3D Polyline Grid Drawings of Graphs with Linear Crossing Number

Timo Poranen*
Dept. of Computer and Information Sciences
University of Tampere, Kanslerinrinne 1
FIN-33014 University of Tampere, Finland

28th January 2003

Abstract

This article studies the properties of three dimensional visibility representations of planar graphs and three dimensional crossing-free polyline grid drawings of non-planar graphs with known crossing number.

First, we show how to construct in linear time a three dimensional polygonal z-visibility representation for planar graph having n vertices with volume $\lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times \lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times (n-1)$. This sharpens earlier results for three dimensional visibility representations for planar graphs.

Second, we show that a planar graph with n-vertices and m-edges, without any restrictions concerning its degree, admits a three dimensional crossing-free polyline grid drawing with volume $\lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times \lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times 3(n-1)$ having at most $2m$ total edge bends.

Third, we give a drawing algorithm for non-planar graphs. Let G be a non-planar graph with n-vertices and m edges and let G_p be the planarized version of G with n vertices and n' dummy vertices. We show how to construct in $O(n+n')$ time three dimensional crossing-free polyline grid drawing of G with volume $2\lceil\sqrt{\lfloor 3(n+n')/2\rfloor - 3}\rceil \times 2\lceil\sqrt{\lfloor 3(n+n')/2\rfloor - 3}\rceil \times 3(n+n'-1)$ having at most $4m + 19n'$ edge bends. It follows that a n-vertex non-planar graph with $O(n)$ crossings admits a three dimensional crossing-free polyline grid drawing with $O(n^2)$ volume.

1 Introduction

Three dimensional drawings of graphs are needed in VLSI-design [32], modeling VRML worlds and in user interface design. There is also a great theoretical interest to learn properties of the three dimensional drawings of graphs. Moreover, the experimental results gives a strong motivation to study three dimensional graph drawing [37].

The first theoretical results for the three dimensional graph drawing problem appeared in [7]. It was proved there that a complete graph with n vertices has a

*Work funded by the Tampere Graduate School in Information Science and Engineering (TISE) and supported by the Academy of Finland (Project 51528), e-mail: tp@cs.uta.fi

three dimensional straight-line crossing-free grid drawing with volume $2n \times 2n \times n$ and that the lower bound of the drawing of complete graph is also $\Omega(n^3)$. It was also shown that every planar graph with maximum degree 4 admits a three dimensional polyline drawing with $\sqrt{n} \times \sqrt{n} \times n$ volume with at most $2n + 4$ edge bends. It was conjectured that there are other classes of graphs allowing a smaller volume than the general case.

The next refinement was given in [6], where it was shown that 2-, 3- and 4-colorable graphs can be drawn with volume $O(n^2)$. In [30] it was pointed out that for any fixed $C \geq 2$, every C-colorable graph can be drawn with volume $O(n^2)$ with integer coordinates and that the order of magnitude of the bound cannot be improved. Since each planar graph is 4-colorable [18], this result implies that each planar graph admits a $O(n^2)$ volume, crossing-free three dimensional drawing with integer coordinates.

More results appeared in [17], where it was proved that 2- and 3-colorable graphs can be drawn with volume $O(n^{3/2})$ and any C-colorable graph admits a straight-line drawing with volume $O(C^4 n^{3/2})$ with rational coordinates.

Recently, Felsner, Liotta and Wismath gave more results for outerplanar graphs [15], showing that outerplanar graphs admit linear volume drawing. See also [10, 11, 38] for other recent results concerning the relationship of three dimensional graph drawing and tree-width, path-width and queue-number. Bose et al. [4] have found the maximum number of edges for a drawing of a given volume. For other three dimensional drawing conventions, see for example [3] and references given there or a recent survey by Landgraf [24].

Fundamental results for three dimensional visibility representations are collected in [5], where it was shown that every planar graph admits a rectangle z-visibility representation. It was also proved an upper bound $n = 56$ for the largest representable complete graph K_n and showed by construction that at least K_{22} admits a z-visibility representation.

In this paper we show first how to construct a three dimensional polygonal z-visibility representation from a two dimensional visibility representation. Our three dimensional visibility representation allows vertices to be associated with polygonal regions parallel to x, y-plane. The volume of this representation is at most $\lceil \sqrt{x} \rceil \times \lceil \sqrt{x} \rceil \times y$, where x and y are the dimensions of the corresponding two dimensional representation.

Second, we describe a method that produces three dimensional drawings from three dimensional visibility representations. Our method is similar to those for two dimensional graph drawing [9]. Then we show that a planar graph with n vertices and m edges admits a three dimensional polyline crossing-free grid drawing with volume $\lceil \sqrt{\lfloor 3n/2 \rfloor - 3} \rceil \times \lceil \sqrt{\lfloor 3n/2 \rfloor - 3} \rceil \times 3(n-1)$ and with at most $2m$ total edge bends. The drawing can be constructed in $O(n)$ time.

Third, we introduce a new drawing method for non-planar graphs. Suppose that G is a non-planar graph with n-vertices and m edges and G_p is the planarized version of G with n vertices and n' dummy vertices. Then our method constructs in linear time three dimensional crossing-free polyline grid drawing of G with volume $2\lceil \sqrt{\lfloor 3(n+n')/2 \rfloor - 3} \rceil \times 2\lceil \sqrt{\lfloor 3(n+n')/2 \rfloor - 3} \rceil \times 3(n+n'-1)$ having at most $4m + 19n'$ edge bends. From this result it follows that a n-vertex non-planar graph with $O(n)$ crossings admits a three dimensional crossing-free polyline grid drawing with $O(n^2)$ volume. This is the main contribution of this paper.

The rest of this paper is organized as follows. In Section 2 we give some

preliminaries for three dimensional graph drawing and visibility representation in two and three dimensions. In Section 3 we introduce our drawing algorithm for planar graphs and analyze its time complexity and other properties. In the fourth section we apply the drawing method introduced in Section 3 to non-planar graphs. In the last section we summarize our results.

2 Preliminaries

For the basic graph-theoretical concepts we refer to [34] and for graph algorithms and their complexity to [9, 12]. Consult [26] for a survey on planarity of graphs and theoretical results for crossing number of non-planar graphs and [8] for the basic terminology concerning computational geometry. Notations and definitions for the three dimensional graph drawing are taken from [6, 7, 17, 30] and for the three dimensional visibility representation we use the approach of [5, 13], with slight modifications.

A graph is *planar* if it admits a plane drawing where no two distinct edges intersect. Otherwise the graph is *non-planar*. Planarity testing can be done in linear time [19, 25].

The crossing number of a graph G is the smallest number k so that G can be drawn in the plane with at most k edge intersections [26, 36]. Determining the crossing number of an arbitrary graph is NP-complete [16], but for sparse graphs with small number of vertices many practical methods are known [9, 20, 21, 27, 28]. A two dimensional graph drawing technique for non-planar graphs is to add dummy vertices for each edge intersection and then apply drawing algorithm for planar graphs. Finally dummy vertices are replaced by a crossing. Therefore, finding small crossing numbers improves the quality of a two dimensional drawing.

A three dimensional *polyline* drawing of a graph is a drawing where the vertices are distinct points in three dimensional space and edges are drawn as a polygonal chain connecting the vertices. The drawing is *straight-line*, if each edge is drawn as one straight line without bends. The drawing is *crossing-free*, if there is no crossings of edges. If the coordinates of the vertices and all edge bends are integer grid points, the drawing is called a *grid* drawing.

Let $\mathcal{D}$ be a three dimensional drawing. The *rectangular hull* of $\mathcal{D}$ is the smallest rectangular box with sides parallel to coordinate axis containing the whole drawing. The *volume* of $\mathcal{D}$ is the product of the lengths of the three sides of the rectangular hull of $\mathcal{D}$.

The *aspect ratio* of a drawing $\mathcal{D}$ is $\max\{x/y, x/z, y/x, y/z, z/x, z/y\}$, where x, y and z are the dimensions of $\mathcal{D}$. Notice that for a two dimensional drawings we assume that $z = 1$.

Next we introduce two and three dimensional *visibility representations*. A visibility representation maps the vertices of the graph to geometrical objects. An edge is represented by the visibility relationship between corresponding objects. If two objects are visible to each other, then there is an edge between corresponding vertices.

Usually two dimensional visibility representation is defined as follows. Vertices are associated to closed disjoint lines parallel to x-axis. Two lines are visible if they can be joined by a line segment parallel to y-axis that does not intersect any other object. This kind of visibility is called y-visibility, since the

joining line segment (if it exists) is parallel to the y-axis. On the left in Figure 1, line segments 1 and 2 are visible to each other and segments 2 and 3 are not visible. It is known that a graph admits a two dimensional visibility representation if and only if it is planar [35, 33]. See [9], for an introduction and references to two dimensional visibility representations and linear time algorithms for constructing the representation. Visibility representation can be constructed also using the canonical numbering of the vertices [29]. See an article by Kant [22] for a method constructing compact visibility representation for planar graphs and Kant and He [23] for 4-connected planar triangular graphs.

