Compact Encodings of Planar Orthogonal Drawings*

Amrita Chanda and Ashim Garg

Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260
{achanda,agarg}@cse.buffalo.edu

Abstract. We present time-efficient algorithms for encoding (and decoding) planar orthogonal drawings of degree-4 and degree-3 biconnected and triconnected planar graphs using small number of bits. We also present time-efficient algorithms for encoding (and decoding) turn-monotone planar orthogonal drawings.

1 Introduction

It is important to compress the representation of planar orthogonal drawings to reduce their storage requirements and transmission times over a network, like Internet. The encoding problem is also interesting from a theoretical viewpoint.

A *degree-d* graph is one, where each vertex has at most d edges incident on it. A *planar* drawing is one with no edge-crossings. An *orthogonal* drawing is one, where each edge is drawn as an alternating sequence of horizontal and vertical line-segments. A *bend* (*turn*) is defined as the meeting point of two consecutive line-segments of an edge in a drawing.

In this paper, we investigate the problem of encoding planar orthogonal drawings of degree-4 and degree-3 biconnected and triconnected planar graphs using small number of bits, and present several results.

Let d be a planar orthogonal drawing, with b turns (bends), of a planar graph G with n vertices, m edges, and f internal faces. Suppose each line-segment of d has length at most W. Through out the paper, to avoid triviality, we will assume that $n \geq 3$. A simple drawing-description format that stores the underlying graph using adjacency-list representation, and stores the coordinates of the vertices and edge-bends would require $\Omega(n + m \log_2 n + n \log_2 (b+n)W + b \log_2 (b+n)W)$ bits in the worst case. More complex formats may require even more bits. We are not aware of any work focusing explicitly on encoding planar orthogonal drawings.

In Sections 2, 3, and 4, respectively, we give our result, previous work, and some definitions. In Sections 5, 6, and 7, respectively, we show how to encode (and decode) degree-3 and degree-4 plane graphs, orthogonal representations, and lengths of the line-segments of a planar orthogonal drawing. In Section 8, we give our overall algorithm for encoding (and decoding) a planar orthogonal drawing.

* Research supported by NSF CAREER Award IIS-9985136 and NSF CISE Research Infrastructure Award No. 0101244.

M.T. Goodrich and S.G. Kobourov (Eds.): GD 2002, LNCS 2528, pp. 174–186, 2002.

2 Our Result

Our results can be summarized as follows (with G, n, m, f, d, W, and b as defined above): Let $p = (2 + b + m - f)\log_2(W + 1)$. Let $q = n + b$.

- If G is a degree-4 biconnected graph, then we can encode (and decode) d using at most $4.74m+2.42n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $5.01m + 2.33n + 1.67b + p + O(\log q)$ bits in $O(n+b+p)$ time.
- If G is a degree-4 triconnected graph, then we can encode (and decode) d using at most $3.58m+2.59n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $3.67m + 2.67n + 1.67b + p + O(\log q)$ bits in $O(n+b+p)$ time.
- If G is a degree-3 biconnected graph, then we can encode (and decode) d using at most $4.74m+1.23n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $5.01m + 1.33n + 1.67b + p + O(\log q)$ bits in $O(n+b+p)$ time.
- If G is a degree-3 triconnected graph, then we can encode (and decode) d using at most $3.58m + n + 1.58b + p + O(\log q)$ bits in $O((n + b + p)^2)$ time, and using at most $3.67m + n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.

Several drawing algorithms [7,8], that try to minimize the number of bends, produce *turn-monotone* planar orthogonal drawings, i.e., where for each edge $e = (u, v)$, if we travel from u to v along e, then we will either make left turns or right turns, but not both (see Figure 1(a)). Turn-monotone planar orthogonal drawings are very common in practice. We show that a turn-monotone planar orthogonal drawing d can be encoded even more succinctly:

- If G is a degree-4 biconnected graph, then we can encode (and decode) d using at most $3.16m+4n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $3.34m + 4n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.
- If G is a degree-4 triconnected graph, then we can encode (and decode) d using at most $2m+4.17n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $2m + 4.34n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.
- If G is a degree-3 biconnected graph, then we can encode (and decode) d using at most $3.16m+2.81n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $3.34m + 3n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.
- If G is a degree-3 triconnected graph, then we can encode (and decode) d using at most $2m+2.58n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time,, and using at most $2m + 2.67n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.

As a by-product, our technique also encodes orthogonal representations, which are important intermediate constructs used by several drawing algorithms [7,4].

3 Related Work

As mentioned above, we are not aware of any previous work focusing explicitly on encoding planar orthogonal drawings. However, a lot of work has been done on encoding planar graphs. Let G be a planar graph with n vertices and m edges. If G is biconnected (triconnected, triangulated, respectively), then it can be encoded using at most $2n + 1.58m$ bits [1] ($1.58(n+m)$ bits [1], $1.33m$ bits [1]). If G is a triangulated graph, then any encoding of G requires at least $1.08m$ bits [9]. [5] presents a technique for encoding G in asymptotically the minimum number of bits in $O(n \log n)$ time. For more results on graph encoding, see [1].

