New Approach for Drawings of 3-Planar Graphs

Shimaa E. Waheed[1,3], Amal Ali M. Mady[1], Mohamed A. El-Sayed[2,4], S. Abdel-Khalek[1]

[1] Department of Math, Faculty of Science, Taif University, KSA;
[2] Department of CS, Computer and IT College, Taif University, KSA;
[3] Department of Math, Faculty of Science, Benha University, Egypt;
[4] Department of Math, Faculty of Science, Fayoum University, Egypt.

Abstract

The field of graph drawing is concerned with finding algorithms to draw graph in an aesthetically pleasant way, based upon a certain number of aesthetic criteria that define what a good drawing, (synonyms: diagrams, pictures, layouts), of a graph should be. This problem can be found in many such as in the computer networks, data networks, class interrelationship diagrams in object oriented databases and object oriented programs, visual programming interfaces, database design systems, software engineering…etc.

Given a plane graph *G*, we wish to find a drawing of *G* in the plane such that the vertices of *G* are represented as grid points, and the edges are represented as straight-line segments between their endpoints without any edge-intersection. Such drawings are called planar straight-line drawings of *G*. An additional objective is to minimize the area of the rectangular grid in which *G* is drawn. In this paper we introduce a new algorithms that finds an embedding of 3-planar graph in linear time $O(n)$.

Keywords: *3- Planar Graph; Graph Drawing; drawing on grid.*

1. Introduction

Since Euler's Königsberg bridge problem dating back to 1736, planar graphs have provided interesting problems in theory and in practice. The drawing of graphs is widely recognized as a very important task in diverse fields of research and development. Examples include VLSI design, plant layout, software engineering and bioinformatics. Using the elaborate techniques of a canonical ordering and Schnyder realizers, every planar graph can be drawn on a grid of quadratic size, and such drawings can be computed in linear time [1, 2]. Large and complex graphs are natural ways of describing real world systems that involve interactions between objects: persons and/or organizations in social networks, articles incitation networks, web sites on the World Wide Web, proteins in regulatory networks, etc [3, 4].

Graphs that can be drawn without edge crossings (i.e. planar graphs) have a natural advantage for visualization. When we want to draw a graph to make the information contained in its structure easily accessible, it is highly desirable to have a drawing with as few edge crossings as possible[1, 5].

A straight-line embedding of a plane graph *G* is a plane embedding of *G* in which edges are represented by straight-line segments joining their vertices, these straight line segments intersect only at a common vertex.

A straight-line drawing is called a convex drawing if every facial cycle is drawn as a convex polygon. Note that not all planar graphs admit a convex drawing. A straight-line drawing is called an inner-convex drawing if every inner facial cycle is drawn as a convex polygon [6].

A strictly convex drawing of a planar graph is a drawing with straight edges in which all faces, including the outer face, are strictly convex polygons, i. e., polygons whose interior angles are less than 180 [7,8]. However, a problem with graph layout methods which are capable of producing satisfactory results for a wide range of graphs is that they often put an extremely high demand on computational resources [9]. Visualizing graphs using virtual physical models is probably the most heavily used technique for drawing graphs in practice. There are many techniques to produce length-sensitive drawings for large graphs by reformulating the energy function [10,11,12].

One of the most popular drawing conventions is the straight-line drawing, where all the edges of a graph are drawn as straight-line segments. Every planar graph is known to have a planar straight-line drawing [13]. A straight-line drawing is called a convex drawing if every facial cycle is drawn as a convex polygon. Note that not all planar graphs admit a convex drawing. Tutte [14] gave a necessary and sufficient condition for a triconnected plane graph to admit a convex drawing. Thomassen [15] also gave a necessary and sufficient condition for a biconnected plane graph to admit a convex drawing. Based on Thomassen's result, Chiba et al. [16] presented a linear time algorithm for finding a convex drawing (if any) for a biconnected plane graph with a specified convex boundary. Tutte [14] also showed that every triconnected plane graph with a given boundary drawn as a convex polygon admits a convex drawing using the polygonal boundary. That is, when the vertices on the boundary are placed on a convex polygon, inner vertices can be placed on suitable positions so that each inner facial cycle forms a convex polygon. The canonical decomposition is a

generalization of the canonical ordering of De Fraysseix et al. [17].

We also presented a linear time algorithm for computing an inner-convex drawing of a triconnected plane graph with a star-shaped boundary [13].

Rosenstiehl and Tarjan [18] posed the question of whether it is always possible to find such an embedding into a polynomial-size grid. Later, de Fraysseix, Pach and Pollack [19] indeed gave a method that embeds an n-vertex planar graph into the $(2n\text{-}4)\times(n\text{-}2)$ grid in an $O(n \log n)$ time. Kant [20] developed a method for constructing convex grid drawing of 3-connected plane graphs in linear-time. His algorithm, related to those of Refs. [21] and [4a], uses a $(2n\text{-}4)\times(n\text{-}2)$ grid, and the grid size was improved to $(n\text{-}2)\times(n\text{-}2)$ by Schnyder and Totter [22] and Chrobak and Kant [20], independently. All these algorithms can be implemented in linear time.

