On the Computational Complexity of Upward and Rectilinear Planarity Testing*

(Extended Abstract)

Ashim Garg and Roberto Tamassia

Department of Computer Science
Brown University
Providence, RI 02912–1910, USA

Abstract. A directed graph is upward planar if it can be drawn in the plane such that every edge is a monotonically increasing curve in the vertical direction, and no two edges cross. An undirected graph is rectilinear planar if it can be drawn in the plane such that every edge is a horizontal or vertical segment, and no two edges cross. Testing upward planarity and rectilinear planarity are fundamental problems in the effective visualization of various graph and network structures. In this paper we show that upward planarity testing and rectilinear planarity testing are NP-complete problems. We also show that it is NP-hard to approximate the minimum number of bends in a planar orthogonal drawing of an n-vertex graph with an $O(n^{1-\epsilon})$ error, for any $\epsilon > 0$.

1 Introduction

Graph drawing addresses the problem of constructing geometric representations of abstract graphs and networks [5]. It is an emerging area of research that combines flavors of topological graph theory and computational geometry. The automatic generation of drawings of graphs has important applications in key computer technologies such as software engineering, database design, visual interfaces, and computer-aided-design.

Various graphic standards have been proposed for the representation of graphs in the plane. Usually, vertices are represented by points, and each edge (u, v) is represented by a simple open Jordan curve joining the points associated with the vertices u and v. A *straight-line* drawing maps each edge into a straight-line segment. An *orthogonal* drawing maps each edge into a chain of horizontal and vertical segments. A *rectilinear* drawing is an orthogonal straight-line drawing, i.e., a drawing where every edge is either a horizontal or a vertical segment. A drawing is *planar* if no two edges cross. A graph (or digraph) is planar if it admits a planar drawing. A graph is *rectilinear planar* if it admits a planar rectilinear drawing. A drawing of a digraph is *upward* if every edge is

* Research supported in part by the National Science Foundation, by the U.S. Army Research Office, and by the Advanced Research Projects Agency. Email addresses of the authors: {ag,rt}@cs.brown.edu.

monotonically nondecreasing in the y-direction. A digraph is *upward planar* if it admits a planar upward drawing.

Testing upward planarity and rectilinear planarity are fundamental problems in the effective visualization of various graph and network structures. For example, upward planarity is useful for the display of order diagrams and subroutine-call graphs, while rectilinear planarity is useful for the display of circuit schematics and entity-relationship diagrams. In this paper we show that the following two problems are NP-complete:

Upward planarity testing: testing whether a digraph is upward planar.

Rectilinear planarity testing: testing whether a graph is rectilinear planar.

These problems have challenged researchers in order theory, topological graph theory, computational geometry, and graph drawing for many years. Our intractability results motivate the following observations:

- Testing whether a graph admits a planar drawing or an upward drawing can be done in linear time. Combining the two properties makes the problem NP-hard.
- Every planar graph admits a planar straight-line drawing. Hence, we can say that planarity is equivalent to straight-line planarity, and both properties can be verified in linear time. We can view upward and rectilinear planarity as derived from straight-line planarity by adding further constraints, which make the problem become apparently much more difficult.

We also show that it is NP-hard to approximate the minimum number of bends in a planar orthogonal drawing of an n-vertex graph with an $O(n^{1-\epsilon})$ error, for any $\epsilon > 0$.

Previous results on upward and rectilinear planarity testing are summarized below. In the rest of this section, we denote with n the number of vertices of the graph being considered.

Combinatorial results on upward planarity of covering digraphs of lattices were first given in [15, 22]. Further results on the interplay between upward planarity and ordered sets are surveyed by Rival [23]. Lempel, Even, and Cederbaum [16] relate the planarity of biconnected undirected graphs to the upward planarity of st-digraphs. A combinatorial characterization of upward planar digraphs is provided in [8, 14]: namely, a digraph is upward planar if and only if it is a spanning subgraph of a planar *st-digraph*. This characterization implies that upward planarity testing is in NP.

Di Battista, Liu, and Rival [7] show that every planar bipartite digraph is upward planar. Papakostas [21] gives a polynomial-time algorithm for upward planarity testing of outerplanar digraphs. Bertolazzi, Di Battista, Liotta, and Mannino [1, 2] give a polynomial-time algorithm for testing upward planarity of triconnected digraphs and of digraphs with a fixed embedding. Concerning single-source digraphs, Thomassen [28] characterizes upward planarity in terms of forbidden circuits. Hutton and Lubiw [12] combine Thomassen's characterization with a decomposition scheme to test upward planarity of a single-source digraph in $O(n^2)$ time. Bertolazzi, Di Battista, Mannino, and Tamassia [3] show that upward planarity testing of a single-source digraph can be done optimally in $O(n)$ time. They also give a parallel algorithm that runs in $O(\log n)$ time on a CRCW PRAM with $n \log \log n / \log n$ processors.

