Compactly Encoding and Decoding the Connectivity of a Plane Pseudograph in Linear Time

Raquel Viaña*

Abstract

A *plane pseudograph* is a plane graph allowing both loops and multiple edges. The encoding technique we propose belongs to a class of compression methods based on a deterministic graph traversal, which is encoded as a bit string. Arrays of vertices and edges stored in the order defined by this traversal, together with the bit string, allow the retrieval of the original graph. Within this framework, the most general methods to encode plane graphs allow obtaining the cyclic ordering of the edges incident to each vertex. Provided any directed edge incident to the infinite face is specified, the plane graph is uniquely defined. However, this information may not be sufficient to re-create the faces of the original graph when loops are allowed. In this paper, we analyse what information must be encoded in the bit string to retrieve correctly all incidences among the vertices, edges and faces of any plane pseudograph. Let $\mathcal{G}$ be a connected plane pseudograph with V vertices, E edges and F faces. The most common representation of $\mathcal{G}$ in computational geometry is the half-edge data structure, which requires $2E \log V + (V + 4E + F) \log(2E) + 2E \log F$ bits to store the connectivity of the graph. The compression method proposed in this paper encodes the graph connectivity in $4E + 1$ bits, and allows encoding-decoding the data structure representing the graph in $O(E)$ time.

1 Introduction

Plane pseudographs are widely employed to model 2D geographic maps in vector format, dual graphs of communication or stream networks, etc. Web applications in the context of GIS require the transmission of these usually huge plane graphs, what under current transmission rates makes compression techniques essential.

Encoding methods have been broadly studied for triangulations [2, 3, 7, 11, 16, 17], and compression ratios achieving the theoretically optimal rate of $\log_2 \frac{256}{27} \approx 3.245$ bits per vertex [19] have been obtained [14]. Techniques to encode general polygonal meshes have also been studied [6, 12, 8, 10]. Provided a plane pseudograph could be triangulated or polygonated, the previously cited techniques could be applied to compress it. This solution, besides increasing both storage and time complexity, is not valid for certain classes of graphs (e.g., graphs containing loops or multiple edges).

Research has also been undertaken in the field of discrete mathematics and information theory to encode planar graphs. In [15], it is shown that general unlabeled graphs can be encoded by $\binom{V}{2} - V log_2 V + O(V)$ bits in $O(V)$ time. Several schemes to succinctly represent plane graphs use particular spanning trees based on Schnyder trees [4, 1]. An encoding for graphs with information-theoretically minimum number of bits in $O(V \log V)$ time is proposed in [5]. Lu [13] improves the previous time complexity to $O(V)$. The drawback of these methods is the constant hidden in the big-O notation. Turán proposed in [18] an scheme which uses $4E$ bits to encode a plane graph. In [9] this bit count was reduced to $3.58E$, at the cost of not encoding the faces in the plane graph. The compression technique we propose uses the deterministic graph traversal proposed by Turán [18], which will be reviewed next.

Let $\mathcal{G}$ be a plane graph and $\mathcal{T}$ be a rooted spanning tree of $\mathcal{G}$ with a distinguished edge incident to its root (see Figure 1). Graph $\mathcal{G} - \mathcal{T}$ results from removing the edges of $\mathcal{T}$ from $\mathcal{G}$. Assume each edge of $\mathcal{T}$ is decomposed into two companion half-edges of opposite directions, such that the face a half-edge bounds is to its right. A half-edge is directed from its *origin* vertex to its *destination* vertex. Let us call the half-edge of the distinguished edge of $\mathcal{T}$ which has its origin at the root v_0 of $\mathcal{T}$ the *first half-edge* of $\mathcal{G}$. Consider a traversal of consecutive half-edges of $\mathcal{T}$ starting at the first half-edge. The origin vertices of the consecutive half-edges form a cyclic sequence of possibly repeated vertices. In the graph of Figure 1 such sequence would be $v_0v_1v_2v_1v_0v_3v_4v_3v_5v_3$. Let us call *current half-edge* to the half-edge indicating the position of the traversal, denoted e_c. All edges e of $\mathcal{G} - \mathcal{T}$ incident to the origin of e_c, and placed between e_c and the half-edge previous to e_c are traversed in counterclockwise sense. Each time one edge e of $\mathcal{G} - \mathcal{T}$ appears, a parentheses symbol is interleaved in the cyclic sequence of vertices. The first time e is reached, an open-bracket (will be added to the sequence. The second time, we will insert a close-bracket) symbol. From the cyclic sequence of vertices, tree $\mathcal{T}$ can be retrieved. Turán proved [18] that the