In this paper, we will describe a new technique for graph layout that attempts to satisfy edge length constraints. This technique uses a modified Kant approach of convex drawing. In this paper we will show how to construct convex drawings of 3-connected plane graphs into a smaller, $(n\text{-}3)\times(n\text{-}3)$, grid in linear time. In addition, The paper present a different techniques for orthogonal drawing of 3- planar graph aiming to improve them to get the optimal upper and lower area bounds.

The remainder of the paper is organized as follows. In section 2, we give some definitions in graph drawing, specially, the canonical decomposition of plane graph . In sections 3, we introduce an algorithm that finds an embedding of G into a grid, $(n\text{-}2)\times(n\text{-}2)$. In sections 4, We will show how to modify the previous algorithm in order to reduce the grid size to $(n\text{-}3)\times(n\text{-}3)$. Section 5 present a new algorithm of 3-planar graph in orthogonal drawing. In section 6,we improve the grid size of orthogonal drawing into a smaller grid in linear time.

2. The Canonical Decomposition

In this section we present the concept of canonical decomposition for triconnected planar graphs. The canonical decomposition is a generalization of the canonical ordering of De Fraysseix et al. [23]. Define a plane graph G to be *internally 3-connected* if (a) G is 2-connected, and (b) if removing two vertices u,v disconnects G then u, v belong to the outer face and each connected component of $G\text{-}\{u, v\}$ has a vertex of the outer face. In other words, G is internally 3-connected iff it can be extended to a 3-connected graph by adding a vertex and connecting it to all vertices on the outer face. Let G be an n-vertex 3-connected plane graph with an edge $e(v_1,v_2)$ on the outer face.

Let $\pi=(V_1,...,V_m)$ be an ordered partition of V, that is , $V_1\cup...\cup V_m = V$ and for $V_i \cap V_j \neq \Phi$ for $i\neq j$. Define G_k to be the subgraph of G induced by $V_1\cup...\cup V_k$, and denote by C_k the external face of G_k. We say that π is a *canonical decomposition* of G with bottom edge $e(v_1,v_2)$ if:

(CD1) V_m is a singleton, $\{z_0\}$, where z_0 lies on the outer face and $z_0 \notin \{v_1,v_2\}$.

(CD2) C_1 is a face of G, and each C_k is a cycle containing $e(v_1,v_2)$.

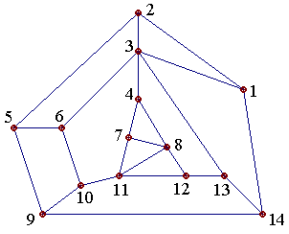
(CD3) Each G_k is 2-connected and internally 3-connected.

(CD4) For each $2\leq k \leq m\text{-}1$, one of the two following conditions holds:

(i) V_k is a singleton, $\{z\}$, where z belongs to C_k and has at least one neighbor in $G\text{-}G_k$.

(ii) V_k is a chain, $(z_1, z_2,..., z_t)$, where each z_i has at least one neighbor in $G\text{-}G_k$, and where z_1 and z_t each have one neighbor on G_{k-1} and these are the only two neighbors of V_k in G_{k-1}.

By an ordered plane graph (G,π) we will understand a plane graph G with a given canonical decomposition $\pi=V_1,...,V_m$. By the *contour* of G_k we will mean its outer face written as C_k.We will commonly view C_k as a path $(w_1,w_2,...,w_j)$ starting with $w_1=v_1$ and ending with $w_j=v_2$, ignoring the edge $e(v_1,v_2)$. We will also view C_k as being ordered from "left" to "right", where w_1 is the leftmost and w_j is the rightmost vertex on C_k. Let w_p be the leftmost and w_q be the rightmost neighbors of v in C_k, we will say that the vertex v *covers* the vertices $w_{p+1},..., w_{q-1}$. Throughout the rest of the paper we will call a plane graph *internally convex* if all its internal faces are convex.



k	V_k	C_k
1	9-14	9-14
2	8	9,10,11,8,12,13,14
3	7	9,10,11,7,8,12,13,14
4	5,6	9,5,6,10,11,7,8,12,13,14
5	4	9,5,6,10,11,7,4,8,12,13,14
6	3	9,5,6,3,13,14
7	2	9,5,2,3,13,14
8	1	9,5,2,1,14

Figure 1: The canonical decomposition with bottom edge $e(9,14)$

We will use the following lemma proved by Kant in [21], and our explanation is similar to the one given by Chrobak and Kant [20]:

Lemma 1: Each 3-connected plane graph has a canonical decomposition.

As it was shown by Kant [21] (Theorem 2.3) a canonical decomposition can be constructed in linear time. In Figure 1 an example (which is given in [20]) of a canonical decomposition of a triconnected planar graph given, with bottom edge $e(9,14)$.

By $P(v)$ we will denote the current position of vertex v in the grid, i.e., $P(v):=(x(v),y(v))$. By $P(u,v)$ we denote the embedding of edge $e(u,v)$, that is, the line segment that connects $P(u)$ with $P(v)$. To each vertex w we assign a set of vertices, $U(w)$, that will contain certain vertices that are located below w and have to be shifted right whenever w is shifted right.

We will describe first an algorithm that uses the $(n-2)\times(n-2)$ grid, $n\geq 3$, and then show how to improve it to $(n-3)\times(n-3)$, $n>3$. The algorithm is enhanced to $(f-1)\times(f-1)$ grid.