Regarding rectilinear planarity testing, Shiloach [24] and Valiant [29] show that any planar graph of degree at most 4 admits a planar orthogonal drawing. Vijayan and Wigderson [30] study structural properties of rectilinear planar drawings. From their results, the membership of rectilinear planarity testing in NP is easy to establish. Storer [25], Tamassia and Tollis [27], Liu, Marchioro, Morgana, Petreschi, and Simeone [18, 19, 20, 17], Even and Granot [9], and Biedl and Kant [13, 4] give various techniques for constructing planar orthogonal drawings with $O(n)$ bends. Tamassia [26] gives an $O(n^2 \log n)$-time algorithm that constructs a planar orthogonal drawing with the minimum number of bends for an embedded planar graph. Di Battista, Liotta, and Vargiu [6] give polynomial time algorithms for minimizing bends in planar orthogonal drawings of series-parallel and cubic graphs. The latter two results show that rectilinear planarity testing can be done in polynomial time for a fixed embedding or for special classes of graphs.

Our proof techniques are based on a two-phase reduction from the known NP-complete problem NOT-ALL-EQUAL-3-SAT. In the first phase, we reduce NOT-ALL-EQUAL-3-SAT to an auxiliary undirected flow problem. In the second phase, we reduce this undirected flow problem to the upward (or rectilinear) planarity testing of a special class of digraphs. The latter reduction is interesting in its own and provides new insights on the characterization by flow networks of the angles formed by the edges of upward planar drawings [1, 2] and orthogonal drawings [6, 26].

The rest of this paper is organized as follows. Preliminary definitions and results are provided in Section 2. The reduction from NOT-ALL-EQUAL-3-SAT to the auxiliary flow problem is given in Section 3. Sections 4 and 5 describe the reductions from the auxiliary flow problem to upward and rectilinear planarity testing, respectively. In this extended abstract, proofs and technical details are omitted. They can be found in the full paper [11].

2 Preliminaries

We assume standard concepts and definitions on NP-completeness [10]. Our results use reductions from the following well-known NP-complete problem:

NOT-ALL-EQUAL-3-SAT Given a set of clauses with three literals each, is there a truth assignment such that each clause has at least one true literal and one false literal?

An *embedding* of a planar graph is the collection of circular permutations of the edges incident upon each vertex in a planar drawing of the graph. An *embedded graph* is a planar graph equipped with an embedding. The *angles* of an embedded graph are the pairs of consecutive edges incident on the same vertex. Such angles are mapped to geometric angles in a straight-line drawing of the graph.

A *rectilinear embedding* of a graph G is an embedding of G where each angle is assigned a label in the set $\{1, 2, 3, 4\}$, such that there exists a rectilinear drawing of G where each angle labeled ℓ in the embedding measures $\ell\pi/2$ in the drawing.

An *upward embedding* of a digraph $\vec{G}$ is an embedding of $\vec{G}$ where each angle formed by pairs of incoming or outgoing edges is assigned a label in the set $\{small,\ large\}$, such that there exists a planar straight-line upward drawing of $\vec{G}$ where each angle labeled *small* has measure $< \pi$ and each angle labeled *large* has measure $> \pi$.

In the rest of this section, we define several graphs that will be used as gadgets in our reductions.

We show in Fig. 1(a) *tendril* T_k ($k \geq 1$), which is an acyclic digraph with $k+1$ sources and $k+1$ sinks. We also define tendril T_0 as a a digraph consisting of a single edge. Tendril T_k ($k \geq 0$) has a designated source and a designated sink, called the *poles* of T_k. We shall consider transformations where a directed edge (u, v) of a digraph is replaced with a tendril T_k, where the source is identified with u and the sink with t.

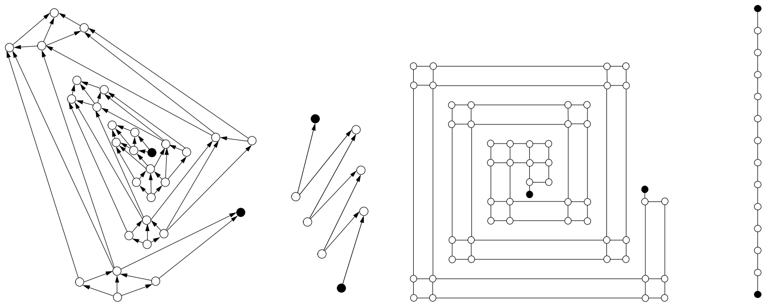

Figure 1: (a) Tendril T_3. **(b)** Wiggle W_3. **(c)** Rectilinear tendril T_3. **(d)** Rectilinear wiggle W_3. The poles are drawn as black-filled circles.

Lemma 1. *Tendril T_k is upward planar and admits a unique upward planar embedding.*

In the upward planar embedding of T_k, the external face consists of two paths between s and t. One such path, called *outer path*, has $2k$ large angles and no small angles, and the other path, called *inner path*, has $2k$ small angles and no large angles. When a tendril replaces an edge of an embedded planar digraph, the outer path becomes a subpath of a face, and we say that the *contribution* of the outer path to the face is $+2k$. Similarly, we say that the contribution of the inner path to its face is $-2k$.