Our encoding technique is based on the graph encoding technique of [1], and on the concept of canonical orderings of planar graphs [2,6,1].

4 Preliminaries

We use standard definitions of graph-theoretic terms. A *plane* graph G is a planar graph equipped with an embedding. Two vertices of G are *neighbors* if they are connected by an edge. Let $u_1, u_2, \ldots, u_k$ be some vertices of G. The plane graph *induced* by $u_1, u_2, \ldots, u_k$ is the maximal subgraph of G that consists of these vertices and their incident edges. Suppose G has n vertices. An *ordering* $v_1, v_2, \ldots, v_n$ of the vertices of G is an assignment of unique integers in the range $[1, n]$ to the vertices, such that the i^{th} vertex v_i in the order is assigned number i.

Let G be a degree-4 plane graph. Two planar orthogonal drawings Γ_1 and Γ_2 of G are *shape equivalent* if: (1) for each vertex v, consecutive edges incident to v form the same angle at v in Γ_1 and Γ_2, and (2) for each edge (u, v), the sequence of left and right turns encountered while walking from u to v along the polygonal chain representing (u, v) is the same in Γ_1 and Γ_2. An *orthogonal representation* Γ of G defines an equivalence class of shape equivalent planar orthogonal drawings of G. Γ is a *turn monotone* representation if each edge is represented as a polygonal chain consisting of only left or right turns, but not both (see Figure 1(a)).

An important concept used by our encoding technique is that of a *canonical ordering* of a plane graph (see Figure 1(b)). This concept has been defined and used in [2,6,1]. Let $G = (V, E)$ be a simple biconnected plane graph with n vertices,

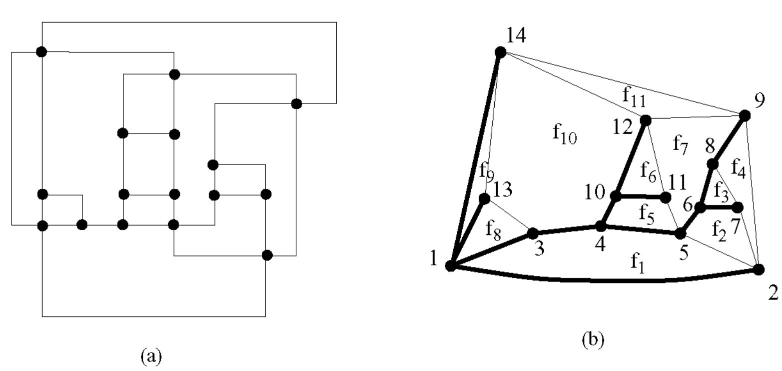

Fig. 1. (a) A turn-monotone planar orthogonal drawing. (b) corresponding plane graph G, a rightmost canonical ordering c of G and the canonical spanning tree T_c corresponding to c. Here, each vertex is labeled by its number in c, the edges of T_c are shown as dark lines. Each face is also labeled by its number in corresponding ordering of faces.

and m edges. Let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices of G. Let G_i be the plane graph induced by vertices $v_1, v_2, \ldots, v_i$. Let H_i be the external face of G_i.

Definition 1 ([1]). *Let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices of a biconnected plane graph $G = (V, E)$, where v_1 and v_2 are two arbitrary vertices on the external face of G with $(v_1, v_2) \in E$. The ordering is* canonical *if there exist ordered partitions $I_1, I_2, \ldots, I_K$ of the interval $[3, n]$ such that the following properties hold for every $1 \leq j \leq K$: Suppose $I_j = [k, k+q]$. Let C_j be the path $(v_k, v_{k+1}, \ldots, v_{k+q})$ (Note, C_j is a single vertex if $q = 0$). Suppose we say that a vertex u of G_{k-1} is a* neighbor *of C_j if a vertex of C_j is a neighbor of u. Then:*

- *The graph G_{k+q} is biconnected. Its external face H_{k+q} contains the edge (v_1, v_2), and the path C_j. C_j has no chord in G, i.e., G does not contain any edge (v_s, v_t) with $|s - t| > 1$ and $k \leq s, t \leq k + q$.*
- *C_j has at least two neighbors in G_{k-1}, all of which are vertices of H_{k-1}. The leftmost neighbor is a neighbor of v_k and the rightmost neighbor is a neighbor of v_{k+q}. Moreover, if $q > 0$, then v_l and v_r are the only neighbors of C_j in G_{k-1}. Here, the leftmost and rightmost neighbors of C_j in G_{k-1} are defined as follows: Vertices v_1 and v_2 divide H_{k-1} into two paths: a path consisting only of edge (v_1, v_2), and another path $v_1(= u_1)u_2 \ldots u_s(= v_2)$ that connect v_1 and v_2, and that does not contain the edge (v_1, v_2). We say that a vertex u_i is* left (right) *of another vertex u_z, if $i < z$ ($i > z$). The* leftmost (rightmost) *neighbor of C_j in G_{k-1} is the vertex u_b such that u_b is a neighbor of C_j, and there is no other vertex u_t such that u_t is a neighbor of C_j and u_t is left (right) of u_b.*

A *canonical* ordering for a triconnected plane graph is defined the same as for a biconnected plane graph, except that it has the following additional property:

Property 1 ([1]). Every vertex v_k, where $1 \leq k \leq n-1$ has at least one neighbor v_p with $k < p$.