3. ConvexDraw Algorithm

The algorithm will be to add sets V_k, one by one, in forward order V_1, ,..., V_m, adjusting the embedding at every step. For z_i, $i=1,2,\ldots,t$, $P(z_i)$:=($x(z_i)$, $y(z_i)$), since $x(z_i)$ and $y(z_i)$ are integers so $P(z_i)$ is always a grid point.

Let (G,π) be a given ordered plane graph with n vertices, where $\pi= V_1,\ldots, V_m$ and $n\geq 3$. Suppose that $2\leq k\leq m$ and that we are about to add V_k to G_{k-1}.

Algorithm ConvexDraw

Input: A convex graph G with β vertices and m contours.

Output: Outline graph embedded in $(\beta-2)\times(\beta-2)$ grid.

Begin

We initialize the embedding by drawing $C_1 =(v_1=z_1, z_2,\ldots, z_t= v_2)$, as follows :

- $P(z_1):=(0,0)$;
- $P(z_t):=(t-2,0)$;
- $P(z_i):=(i-2,1)$, for all $i=2,\ldots,t-1$;
- $U(z_i)=\{z_i\}$, $i=1,2,3,\ldots,t$.

Then, for each $k= 2,\ldots,m$, we do the following:

- Let $C_{k-1}= (v_1=w_1,w_2,\ldots,w_j= v_2)$ be the contour of G_{k-1} .
- Let $V_k =(z_1, z_2,\ldots, z_t)$, and V_k may be a singleton or a chain.
- Let w_p be the leftmost and w_q be the rightmost neighbors of V_k in C_{k-1}.

We now execute the following steps:

Step 1: (Shift operation) for each vertex v is belong to $\{U(w_i), i=p+1,\ldots,j\}$ do

$$x(v) = x(v) + t; \quad (1)$$

Step 2: (Install operation) For each $i=1,\ldots,t$, let $P(z_i)$ be defied by :

$$x(z_i) = x(w_p)+i-1, \quad (2)$$

$$y(z_i) = y(w_q) + x(w_q) - x(w_p) - t + 1; \quad (3)$$

Step 3: (Update operation) :

$U(z_1)=\{z_1\}\cup\{U(w_i), i=p+1,\ldots,q-1\}$ and $U(z_i)=\{z_i\}$, $i=2,3,\ldots,t$.

End

In the other words, in step 2, we draw the V_k horizontally, in such a way that the slope of the segment $P(z_t,w_q)$ is -45^o. Vertex z_1 is placed above w_p, that the slope of the segment $P(w_p,z_1)$ is 90^o. Note that by moving some of the points $P(w_i)$ in step 1, we ensure that all neighbors of V_k will be visible from $P(z_i)$ for $i=1,2,\ldots,t$.

Lemma 2: *Let* $1\leq k\leq m$, *and* $C_k=(w_1=v_1,w_2,\ldots,w_j=v_2)$ *and* β *is the number of vertices of* G_k . *Then* $P(v_1)=(0,0)$, $P(v_2)=(\beta-2,0)$, *and all contour segments* $e(w_i,w_{i+1})$, $i=1,2,\ldots,j-1$, *have slopes in* $\{-45^o,0^o,90^o\}$.

Proof: the proof is by induction on k. For G_1 the lemma is obvious, the segment $e(w_1,w_2)$ has slope of 90^o, the segments $e(w_i,w_{i+1})$, $i=2,3,\ldots,j-2$ have slope of 0^o, and the segment $e(w_{j-1},w_j)$ has slope of -45^o, and $P(v_2)=(j-2,0)$.

So suppose that it holds for G_{k-1}. As in the algorithm, before installing V_k, the contour $C_{k-1}=(w_1=v_1,w_2,\ldots,w_j=v_2)$, $P(v_1)=(0,0)$.and $P(v_2)=(\alpha-2,0)$ where α is the number of vertices in G_{k-1} . Let $w_p,w_{p+1},\ldots, w_q$ be the neighbors of V_k in C_{k-1}.

When we are going to install V_k, we always have $w_j=v_2$, $x(w_j)= \alpha-2$ and from (1), by moving all the vertices $w_{p+1},\ldots,w_j$ by t to the right, $x(w_j)= \alpha-2+t$, but $\alpha+t$ equal to the number of vertices in G_k , hence $P(v_2)=(\beta-2,0)$. Let $w_p,w_{p+1},\ldots, w_q$ be the neighbors of V_k in C_{k-1}.After installing V_k we can divide the segments of the contour C_k into three intervals, the first interval is $\{e(w_i,w_{i+1}), i:=1,2,..,p-1\}$, the second interval is $\{e(w_p,z_1),e(z_1,z_2), \ldots,e(z_{t-1},z_t), e(z_t,w_q)\}$ and the third interval is $\{e(w_i,w_{i+1}), i:=q,..,j-1\}$.