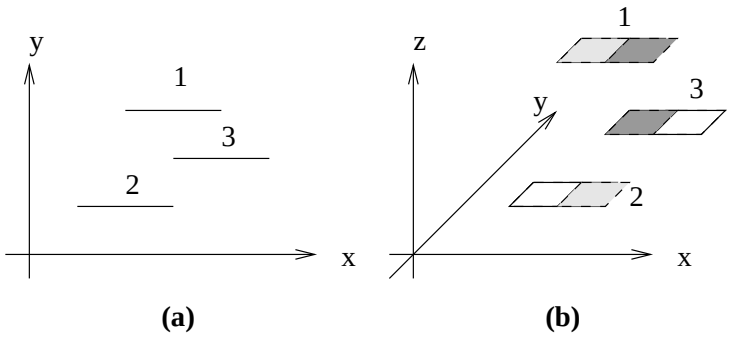

Figure 1: Two and three dimensional visibility representations. Colored areas on the right denote the visibility of regions.

Next we define three dimensional visibility representations. Consider a set of polygonal disjoint regions in $\mathcal{R}^3$ such that the planes determined by the regions are perpendicular to the z-axis. Two regions R_i and R_j are *z-visible* [5] if there is a closed cylinder C of positive length and radius such that the ends of C are contained in R_i and R_j, the axis of C is parallel to z-axis, and the intersection of C with any other region is empty. A graph with n vertices admits a *three dimensional visibility representation* (shortly representation) if and only if its vertices can be associated with n disjoint polygonal regions parallel to x,y-plane such that if vertices v_i and v_j are adjacent in G then their corresponding polygonal regions R_i and R_j are z-visible. See an example on the right in Figure 1, where regions 1 and 2 are visible and regions 2 and 3 are not.

Note that our definition for the three dimensional visibility differs from that in [5, 13], since we allow vertices to be polygonal regions, not only rectangles. Fekete and Meier [13] consider also box visibility representations. Bose et al. [5] show that every planar graph admits a z-visibility representation, but no exact volume was given for the representation. See [1, 2, 14], for other approaches and results of the three dimensional visibility representations.

For the basic geometrical objects (points, lines and regions) we use their normal notations. Since rectangles and squares play important role when constructing visibility representations, we need some special notations and definitions. A rectangle R with sides parallel to x and y axes is a quartet (x_1, y_1, x_2, y_2), where x_1, y_1 are the coordinates of the down left point and x_2, y_2 are the coordinates of the right up point of rectangle. A k-square is a rectangle $(x_1, y_1, x_1 + k, y_1 + k)$, where $k \in \mathcal{Z}$. Rectangle (x_1, y_1, x_2, y_2) with integer dimensions $w = \mid x_2 - x_1 \mid$ and $h = \mid y_2 - y_1 \mid$ can be naturally divided into wh unit squares, h rows and w

columns. In Sections 3 and 4 of this paper we usually assume that rectangles are placed in the plane in such a way that there are integer grid points in the middle of these unit squares. These integer points inside a rectangle R are denoted as point (x_i, y_i) of R, where $x_1 \leq x_i < w$ and $y_1 \leq y_i < h$.

Let S_1, S_2 and S_3 be 2-squares in a plane and p_1, p_2 and p_3 be the midpoints of these squares. Let l be a line that goes through points p_1 and p_2. If p_3 lies also on the line l, we say that squares lie on the same line. If the slope of l is positive, we say that squares are ascending and otherwise in descending order. If S_3 is not on l, then it is on the left side (above) or right side (below) of l.

3 Polyline drawing for planar graphs

This section introduces a technique for converting a two-dimensional visibility representation to a three dimensional one and a method how to use this representation to obtain a 3D polyline drawing.

To convert two dimensional visibility representation into three dimensions, we "stretch" and "roll up" line segments and empty space between those segments which lie on the same line parallel to x-axis to obtain a region parallel to x, y-plane. Then we put the regions one on the other, with the same z-coordinate as the corresponding y-coordinate was in the two dimensional representation, to obtain a three dimensional visibility representation. See [7], for a quite similar method for converting a two-dimensional orthogonal drawing to a three dimensional polyline crossing-free grid drawing.

After this conversion, we route edges between visible regions without edge crossings.

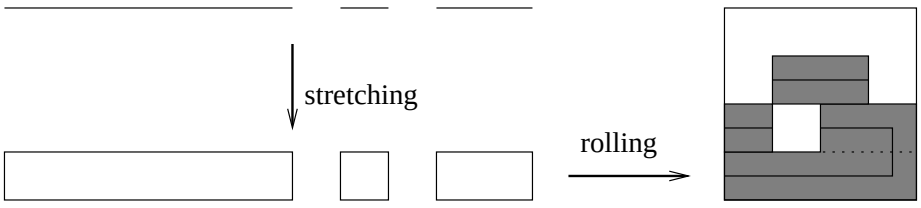

Figure 2: An example on stretching and rolling line segments to obtain polygonal regions with stretching function σ and rolling function δ_4.

Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_1)$ be points in a line parallel to x axis with $0 \leq x_1 < x_2$. Stretching function σ maps the two dimensional line segment $\overline{p_1 p_2}$ parallel to x axis to a two dimensional rectangle with height 1:

$$\sigma(x, y, length) = (x, y_1, x_2, y_1 + 1).$$

Rolling function δ_r, where $r \in \mathcal{Z}_+$, maps the two dimensional rectangle (x_1, y_1, x_2, y_2) to a two dimensional polygonal region such that, for $x_1 \leq x \leq x_2$, $y_1 \leq y \leq y_2$, $x_1 < r$ and $x, y \in \mathcal{N}$,

$$\delta_r(x, y) = \begin{cases} (\lfloor x \mod r \rfloor, y + \lfloor x/r \rfloor), & \text{if } \lfloor x/r \rfloor \text{ is even} \\ (r - (x \mod r) - 1, y + \lfloor x/r \rfloor), & \text{if } \lfloor x/r \rfloor \text{ is odd.} \end{cases}$$

See Figure 2 for an example of using stretching and rolling functions. Next we introduce a lemma that describes the properties of the two dimensional visibility representations. This lemma is needed to prove the properties of the three dimensional representation. We omit the proof since it can be found elsewhere.

Lemma 3.1. *[22] Let G be a planar graph with n-vertices. Then G admits a two dimensional visibility representation Θ which can be constructed $O(n)$ time, line segments and lengths have integer coordinates and the area is at most $(\lfloor 3n/2 \rfloor - 3) \times (n-1)$.*

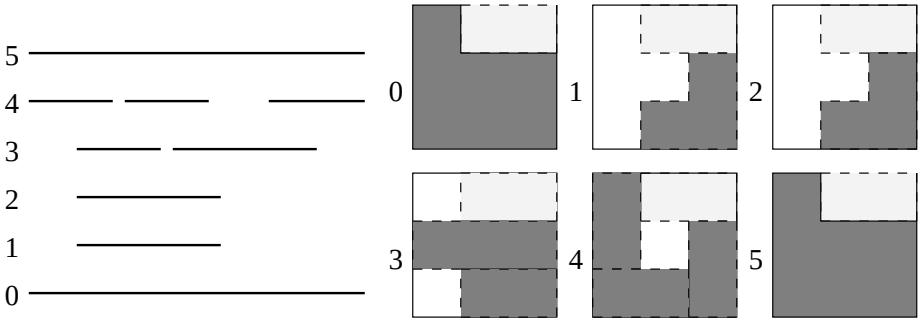

Figure 3: A two dimensional visibility representation and corresponding three dimensional visibility representations constructed by Algorithm 1 with stretching function $\delta_{\lceil\sqrt{9}\rceil} = \delta_3$).

Algorithm 1 converts two dimensional visibility representation to three dimensional visibility representation as follows. First it stretches and rolls up all line segments on the same line parallel to x axis. Then the planes determined by these polygonal regions are placed one on the other. By Lemma 3.1, we can assume that the down left corner of the two dimensional visibility representation is located at point $(0,0)$ in the plane, and after rolling and stretching, the down left point of level i is located at point $(0,0,i)$ in the three dimensional coordinates. The step 4 of the Algorithm 1 decreases the x and y coordinates by $1/2$ of the three dimensional visibility representation. This is done to get integer grid points in the middle of unit squares inside different regions. Throughout this paper, we assume that one unit square inside region contains exactly one integer grid point.

Algorithm 1 3D planar visibility
Input: A n-vertex planar graph $G = (V, E)$ and a two dimensional visibility representation Θ of G.
Output: A 3D visibility representation Θ' of G.
For all lines $y = k$, $k = 0, 1, \ldots,$ containing line segments in Θ **do**
1. Stretch line segments to obtain rectangles by using stretch function σ.
2. Let $r = \lceil\sqrt{n}\rceil$. Roll up rectangles to obtain polygonal regions by using rolling function δ_r.
3. Place polygonal regions to three dimensional space with coordinate $z = k$.

4. Decrease all x and y coordinates by $1/2$.

See Figure 3 for an example of the use of Algorithm 1. The asymptotic volume of this new visibility representation remains unchangeable, but the aspect ratio of this representation gets lower.

The proof of Theorem 3.1 follows from the properties of Algorithm 1.