A *rightmost canonical (rmc) ordering* for a biconnected plane graph G is defined as follows (see Figure 1(b)):

Definition 2 ([1]). *Let $v_1, v_2, \ldots, v_n$ be a canonical ordering for G, where $I_1, I_2, \ldots, I_K$ are its corresponding interval partitions. We say that $v_1, v_2, \ldots, v_n$ is a rightmost canonical (rmc) ordering for G if the following property holds for every interval I_j, where $1 \leq j \leq K$:*
Suppose $I_j = [k, k+q]$. Let $v_1, v_2, \ldots, v_{k-1}, u_k, u_{k+1}, \ldots, u_n$ be any canonical ordering for G whose first $j-1$ interval partitions are exactly $I_1, I_2, \ldots, I_{j-1}$ (Clearly, the G_{k-1} and H_{k-1} with respect to both canonical orderings are the same). Let v_l be the leftmost neighbor of v_k on H_{k-1}. Let u_l be the leftmost neighbor of u_k on H_{k-1}. Then, v_l is to the right of u_l on H_{k-1}.

Theorem 1. *[1] Every biconnected plane graph G with n vertices admits a rightmost canonical (rmc) ordering, which can be constructed in $O(n)$ time.*

Any canonical ordering $c = v_1, v_2, \ldots, v_n$ of a biconnected plane graph G and its corresponding interval partitions $I_1, I_2, \ldots, I_K$ defines a *canonical spanning tree* T_c that consists of the edge (v_1, v_2) plus the union of the paths $v_l v_k v_{k+1} ... v_{k+q}$ over all intervals $I_j = [v_k, v_{k+q}]$, where $1 \leq j \leq K$ and v_l is the leftmost neighbor of v_k on H_{k-1} (see Figure 1(b)). Suppose we root T_c at v_1. A *tree* edge of a vertex v is one that also belongs to T_c, and a *non-tree* edge is one that does not. The *incoming* tree edge of v is the edge that connects v to its parent in T_c, and an *outgoing* tree edge is one that connects v to a child in T_c. Suppose vertex v belongs to interval $I_j = [v_k, v_{k+q}]$. The *incoming* non-tree edges of v are those that connect v to neighbors in G_{k-1}, and the *outgoing* non-tree edges are the remaining non-tree edges. Note that by the definition of canonical ordering, each outgoing non-tree edge of v will be of the form (v, v_s), where $s > k + q$. Also, note that for each incoming non-tree edge (u, v) of v, u will be on H_{k-1}. For example, in Figure 1(b), Vertex 12 has incoming tree edge $(10, 12)$, no outgoing tree edge, incoming non-tree edges $(11, 12)$ and $(9, 12)$, and outgoing non-tree edge $(12, 14)$.

Properties 2 and 3 follow easily from the definition of canonical ordering:

Property 2. Let $v \neq v_1$ be a vertex of G. Then, v has exactly one incoming tree edge, and at least one outgoing tree-edge or incoming non-tree edge. Vertex v_1 has only outgoing tree edges.

Property 3. If G is a triconnected graph, then for every vertex $v \neq v_1, v_n$ of G, v has Property 2. Also, it has either at least one outgoing tree-edge, or at least one outgoing non-tree edge. Vertex v_1 has only outgoing tree edges. Vertex v_n has exactly one incoming tree edge and all its other edges are incoming non-tree edges.

Theorem 2. *Let G be a biconnected plane graph with n vertices. Suppose, we are given a rightmost canonical ordering $c = v_1, v_2, \ldots, v_n$ of the vertices of G, along with the number of outgoing tree edges, incoming non-tree edges, and outgoing non-tree edges of each vertex as defined by c. Then, we can determine in $O(n)$ time, all the edges in G, as well as its embedding. In other words, given this information, we can determine the entire structure of graph G.*

Sketch of Proof. Starting from the empty graph G_0, we reconstruct G iteratively by constructing a sequence of plane graphs $G_0, G_1, G_2, \ldots, G_p = G$, where in each iterative step i, we obtain G_i from G_{i-1} by inserting some vertices and edges into it. This is done as follows: For each vertex v_k, where $1 \leq k \leq n$, maintain two counters, $TCount(v_k)$ and $Ncount(v_k)$, that initially store the number of outgoing tree edges and outgoing non-tree edges, respectively, of v_k.