In the first interval, if it contains any line-segment, the slope will be the same as its slope at the contour C_{k-1}. But for the second interval, the line-segment $e(w_p,z_1)$, has

slope $[y(z_1)-y(w_p)]/[x(z_1)-x(w_p)]$, from (2) the denominator $x(z_1)-x(w_p)=0$, from (3) the numerator $y(z_1)-y(w_p)$ greater than zero and less than infinity , so the line-segment $e(w_p,z_1)$ has the slope equal to 90°. The line-segments $e(z_i,z_{i+1})$, i=1,2,…,t-1, have the slope equal to 0°, because $y(z_{i+1})-y(z_i)$ equal to zero from (3). But for the line-segments $e(z_t,w_q)$, $y(w_q)-y(z_t)= -\{x(w_q)-x(z_t)\}$, i.e., the line-segments $e(z_t,w_q)$ has the slope equal to -45°. For the third interval, the line-segments has the same slopes as C_{k-1} because the only change that we have shifted vertices $w_q,...,w_j$ to the right by t and this will not effect the slopes of the line-segments from C_{k-1} to C_k. Hence, the contour segments $e(w_i,w_{i+1})$, i=1,2,...,j-1 of G_k have slopes in {-45°,0°,90°}.□

The lemma above implies immediately that adding V_k does not destroy the embedding, as stated in the corollary below.

Corollary 1: *For each* $1\le k\le m$, *when we add* V_k, *then after applying the shift operation*, *all neighbors of* V_k *are visible*, *that the edges between* V_k *and* C_{k-1} *do not intersect themselves or edges in* C_{k-1}.

What remains to show is that do destroy the planarity property and convexity when we apply the shift operation. This is proven in the next lemma.

Lemma 3: *Let* G_k *be straight-line embedded and internally convex. Additionally, it has the following property: Suppose* $C_k=(v_1=w_1, w_2,...,w_j=v_2)$, *and any integer t. if we shift all nodes in* $\{U(w_i), i=p+1, ...,j\}$ *by t to the right, then* G_k *remains straight-line embedded and internally convex.*

Proof: the proof is by induction on k. For G_1 the lemma is obvious, by inspection. Assume the lemma holds for G_{k-1}, we will show that the lemma properties are preserved when we add V_k. As in the algorithm, before installing V_k, the contour $C_{k-1}= (w_1=v_1,w_2,...,w_j=v_2)$ and w_p be the leftmost and w_q be the rightmost neighbors of V_k in C_{k-1}. When we are going to install V_k, from (1) by moving all the vertices $U(w_{p+1}),...,U(w_j)$ by t to the right, we have three classes of faces in G_{k-1}. First class, all vertices of the face are belong to $U(w_1),...,U(w_p)$, there is no any shift. Therefore, all faces of this type are not change, and its properties in G_k will be the same as in G_{k-1}. Second class, all vertices of the face are belong to $U(w_{p+1}),...,U(w_j)$, so, all vertices shifted by t to the right. Therefore, all faces of this type are moved by t to the right and its properties in G_k will be the same as in G_{k-1}. Third class, the vertices of a face classified two to sets, the first set are belong to $U(w_1),...,U(w_p)$, they not moved to the right, the second set are belong to $U(w_{p+1}),...,U(w_j)$, they moved to the right by t, in this case, any edge of the considerable face which has one vertex element in the first set and the second element lies in the second set will be stretched, and this will not affect its properties.

Let us assume now that V_k. is a singleton, $V_k=\{z_1\}$. Let z_1 have λ neighbors among $w_p,w_{p+1},...,w_q$ and let $F_1,F_2,...,F_{\lambda-1}$ be the faces created when adding V_k. From the algorithm all these faces preserved the lemma properties. The proof when V_k is a chain is very similar. □

4. Improving the Grid Size

Now we sketch how to modify the algorithm ConvexDraw in order to reduce the grid size to (n-3)×(n-3). First we pick $V_m=\{z_0\}$ to be the neighbor of v_2 different from v_1 on the outer face of G. We construct a canonical decomposition and run the algorithm ConvexDraw for m-1 steps. In the last step, having already embedded G_{m-1}, we set $P(z_0):=(1,n-3)$ and we do not shift any vertices to the right.

Algorithm MConvexDraw

Input: A convex graph G with β vertices and m contours.

Output: Outline graph embedded in (β-2)×(β-2) grid.

Begin

We initialize the embedding by drawing $C_1=(v_1=z_1, z_2,..., z_t=v_2)$, as follows :

- $P(z_1):=(0,0)$;
- $P(z_t):=(t-2,0)$;
- $P(z_i):=(i-2,1)$, for all i=2,…,t-1;
- $U(z_i)=\{z_i\}$, $i=1,2,3,...,t$.

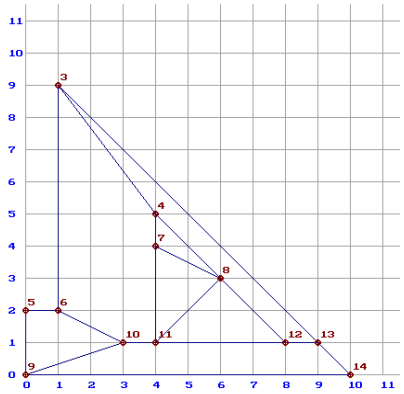
For each k= 2,…,m-1, we do the following:

- Let $C_{k-1}= (v_1=w_1,w_2,...,w_j=v_2)$ be the contour of G_{k-1} .
- Let $V_k=(z_1, z_2,..., z_t)$, and V_k may be a singleton or a chain.
- Let w_p be the leftmost and w_q be the rightmost neighbors of V_k in C_{k-1}.
- Calculate the shift operation.
- Install operation.
- Execute the update operation.