Theorem 3.1. *Let G be graph that admits a two dimensional visibility representation Θ with volume $x \times y$. Then Algorithm 1 constructs in $O(n)$ time a three dimensional visibility representation Θ' of G with bounding box $\lceil\sqrt{x}\rceil \times \lceil\sqrt{x}\rceil \times y$.*

Proof. Algorithm 1 clearly works in linear time. Let G be a graph with n vertices and let Θ be the two dimensional visibility representation of G. Suppose that there is an edge between vertices v_i and v_j. Then line segments associated to v_i and v_j are visible in Θ. By the construction of Algorithm 1, the corresponding three dimensional regions are visible in representation Θ'. By the construction of Algorithm 1, the dimensions of the bounding box are $\lceil\sqrt{x}\rceil \times \lceil\sqrt{x}\rceil \times y$. The theorem follows. □

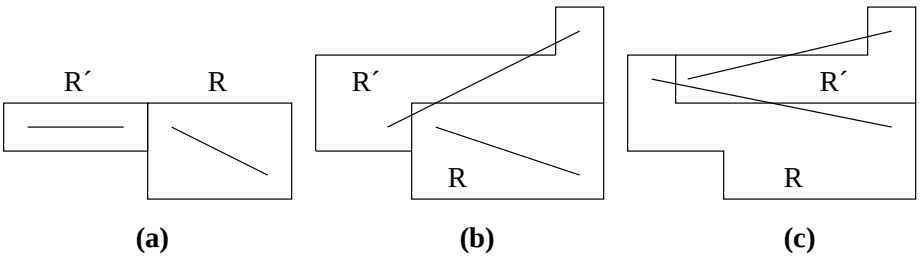

Figure 4: Illustration of the Lemma 3.2.

Next we give a simple lemma, which is needed to construct crossing-free drawings for planar graphs from their visibility representations.

Lemma 3.2. *Let R and R' be polygonal regions produced by Algorithm 1 having common boundary and let p_1 and p_2 be distinct integer points inside R and p_3 and p_4 integer points inside R'. Then following properties hold:*

1. *Straight line segments $l_1 = \overline{p_1p_2}$ and $l_2 = \overline{p_3p_4}$ do not intersect.*
2. *If line segment l crosses the outer boundary of R and lies over R', then it does not cross with any integer point inside R'.*

Proof. Let R and R' be polygonal regions produced by Algorithm 1. If R and R' are rectangles, then any line segment inside one region can not cross with line segment of the other region. Also the property 2 holds trivially (see Figure 4 (a)).

Suppose that R or R' is not a rectangle (see Figures 4 (b) and (c)). Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be distinct integer points inside R, $p_3 = (x_4, y_4)$ and $p_4 = (x_4, y_4)$ be distinct integer points inside R' and line segments $l_1 = \overline{p_1p_2}$

connect points p_3 and p_4 and $l_2 = \overline{p_3 p_4}$ connect points p_3 and p_4. Without loss of generality, we can assume that line segment in the two dimensional visibility representation corresponding to R appears before R' and $y_1 \leq y_2$ and $y_3 \leq y_4$. By the construction of Algorithm 1, we have $y_1 \leq y_3$ and $y_2 \leq y_4$.

If $y_2 \leq y_3$, there can not be any crossings. Therefore assume that $y_2 = y_3$. If the other endpoint of the line segment l_1 (respectively l_2) has higher y coordinate (respectively lower y coordinate), segments can not cross. Suppose then than also $y_2 = y_4$. But now by the construction of Algorithm 1, both lines lie inside their regions. Hence property 1 holds.

Suppose that $y_2 > y_3$. By the properties of Algorithm 1, line segment l_1 can not cross the boundary of R, if $y_1 = y_2$. Therefore, we can assume that $y_1 < y_2$. For a contradiction, suppose that line segment l_1 crosses an integer point $p' = (x', y')$ outside R and l is increasing. Since the endpoints of l_1 belongs to R, it holds $y_1 < y' < y_2$. Now the point p' is inside the region R, but by the construction method of the regions, this can not be true. The case when l_1 is decreasing, can be shown with similar reasoning. The case holds and the lemma follows. □

Now we are ready to give an algorithm that produces a three dimensional crossing-free polyline drawing from the three dimensional visibility representation. First, Algorithm 2 modifies the three dimensional representation adding space between different levels. This increases the height of the drawing to $3n$. Then edges are routed in such a way that the crossings are avoided. Second, Algorithm 2 places a vertex associated to a polygonal region in some integer point inside each region. Then edges are routed without crossings.

Algorithm 2 3D Planar Draw
Input: A 3D visibility representation Θ' of G constructed by Algorithm 1.
Output: A 3D polyline crossing-free grid drawing $\mathcal{D}$ of G.
1. **For** each region R_i in Θ' **do**
   Increase corresponding z coordinate by $3z$.
   Insert a vertex v_i to any integer point inside R_i
2. **For** each regions R_i and R_j at levels $z(R_i)$ and $z(R_j)$ that are visible **do**
   Let R_i^{xy} and R_j^{xy} be the projections of R_i and R_j to the x, y-plane and $P = \{(x, y) \in R_i^{xy} \cap R_j^{xy} \mid (x, y) \in \mathcal{Z}\}$.
   Choose a point (x, y) from P. Route edge with a chain of straight lines from v_i to v_j with bendings in the coordinates $(x, y, z(R_i) + 1)$ and $(x, y, z(R_j) - 1)$.

Next we prove that Algorithm 2 works correctly.

Theorem 3.2. *Let G be a graph with n-vertices and m edges and let Θ' be the three dimensional polygonal visibility representation of G with volume $x \times y \times z$. Then Algorithm 2 constructs in $O(n)$ time a three dimensional polyline crossing-free grid drawing of G with bounding box $x \times y \times 3z$, with integer coordinates and with at most $2m$ total edge bends.*

Proof. By the construction of Algorithm 2, the following properties holds trivially: algorithm works in linear time, the volume of the bounding box is $x \times y \times 3z$, and the vertices and edge bends have integer coordinates.

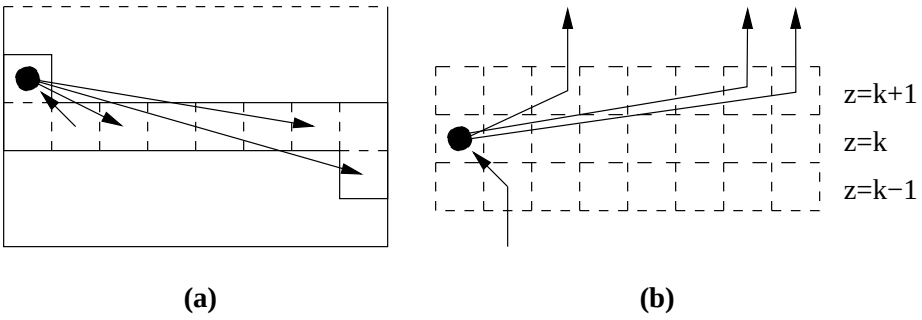

Figure 5: Routing edges between visible regions in Algorithm 2: (a) projection to the x, y-plane; (b) projection to the x, z-plane. The arrows denote the increasing z-coordinate.

Next we show that there is no crossings in the drawing (see Figures 5 (a) and (b)). Let R_i and R_j be regions that are visible with vertices v_i and v_j placed at integer coordinates (x_i, y_i, z_i) and (x_j, y_j, z_j), respectively. Let R_i^{xy} and R_j^{xy} be the projections of R_i and R_j to the x, y-plane and $P = \{(x, y) \in R_i^{xy} \cap R_j^{xy} \mid (x, y) \in \mathcal{Z}\}$. By the properties of the visibility representations, we have that $P \neq \emptyset$.

Algorithm 2 routes edges from v_i to v_j with a chain of straight lines with bendings in the coordinates $(x, y, z(R_i) + 1)$ and $(x, y, z(R_j) - 1)$ where $(x, y) \in P$. We consider now outgoing edges from v_i (incoming edges to v_j can be considered similarly). Since all other line segments from v_i share their starting point (x_i, y_i, z_i) and their endpoints have distinct x and y coordinates with same z coordinate, they cannot cross with each other.

By Lemma 3.2, there is no crossings with other line segments from other regions and no integer point from other regions are crossed. By the properties of the visibility representations, the line segment between $(x, y, z(R_i) + 1)$ and $(x, y, z(R_j) - 1)$ can not cross with any other line segment going strictly upward. Therefore there is no crossings in the drawing.

Finally, since there are at most 2 bends for an edge, there are total by at most $2m$ edge bends. The theorem follows. □

Using Theorems 3.1 and 3.2 for planar graphs with the result of Lemma 3.1, we can deduce the following theorem for the volume of three dimensional polyline grid drawing of planar graphs.

Theorem 3.3. *A three dimensional crossing-free polyline grid drawing of a planar graph with n vertices and m edges can be constructed in $O(n)$ time. The volume of the obtained drawing is at most $\lceil\sqrt{\lfloor 3n/2 \rfloor - 3}\rceil \times \lceil\sqrt{\lfloor 3n/2 \rfloor - 3}\rceil \times 3(n-1)$ and there are at most $2m$ total edge bends.*

4 Polyline drawing for non-planar graphs

This section studies non-planar graphs and their three dimensional crossing-free polyline grid drawings. A well known approach for drawing non-planar graphs in two dimensions is to replace each edge intersection with a dummy vertex

and then apply some graph drawing algorithm for planar graphs [9]. After applying the given drawing algorithm, dummy vertices are replaced back to edge crossings. This drawing paradigm is also suitable for three dimensional graph drawing. Fortunately, three dimensional space gives more freedom than two dimensional space, and instead of replacing dummy vertices by a crossing, it is possible to route edges in such a way that they do not cross. To remove dummy vertices and corresponding edge crossings, we have to again add some space between planes of a three dimensional visibility representation and make polygonal regions slightly larger. This increases the volume of the drawing. Then edges are routed in such a way that crossings are avoided.

