrich, S. Kobourov, G. Liotta, P. Mutzel, Selected open problems in graph drawing, in: G. Liotta (Ed.), Graph Drawing (Proc. GD 2003), in: Lecture Notes in Computer Science, vol. 2912, Springer-Verlag, Berlin, 2004, pp. 515–539.
[8] M. Closson, S. Gartshore, J. Johansen, S.K. Wismath, Fully dynamic 3-dimensional orthogonal graph drawing, J. Graph Algorithms Appl. 5 (2) (2001) 1–34.
[9] S.A. Cook, The complexity of theorem-proving procedures, in: Proc. 3rd Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 1971, pp. 151–158.
[10] E.D. Demaine, J.S.B. Mitchell, J. O'Rourke (Eds.), The Open Problems Project, http://cs.smith.edu/~orourke/TOPP/Welcome.html.
[11] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing, Prentice-Hall, Upper Saddle River, NJ, 1999.
[12] G. Di Battista, G. Liotta, A. Lubiw, S. Whitesides, Orthogonal drawings of cycles in 3D space, in: J. Marks (Ed.), Graph Drawing (Proc. GD'00), in: Lecture Notes in Computer Science, vol. 1984, Springer-Verlag, Berlin, 2001.
[13] G. Di Battista, G. Liotta, A. Lubiw, S. Whitesides, Embedding problems for paths with direction constrained edges, Theoret. Comput. Sci. 289 (2002) 897–917.
[14] G. Di Battista, M. Patrignani, F. Vargiu, A split&push approach to 3D orthogonal drawing, J. Graph Algorithms Appl. 4 (3) (2000) 105–133.
[15] E. Di Giacomo, G. Liotta, M. Patrignani, A note on 3D orthogonal drawings with direction constrained edges, Inform. Process. Lett. 90 (2004) 97–101.
[16] P. Eades, C. Stirk, S. Whitesides, The techniques of Kolmogorov and Barzdin for three-dimensional orthogonal graph drawings, Inform. Process. Lett. 60 (1996) 97–103.
[17] P. Eades, A. Symvonis, S. Whitesides, Two algorithms for three-dimensional orthogonal graph drawing, in: S. North (Ed.), Graph Drawing (Proc. GD'96), in: Lecture Notes in Computer Science, vol. 1190, Springer-Verlag, Berlin, 1997, pp. 139–154.
[18] P. Eades, A. Symvonis, S. Whitesides, Three-dimensional orthogonal graph drawing algorithms, Discrete Appl. Math. 103 (1–3) (2000) 55–87.
[19] P. Eades, S. Whitesides, The logic engine and the realization problem for nearest neighbor graphs, Theoret. Comput. Sci. 169 (1) (1996) 23–37.
[20] P. Eades, S. Whitesides, The realization problem for Euclidean minimum spanning trees is NP-hard, Algorithmica 16 (1996) 60–82.
[21] A. Garg, R. Tamassia, On the computational complexity of upward and rectilinear planarity testing, SIAM J. Comput. 31 (2) (2001) 601–625.
[22] K. Hagihara, N. Tokura, N. Suzuki, Graph embedding on a three-dimensional model, Systems-Comput.-Controls 14 (6) (1983) 58–66.
[23] M. Kitching, S. Whitesides, The three dimensional logic engine, in: J. Pach (Ed.), Graph Drawing (Proc. GD'04), in: Lecture Notes in Computer Science, vol. 3383, Springer-Verlag, Berlin, 2004, pp. 329–339.
[24] A.N. Kolmogorov, Y.M. Barzdin, On the realization of nets in 3-dimensional space, Problems in Cybernetics 8 (1967) 261–268.
[25] B.Y.S. Lynn, A. Symvonis, D.R. Wood, Refinement of three-dimensional orthogonal graph drawings, in: J. Marks (Ed.), Graph Drawing (Proc. GD'00), in: Lecture Notes in Computer Science, vol. 1984, Springer-Verlag, Berlin, 2001, pp. 308–320.

[26] A. Papakostas, I.G. Tollis, Algorithms for incremental orthogonal graph drawing in three dimensions, J. Graph Algorithms Appl. 3 (4) (1999) 81–115.
[27] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, third ed., Springer-Verlag, Berlin, 1990.
[28] R. Tamassia, On embedding a graph in the grid with the minimum number of bends, SIAM J. Comput. 16 (3) (1987) 421–444.
[29] R. Tamassia, G. Di Battista, C. Batini, Automatic graph drawing and readability of diagrams, IEEE Trans. Syst. Man Cybern. SMC-18 (1) (1988) 61–79.
[30] G. Vijayan, A. Wigderson, Rectilinear graphs and their embeddings, SIAM J. Comput. 14 (1985) 355–372.
[31] D.R. Wood, On higher-dimensional orthogonal graph drawing, in: J. Harland (Ed.), Proc. Computing: the Australasian Theory Symposium (CATS '97), vol. 19, Australian Computer Science Commission, 1997, pp. 3–8.
[32] D.R. Wood, Three-dimensional orthogonal graph drawing, PhD thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, 2000.
[33] D.R. Wood, Lower bounds for the number of bends in three-dimensional orthogonal graph drawings, J. Graph Algorithms Appl. 7 (2003) 33–77.
[34] D.R. Wood, Optimal three-dimensional orthogonal graph drawing in the general position model, Theoret. Comput. Sci. 299 (2003) 151–178.
[35] D.R. Wood, Minimising the number of bends and volume in 3-dimensional orthogonal graph drawings with a diagonal vertex layout, Algorithmica 39 (2004) 235–253.