*Department of Mathematics, University of Alcala, raquel.viana@uah.es

position of edges belonging to $\mathcal{G} - \mathcal{T}$ can be obtained from the cyclic sequence enriched with the parenthesis symbols (see Figure 1). When a (symbol is found, an edge e belonging to $\mathcal{G} - \mathcal{T}$ has been reached the first time. Thus, one of the vertices v incident to e can be retrieved. All edges incident to v are retrieved in counterclockwise order. Reaching a) means that the other vertex incident to the last edge affected by a (has been reached. By encoding the cyclic sequence of vertices with + and − symbols, depending on whether each traversed vertex is closer or further from the root, 4 bits per edge are required to encode $\mathcal{G}$.

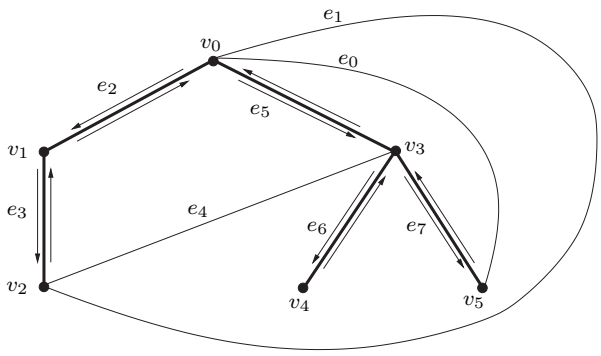

Figure 1: The edges of a vertex spanning tree $\mathcal{T}$ of plane graph $\mathcal{G}$ have been depicted with bold lines. The root of $\mathcal{T}$ is v_0, and e_2 is chosen as the distinguished edge. Deterministic traversal of $\mathcal{G}$ is encoded as sequence $((v_0v_1)(v_2v_1v_0)v_3v_4v_3)v_5v_3$, or equivalently $((++)(--+)+-+)--$. The arrays of vertices and edges are $v_0v_1v_2v_3v_4v_5$ and $e_0e_1e_2e_3e_4e_5e_6e_7$ respectively.

The theoretical aspect of encoding a plane pseudograph in as less number of bits as possible has its practical counterpart in the efficient implementation of algorithms to encode and decode the data structure used to store the graph. A measure of the efficiency of a data structure is the capability of traversal according to any criterion. Usual guides are traversal of faces or edges around a vertex and traversal of edges around a face.

The *FE data structure* [21] (usually known as *half-edge* data structure) is the most common computational representation of a plane pseudograph. Each edge is decomposed into two companions half-edges, one for each possible direction, such that the face they bound is to their right (or to the left). The specification of the FE data structure follows:

- for each vertex v, its coordinates, and one of the half-edges with origin at v;
- for each half-edge e (see Figure 2), its origin vertex v_o; its incident face f; its companion half-edge e', the half-edge e_{prec} coming into its origin vertex, and the half-edge e_{post} outgoing its destination vertex (3 half-edges in total);
- for each connected component belonging to the boundary of a face, called a *loop*, a reference to one of its half-edges (or the vertex in case there is no edge); and a reference to next loop inside the face;
- for each face f, its outer loop.

The half-edge data structure is *sufficient* [21], i.e. it allows the retrieval of all incidences among the vertices, edges and faces of the plane pseudograph it represents. Besides, all entities incident to a given vertex, edge or face can be retrieved in time linear in the output size. Disregarding the geometry of vertices and edges, this data structure requires $2E$ references to vertices, $2E$ references to faces, and $V+6E+F$ references to half-edges. However, by storing each pair of companion half-edges in consecutive locations, only $V + 4E + F$ references to half-edges must be stored. Hence, a total amount of $V \log(2E) + E(2 \log V + 4 \log(2E) + 2 \log F) + F \log(2E)$ bits are required to store the connectivity of a connected plane pseudograph using this data structure.

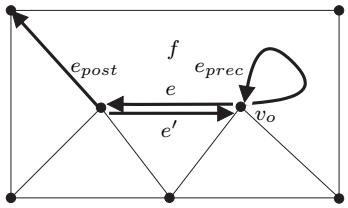

Figure 2: In the half-edge data structure, each half-edge e stores references to its origin vertex v_0; incident face f; half-edge e_{prec} previous to it in f; half-edge e_{post} posterior to it in f; and its companion half-edge e'.