In what follows, we assume that exactly one crossing in a drawing of non-planar graph is replaced by a one dummy vertex. If one dummy is used to remove several crossings at the same time, it is easy to add more dummies to get our assumption hold. There are methods for constructing visibility representation assuming that given planar graph is maximalized [29] using Read's maximalization algorithm [31]. Since we are interested only in the method for removing edge intersection, we just do not draw these dummy edges possibly added in the maximalization phase when constructing crossing-free polyline drawing.

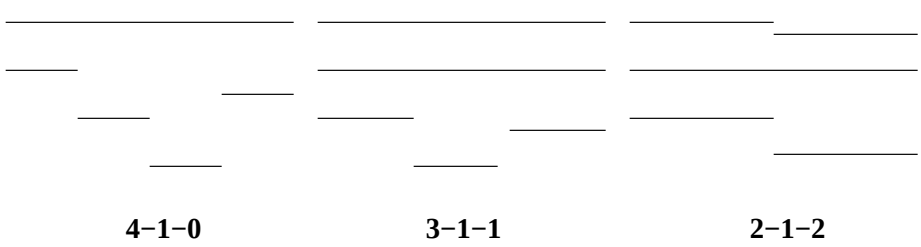

Figure 6: Different types of dummy vertices.

Next we give some simple preliminary results for the dummy vertices of a planarized graph and their relation to the visibility representation. The first property follows from the assumption that one vertex replaces only one edge crossing and from the fact that dummy edges possible added in the maximalization are omitted.

Property 1. *The degree of a dummy vertex is* 4.

There is three different types of dummy vertices in the two dimensional visibility representation, if we do not take account edges that were possibly added in the maximalization phase before construction visibility representation.

Definition 1. *Let G be a non-planar graph and let G_p be a planarized version of G, obtained by adding dummy vertices for each edge crossing. Let Θ be the two dimensional visibility representation of G_p.*

- *A dummy vertex v of G_p is of type $4-1-0$, if there are four visible line segments below the line segment corresponding to v in Θ.*
- *A dummy vertex v of G_p is of type $3-1-1$, if there are three visible line segments below the line segment corresponding to v in Θ and there is one visible line segments above the line segment corresponding to v in Θ.*

- *A dummy vertex v of G_p is of type $2-1-2$, if there are two visible line segments below the line segment corresponding to v in Θ and there are two visible line segments above the line segment corresponding to v in Θ.*

The cases where word "below" is changed to "above" and word "above" to "below" in the previous definition, are called types $0-1-4$ and $1-1-3$, respectively. See Figure 6 for an example of different dummy types. The long connected segment (in the case $3-1-1$, the segment in the middle) denotes the segment corresponding to a dummy vertex. Notice, that the vertical distances between dummy vertex and its adjacent vertices might be different and there might be longer vertical space between horizontal segments, since the dummy edges possibly added in the maximalization phase are omitted.

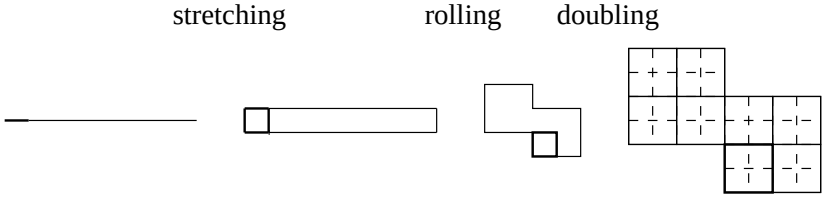

Figure 7: Algorithm 3 with rolling function δ_4. A line segment is stretched, rolled and doubled. The bolded segment clarifies the effect of the doubling procedure. The small dashed square of the polygonal region denote the location of the distinct integer points after doubling.

The following Algorithm 3 modifies the two dimensional visibility representation to have enough free space inside regions and between different levels to route edges without intersections. It doubles the x and y dimensions of the polygonal regions obtained from Algorithm 1. As in the previous section, we assume that after applying Algorithm 3 (which calls Algorithm 1) the unit squares inside regions contain exactly one integer point, and this point is located in the middle of the unit square. See Figure 7 for an illustration of the effects of Algorithm 3.

Algorithm 3 3D non-planar visibility
Input: A n-vertex non-planar graph $G = (V, E)$, planarized version G_p of G and a two dimensional visibility representation Θ of G_p.
Output: A 3D visibility representation Θ'' of G_p.
1. Use Algorithm 1 to obtain three dimensional visibility representation Θ' of G_p.
2. Increase all x and y coordinates of regions in Θ' by $2x$ and $2y$, respectively.
3. Increase all z coordinates of regions in Θ' by $3z$.

In what follows, we call the regions corresponding to the dummy vertices constructed by Algorithm 3 shortly dummy regions. Next we give further simple properties of dummy regions. Property 2 shows the connection of a unit length line segment of a two dimensional visibility representation Θ and 2-squares inside a three dimensional visibility representation constructed from Θ by Algorithm 3.

Property 2. *Algorithm* 3 *maps every unit length line segment of a two dimensional visibility representation to a* 2*-square containing* 4 *unit squares.*

Property 3 follows directly from the properties of the visibility representation.

Property 3. *Every* 2*-square corresponding to a unit length line segment of a two dimensional visibility representation of a dummy region is visible on upward to at most one region and downward to at most one region.*

The next property is derived from Properties 1 and 3. It gives a lower bound for the area of a dummy region.

Property 4. *Every dummy region contains at least two adjacent* 2*-squares.*

Since the mapping of a two dimensional visibility representation to a three dimensional visibility representation applying Algorithm 3 is one-to-one, every unit length line segment is mapped to distinct 2-squares.

Property 5. *Let* S_1 *and* S_2 *be* 2*-squares corresponding to a unit length line segment of a two dimensional visibility representation. If* $S_1 \cap S_2 \neq \emptyset$*, then* $S_1 = S_2$.

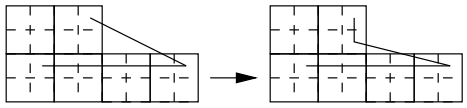

Figure 8: Algorithm $2'$ makes the correct routing for the doubled regions constructed by Algorithm 3. The upperline segment is routed with a bend to avoid intersecting integer points outside the region.

If Algorithm 2 is used straightly to construct crossing-free polyline grid drawing for three dimensional visibility representation constructed by Algorithm 3, there might appear crossings since the Lemma 3.2 can not be applied without slight modifications to the routing procedure of the Algorithm 2.

We give next a modified version of Algorithm 2 for doubled regions. The main idea behind Algorithm $2'$ is to take account the possibility that if there is more than 2 rows in a polygonal region, then the first and last row should be avoided. Incoming edges to these rows are routed by adding a bend to the integer grid point with one lower or higher y coordinate. Property 2 guarantees that there are such a free grid point. After this modification, we can again apply Lemma 3.2 to show the correctness of Algorithm $2'$. The number of bends for an edge increases to four. We omit the proof since it is similar to the one given for Theorem 3.2. See Figure 8 for an illustration of Algorithm $2'$ that works correctly with doubled regions.

Algorithm 2' 3D Doubled-Region-Draw
Input: A 3D visibility representation Θ' of G constructed by Algorithm 3.
Output: A 3D polyline crossing-free grid drawing $\mathcal{D}$ of G.
1. For each region R_i in Θ' with r rows **do**
if $r > 2$
Insert a vertex v_i to integer point p_i in rows 2 to $r-1$ inside R_i
else $(r = 2)$
Insert a vertex v_i to integer point p_i inside R_i **od**
2. For each visible regions R_i and R_j with $z(R_i) < z(R_j)$ having r_i and r_j rows **do**
Let R_i^{xy} and R_j^{xy} be the projections of R_i and R_j to the x,y-plane and $P = \{(x,y) \in R_i^{xy} \cap R_j^{xy} \mid (x,y) \in \mathcal{Z}\}$ Choose a point (x,y) from P. Route an edge with a chain of straight lines from p_i to p_j with bendings in the coordinates $(x,y,z(R_i)+1)$ and $(x,y,z(R_j)-1)$
if p is located on the row 1 or r_i in R_i
add a bend to the coordinate $(x,y+1,z(R_i)+1)$ or $(x,y-1,z(R_i)+1)$
if p is located on the row 1 or r_j in R_j
add a bend to the coordinate $(x,y+1,z(R_j)-1)$ or $(x,y-1,z(R_j)-1)$

Lemma 4.1. *Let G be a graph with n-vertices and m edges and let Θ' be the three dimensional polygonal visibility representation of G with volume $x \times y \times z$ constructed by Algorithm 3. Then Algorithm 2' constructs in $O(n)$ time a three dimensional polyline crossing-free grid drawing of G with bounding box $2x \times 2y \times 3z$, with at most $4m$ total edge bends.*

Next we give three routing lemmas, which are used in Algorithm 4 for constructing crossing free drawings for non-planar graphs. By the properties 1 to 5, it is enough to show that given any four suitable chosen 2-squares in a dummy region, we can place four points to integer coordinates inside these 2-squares, and connect any two of these points with a polygonal chain of line segments in such a way that these chains do not cross. We will show later that these polygonal chains do not cross with other line segments corresponding to other regions.

The following lemmas give a routing for the edges of a dummy region in such a way that all line segments which are added to avoid edge crossings are parallel to x,y-plane. Only the line segments connecting visible regions go strictly upward or downward. It is obvious that there are also other possible routings, but we construct only this one.