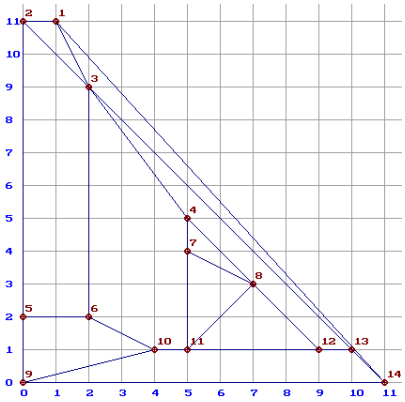
Finally , for $k=m$, we put $P(V_m=\{z_0\})=(1,n-3)$

End

Let us call this algorithm MConvexDraw. In order to show correctness, we only need to show that adding $z_0$will result in a correct, convex embedding. By lemma 2 and the algorithm, before adding $z_0$we have $x(w_1)=x(w_2)=\ldots=x(w_p)=0$, and $x(w_q)=n-3$, where $w_q=v_2$. The edge with slope -45° from v_2 contains the point $(1,n-4)$. This implies that all vertices $w_p,\ldots,w_q$ are visible from $(1,n-3)$. The convexity of the outer face follows from the choice of z_0. Consequently, we obtain the following theorems:



(a) G_{m-2}



(b) G_m

Figure 2: The drawing of the graph G

Table 1: The values of the different variables in ConvexDraw.

k	V_k	w_p	w_q	x-coordinates of vertices													
				1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	9-14	-	-									0	0	1	2	3	4
2	8	11	12								1	0	0	1	3	4	5
3	7	11	8							1	2	0	0	1	4	5	6
4	5,6	9	10					0	1	3	4	0	2	3	6	7	8
5	4	7	8				3	0	1	3	5	0	2	3	7	8	9
6	3	6	13			1	4	0	1	4	6	0	3	4	8	9	10
7	2	5	3				0	2	5	0	2	5	7	0	4	5	9	10	11
8	1	2	14	1			0	2	5	0	2	5	7	0	4	5	9	10	11
y-coordinates				11	11	9	5	2	2	4	3	0	1	1	1	1	0

Theorem 1: *Given a 3-connected plane graph G, algorithm MConvexDraw constructs a straight-line convex embedding of G into a* $(n-3) \times (n-3)$ *grid.*

Theorem 2: *Given a plane graph G, the above algorithm MConvexDraw computes a convex embedding of G into the* $(n-3) \times (n-3)$ *grid in* $O(n)$ *time*.

In Figure 2 an example of a drawing is given. After adding vertex 3, we have $U(w)=\{w\}$ for $w\in\{5,6,9,13,14\}$, $U(3)=\{3,4,7,8,10,11,12\}$. Therefore, when adding vertex 2, the vertices in $U(3)\cup U(6)\cup U(13)\cup U(14)=\{3,4,6,7,8,10,11,12,13,14\}$ will be shifted right. After adding vertex 2, we have $U(w)=\{w\}$ for $w\in\{5,9,13,14\}$, $U(3)=\{3,4,7,8,10,11,12\}$ and $U(2)=\{2,6\}$. Table 1 show the values of the different variables in ConvexDraw. Notice that the drawing is not *strictly convex*, i.e. there are angles of size 180°.

The 3-regular plane graphs are plane graphs where every vertex has exactly 3 neighbours. Especially in the mathematical literature 3-regular graphs are also called "cubic" graphs.

Lemma 4 *Let* (G,π) *given a 3-plane graph . Algorithm MConvexDraw constructs a straight-line convex embedding of G at most in* $(2f-7) \times (2f-7)$ *grid.*

Proof: Assume first that G is 3-plane graph. By Euler's formula, N is even, number of edges $M=3N/2$ and $f=N/2+2$. Let a canonical decomposition of G be given. Science $N=2f-4$, and from Theorem 1, the grid size is at most *in* $(2f-7) \times (2f-7)$. □

Theorem 3: *Let* (G,π) *given a 3-connected plane graph with M (the number of* V_i*). Algorithm MConvexDraw can be constructs a straight-line convex embedding of G at most in* $(f-1) \times (f-1)$ *grid.*

Proof: That is easy ,where *MConvexDraw* modified in some steps as the following:

- Let $C_{k-1}=(w_1,w_2,\ldots,w_j)$ be the contour of G_{k-1} .
- Let $V_k=(z_1, z_2,\ldots, z_t)$. V_k may be a singleton or a chain .
- Let w_p be the leftmost and w_q be the rightmost neighbors of V_k in C_{k-1} .