Figure 1(b) shows a *wiggle* W_k, which is an acyclic digraph consisting of a chain of $2k+1$ edges whose orientation alternates along the chain. The extreme vertices of W_k, a source and a sink, are called the *poles* of W_k. We shall consider transformations where a directed edge (u, v) of an embedded digraph is replaced with wiggle W_k, where s is identified with u and v with t. Given an upward embedding of W_k, we say that the *contribution* of W_k to a face f containing

W_k is the number of large angles minus the number of small angles of W_k in f. Clearly, W_k can be upward embedded to give to f any contribution $2i$ with $0 \leq i \leq k$. Note that if G_k gives contribution c to a face, it gives contribution $-c$ to the other face it belongs to.

We show in Fig. 1(c) an undirected graph called *rectilinear tendril* T_k, which also has two designated poles denoted by s and t. We also define rectilinear tendril T_0 as a a graph consisting of a single edge. We show in Fig. 1(d) *rectilinear wiggle* W_k, which is a chain of $4k+1$ edges.

The *contribution* of a rectilinear tendril (or wiggle) to a face containing it is the number of angles of the tendril (or wiggle) labeled 3 minus the number of angles labeled 1 that lie in the face. Rectilinear tendril T_k contributes $4k$, $4k+1$, or $4k+2$ to one face, and the opposite value to the other face. Rectilinear wiggle W_k contributes to one face any value between 0 and $4k$, and the opposite value to the other face.

3 An Auxiliary Undirected Flow Problem

In this section we define two auxiliary flow problems and show that they are equivalent to NOT-ALL-EQUAL-3-SAT under polynomial-time reductions.

A *switch-flow network* is an undirected flow network $\mathcal{N}$ where each edge is labeled with a range $[c' \cdots c'']$ of nonnegative integer values, called the *capacity range* of the edge. For simplicity, we denote the capacity range $[c \cdots c]$ with $[c]$. A *flow* for a switch-flow network is an orientation of and an assignment of integer "flow" values to the edges of the network. A *feasible flow* is a flow that satisfies the following two properties:

Range property: the flow assigned to an edge is an integer within the capacity range of the edge.

Conservation property: the total flow entering a vertex from the incoming edges is equal to the total flow exiting the vertex from the outgoing edges.

Starting from an instance $\mathcal{S}$ of NOT-ALL-EQUAL-3-SAT, we construct a switch-flow network $\mathcal{N}$ as follows (see Fig. 2). Let the literals of $\mathcal{S}$ be denoted with $x_1, y_1, \cdots, x_n, y_n$, where $y_i = \overline{x_i}$, and the clauses of $\mathcal{S}$ be denoted with $c_1, \cdots, c_m$. Let θ be a positive integer parameter. We denote with α_i and β_i $(i = 1, \cdots, n)$ the number of occurrences of literals x_i and y_i in the clauses of $\mathcal{S}$, respectively. Note that $\sum_{i=1}^{n}(\alpha_i + \beta_i) = 3m$. Also, we define $\gamma_i = (2i-1)\theta$ and $\delta_i = 2i\theta$ $(i = 1, \cdots, n)$. Network $\mathcal{N}$ has a *literal vertex* for each literal of $\mathcal{S}$ and a *clause vertex* for each clause of $\mathcal{S}$, plus a special dummy vertex z. There are three types of edges in $\mathcal{N}$:

Literal edges joining pairs of literals associated with the same boolean variable. The capacity range of literal edge (x_i, y_i) is $[\alpha_i\gamma_i + \beta_i\delta_i]$.

Clause edges joining each literal to each clause. The capacity range of clause edge (x_i, c_j) is $[\gamma_i]$ if $x_i \in c_j$, and $[0]$ otherwise. The capacity range of clause edge (y_i, c_j) is $[\delta_i]$ if $y_i \in c_j$, and $[0]$ otherwise.

Dummy edges joining each literal and each clause to the dummy vertex. The capacity ranges of dummy edges (z, x_i) and (z, y_i) are $[\beta_i\delta_i]$ and $[\alpha_i\gamma_i]$, respectively. The capacity range of dummy edge (z, c_j) is $[0 \cdots \eta_j - 2\theta]$, where η_j is the sum of the capacities of the clause edges incident on c_j.