The first lemma handles the case when all 2-squares in a polygonal region are distinct and remaining two lemmas considers cases where some of the 2-squares are same.

Lemma 4.2. *Let R be a polygonal region produced by Algorithm 3 and let S_1, S_2, S_3 and S_4 be distinct 2-squares inside R corresponding to unit length line segments of a two dimensional visibility representation such that they are projected to R and no two of these 2-squares are the same. Then there exists a placement of points p_1, p_2, p_3 and p_4 into integer coordinates inside squares S_1, S_2, S_3 and S_4, respectively, and polygonal chains l_1 connecting points p_1 and p_2 and l_2 connecting points p_3 and p_4 that are parallel to the x,y-plane and there are no intersections.*

Proof. There are three different base cases depending on the placement of the four squares inside R. We prove the lemma by constructing polygonal chains connecting distinct points with the claimed properties. We enumerate different cases to make it easier to refer them later. Let the points (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) denote the down left point of 2-squares S_1, S_2, S_3 and S_4 inside R.

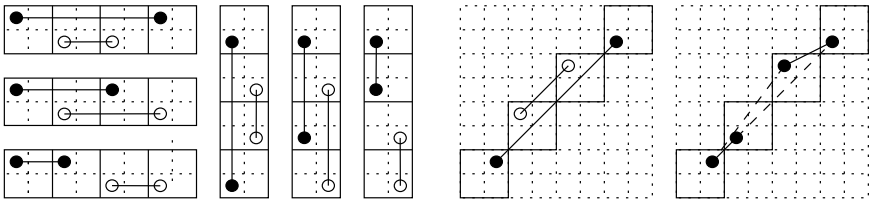

Figure 9: Cases $1(a)$ and $1(b)$ of Lemma 4.2

1. Suppose that all S_1, S_2, S_3 and S_4 lie on the same line. See Figure 4 for an illustration of the routing procedure.

   (a) If 2-squares are parallel to x-axis (respectively y-axis), let $p_1 = (x_1, y_1 + 1)$ (resp. $p_1 = (x_1, y_1)$), $p_2 = (x_2, y_2 + 1)$ (resp. $p_2 = (x_2, y_2)$), $p_3 = (x_3, y_3)$ (resp. $p_3 = x_3 + 1, y_3)$) and $p_4 = (x_4, y_4)$ (resp. $p_4 = (x_4 + 1, y_4)$).

   (b) If the squares are not parallel to x or y axes, suppose that they are in the ascending order. Let S_i, S_j, S_k and S_l be the ordering of S_1, S_2, S_3 and S_4 induced by increasing y coordinate. Place the point corresponding to the square S_i to the point $(x_i + 1, y_i + 1)$ and the point corresponding to the square S_l to the point (x_l, y_l). If the points of S_i and S_l are to be connected, place remaining points to $(x_j, y_j + 1)$ and $(x_j, y_j + 1)$, otherwise to (x_j, y_j) and $(x_j, y_j + 1)$.

   (c) Descending rectangles can be handled similarly to the case (b).
   with the similar reasoning as in the previous case.

2. Suppose that S_i, S_j and S_k lie on the same line and the S_l is not on the same line as the other squares. See Figures 10, 11 and 12 for illustrations of the routing procedure for the case 2.

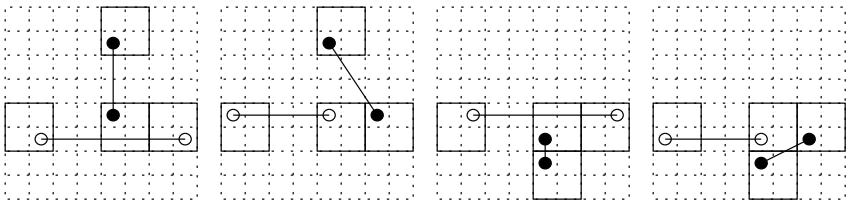

Figure 10: Case $2(a)$ of Lemma 4.2.

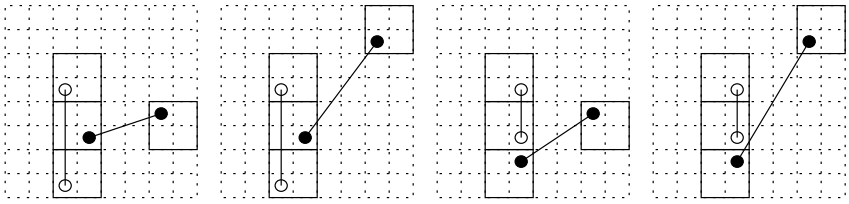

Figure 11: Case 2(b) of Lemma 4.2.

(a) If S_i, S_j and S_k, ordered by the increasing x-coordinate, lie on the same line parallel to x-axis and the fourth square S_l is above (respectively below) of the other three. Place the point s_j corresponding to S_j to $(x_j, y_j + 1)$ (resp. (x_j, y_j)) and s_l to (x_l, y_l) (resp. $(x_l, y_l + 1)$). If s_l is to be connected with s_j, place remaining points to (x_i, y_i) (resp. $(x_i, y_i + 1)$) and (x_k, y_k) (resp. $(x_k, y_k + 1)$), otherwise to points $(x_i, y_i + 1)$ (resp. (x_i, y_i)) and $(x_k, y_k + 1)$ (resp. (x_k, y_k)). See Figure 10 for an example.

(b) If S_i, S_j and S_k, ordered by the increasing y-coordinate, lie on the same line parallel to y-axis, and the fourth square S_l is on the right side (respectively left) of the other three. Place the point s_j corresponding to S_j to $(x_j + 1, y_j)$ (resp. (x_j, y_j)) and if $x_l \leq x_k$, then place s_l to $(x_l, y_l + 1)$ (resp. $(x_l + 1, y_l + 1)$, otherwise to (x_l, y_l) (resp. $(x_l + 1, y_l)$). If s_l is to be connected with s_j, place remaining points to (x_i, y_i) (resp. $(x_i + 1, y_i)$) and (x_k, y_k) (resp. $(x_k + 1, y_k)$), otherwise to points $(x_i + 1, y_i)$ (resp. (x_i, y_i)) and $(x_k + 1, y_k + 1)$ (resp. $(x_k, y_k + 1)$). See Figure 11 for an example.

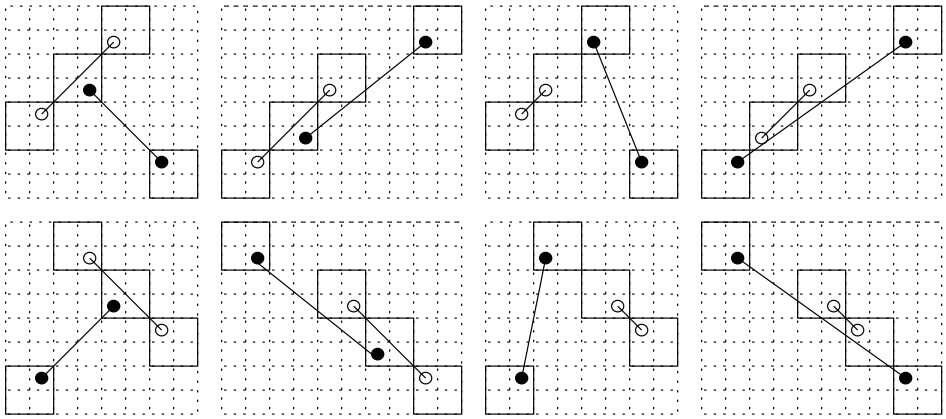

Figure 12: Cases 2(c) and 2(d) of Lemma 4.2.

(c) If S_i, S_j and S_k, ordered by the increasing y-coordinate, lie on the same ascending line not parallel to x or y-axis, and the fourth square S_l is below (respectively above) of the other three. If s_j is going

to be connected with s_k, then place s_j to (x_j+1, y_j+1) (resp. (x_j+1, y_j+1)), otherwise to (x_j, y_j+1) (resp. (x_j+1, y_j+1). If $x_l \leq x_k$, then place s_l to (x_l, y_l+1) (resp. (x_l+1, y_l+1), otherwise to (x_l, y_l) (resp. (x_l+1, y_l)). Place remaining points to (x_i+1, y_i+1) (resp. (x_i, y_i+1)) and (x_k, y_k) (resp. (x_k+1, y_k)). See Figure 12 for an example.

(d) If S_i, S_j and S_k, ordered by the decreasing y-coordinate, lie on the same descending line not parallel to x or y-axis, and the fourth square S_l is below (respectively above) of the other three. If s_j is going to be connected with s_k, then place s_j to (x_j, y_j) (resp. (x_j+1, y_j+1)), otherwise to (x_j+1, y_j) (resp. (x_j, y_j). If $x_l \leq x_i$, then place s_l to (x_l+1, y_l+1) (resp. (x_l, y_l+1)), otherwise to (x_l+1, y_l) (resp. (x_l, y_l)). Place remaining points to (x_i+1, y_i) (resp. (x_i, y_i)) and (x_k, y_k+1) (resp. (x_k, y_k+1)). See Figure 12 for an example.

3. Suppose S_1, S_2, S_3 and S_4 are 2-squares with the property that no more than two of them lie on the same line. If for all possible placements of points s_1, s_2, s_3 and s_4 inside squares S_1, S_2, S_3 and S_4, three points are the boundary points of the convex hull C of these points, and the fourth point is the interior point of C, then any pair of points can be connected with a straight line without crossings. For all squares S_i having the lowest y coordinate place s_i to (x_i+1, y_i+1) and for other squares S_j place s_i to (x_i, y_i).