In the first iterative step, insert into G_0, vertices $v_1, v_2, \ldots, v_r$, where v_r is the lowest numbered vertex in c with an incoming non-tree edge. Also, insert the edges $(v_1, v_2), (v_1, v_3), (v_3, v_4), \ldots, (v_{r-1}, v_r), (v_r, v_2)$. Note that these edges will form a single simple cycle. Reduce $Ncount(v_2)$ by 1. For each j, where $1 \leq j \leq r - 1$ and $j \neq 2$, reduce $TCount(v_j)$ by 1.

In a general iterative step i, we obtain G_i from G_{i-1} as follows: Suppose G_{i-1} consists of vertices $v_1, v_2, \ldots, v_{k-1}$. Let $v_k, v_{k+1}, \ldots, v_{k+q}$ be the sequence of consecutive vertices in c such that vertices $v_k, v_{k+1}, \ldots, v_{k+q-1}$ do not have any incoming non-tree edge, but v_{k+q} does. Let H_{k-1} be the external face of G_{k-1}. Let u be the rightmost vertex on H_{k-1} with $TCount(u) > 0$. We have 2 cases:

- $q = 0$: Suppose v_k has t incoming non-tree edges. Let $u_1, u_2, \ldots, u_t$ be the first t vertices to the right of u on H_{k-1}, such that, for each u_i, $NCount(u_i) > 0$. Insert vertex v_k and the edges (u, v_k), $(v_k, u_1), (v_k, u_2), \ldots, (v_k, u_t)$ into G_{k-1} to obtain G_k. For each i, where $1 \le i \le t$, reduce $NCount(u_i)$ by 1 (Note that except for u_t, $NCount(u_i)$ for each u_i will become 0 now). Reduce $TCount(u)$ by 1.
- $q > 0$: In this case, v_{k+q} will have only one neighbor u' on H_{k-1}, where u' is the first vertex right of u with $NCount(u') > 0$. Insert vertices $v_k, v_{k+1}, \ldots,$ v_{k+q} and edges $(u, v_k), (v_k, v_{k+1}), (v_{k+1}, v_{k+2}), \ldots, (v_{k+q-1}, v_{k+q}), (v_{k+q}, u')$, into G_{k-1} to obtain G_k. Reduce $NCount(u')$ by 1. Reduce $TCount(u)$ by 1. Also, for each i, where $k \le i \le k + q - 1$, reduce $TCount(v_i)$ by 1.

5 Encoding Degree-3 And Degree-4 Plane Graphs

The algorithms of [1] will encode a biconnected (triconnected) degree-4 plane graph using $5.17n$ bits ($4.74n$ bits), and a degree-3 biconnected (triconnected) plane graph using $4.37n$ ($3.95n$) bits. But, these algorithms do not consider the degrees of vertices while encoding a graph. (The algorithm of [5] will construct an asymptotically bit-minimum encoding of these graphs, but it is practical only for very large graphs.) Here, we show that we can get a better encoding for degree-3 and degree-4 plane graphs by considering the degrees of their vertices.

The basic idea is simple. Suppose we construct a rightmost canonical ordering $c = v_1, v_2, \ldots, v_n$ of the vertices of a biconnected plane graph G. Then, to encode G, from Theorem 2, it is sufficient to encode, for each vertex, how many outgoing tree edges, incoming non-tree edges, and outgoing non-tree edges the vertex has.

Suppose G is a degree-3 graph. From Property 2, a vertex $v \neq v_1$ can only be of one of the 7 types, A to G, based on its degree, and the number and types of its edges. v is of one of types A to F, if it has degree 3, and of type F or G if it has degree 2. v is of (a) *Type A:* if it has two outgoing tree edges, (b) *Type B:* if it has one outgoing tree edge and one incoming non-tree edge, (c) *Type C:* if it has one outgoing tree edge and one outgoing non-tree edge, (d) *Type D:* if it has one incoming and one outgoing non-tree edge, (e) *Type E:* if it has two incoming non-tree edges, (f) *Type F:* if it has one incoming non-tree edge, and (g) *Type G:* if it has one outgoing tree edge. Note that vertex v_1 will have either two or three outgoing tree edges. Thus, we encode G by a string $S = s_1 s_2 \ldots s_n$, where

- s_1 represents the number of outgoing tree edges of v_1, and is equal to 0 if v_1 has two outgoing edges, and is equal to 1 if v_1 has three outgoing edges.
- Each symbol s_i, $2 \le i \le n$, represents the type of vertex v_i, and is equal to A, B, C, D, E, F, or G.