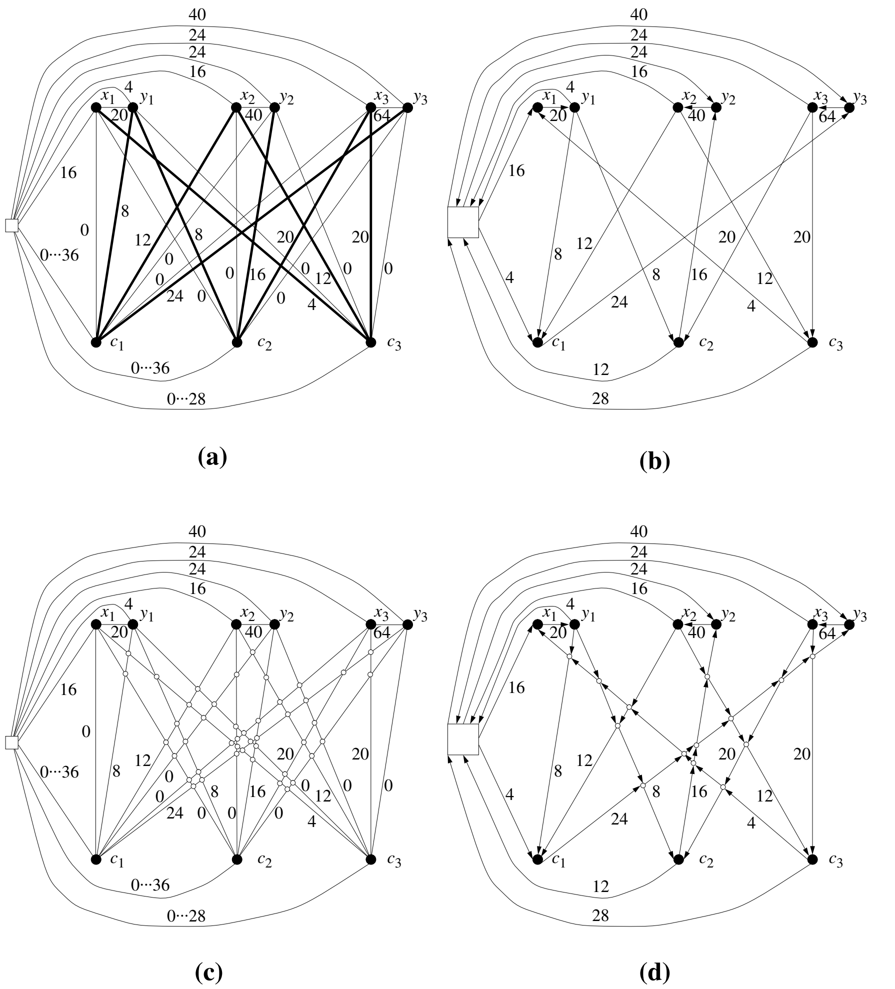

Figure 2: **(a)** Switch-flow network $\mathcal{N}$ with parameter $\theta = 4$ associated with the the NOT-ALL-EQUAL-3-SAT instance $\mathcal{S}$ with clauses $c_1 = y_1 x_2 y_3$, $c_2 = y_1 y_2 x_3$, and $c_3 = x_1 x_2 x_3$. The clause edges with nonzero capacity range are shown with thick lines. **(b)** Feasible flow for $\mathcal{N}$ corresponding to the satisfying truth assignment (y_1, x_2, x_3) for $\mathcal{S}$. Only the edges with nonzero flow are shown. **(c)** Planar switch-flow network $\mathcal{P}$ associated with $\mathcal{S}$. **(d)** Feasible flow for $\mathcal{P}$ corresponding to the satisfying truth assignment (y_1, x_2, x_3) for $\mathcal{S}$. Only the edges with nonzero flow are shown.

The construction of network $\mathcal{N}$ from $\mathcal{S}$ is straightforward, and we have:

Lemma 2. *Given an instance $\mathcal{S}$ of* NOT-ALL-EQUAL-3-SAT *with n variables and m clauses, the associated switch-flow network $\mathcal{N}$ has $O(n + m)$ vertices and*

$O(nm)$ edges, and can be constructed in $O(nm)$ time.

A feasible flow in network $\mathcal{N}$ corresponds to a satisfying truth assignment for $\mathcal{S}$. Namely, we have that a literal is true whenever its incident literal edge is incoming in the feasible flow (see Fig. 2(b)) and its incident clause edges with nonzero capacity range are outgoing. Also, the three clause edges with nonzero capacity range incident on a clause vertex c_j cannot be all incoming or all outgoing because of the conservation property at vertex c_j and the choice of capacity range for the dummy edge incident on c_j. We obtain:

Lemma 3. *An instance $\mathcal{S}$ of* NOT-ALL-EQUAL-3-SAT *is satisfiable if and only if the associated switch-flow network $\mathcal{N}$ admits a feasible flow. Also, given a feasible flow for $\mathcal{N}$, a satisfying truth assignment for $\mathcal{S}$ can be computed in time $O(nm)$, where n and m are the number of variables and clauses of $\mathcal{S}$, respectively.*

Now, starting from $\mathcal{N}$, we construct a planar switch-flow network $\mathcal{P}$ as follows (see Fig. 2). First, we construct a drawing of $\mathcal{N}$ such that the literal vertices and the clause vertices are arranged on two parallel lines, and crossings occur only between clause edges. Next, we replace the crossings formed by the clause edges with new vertices, called *crossing* vertices. We call *fragment edges* the edges originated by the splitting of the clause edges. Each fragment edge inherits the capacity range of the originating clause edge.