4. Suppose that the previous case holds with the exception that there exists a placement for points inside squares that all points are boundary points of the convex hull of these points, then sort squares in increasing order by their y coordinates and if there is two squares with the same y coordinate, order them in increasing order by their x coordinates. Let squares S_i, S_j, S_k and S_l be in order. We have that $y_i \leq y_j \leq y_k \leq y_l$ with the property that at most two equalities hold in inequality chain and there is no two consecutive equality. This implies 5 distinct subcases.

   (a) If $y_i = y_j < y_k = y_l$ and $x_i < x_i$ and $x_k < x_l$, then set $s_i = (x_i+1, y_i+1)$, $s_j = (x_j, y_j+1)$, $s_k = (x_k+1, y_k)$ and $s_l = (x_l, y_l)$. If points can be connected with straight line segments without crossings, do so. Otherwise connect first points s_j and s_k with a straight line and route the other edge starting from s_i with bendings at points (x_k, y_k) and (x_k+1, y_k+1) following a straight line to s_l.

   (b) If $y_i = y_j < y_k < y_l$ and $x_i < y_i$, then set $s_i = (x_i+1, y_i+1)$, $s_j = (x_j, y_j+1)$ and $s_l = (x_l, y_l)$. If at least one of the points of S_k is on the left side of the line going through points s_i and s_j, then set $s_k = (x_k+1, y_k)$, otherwise set $s_k = (x_l, y_l)$ (notice that S_k is not inside triangle with boundary points s_i, s_j, s_l). If points can be connected with straight line segments without crossings, do so. Otherwise, if s_k was on the left side, connect s_k and s_j with a straight line and route the other edge starting from s_i with bendings at points (x_k, y_k) and (x_k+1, y_k+1) following a straight line to s_l. If s_k was on the right side, connect s_i and s_k with a straight line

and route the other edge starting from s_j with bendings at points (x_k+1, y_k) and (x_k, y_k+1) following a straight line to s_l.

(c) If $y_i < y_j = y_k < y_l$ and $x_j < y_k$, then set $s_i = (x_j, y_j+1)$, $s_j = (x_j+1, y_j)$, $s_k = (x_k, y_k)$ and $s_l = (x_l, y_l)$. If points can be connected with straight line segments without crossings, do so. Otherwise connect points s_j and s_k with a straight line segment and route the other edge starting from s_i with bendings at points (x_j, y_j) and (x_j+1, y_j+1) following a straight line to s_l.

(d) Suppose that $y_i < y_j < y_k < y_l$. Set $s_i = (x_i, y_i+1)$ and $s_l = (x_l, y_l)$. Let l be a line going through points s_i and s_l. Now three distinct subcases arise. If S_j and S_k are both on the left side of l, right side of l or one of them, is on the left side and the other on the right side.

Consider the first subcase. Set $s_j = (x_j+1, y_j+1)$ and $s_k = (x_k, y_k)$. If points can be connected with straight line segments without crossings, do so. Otherwise connect points s_j and s_l with a straight line segment and route the other edge starting from s_i with bendings at points (x_j+1, y_j) and (x_j, y_j+1) following a straight line to s_k.

Consider the second subcase. Set $s_j = (x_j, y_j)$ and $s_k = (x_k, y_k)$. If points can be connected with straight line segments without crossings, do so. Otherwise connect points s_j and s_l with a straight line segment and route the other edge starting from s_i with bendings at points (x_j+1, y_j) and (x_j, y_j+1) following a straight line to s_k.

Consider the third subcase. If S_j is on the left side and S_k is on the right side set $s_j = (x_j+1, y_j+1)$ and $s_k = (x_k, y_k)$, otherwise set $s_j = (x_j, y_j+1)$ and $s_k = (x_k+1, y_k)$. If points can be connected with straight line segments, do so. Otherwise connect points s_j and s_k with a straight line and route the other edge starting from s_i with bendings at points (x_j+1, y_j) and (x_j, y_j+1) following a straight line to s_l if S_j was on the left side and otherwise route the edge with bendings at points (x_k, y_k) and (x_k+1, y_k+1) following a straight line to s_l.

(e) If $y_i < y_j < y_k = y_l$ and $x_k < y_k$, then set $s_i = (x_i+1, y_i+1)$, $s_j = (x_k+1, y_k)$ and $s_l = (x_l, y_l)$. If S_j is on the left side of the line going through points s_i and s_k, then set $s_j = (x_j+1, y_j)$, otherwise set $s_j = (x_j, y_j)$ (notice that S_j is not inside triangle with boundary points s_i, s_k, s_l). If points can be connected with straight line segments without crossings, do so. Otherwise, if s_l was on the left side, connect s_j and s_l with a straight line and route the other edge starting from s_i with bendings at points (x_j, y_j) and (x_j+1, y_j+1) following a straight line to s_k. If s_j was on the right side, connect s_j and s_k with a straight line and route the other edge starting from s_i with bendings at points (x_j+1, y_j) and (x_j, y_j+1) following a straight line to s_l.

□

Lemma 4.2 did not cover the possibility that two 2-squares are the same. This situation may happen when there is visibility relationships in two directions, namely upward and downward, from the same unit length line segment

of a two dimensional visibility representation. Since every 2-square has four integer points, we can easily choose among them points where incoming edges are routed. Therefore, we give two more routing lemmas for these special cases. The following lemma handles the case when there is only three squares for four points.

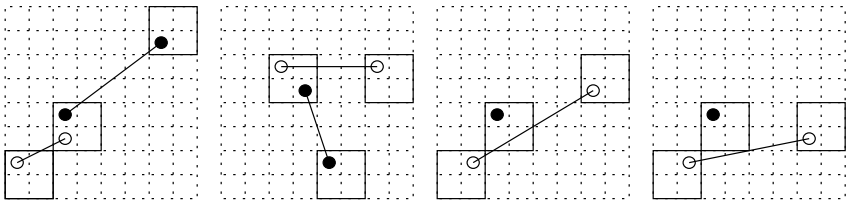

Figure 13: Case 2 of Lemma 4.3.

Lemma 4.3. *Let R be a polygonal region produced by Algorithm 3 and $P = \{p_1, p_2, p_3, p_4\}$ be a set of points and S_1, S_2 and S_3 be distinct 2-squares inside R corresponding to unit length line segments of a two dimensional visibility representation. Then there exists a placement of points in P to integer points inside squares S_1, S_2 and S_3 in such a way there are polygonal chains l_1 and l_2, possibly of zero length, connecting points p_1 with p_2 and p_3 with p_4 without intersections for any mapping function f, with $f : P \to S_i$, where $i \in \{1, 2, 3\}$ and f is a surjection.*

Proof. There are two different cases depending on the placement of these three 2-squares inside the R. We prove the lemma by constructing polygonal chain connecting distinct points for all possible forms of function f.

1. Suppose that all three 2-squares lie on the same line. Sort squares in increasing order by their y coordinates and if there is three squares with the same y coordinate, order them in increasing order by their x coordinates. Assume that S_i, S_j, S_k are sorted in this order.

   (a) Suppose first that one of the squares contains points p_1 and p_2 (the case where one of the squares contains points p_3 and p_4 can be handled similarly). If $x_i \leq x_j$, then for the square S_*, where $* \in \{i, j, k\}$, containing the two points, set $p_1 = p_2 = (x_* + 1, y_*)$ and place the other two points to the upper right and down left corners and connect them with a straight line. Otherwise ($x_1 > x_2$) S_* contain the two points, set $p_1 = p_2 = (x_*, y_*)$ and, for the other two points, place to the upper left and down right corners and connect them with a straight line.

   (b) Suppose now that there is no square having points p_1 and p_2 (and no square containing p_3 and p_4). If $x_i \leq x_j$ and S_j contains two points, set those points to the upper left and down right points, and place remaining two points to the upper right and down left points, respectively. Now points can be connected with straight lines without intersections. If $x_i > x_j$ and S_j contains two points, set those points

to the down left and upper right points, and place remaining two points to the upper left and down right points, respectively. Now points can be connected with straight lines without intersections.

2. Suppose that squares do not lie on the same line. Sort squares in increasing order by their y coordinates and if there are same y coordinates, use increasing x coordinate to sort them. Assume that S_i, S_j, S_k are sorted in this order. Let l be a line that goes through midpoints of S_i and S_k. See Figure 13 for an illustration of the following routing procedure.

   (a) Suppose first that S_j lies on the left side of line l. If S_*, where $* \in \{i, j, k\}$, contains both p_1 and p_2, set $p_1 = p_2 = (x_*, y_*)$ (the case where one of the squares contains points p_3 and p_4 can be handled similarly). Place the remaining two points to two of the following three unoccupied points $(x_i, y_i + 1)$, $(x_j + 1, y_j)$ and (x_k, y_k). If S_i contains p_1 and p_3 (p_2 and p_4 with similar reasoning) place these points to $(x_i, y_i + 1)$ and $(x_i + 1, y_i + 1)$, if S_j contain p_1 and p_3 place these points to $(x_j + 1, y_j)$ and $(x_j + 1, y_j + 1)$ and if S_k contain them, place these points to (x_k, y_k) and $(x_k + 1, y_k)$. For the remaining two points, place them to two of the following three points, depending on the square that contains them, to $(x_i, y_i + 1)$, $(x_j + 1, y_j)$ or (x_k, y_k). Connect now corresponding points with a straight line segments. If there is now an intersection, change the places of p_1 and p_3 to remove the crossing.