Since each s_i, where $2 \le i \le n$, can have 7 possible values, we can encode the substring $S' = s_2 \ldots s_n$ using $(n-1)\log_2 7 \approx 2.81(n-1)$ bits by converting the corresponding Base-7 number into binary representation in $O(n^2)$ time. Using Huffman encoding, we can encode S' using at most $3(n-1)$ bits in $O(n)$ time. This, combined with Theorems 1 and 2, gives us the following lemma:

Lemma 1. *Given a degree-3 biconnected plane graph G with n vertices, we can encode it using less than $2.81n$ bits in $O(n^2)$ time and decode the encoding to reconstruct G in $O(n^2)$ time. We can also encode G using at most $3n-2$ bits and decode the encoding to reconstruct G in $O(n)$ time.*

If G is a degree-3 triconnected graph, then, using Property 3, we can show that each vertex $v \neq v_1, v_n$ can only be of 4 types. This gives a shorter encoding for G:

Lemma 2. *Given a degree-3 triconnected plane graph G with n vertices, we can encode it using at most $2n-2$ bits in $O(n)$ time (using Huffman Encoding). This encoding can be decoded in $O(n)$ time to reconstruct G.*

If G is a degree-4 biconnected (triconnected) graph, then each vertex v, where $v \neq v_1$ ($v \neq v_1, v_n$), can be of 16 (12) types. This gives Lemma 3 (Lemma 4):

Lemma 3. *Given a degree-4 biconnected plane graph G with n vertices, we can encode it using at most $4n-2$ bits in $O(n)$ time (using Huffman Encoding) and decode the encoding to reconstruct G in $O(n)$ time.*

Lemma 4. *Given a degree-4 triconnected plane graph G with n vertices, we can encode it using $2+(n-2)\log_2 12+1 < 3.59n$ bits in $O(n^2)$ time and decode the encoding to reconstruct G in $O(n^2)$ time. We can also encode G using at most $2+3.67(n-1)+1 < 3.67n$ bits (using Huffman encoding) and decode the encoding to reconstruct G in $O(n)$ time.*

6 Encoding an Orthogonal Representation

We will use the following properties of an orthogonal representation:

Property 4. The sum of angles around any vertex is equal to 360°.

Property 5. The sum of interior angles of the polygon p representing any internal face is equal to $(k-2)180°$, where k is the total number of line-segments in p.

We can encode an orthogonal representation Γ of a biconnected plane graph G by:

- *encoding structure:* encoding the structure of graph G,
- *encoding angles:* for each vertex, encoding the angles between consecutive edges incident on it, and
- *encoding turns:* for each edge $e=(u,v)$, encoding the sequence of left and right turns encountered while walking from u to v along e.

To encode angles, suppose G has n vertices and m edges. Each angle of Γ can be either 90°, 180°, or 270°. Suppose we have already constructed a rightmost canonical ordering $c = v_1, v_2, \ldots, v_n$ of the vertices of G. Let v_i be a vertex of G. Let $e_1 e_2 \ldots e_k$, where $k \leq 4$ be the counterclockwise order of edges incident on v_i, where, if $v_i \neq v_1$, then e_1 is the incoming tree edge of v, and if $v_i = v_1$, then e_1 is the edge (v_1, v_2). Let s_i^* be the string $a_1 a_2 \ldots a_k$, where a_j represents the counterclockwise angle between edges e_j and e_{j+1} at vertex v_i. a_j is equal to A, B, or C, respectively, if the magnitude of the corresponding angle is equal to 90°, 180°, or 270°, respectively. Then, we can construct a string $S^* = s_1^* s_2^* \ldots s_n^*$, that encodes all the angles of G. Total number of symbols in S^* is equal to number of angles in Γ, which is equal to $2m$.

Using Property 4, we can encode S^* using even fewer bits. Property 4 implies that, for each vertex v_i, it is sufficient to encode angles $a_1, a_2, \ldots, a_{k-1}$ since the value of angle a_k can be obtained from them. Thus, for v_i, it is sufficient to construct the string $s_i^* = a_1 a_2 \ldots a_{k-1}$. So, the overall number of symbols in string S^* can be reduced to $2m - n$. We therefore have the following lemma:

Lemma 5. *Given an orthogonal representation of a degree-4 biconnected graph G with n vertices, we can encode its angles using $(2m-n)\log_2 3 \approx 1.58(2m-n) \leq 4.74n$ bits in $O(n^2)$ time and at most $1.67(2m-n) \leq 5.01n$ bits in $O(n)$ time (using Huffman Encoding). Moreover, during decoding, if we already knew the degree of each vertex, then we can decode these encodings to obtain the angles in $O(n^2)$ and $O(n)$ time, respectively.*

If G is a triconnected graph, then each vertex has at least 3 angles around it, and so each angle can be either 90°, or 180°. Therefore:

Lemma 6. *Given an orthogonal representation of a degree-4 triconnected graph G with n vertices, we can encode its angles using $2m - n$ bits in $O(n)$ time. Moreover, during decoding, if we already knew the degree of each vertex, then we can decode the encoding to obtain the angles in $O(n)$ time.*