$\Delta x = x(w_q) - x(w_P)$; $\Delta y = y(w_p) - y(w_q)$

Step 1 : Shift operation :

If $\Delta y \le 0$ then

If $t > \Delta x$ then *shift*(w_{p+1}) by $t-\Delta x$

Else *no shift*

Else

If $t > \Delta x - \Delta y$ then *shift*(w_q) by $t - \Delta x + \Delta y$

Else *no shift*

Step 2: Install operation:

For each i=1,…,t, let $P(z_i)$ be defied by :

$x(z_i) := x(w_p) + i - 1$,

$y(z_i) := y(w_q) + x(w_q) - x(w_p) - t + 1$;

Step 3: Update operation;

$$Under(z_1) := \{z_1\} \cup \bigcup_{i=p+1}^{q-1} Under(w_i).$$

$$Under(z_i) := \{z_i\}, \; i = 2,3,...,t.$$

5. Linear-Time Algorithm

The linear-time implementation is achieved by representing the sets *Under*(v) using a binary tree T. When we embed V_k, it is not really necessary to know exact positions of w_p and w_q. If we know only their y-coordinates and their relative x-coordinates (that is, $x(w_q)$-$x(w_p)$), then we can compute $y(v)$, $v \in \{z_1, z_2, \ldots, z_t\}$, and the x-coordinate of v relative to w_p, that is $x(v)$-$x(w_p)$. For each vertex v, the *x-offset* of v is defined as $\Delta x(v) = x(v) - x(w)$, where w is the T-father of v. More generally, if w is an ancestor of v, the x-offset between w and v is $\Delta x(w,v) = x(v) - x(w)$.

By $T(v)$ we denote the subtree of T rooted at v. With each vertex v we store the following information :

Left(v)= the left T-child of v,
Right(v)= the right T-child of v,
$\Delta x(v)$= the x-offset of v from its T-father,
$\Delta y = y(w_p) - y(w_q)$
$x(v)$= the x-coordinate of v, and
$y(v)$= the y-coordinate of v.

The root of T is v_1. *Right*(v) is the next node in the contour. Thus the path $C_k = (w_1, w_2, ..., w_j)$ consists of: v_1, *Right*(v_1), *Right*(*Right*(v_1)),....., etc. If v is in the contour then *Left*(v) is the node u such that $T(u) = Under(v) - \{v\}$. *Under*(w_i) consists of w_i and its T-subtree rooted at *Left*(w_i). Thus we have the relationship: $T(w_i) := \bigcup_{a=i}^{j} Under(w_a)$.

The algorithm consists of two phases. In the first phase we add new vertices one by one, and each time we add a vertex we compute its x-offset and y-coordinate, and update the x-offsets of one or two other vertices. In terms of our T, when we add V_k, we need to shift $T(w_{p+1})$ to the right. The crucial observation that leads to the linear-time algorithm is that it is not really necessary to know the exact positions of w_p and w_q at the time when we install $V_k = (z_1, z_2, \ldots, z_t)$. If we only know their y-coordinates and offset $\Delta x(w_p, w_q)$ then for each i>1 we can compute $y(z_i)$ and the x-offset of z_i relative to z_{i-1}, the x-offset of z_1 relative to w_p, and the x-offset of w_q relative to z_t. In the second phase, we traverse the tree and compute final x-coordinates by accumulating offsets.

The first phase is implemented as follows:

First, we use the initial values of the vertices $z_1, z_2, \ldots, z_t$ of V_1

Initialize:

$\Delta x(z_1)$, $y(z_1)$, *Right*(z_1), *Left*(z_1):= 0, 0, z_2, *Nil;*

FOR i:=2 TO t-1 DO BEGIN $\Delta x(z_i)$, $y(z_i)$, *Right*(z_i), *Left*(z_i):=1, 1, z_{i+1}, *Nil;* END;

$\Delta x(z_t)$, $y(z_t)$, *Right*(z_t), *Left*(z_t):= 1, 0, *Nil*, *Nil;*

Then, we embed other vertices, one by one:

FOR k:= 2 TO M-1 DO
BEGIN

Notation:

Let $C_{k-1} = (v_1 = w_1, w_2, ..., w_j = v_2)$ be the contour of G_{k-1}.

Let $V_k = (z_1, z_2, ..., z_t)$.

Let w_p be the leftmost, and w_q be the rightmost neighbors of V_k in G_{k-1}.

Stretch gaps:

$\Delta x(w_p, w_q) := \Delta x(w_{p+1}) + \ldots + \Delta x(w_q)$;
$\Delta y = y(w_p) - y(w_q)$;
IF $\Delta y <= 0$ THEN
IF $(t - \Delta x(w_p, w_q) > 0)$ THEN
$x(w_{p+1}) = \Delta x(w_{p+1}) + t - \Delta x(w_p, w_q)$;
ELSE
IF $(t - \Delta x(w_p, w_q) + \Delta y > 0)$ THEN
$x(w_q) = \Delta x(w_q) + t - \Delta x(w_p, w_q) + \Delta y$;

Adjust offsets:

$\Delta x(z_1) := 0$;
$\Delta x(w_p, w_q) := \Delta x(w_{p+1}) + \ldots + \Delta x(w_q)$;
$y(z_1) := y(w_q) + \Delta x(w_p, w_q) - t + 1$;
FOR i:=2 TO t DO BEGIN
$\Delta x(z_i) := 1$, $y(z_i) := y(z_1)$;
END;
$\Delta x(w_q) := \Delta x(w_p, w_q) - (t-1)$;

Install v_k:

Right(w_p):= z_1;
FOR i:=2 TO t DO *Right*(z_{i-1}):= z_i;
Right(z_t):= w_q;
IF $p+1 \neq q$ THEN
IF $\Delta y <= 0$ THEN

BEGIN
Left(w_q):=w_{p+1} ;
Right(w_{q-1}):=*Nil;*
$\Delta x(W_{p+1}):= \Delta x(W_{p+1})-\Delta x\ (w_p,w_q)$;
END
ELSE
BEGIN
Left(w_p):=w_{p+1} ; *Right*(w_{q-1}):=*Nil;*
END
ELSE *Left*(z_1):= *Nil;*
END {FOR}

Finally, we put the vertex z_0 of V_M at the position $x(z_0):= 1$, $y(z_0):= x(w_j)$.