   (b) Suppose that S_j lies on the right side of line l. If S_*, where $* \in \{i, j, k\}$, contains both p_1 and p_2, set $p_1 = p_2 = (x_*, y_*)$ (the case where one of the squares contains points p_3 and p_4 can be handled similarly). Place the remaining two points to two of the following three unoccupied points $(x_i, y_i + 1)$, $(x_j + 1, y_j + 1)$ and (x_k, y_k). If S_i contain p_1 and p_3 (p_2 and p_4 with similar reasoning) place these points to $(x_i, y_i + 1)$ and $(x_i + 1, y_i + 1)$, if S_j contain p_1 and p_3 place these points to (x_j, y_j) and $(x_j, y_j + 1)$ and if S_k contain them, place these points to (x_k, y_k) and $(x_k + 1, y_k)$. Place the remaining two points to two of the following three points, depending on the square that contains them, to $(x_i, y_i + 1)$, $(x_j + 1, y_j)$ or (x_k, y_k). Connect now corresponding points with a straight line segments. If there is now an intersection, change the places of p_1 and p_3 to remove the crossing.

□

Third routing lemma considers the case when four points are mapped to two squares.

Lemma 4.4. *Let R be a polygonal region produced by Algorithm 3 and $P = \{p_1, p_2, p_3, p_4\}$ be a set of points and S_1 and S_2 be distinct 2-squares inside R corresponding to unit length line segments of a two dimensional visibility representation. Then there exists a placement for points of P to integer points inside squares S_1 and S_2 in such a way there are polygonal chains l_1 and l_2, possible of zero length, connecting points p_1 with p_2 and p_3 with p_4 without*

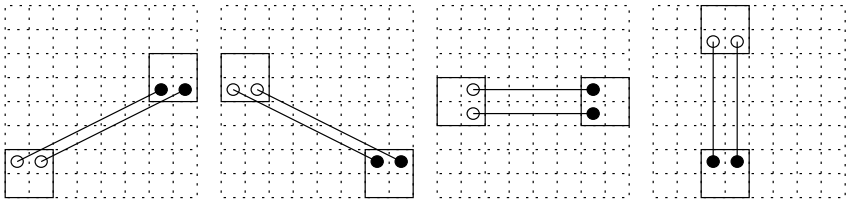

Figure 14: Case 1 of Lemma 4.4.

intersections for any mapping function f, which maps two points to S_1 and two points to S_2.

Proof. There are two different cases depending on the placement of four points in 2-squares. In the first case points p_1 and p_2 are mapped to the same square and in the second case points p_1 and p_2 are mapped to distinct squares. We prove the lemma by constructing polygonal chain connecting distinct points.

1. Suppose first that S_1 contains points p_1 and p_3 and S_2 points p_2 and p_4. Suppose that S_1 is below S_2 (the case when S_1 above S_2 can be handled similarly). Set $p_1 = (x_1, y_1 + 1)$, $p_3 = (x_1 + 1, y_1 + 1)$, $p_2 = (x_2, y_2)$, $p_4 = (x_2 + 1, y_2)$ and connect adjacent points with a straight line. If S_1 and S_2 have the same y coordinate, suppose that S_1 is on the left side of S_2 (if S_1 is on the right side of S_2, similar reasoning can be used). Set $p_1 = (x_1 + 1, y_1)$, $p_3 = (x_1 + 1, y_1 + 1)$, $p_2 = (x_2, y_2)$, $p_4 = (x_2, y_2 + 1)$ and connect adjacent points.

2. Suppose now that S_1 contains points p_1 and p_2 and S_2 points p_3 and p_4. Set $p_1 = p_2 = (x_1, y_1)$ and $p_3 = p_4 = (x_2, y_2)$.

□

Next we give an important property of the previous three routing lemmas. It shows that given any polygonal chain of line segments constructed by Lemma 4.2, it does not cross with any integer point outside its region.

Lemma 4.5. *Let R be a polygonal region with r rows constructed by Algorithm 3 with polygonal chains l_1 and l_2 constructed by Lemma 4.2, 4.3 or 4.4. Then it holds that l_1 and l_2 do not cross with any integer point outside R.*

Proof. The rows 1 and r are used in the routing only if $r = 2$ or if the convex hull containing squares where points are assigned is a rectangle. by Lemma 3.2, it follows that there are no crossings between polygonal chains in the adjacent regions. □

Now we are ready to introduce an algorithm that produces three dimensional crossing-free polyline drawings for non-planar graphs. Our algorithm gets as input a non-planar graph, the planarized version of it and the two dimensional visibility representation of it. First it calls Algorithms 1 to convert the input to the three dimensional visibility representation, then it calls Algorithm 3 to double the representation and after that it creates a three dimensional drawing

using Algorithm $2'$. Finally, dummy vertices are removed and the adjacent edges of the dummies are routed without crossings using Lemmas 4.2, 4.3 and 4.4. The type of the dummy vertex implies the direction where from edges arrive to the dummy region. If the type of the dummy vertex is $4-1-0$ ($0-1-4$), then there arrives four lines to the region of the dummy vertex from the down (up) and if the type of the dummy vertex is $3-1-1$ ($1-1-3$), there arrives three lines from down (up) and one line from the up (down) of the region. In the case $2-1-2$ there arrives two lines from down and two lines from the up of the region.

Algorithm 4 3D Planar-Draw
Input: A non-planar graph $G=(V,E)$ with n vertices and m edges, and a planarized version G_p of G with n' added dummy vertices.
Output: A 3D polyline crossing-free grid drawing $\mathcal{D}$ of G.
1. Use Algorithm 1 to obtain a 3D visibility representation Θ' of G_p.
2. Use Algorithm 3 to modify Θ' to obtain visibility representation Θ'' of G_p.
3. Use Algorithm $2'$ to obtain 3D grid drawing drawing $\mathcal{D}$ of G_p from Θ''.
4. **For** each dummy region R_i of vertex v_i at level m, $m=0,1,2,\ldots,$ in Θ'' **do**
Remove all drawn edges incident v_i between levels $m-1$ and $m+1$, but leave the endpoints of the incident edges at levels $m-1$ and $m+1$.
5. **For** each dummy region R_i of vertex v_i at level m, $m=0,1,2,\ldots$ in Θ'' **do**
Let R_1^{xy}, R_2^{xy}, R_3^{xy} and R_4^{xy} be the projections of R_1, R_2, R_3 and R_4 to the $y=0$ plane that are visible to R ordered by the increasing z coordinate. Choose a 2-square corresponding to a unit length line segment in the two dimensional visibility representation from each R^{xy} and denote these squares by S_1, S_2, S_3 and S_4.
If $|\{S_1\cup S_2\cup S_3\cup S_4\}|=4$ **do**
Apply Lemma 4.2 to find points $p_i=(x_i,y_i,m)\in S_i$, where $i\in\{1,2,3,4\}$ and a polygonal chain l_1 and l_2 connecting points p_1 with p_2 and p_3 with p_4.
else if $|\{S_1\cup S_2\cup S_3\cup S_4\}|=3$ **do**
Apply Lemma 4.3 to find points $p_i=(x_i,y_i,m)\in S_i$, where $i\in\{1,2,3,4\}$ and a polygonal chain connecting points p_1 with p_2 and p_3 with p_4.
else (now $|\{S_1\cup S_2\cup S_3\cup S_4\}|=2$) **do**
Apply Lemma 4.4 to find points $p_i=(x_i,y_i,m)\in S_i$, where $i\in\{1,2,3,4\}$ and a polygonal chain connecting points p_1 with p_2 and p_3 with p_4.
Draw polygonal chains l_1 and l_2. Let p_1', p_2', p_3' and p_4' be the incoming points of the edges at levels $m-1$ and $m+1$ corresponding to the regions R_1, R_2, R_3 and R_4 depending on the type of the dummy vertex.
Draw straight lines $\overline{p_1'p_1}$, $\overline{p_2p_2'}$, $\overline{p_3',p_3}$ and $\overline{p_4p_4'}$.
od

Theorem 4.1. *Let G be a non-planar graph with n-vertices and m edges and let G_p be the planarized version of G with n vertices and n' dummy vertices. Then Algorithm 4 constructs in $O(n+n')$ time a three dimensional polyline crossing-free grid drawing of G with bounding box $2\lceil\sqrt{\lfloor 3(n+n')/2\rfloor-3}\rceil\times 2\lceil\sqrt{\lfloor 3(n+n')/2\rfloor-3}\rceil\times 3(n+n')$ and there is at most $4m+19n'$ total edge bends.*

Proof. The properties of Algorithm 3 guarantee that the claimed upper bound

for the volume of the drawing holds. To show that Algorithm 4 runs in $O(n+n')$ time, it is enough to consider the running time to detect different cases for routing, since Algorithms 1, 3 and $2'$ that are called from Algorithm 4 run in $O(n+n')$ time and all other operations of Algorithm 4 are executed once for each dummy vertex. Each subcase of Lemmas 4.2, 4.3 and 4.4 can be performed in constant time and to test wheter any two straight line segments cross can be done in constant time. Therefore, the total running time of Algorithm 4 is $O(n+n')$.