Now consider the problem of encoding the turns of Γ. Given a rightmost canonical ordering $c = v_1, v_2, \ldots, v_n$ of G, and the associated canonical tree T_c, we first construct an ordering $o = e_1, e_2, \ldots, e_m$ of the edges of G by putting the incoming tree and non-tree edges of the vertices $v_1, v_2, \ldots, v_n$ into o, such that the edges of v_i precede those of v_j if $i < j$, and for each vertex v_i, we first put its incoming tree edge, and then its incoming non-tree edges in the same order as their counter-clockwise order around v_i. Next, for each edge $e_i = (v_j, v_k)$ in o, where $j < k$, we construct a (possibly empty) string s_i^+ consisting of symbols L and R, where a symbol L (R) denotes a left (right) turn encountered while walking from v_j to v_k along e_i. These symbols are placed in s_i^+ in the order the corresponding turns are encountered while walking from v_j to v_k. Finally, we construct a string $S^+ = s_1^+ \# s_2^+ \# \ldots \# s_n^+$ consisting of all the s_i^+'s separated by a symbol $\#$.

Lemma 7. *Given an orthogonal representation Γ with b turns (bends) of a degree-4 biconnected graph G with n vertices, we can encode its turns using $(b+m-1)\log_2 3 \approx 1.58(b+m-1)$ bits in $O((b+m)^2)$ time, and at most $1.67(b+m-1)$*

bits in $O(b+m)$ time (using Huffman Encoding). These encodings can be decoded in $O((b+m)^2)$ and $O(b+m)$ time, respectively, to obtain the turns of Γ.

If Γ is a turn-monotone orthogonal representation, then we can reduce the length of S^+ by using Property 5 as follows: c induces an ordering $f_1, f_2, \ldots, f_p$ of the internal faces of G, such that when we reconstruct G using c, as in the proof of Theorem 2, starting from an initially empty graph, the faces will get inserted into it in the same order. Let $I_1, I_2, \ldots, I_K$ be the corresponding intervals of c. The ordering $f_1, f_2, \ldots, f_p$ is defined as follows (see Figure 1(b)): Let $I_1 = [v_3, v_{3+q'}]$, where $q'' \geq 0$. Face f_1 is the face consisting of the vertices $v_1, v_3, \ldots, v_{3+q'}, v_2$. In general, suppose we have already constructed the partial ordering $f_1, f_2, \ldots, f_s$ of the faces, using intervals $I_1, I_2, \ldots, I_{k-1}$. Let $I_k = [v_k, v_{k+q}]$, where $q \geq 0$. Let $P = v_1(= u_1)u_2 \ldots u_s(= v_2)$ be the subpath of H_{k-1} that we obtain by removing the edge (v_1, v_2) from H_{k-1}. Let C_j be the path $v_k v_{k+1} \ldots v_{k+q}$. We have 2 cases:

- $q > 0$: Then, C_j has exactly two neighbors v_l and v_r in H_{k-1}. Let $x_1 = (v_l)x_2, \ldots, x_t(= v_r)$ be the subpath of P that connects v_l and v_r. Then, f_{s+1} is the internal face of G consisting of the vertices $v_l(= x_1), v_k, \ldots, v_{k+q}, v_r(= x_t), x_{t-1}, x_{t-2}, \ldots, x_2$. We say that face f_{s+1} *belongs to* Interval I_k.
- $q = 0$: Then C_j consists of exactly one vertex v_k. Let $u'_1(= v_l), u'_2, \ldots, u'_t(= v_r)$ be the left-to-right order of the neighbors of v_k in H_{k-1}. Let P'_i, where $1 \leq i \leq t-1$, be the subpath of P that connects u'_i and u'_{i+1}. Then, each face f_{s+i}, where $1 \leq i \leq t-1$, is the internal face that consists of the vertex v_k and the vertices of path P'_i. We say that face f_{s+i} *belongs to* Interval I_k.

(Figure 1(b) also shows the ordering of the faces given by the rightmost canonical ordering shown in Figure 1(b).) Let T_c be the canonical spanning tree associated with c. For each face f_i, a *tree* edge of f_i is one that is also an edge of T_c.

Fact 1. *Except for one non-tree edge e, all the non-tree edges of each face f_i are already contained in the faces $f_1, f_2, \ldots, f_{i-1}$. We will call edge e as the* non-tree completion edge *of f_i.*

Intuitively, we call the edge non-tree completion edge because, while reconstructing G using c, this is the only non-tree edge that we need to add to the already constructed graph to add face f_i to it (of course, we will need to add the tree edges of f_i also). For example, in Figure 1, edge $(14, 12)$ is the non-tree completion edge of face f_{10}. For the face f_s, in the case $q > 0$ given above, the non-tree completion edge is the edge (v_{k+q}, v_r). For each face f_{s+i}, in the case $q = 0$ given above, the non-tree completion edge is the edge (v_k, u_{i+1}).