Lemma 4. *Given network $\mathcal{N}$ representing an instance $\mathcal{S}$ of* NOT-ALL-EQUAL-3-SAT *with n variables and m clauses, the associated planar switch-flow network $\mathcal{P}$ has $O(n^2m^2)$ vertices and edges, and can be constructed in $O(n^2m^2)$ time. Also, network $\mathcal{N}$ admits a feasible flow if and only if network $\mathcal{P}$ admits a feasible flow, and a feasible flow for $\mathcal{N}$ can be computed from a feasible flow for $\mathcal{P}$ in $O(n^2m^2)$ time.*

By combining Lemmas 3 and 4, we obtain the main result of this section.

Theorem 5. *Given an instance $\mathcal{S}$ of* NOT-ALL-EQUAL-3-SAT *with n variables and m clauses, the associated planar switch-flow network $\mathcal{P}$ has $O(n^2m^2)$ vertices and edges, and can be constructed in $O(n^2m^2)$ time. Instance $\mathcal{S}$ is satisfiable if and only if network $\mathcal{P}$ admits a feasible flow. Also, given a feasible flow for $\mathcal{P}$, a satisfying truth assignment for $\mathcal{S}$ can be computed in time $O(n^2m^2)$.*

4 Upward Planarity Testing

In this section we show how to reduce the problem of computing a feasible flow in the planar switch-flow network $\mathcal{P}$ associated with a NOT-ALL-EQUAL-3-SAT instance $\mathcal{S}$ to the problem of testing the upward planarity of a suitable digraph. We set parameter θ equal to 4.

Lemma 6. *Let $\mathcal{S}$ be a* NOT-ALL-EQUAL-3-SAT *instance with at least three variables and three clauses. The planar switch-flow network $\mathcal{P}$ associated with $\mathcal{S}$ is triconnected.*

Now, we construct on orientation $\vec{\mathcal{P}}$ of $\mathcal{P}$ as follows (see Fig. 3):

- Every literal edge (x_i, y_i) is oriented from x_i to y_i.
- Every fragment edge is oriented away from the clause vertex and towards the literal vertex.

- Every dummy edge incident on a literal vertex is oriented towards the dummy vertex, and every dummy edge incident on a clause vertex is oriented towards the clause vertex.

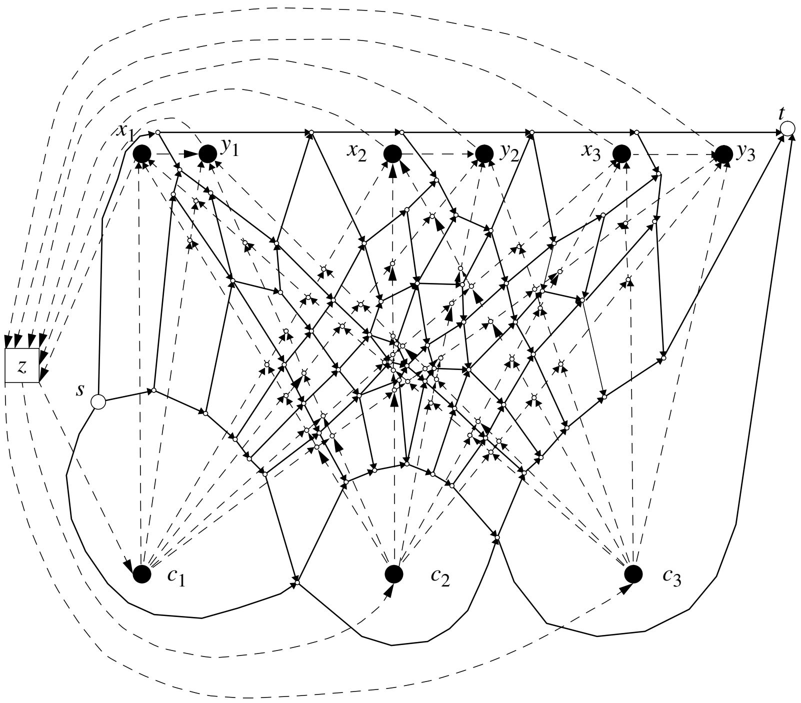

Figure 3: Orientation $\vec{\mathcal{P}}$ (drawn with dashed lines) of the network $\mathcal{P}$ shown in Fig. 2(c) and dual digraph $\vec{\mathcal{D}}$ (drawn with solid lines) of $\vec{\mathcal{P}}$.

Lemma 7. *In digraph $\vec{\mathcal{P}}$, every vertex has at least one incoming and one outgoing edge, every directed cycle contains the dummy vertex, and there are exactly two faces that are directed cycles.*

By Lemma 6, the planar embedding of $\mathcal{P}$ and the dual graph $\mathcal{D}$ of $\mathcal{P}$ are unique. We construct the dual digraph $\vec{\mathcal{D}}$ of $\vec{\mathcal{P}}$ by orienting every dual edge of $\mathcal{D}$ from the face on the left to the face on the right of the primal edge (see Fig. 3).