Theorem 4.1 guarantees that all edges between non-dummy vertices are drawn without intersections and Lemmas 4.2, 4.4 and 4.4 guarantee that there are no intersections inside dummy regions. By Lemma 3.2, there are no crossings with other line segments from other regions and no integer point from other regions is crossed.

To prove the upper bound for the total edge bends, let R_1, R_2, R_3 and R_4 be regions that are joined with an intersection on a dummy vertex before step 4 of Algorithm 4. By Algorithm $2'$, there is at most two bends before arriving to point p_i' just one level before or after a dummy region. One bend is possibly added when connecting points p_i' and p_i, where p_i is a point located at the dummy region. The worst case of Lemmas 4.2, 4.3 and 4.4 adds at most 5 bends (case $4(b)$ of Lemma 4.2) when connecting points p_1 with p_2 (or p_3 with p_4) in the dummy region, but on the other hand, the points p_3 and p_4 (or p_1 and p_2) are always connected with a straight line segment. Also the points p_3 and p_4 might add the total number of bends by 2. Since one dummy vertex can cause up to 19 edge bends and an edge constructed by Algorithm $2'$ have at most 4 edge bends, we have at most $4m + 19n'$ edge bends. The theorem follows.

□

From the Theorem 4.1 following result can be derived.

Corollary 4.1. *Let G be a non-planar graph with n vertices. If the crossing number of G is $f(n)$ then G admits a three dimensional crossing-free polyline grid drawing with volume $O(\sqrt{n+f(n)}) \times O(\sqrt{n+f(n)}) \times O((n+f(n)))$.*

By Corollary 4.1, a non-planar graph with $O(n)$ crossings admits a polyline crossing-free grid drawing with $O(n^2)$ volume. It is not known whether there exist any non trivial classes of graphs admiting linear number of crossings [36], but on the other hand, many sparse graphs have only few crossings.

5 Conclusions

In this paper we showed that the three dimensional polygonal z-visibility representation for planar graphs can be drawn with volume $\lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times \lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times (n-1)$. The time complexity for this representation is linear in the number of vertices in the graph. We also showed that planar graphs with n vertices admit a three dimensional crossing-free polyline grid drawing with at most two bends for an edge with bounding box $\lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times \lceil\sqrt{\lfloor 3n/2\rfloor - 3}\rceil \times 3(n-1)$. The final result of this paper was that non-planar graphs with n' crossings can be drawn with volume $2\lceil\sqrt{\lfloor 3(n+n')/2\rfloor - 3}\rceil \times 2\lceil\sqrt{\lfloor 3(n+n')/2\rfloor - 3}\rceil \times 3(n+n'-1)$ having at most $2m+19n'$ edge bends. The

drawing algorithm for non-planar graphs runs in $O(n + n')$ time, if planarized version of the G graph is given as input.

References

[1] H. Alt, M. Godau, and S. Whitesides. Universal 3-dimensional visibility representations for graphs. In F.J. Brandenburg, editor, *Proc. of the Symposium on Graph Drawing (GD'95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 8–19. Springer, 1996.

[2] R. Babilon, H. Nyklová, O. Pangrác, and J. Vondrák. Visibility representations of complete graphs. In J. Kratochvíl, editor, *Proc. Graph Drawing: 7th International Symp. (GD'99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 333–341. Springer, 1999.

[3] T. Biedl, J.R. Johansen, T. Shermer, and D.R. Wood. Orthogonal drawings with few layers. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proc. Graph Drawing: 9th International Symp. (GD'01)*, volume 2265 of *Lecture Notes in Computer Science*, pages 297–311. Springer, 2002.

[4] P. Bose, J Czyzowicz, P Morin, and D.R. Wood. The maximum number of edges in a three-dimensional grid-drawing. Technical Report TR-02-04, School of Computer Science, Carleton University, Ottawa, Canada, 2002.

[5] P. Bose, H. Everett, S.P. Fekete, M. Houle, A. Lubiw, H. Meijer, K. Romanik, G. Rote, T.C. Shermer, S. Whitesides, and C. Zelle. A visibility representation for graphs in three dimensions. *J. of Graph Algorithms and Applications*, 2(3):1–16, 1998.

[6] T. Calamoneri and A. Sterbini. Drawing 2-, 3- and 4-colorable graphs in $O(n^2)$ volume. In S. North, editor, *Proc. of the Symposium Graph Drawing (GD'96)*, volume 1190 of *Lecture Notes in Computer Science*, pages 53–62. Springer, 1996.

[7] R.F. Cohen, P. Eades, T. Lin, and F. Ruskey. Three-dimensional graph drawing. *Algorithmica*, 17(2):199–208, 1997.

[8] M. de Berg, M. van Kreveld, M. Overmans, and O. Scharzkopf. *Computational Geometry.* Springer, 2000.

[9] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing.* Prentice Hall, 1999.

[10] V. Dujmović, P. Morin, and D.R. Wood. Path-width and three-dimensional straight-line grid drawings of graphs. In M.T. Goodrich and S.G. Kobourov, editors, *Proc. Graph Drawing: 10th International Symp. (GD'02)*, volume 2528 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2002.

[11] V. Dujmović and D.R. Wood. Tree-partitions of k-trees with applications in graph layout. Technical Report TR-02-03, School of Computer Science, Carleton University, Ottawa, Canada, 2002.

[12] S. Even. *Graph Algorithms.* Computer Science Press, 1979.

[13] S. Fekete and H. Meijer. Rectangle and box visibility graphs in 3D. *International J. of Computational Geometry and Applications*, 9(1):1–27, 1999.

[14] S.P. Fekete, M.E. Houle, and S. Whitesides. New results on a visibility representation of graphs in 3D. In F.J. Brandenburg, editor, *Proc. of the Symposium on Graph Drawing (GD'95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2001.

[15] S. Felsner, G. Liotta, and S. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proc. Graph Drawing: 9th International Symp. (GD'01)*, volume 2265 of *Lecture Notes in Computer Science*, pages 328–342. Springer, 2002.

[16] M.R. Garey and D.S. Johnson. Crossing number is NP-complete. *SIAM J. of Algebraic and Discrete Methods*, 4:312–316, 1983.

[17] A. Garg, R. Tamassia, and P. Vocca. Drawing with colors. In J. Díaz and M.J. Serna, editors, *Proc. 4th Annual European Symposium on Algorithms*, volume 1136 of *Lecture Notes in Computer Science*, pages 12–26. Springer, 1996.

[18] F. Harary. *Graph Theory*. Addison-Wesley, 1971.

[19] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[20] R. Jayakumar, K. Thulasiraman, and M.N.S Swamy. $O(n)$ algorithms for graph planarization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8:257–267, 1989.

[21] G. Kant. An $O(n)$ maximal planarization algorithm based on PQ-trees. Technical Report RUU-CS-92-03, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 1992.

[22] G. Kant. A more compact visibility representation. Technical Report RUU-CS-93-26, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 1993.

[23] G. Kant and X. He. Two algorithms for finding rectangular duals of planar graphs. Technical Report RUU-CS-92-41, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 1992.

[24] B. Landgraf. 3D graph drawing. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*, pages 172–190. Springer, 2001.

[25] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In P. Rosenstiehl, editor, *Theory of Graphs, International Symposium (1966)*, pages 215–232. Gordon and Breach, 1967.

[26] A. Liebers. Planarizing graphs - a survey and annotated bibliography. *J. of Graph Algorithms and Applications*, 5(1):1–74, 2001.

[27] Jünger M. and P. Mutzel. Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica*, 16:33–59, 1996.

[28] E. Nardelli and M. Talamo. A fast algorithm for planarization of sparse diagrams. Technical Report R.105, IASI-CNR, Rome, 1984.

[29] J. Nummenmaa. Constructing compact rectilinear planar layouts using canonical representation of planar graphs. *Theoretical Computer Science*, 99:213–230, 1992.

[30] J. Pách, T. Thiele, and G. Tóth. Three-dimensional grid drawings of graphs. In G. Di Battista, editor, *Proc. Graph Drawing: 5th International Symp. (GD'97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 47–51. Springer, 1997.

[31] R.C. Read. A new method for drawing a planar graph given the cyclic order of the edges at each vertex. *Congressus Numerantium*, 56:31–44, 1987.

[32] A.L. Rosenberg. Three-dimensional VLSI: a case study. *J. ACM*, 30(3):397–416, 1983.

[33] P. Rosenstiehl and R.E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Computational Geometry*, 1(4):343–353, 1986.

[34] M.N.S. Swamy and K. Thulasiraman. *Graphs, Networks and Algorithms.* John Wiley and Sons, 1981.

[35] R. Tamassia and I. Tollis. A unified approach to visibility representations of planar graphs. *Discrete Computational Geometry*, 1(1):321–341, 1986.

[36] I. Vrt'o. Crossing numbers of graphs: A bibliography. Available electronically at ftp://ifi.savba.sk/pub/imrich/crobib.ps.gz.

[37] C. Ware and G. Franck. Viewing a graph in a virtual reality display is three times as good as a 2D diagram. In A.L. Ambler and T.D. Kimura, editors, *Proc. IEEE Symp. Visual Languages (VL'94)*, pages 182–183. IEEE, 1994.

[38] D.R. Wood. Queue layouts, tree-width, and three-dimensional graph drawing. In *Proc. of 22nd Foundations of Software Technology and Theoretical Computer Science (FSTTCS '02)*, Lecture Notes in Computer Science. Springer, to appear.