Since each edge of a turn-monotone orthogonal representation Γ has same kinds of turns only (left or right, but not both), Property 5 implies that for any face of Γ, it is sufficient to encode the turns of all but one edge, namely its non-tree completion edge e, since the turns of e can be deduced from the turns of the other edges. Infact, Lemma 8 says that it is sufficient to encode turns of tree edges:

Lemma 8. *Let Γ be a turn-monotone orthogonal representation of a degree-4 biconnected plane graph G. let c be a rightmost canonical ordering of G. Suppose we construct a string S^+ encoding the turns of Γ as in Lemma 7 using c, except that S^+ encodes the turns of* only *the tree edges of G. Then, by decoding S^+ we can obtain the turns of* all *the edges of Γ.*

Proof. Let $f_1, f_2, \ldots, f_p$ be the ordering of faces that corresponds to c, as defined above. We can easily prove this lemma can using induction:

Base Case: Consider face f_1. Decoding S^+ will give us the turns of all the tree edges of f_1. f_1 has exactly one non-tree edge e (which is its non-tree completion edge). From Property 5, we can determine the turns of e also.

Induction: Suppose we have already determined the turns of all the edges of faces $f_1, f_2, \ldots, f_{i-1}$. Consider face f_i. From Fact 1, except for its non-tree completion edge e, all the other non-tree of f_i are already contained in the faces $f_1, f_2, \ldots, f_{i-1}$. Decoding S^+ will give us the turns of all the tree edges of f_1. Hence, except for e, we would know the turns of all the edges of f_i. From Property 5, we can determine the turns of e also.

□

Since, T_c has exactly $n-1$ edges, we have:

Lemma 9. *Given a turn-monotone orthogonal representation with b turns (bends) of a degree-4 biconnected graph with n vertices, we can encode its turns using at most $(b+n-2)\log_2 3 \approx 1.58(b+n-2)$ bits in $O((b+n)^2)$ time, and at most $1.67(b+n-2)$ bits in $O(b+n)$ time (using Huffman Encoding). These encodings can be decoded in $O((b+n)^2)$ and $O(b+n)$ time, respectively, to obtain the turns.*

To encode an orthogonal representation Γ, we construct a string $S_1 = A'L'A^*L^*S'S^*S^+$, where S', S^*, S^+ are strings encoding structure, angles, and turns, respectively, of Γ, as given by Lemmas 1 (or 2, 3, or 4), 5 (or 6), and 7 (or 9), respectively, L' (L^*) is length of S' (S^*) in binary notation, and $A'(A^*)$ encodes the length of L' (L^*) in unary notation, and consists of $|L'|$ ($|L^*|$) 0's followed by a 1. Note that lengths of A', L', A^*, and L^* are $O(\log n)$ each.

7 Encoding Lengths of Line-Segments

Let d be a planar orthogonal drawing with b turns of a degree-4 biconnected plane graph G with n vertices and m edges. Suppose each line-segment of d has length at most W. Just as we constructed a string S^+ in Section 6 to encode turns, we can construct a string $S'' = s_1''s_2'' \ldots s_m''$, where each s_i contains the lengths of all the line-segments of edge $e_i = (v_j, v_k)$, placed in the order the corresponding line-segments are encountered while traveling along e_i from v_j to v_k, where $j < k$.

We can reduce the length of S'' by using the following property of d: Suppose we orient each horizontal line-segment of d as going "East" or "West", and each vertical line-segment as going "North" or "South", assuming that the line-segment of the edge (v_1, v_2) incident on v_1 goes East. (We can easily do this in $O(n+m+b) = O(n+b)$ time using the angle and turn information contained in d.)

Property 6. In d, for each face of G:

1. The sum of the lengths of all the line-segments going East = the sum of the lengths of all the line-segments going West, and
2. The sum of the lengths of all the line-segments going North = the sum of the lengths of all the line-segments going South.

Property 6 implies that we can omit encoding the length of one horizontal and one vertical line-segment of f_i, and still be able to obtain the lengths of all the line-segments of f_i from an encoding of the lengths of its other line-segments. To decide which line-segments to omit, consider the ordering $f_1, f_2, \ldots, f_m$ of the faces of G that we can obtain from a rightmost ordering c of G, as described in Section 6. Let $I_k = [v_k, v_{k+q}]$ be the interval of c, such that f_i belongs to I_k. Let E_i be the set of all the edges of f_i that are not in the faces $f_1, f_2, \ldots, f_{i-1}$. Note that E_i contains at least one edge, namely, the non-tree completion edge $e = (u, v)$ of f_i. Moreover, the edges of E_i form a connected path p, which connects u with a vertex u', where u is the end-vertex of e that belongs to H_{k-1}, and u' is a vertex common to both f_i and f_{i-1}. The *free horizontal* (*vertical*) line-segment of f_i is defined as the first horizontal (vertical) line-segment encountered while traveling along p from u to u'. Note that f_i will have at least one free line-segment (which can be horizontal or vertical). While encoding the lengths of the line-segments of d, we can omit from S'' all the free line-segments of the faces of G.