Lemma 8. *The dual digraph $\vec{\mathcal{D}}$ of $\vec{\mathcal{P}}$ is upward planar, triconnected, acyclic and has exactly one source and one sink, denoted with s and t.*

Starting from digraph $\vec{\mathcal{D}}$ we construct a new digraph $\vec{\mathcal{G}}$ by replacing the edges of $\vec{\mathcal{D}}$ with subgraphs (tendrils or wiggles), as follows (see Fig. 4):

- Every edge of $\vec{\mathcal{D}}$ that is the dual of a literal edge, fragment edge, or dummy edge incident on a literal vertex is replaced with tendril T_c, where $[c]$ is the capacity range of the dual edge. Note that c is a multiple of parameter θ.
- Every edge of $\vec{\mathcal{D}}$ that is the dual of a dummy edge incident on a clause vertex is replaced with wiggle W_c, where $[0 \cdots c]$ is the capacity range of the dual edge.

The vertices of $\vec{\mathcal{G}}$ that are also in $\vec{\mathcal{D}}$ are called *primary vertices.* The remaining vertices of $\vec{\mathcal{G}}$ are called *secondary vertices.*

Lemma 9. *Digraph $\vec{\mathcal{G}}$ is planar and acyclic. Also, the only primary source vertex is s and the only primary sink vertex is t.*

By Lemma 8 and the construction of digraph $\vec{\mathcal{G}}$, all the embeddings of $\vec{\mathcal{G}}$ are obtained by choosing one of the two possible flips for each tendril.

Lemma 10. *Digraph $\vec{\mathcal{G}}$ is upward planar if and only if the tendrils can be flipped and the wiggles can be arranged such that for every face the total contribution of the tendrils and wiggles is zero.*

The proof of Lemma 10 uses the characterization of upward embeddings by Bertolazzi, Di Battista, Liotta, and Mannino [1, 2].

We establish the following correspondence between digraph $\vec{\mathcal{G}}$ and network $\mathcal{P}$ (see Fig. 4): the faces of $\vec{\mathcal{G}}$ correspond to the vertices of $\mathcal{P}$; the tendrils and

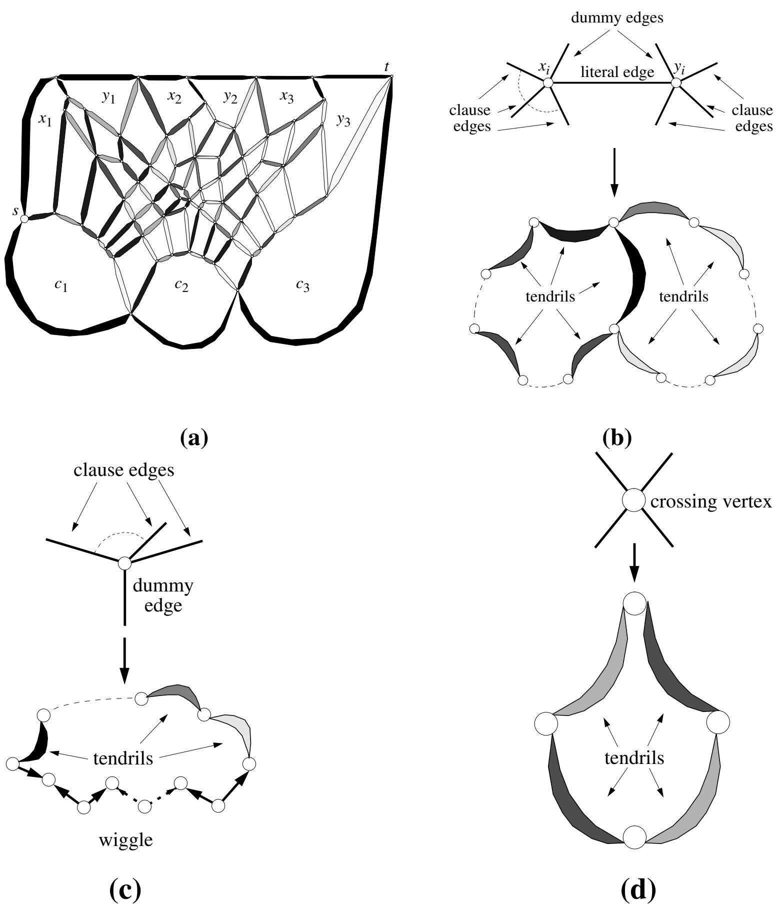

Figure 4: Schematic illustration of: (**a**) digraph $\vec{\mathcal{G}}$ obtained from $\vec{\mathcal{D}}$ by replacing edges with tendrils and wiggles; (**b**) the two faces of $\vec{\mathcal{G}}$ associated with literal vertices x_i and y_i of $\mathcal{P}$; (**c**) the face of $\vec{\mathcal{G}}$ associated with a clause vertex of $\mathcal{P}$; and (**d**) the face of $\vec{\mathcal{G}}$ associated with a crossing vertex of $\mathcal{P}$.

wiggles of $\vec{\mathcal{G}}$ correspond to the the edges of $\mathcal{P}$; flipping a tendril of $\vec{\mathcal{G}}$ corresponds to orienting an edge of $\mathcal{P}$; the contribution of a tendril or wiggle of $\vec{\mathcal{G}}$ corresponds to the flow in an edge of $\mathcal{P}$; the balance of the contributions of the tendrils and wiggles in the faces of $\vec{\mathcal{G}}$ corresponds to the conservation of flow at the vertices of $\mathcal{P}$.