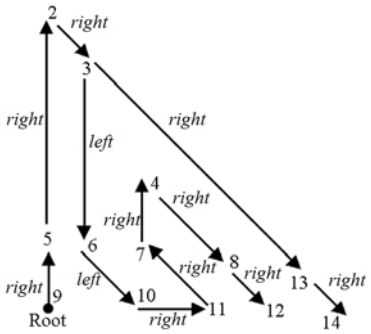


Table 2: The values of the different variables in ConvexDraw.

k	V_k	$\Delta x(v)$													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	9-14									0	1	1	1	1	1
2	8								0	0	1	1	1	1	1
3	7							0	1	0	1	1	1	1	1
4	5,6					0	1	0	1	0	1	1	1	1	1
5	4				0	0	1	0	2	0	1	1	1	1	1
6	3			0	0	0	1	0	2	0	1	1	1	6	1
7	2		0	1	0	0	0	0	2	0	1	1	1	6	1

Figure 3: The tree *T* and $\Delta x(v)$.

At this point all y-coordinates and x-offsets have already been computed. In the second phase, we compute x-coordinates, invoking *AccumalateOffests*, beginning by the root v_1 of *T* and the zero value of its *x*-offset, i.e. we invoke *AccumalateOffests*(v_1,0) , where *AccumalateOffests* is as follows:

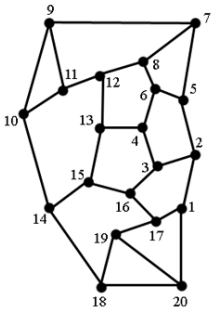
PROCEDURE *AccumalateOffests*(*v*: vertex, δ:integer)
BEGIN
IF *v*≠*Nil* THEN
BEGIN
$x(v):= \Delta x(v)+\delta$;
AccumalateOffests(*Left*(*v*), *x*(*v*));
AccumalateOffests(*Right*(*v*), *x*(*v*))
END;
END.

In figure 3 the construction of the tree and the values of $\Delta x(v)$ are given for the example from Figure 1.

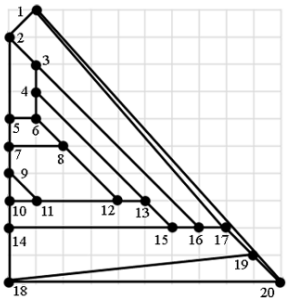
Correctness. In order to prove correctness, since the x-coordinate of a vertex *v* equal to the sum of the offsets on the path from the root v_1 to *v*, it is sufficient to show that all offsets $\Delta x(v)$ are computed correctly. To see that the *Stretch* step works correctly, recall that the *T*-sub tree rooted at w_i consists of $\bigcup_{a=i}^{j} Under(w_a)$. So, incrementing the offset of w_i increments the cumulative offset from v_1 of each member of that *T*-subtree, i.e., shifts them all to the right. During the *adjustment* step of V_k, only the offsets of w_q get changed. But w_p remains an ancestor of each, and their offsets from w_p remain unchanged, by simple algebra. It follows that the cumulative offsets of all vertices already in the graph remain unchanged by the adjustment step.

Complexity. The linear time complexity is achieved by appropriately distributing information in the vertices of the graph, we have already mentioned that the canonical decomposition can be found in time $O(n)$. In the first phase, when we add V_k =(z_1, z_2,…, z_t), the cost is proportional to *q*-*p*+*t*, where w_p and w_q denote, as usual, the leftmost and rightmost neighbors of V_k in C_{k-1} . So the time complexity of the first phase is $O(n)$. Obviously, the second phase runs in linear time.

Theorem 4: *Given a 3-Connected planar graph G, the above Linear-time Algorithm computes a convex embedding of G into the* (*f*-1) × (*f*-1) *grid in O*(*n*) *time.*



(a) 3-planar graph *G* with 20 vertices



(b) The drawing of *G* with area 10×10

Figure 4: Given 3-planar graph and its convex embedding

Lemma 5 *Let* (*G*,π) *given a 3-plane graph. There is an algorithm constructs a straight-line convex embedding of G at most in* $\lfloor n/2 \rfloor+1 \times \lfloor n/2 \rfloor+1$ *grid.*

Proof: Assume first that *G* is tri-connected. By Euler's formula, *n* is even, number of edges *m*=3*n*/2 and *f*=*n*/2+2. Let a canonical decomposition of *G* be given. Science

$M<f$, and from lemma 4, the grid size is at most f-1= $\lfloor n/2 \rfloor+1$. □

Theorem 5: *There is a O(n) time algorithm to draw a 3-planar graph with convex embedding of G into the* $\lfloor n/2 \rfloor+1 \times \lfloor n/2 \rfloor+1$ *grid*.