Having constructed S'', we can construct a string $S_2 = A''T''S''$, where T'' stores the value of W in binary using exactly $\log_2(W+1)$ bits, and A'' contains a sequence of $|T''|$ 0's followed by a 1. String A'' basically encodes the length of T'' in unary notation. We have the following lemma:

Lemma 10. *We can encode the lengths of the line-segments of d using a string S_2 consisting of $1+(2+b+m-f_H-f_V)\log_2(W+1) \leq 1+(2+b+m-f)\log_2(W+1)$ bits in $O(|S_2|)$ time, where f_H and f_V are the number of horizontal and vertical free line-segments, respectively, of d. Assuming that, while decoding, we already know all the angles and turns of d, we can decode S_2 to obtain the lengths of all the line-segments of d in $O(|S_2|)$ time.*

8 Encoding a Planar Orthogonal Drawing

Let d be a planar orthogonal drawing of a degree-4 biconnected planar graph G. Let Γ be the orthogonal representation of G that corresponds to d.

We can encode d by constructing a string $S = BLS_1S_2$, where S_1 is the string constructed in Section 6 that encodes Γ, S_2 is the string constructed using Lemma 10 that encodes the free lengths of the line-segments of d, L is a string, with length $\log_2(|S_1|+1)$, that encodes in binary notation the length of string S_1, and B is a string that contains a sequence of $|L|$ 0's followed by a 1. B encodes the length of L in unary notation.

We can obtain d by decoding S, by first extracting B from it and obtaining the length of L, then extracting L and obtaining the length of S_1, then extracting

S_1 and decoding it to obtain Γ, and finally, extracting S_2 and decoding it to obtain the lengths of the line-segments of d. This is summarized in Theorem 3:

Theorem 3. *Let d be a planar orthogonal drawing, with b turns (bends) of a plane graph G with n vertices, m edges, and f internal faces. Suppose each line-segment of d has length at most W. Let $p = (2 + b + m - f)\log_2(W + 1)$. Let $q = n + b$.*

- *If G is a degree-4 biconnected graph, then we can encode (and decode) d using at most $4.74m+2.42n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $5.01m + 2.33n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*
- *If G is a degree-4 triconnected graph, then we can encode (and decode) d using at most $3.58m+2.59n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $3.67m + 2.67n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*
- *If G is a degree-3 biconnected graph, then we can encode (and decode) d using at most $4.74m+1.23n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $5.01m + 1.33n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*
- *If G is a degree-3 triconnected graph, then we can encode (and decode) d using at most $3.58m + n + 1.58b + p + O(\log q)$ bits in $O((n + b + p)^2)$ time, and using at most $3.67m+n+1.67b+p+O(\log q)$ bits in $O(n+b+p)$ time.*

Moreover, if d is turn-monotone, then we can encode it using fewer bits, as follows:

- *If G is a degree-4 biconnected graph, then we can encode (and decode) d using at most $3.16m + 4n + 1.58b + p + O(\log q)$ bits in $O((n + b + p)^2)$ time, and using at most $3.34m + 4n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*
- *If G is a degree-4 triconnected graph, then we can encode (and decode) d using at most $2m + 4.17n + 1.58b + p + O(\log q)$ bits in $O((n + b + p)^2)$ time, and using at most $2m + 4.34n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*
- *If G is a degree-3 biconnected graph, then we can encode (and decode) d using at most $3.16m+2.81n+1.58b+p+O(\log q)$ bits in $O((n+b+p)^2)$ time, and using at most $3.34m + 3n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*
- *If G is a degree-3 triconnected graph, then we can encode (and decode) d using at most $2m + 2.58n + 1.58b + p + O(\log q)$ bits in $O((n + b + p)^2)$ time,, and using at most $2m + 2.67n + 1.67b + p + O(\log q)$ bits in $O(n + b + p)$ time.*

References

1. R. C. Chuang, A. Garg, X. He, M. Y. Kao, and H. Lu. Compact encoding of planar graphs via canonical ordering and multiple parenthesis. In *Proc. International Colloqium on Automata, Languages and Programming (ICALP)*, pp. 118–129, 1998.
2. H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.
3. G. Di Battista, G. Liotta, and F. Vargiu. Diagram Server. *J. Visual Lang. Comput.*, 6(3):275–298, 1995.

4. G. Di Battista, G. Liotta, and F. Vargiu. Spirality and optimal orthogonal drawings. *SIAM J. Comput.*, 27(6):1764–1811, 1998.
5. Xin He, M.-Y. Kao, and H. Lu. A fast general methodology for information-theoretically optimal encodings of graphs. *SIAM J. Comput.*, 30(3):838–846, 2000.
6. G. Kant. Drawing planar graphs using the *lmc*-ordering. In *Proc. 33th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 101–110, 1992.
7. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.
8. R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Trans. Circuits Syst.*, CAS-36(9):1230–1234, 1989.
9. W. T. Tutte. A census of planar triangulation. *Canad. J. Math.*, 14:21–38, 1962.