Theorem 11. *Given an instance $\mathcal{S}$ of* NOT-ALL-EQUAL-3-SAT *with n variables and m clauses and the associated planar switch-flow network $\mathcal{P}$, digraph $\vec{\mathcal{G}}$ associated with $\mathcal{S}$ and $\mathcal{P}$ has $O(n^3m^2)$ vertices and edges, and can be constructed in $O(n^3m^2)$ time. Instance $\mathcal{S}$ is satisfiable and network $\mathcal{P}$ admits a feasible flow if and only if digraph $\vec{\mathcal{G}}$ is upward planar. Also, given an upward planar embedding for $\vec{\mathcal{G}}$, a feasible flow for $\mathcal{P}$ and a satisfying truth assignment for $\mathcal{S}$ can be computed in time $O(n^3m^2)$.*

From Theorems 5 and 11 we conclude:

Corollary 12. *Upward planarity testing is NP-complete.*

5 Rectilinear Planarity Testing

In this section we show how to reduce the problem of computing a feasible flow in the planar switch-flow network $\mathcal{P}$ associated with a NOT-ALL-EQUAL-3-SAT instance $\mathcal{S}$ to the problem of testing the rectilinear planarity of a suitable graph $\mathcal{G}$. The construction of graph $\mathcal{G}$ is carried out in several stages, where at each stage an intermediate graph is produced.

Given an instance $\mathcal{S}$ of NOT-ALL-EQUAL-3-SAT with n variables and m clauses, let $\mathcal{P}$ be the associated planar switch-flow network with parameter $\theta = 32nm$.

Let $\mathcal{D}$ be the dual graph of $\mathcal{P}$. The edges of $\mathcal{D}$ are classified as literal, clause, and dummy edges according to the type of their dual edge in $\mathcal{P}$. Starting from $\mathcal{D}$, we construct a degree-3 planar graph $\mathcal{F}$ by first replacing every vertex of degree d with a binary tree with d leaves, and then replacing every edge of the resulting graph with a chain of 5 edges. Graph $\mathcal{F}$ has $O(n^2m^2)$ vertices and edges.

Lemma 13. *Graph $\mathcal{F}$ has a unique planar embedding and admits a rectilinear embedding.*

We classify the edges of $\mathcal{F}$ as expansion, literal, clause, and dummy edges, where the edges forming the binary trees replacing the former vertices of $\mathcal{D}$ are expansion edges, and the remaining edges of $\mathcal{F}$ are classified according to the type of the edge of $\mathcal{D}$ that originated them. Note that each edge e of $\mathcal{P}$ is thus associated with exactly 5 edges of $\mathcal{F}$ forming a path, and we call the middle edge the *representative* of e in $\mathcal{F}$.

Finally, we construct graph $\mathcal{G}$ as follows. Let e be an edge of $\mathcal{P}$. If e is a dummy clause edge with capacity range $[0 \cdots c]$, we replace the representative of e in $\mathcal{F}$ with rectilinear wiggle W_c. Else, e is a literal, fragment, or dummy literal edge with capacity range $[c]$, and we replace the representative of e in $\mathcal{F}$ with rectilinear tendril T_c.

By Lemma 13 and the construction of digraph $\mathcal{G}$, all the embeddings of $\vec{\mathcal{G}}$ are obtained by choosing one of the two possible flips for each rectilinear tendril.

Lemma 14. *Graph $\mathcal{G}$ admits a rectilinear planar drawing if and only if the rectilinear tendrils can be flipped and the rectilinear wiggles can be arranged such that for every face the total contribution of the tendrils and wiggles is zero.*

We establish the following correspondence between graph $\mathcal{G}$ and network $\mathcal{P}$: the faces of $\mathcal{G}$ correspond to the vertices of $\mathcal{P}$; the rectilinear tendrils and wiggles of $\mathcal{G}$ correspond to the the edges of $\mathcal{P}$; flipping a rectilinear tendril of $\mathcal{G}$ corresponds to orienting an edge of $\mathcal{P}$; the contribution of a rectilinear tendril or wiggle of $\mathcal{G}$ corresponds to the flow in an edge of $\mathcal{P}$; the balance of the contributions of the rectilinear tendrils and wiggles in the faces of $\mathcal{G}$ corresponds to the conservation of flow at the vertices of $\mathcal{P}$.