6. Conclusion

We showed that 3-planar graphs can be embedded on O(n) × O(n) integer grid, so that edges are drawn as straight-line segments in linear time algorithm. The results produced are good and the algorithm is scalable to large graphs. In addition, The paper present a different techniques for convex drawing of 3- planar graph aiming to improve them to get the optimal upper and lower area bounds. The algorithm improves the lower and upper bounds of $\lfloor n/2 \rfloor+1 \times \lfloor n/2 \rfloor+1$ grid area.

References

[1] Brandenburg, F.J., Eppstein, D., Gleißner, A., Goodrich, M.T., Hanauer, K., Reislhuber, J.: On the density of maximal 1-planar graphs. In: Didimo, W., Patrignani, M. (eds.) Graph Drawing. LNCS, vol. 7704, pp. 327–338. Springer 2013.

[2] Mohamed A. El-Sayed, "GA For Straight-Line Grid Drawings Of Maximal Planar Graphs", Egyptian Informatics Journal Production and Hosting by Elsevier, www.elsevier.com/locate/eij. Vol. 13, No. 1, pp. 9–17, 2012.

[3] Didimo,W.: Density of straight-line 1-planar graph drawings. Information Processing Letter, 113(7), 236–240 , 2013

[4] Fabrice Rossi and Nathalie Villa-Vialaneix" Optimizing an organized modularity measure for topographic graph clustering: A deterministic annealing approach" , Preprint submitted to Neurocomputing October 26, 2009.

[5] Bannister, Michael J.; Cheng, Zhanpeng; Devanny, William E.; Eppstein, David (2013), "Superpatterns and universal point sets", Proc. 21st International Symposium on Graph Drawing (GD 2013), arXiv:1308.0403

[6] Mohamed A. El-Sayed, S. Abdel-Khalek , and Hanan H. Amin "Study of Neural Network Algorithm for Straight-Line Drawings of Planar Graphs", International Journal of Computer Science and Information Security (IJCSIS) ISSN 1947-5500, Vol. 9, No. 9, pp. 13-19, 2011.

[7] Ahmed A. A. Radwan, Mohamed A. El-Sayed, Linear-time Algorithm for Convex Grid Drawings of 3-connected Planar Graph. International Journal of Applied Mathematics, Vol. 2. (11), 1335-1348, 2000.

[8] Imre Bárány and Günter Rote, "Strictly Convex Drawings of Planar Graphs", Documenta Mathematica 11, pp. 369–391, 2006.

[9] Bernd Meyer "Competitive Learning of Network Diagram Layout", Proc. Graph Drawing '98, Montreal, Canada, Springer Verlag LNCS 1547.S. pp. 246–262.

[10] Emden R. Gansner, Yifan Hu, and Shankar Krishnan , COAST: A Convex Optimization Approach to Stress-Based Embedding, Graph Drawing: Lecture Notes in Computer Science Vol 8242, pp 268-279, 2013.

[11] Gansner, E.R., Hu, Y., North, S.C.: A maxent-stress model for graph layout. IEEE Trans. Vis. Comput. Graph. 19(6), 927–940 ,2013.

[12] Khoury, M., Hu, Y., Krishnan, S., Scheidegger, C.: Drawing large graphs by low-rank stress majorization. Computer Graphics Forum 31(3), 975–984 , 2012.

[13] I. Fary, "On straight line representations of planar graphs", Acta Sci. Math. Szeged, 11, pp. 229-233, 1948.

[14] W. T. Tutte, Convex representations of graphs, Proc. of London Math. Soc., 10, no. 3, pp. 304-320, 1960.

[15] C. Thomassen, Plane representations of graphs, in Progress in Graph Theory, J. A. Bondy and U. S. R. Murty (Eds.), Academic Press, pp. 43-69, 1984.

[16] N. Chiba, T. Yamanouchi and T. Nishizeki, Linear algorithms for convex drawings of planar graphs, Progress in Graph Theory, Academic Press, pp. 153-173, 1984.

[17] H. de Fraysseix, J. Pach and R. Pollack, How to draw a planar graph on a grid, Combinatorica, 10, 41-51, 1990.

[18] P. Rosenstiehl and R.E. Tarjan, Rectilinear planer layouts and bipolar orientations of planar graphs, Discrete Comput. Geom. 1, 343-353,1986.

[19] H. de Fraysseix, J. Pach and R. Pollack, Small sets supporting Straight-Line Embeddings of planar graphs, in: proc. 20th Ann. Symp. on Theory of computing 426-433,1988.

[20] M. Chrobak and G. Kant, Convex grid drawings of 3-connected planar graphs, International Journal of Computational Geometry Vol.7. 211-233, 1997.

[21] G. Kant, Drawing Planar Graphs using the lmc-ordering ,in proc. 33rd symp. On Foundations of Computer Science, pp. 101-110, 1992.

[22] W. Schynder, W. Trotter, Convex drawings of planar graphs , Abstract Amer. Math. Soc. 13, 5, 1992.