Theorem 15. *Given an instance $\mathcal{S}$ of* NOT-ALL-EQUAL-3-SAT *with n variables and m clauses, graph $\mathcal{G}$ associated with $\mathcal{S}$ has $O(n^4m^3)$ vertices and edges, and can be constructed in $O(n^4m^3)$ time. Instance $\mathcal{S}$ is satisfiable if and only if graph $\mathcal{G}$ is rectilinear planar. Also, given a rectilinear planar embedding for $\mathcal{G}$, a satisfying truth assignment for $\mathcal{S}$ can be computed in time $O(n^4m^3)$.*

From Theorem 15 we conclude:

Corollary 16. *Rectilinear planarity testing is NP-complete.*

Corollary 17. *Let G be an n-vertex planar graph whose minimum number of bends in any planar orthogonal drawing is b^*. Computing a planar orthogonal drawing of G with $O(b^* + n^{1-\epsilon})$ bends is NP-hard for $\epsilon > 0$.*

Acknowledgements

We would like to thank Giuseppe Di Battista for useful discussions, and Ivan Rival for comments on a preliminary version of this paper.

References

1. P. Bertolazzi and G. Di Battista. On upward drawing testing of triconnected digraphs. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pp. 272–280, 1991.
2. P. Bertolazzi, G. Di Battista, G. Liotta, and C. Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, to appear.
3. P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. In *Proc. European Symposium on Algorithms*, LNCS vol. 726, pp. 37–48. Springer-Verlag, 1993.
4. T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. In *Proc. European Symp. on Algorithms*, LNCS vol. 855, pp. 24–35. Springer-Verlag, 1994.
5. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl.*, to appear. Preprint available by `ftp` from `ftp.cs.brown.edu:pub/papers/compgeo/`.
6. G. Di Battista, G. Liotta, and F. Vargiu. Spirality of orthogonal representations and optimal drawings of series-parallel graphs and 3-planar graphs. In *Proc. Workshop Algorithms Data Struct.*, LNCS vol. 709, pp. 151–162. Springer-Verlag, 1993.
7. G. Di Battista, W. P. Liu, and I. Rival. Bipartite graphs upward drawings and planarity. *Inform. Process. Lett.*, 36:317–322, 1990.
8. G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoret. Comput. Sci.*, 61:175–198, 1988.

9. S. Even and G. Granot. Rectilinear planar drawings with few bends in each edge. Technical Report 797, Computer Science Dept., Technion", 1994.

10. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, NY, 1979.

11. A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. Report CS-94-10, Comput. Sci. Dept., Brown Univ., Providence, RI, 1994.

12. M. D. Hutton and A. Lubiw. Upward planar drawing of single source acyclic digraphs. In *Proc. 2nd ACM-SIAM Sympos. Discrete Algorithms*, pp. 203–211, 1991.

13. G. Kant. Drawing planar graphs using the *lmc*-ordering. In *Proc. 33th Annu. IEEE Sympos. Found. Comput. Sci.*, pp. 101–110, 1992.

14. D. Kelly. Fundamentals of planar ordered sets. *Discrete Math.*, 63:197–216, 1987.

15. D. Kelly and I. Rival. Planar lattices. *Canad. J. Math.*, 27(3):636–665, 1975.

16. A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs: Internat. Symposium (Rome 1966)*, pp. 215–232, New York, 1967. Gordon and Breach.

17. Y. Liu, P. Marchioro, and R. Petreschi. A single bend embedding algorithm for cubic graphs. Manuscript, 1994.

18. Y. Liu, P. Marchioro, R. Petreschi, and B. Simeone. Theoretical results on at most 1-bend embeddability of graphs. Technical report, Dipartimento di Statistica, Univ. di Roma "La Sapienza", 1990.

19. Y. Liu, A. Morgana, and B. Simeone. General theoretical results on rectilinear embeddability of graphs. *Acta Math. Appl. Sinica*, 7:187–192, 1991.

20. Y. Liu, A. Morgana, and B. Simeone. A linear algorithm for 3-bend embeddings of planar graphs in the grid. Manuscript, 1993.

21. A. Papakostas. Upward planarity testing of outerplanar dags. In *Proc. Graph Drawing '94*, 1994.

22. C. Platt. Planar lattices and planar graphs. *J. Combin. Theory Ser. B*, 21:30–39, 1976.

23. I. Rival. Reading, drawing, and order. In I. G. Rosenberg and G. Sabidussi, editors, *Algebras and Orders*, pp. 359–404. Kluwer Academic Publishers, 1993.

24. Y. Shiloach. *Arrangements of Planar Graphs on the Planar Lattice.* PhD thesis, Weizmann Institute of Science, 1976.

25. J. A. Storer. On minimal node-cost planar embeddings. *Networks*, 14:181–212, 1984.

26. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.

27. R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Trans. on Circuits and Systems*, CAS-36(9):1230–1234, 1989.

28. C. Thomassen. Planar acyclic oriented graphs. *Order*, 5(4):349–361, 1989.

29. L. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comput.*, C-30(2):135–140, 1981.

30. G. Vijayan and A. Wigderson. Rectilinear graphs and their embeddings. *SIAM J. Comput.*, 14:355–372, 1985.