Turán stated in [18] that his encoding technique admits an extension to encode connected plane pseudographs in a number of bits linear in the number of edges, and that coding and decoding require polynomial time. In the next section, we provide an encoding scheme which performs both encoding and decoding in linear time and that with only one more bit than Turán's proposal for simple plane graphs allows the presence of loops and multiple edges.

2 Encoding Algorithm

Let $\mathcal{G}$ be a plane pseudograph, and $\mathcal{T}$ be a vertex spanning tree of $\mathcal{G}$. Assume $\mathcal{T}$ contains some edge. Consider a sufficiently small disk centered in the root v_0 of $\mathcal{T}$ which does not contain completely any edge inside it. In Figure 3 (a) we have depicted disk D centered in v_0 with a dashed line. All edges with origin v_0 which do not belong to $\mathcal{T}$ will be traversed from the half-edge previous to the first half-edge until the first half-edge in counterclockwise order as they are found when traversing the

boundary of the disk. After that, the first half-edge will be traversed, and this process is repeated until the traversal of all the half-edges in $\mathcal{T}$ has finished. During this traversal, two arrays will be created. In the array of vertices, they are stored in the order in which they are traversed, i.e., as in a preorder traversal of $\mathcal{T}$. Edges of $\mathcal{G}$ are stored in the array of edges when they are first found. In Figure 3 (a), the array of vertices would be v_0v_1, and the array of edges $e_1e_2e_0$. During the decoding process, a stack of edges in $\mathcal{G}-\mathcal{T}$ is maintained. Each time an edge in $\mathcal{G}-\mathcal{T}$ is found the first time, it is added to the stack. When an edge in $\mathcal{G}-\mathcal{T}$ is reached the second time, it will identified with the top most half-edge in the stack.

It can be easily proved that Turán's encoding adapted to plane pseudographs provides a correct cyclic ordering of the edges of $\mathcal{G}$ around each vertex. However, for the decoding algorithm retrieving the faces of $\mathcal{G}$ correctly, the half-edges bounding the infinite face must be known. If loops are not allowed, it suffices to know one of the half-edges incident to the infinite face. However, for general plane pseudographs this is not sufficient. In Figure 3, two different pseudographs with the same cyclic ordering of edges around v_0 are shown. In both of them, the half-edge of e_2 which determines a counterclockwise loop bounds the infinite face.

Theorem 1 *A connected plane pseudograph $\mathcal{G}$ can be encoded using $4E+1$ bits, where E is the number of edges in $\mathcal{G}$. Both encoding and decoding require $O(E)$ time.*

Proof. Assume there are two or more vertices of $\mathcal{G}$ incident to the infinite face, denoted f_∞. In this case, at least a half-edge which is not a loop exists bounding f_∞. If one of such half-edges is chosen as the first half-edge e of $\mathcal{G}$, the half-edge e_{prec} previous to e is uniquely defined from the cyclic ordering of edges around the origin of e, and e_{prec} will also bound f_∞. This way, moving from each edge to its previous one all the half-edges bounding f_∞ can be retrieved.

If only one vertex v_0 of $\mathcal{G}$ is incident to f_∞, this implies that a set of loops incident to v_0 exist which contain all edges of any vertex spanning tree $\mathcal{T}$ (see Figure 3). In this case, given a half-edge e bounding f_∞, and given the cyclic sequence of edges around v_0, the half-edge e_{prec} previous to e is not uniquely defined. The boundary of disk D centered in v_0 contains two occurrences of e, and e_{prec} must be specified. Two possibilities arise: either $e_{prev}=e$ or $e_{prev}\neq e$. In the first case, e is the only half-edge bounding f_∞. In the second case, e and e_{prec} uniquely determine the boundary of f_∞.

There is a total of three possible cases which the bit string must encode to correctly retrieve the infinite face of $\mathcal{G}$, namely (i) a non-loop edge exists bounding f_∞; (ii) there is one and only one loop bounding f_∞; (iii) face f_∞ is only bounded by either two or more than two loops. The traversal of $\mathcal{G}$ can start at any edge of $\mathcal{G}$. Taking this into account, cases (i) and (iii) can be reduced to one only case: there are two or more than two half-edges bounding f_∞. The traversal should then start at an edge (or occurrence of a loop) bounding f_∞ such that the edge posterior to it in counterclockwise sense also bounds f_∞. Provided there is one and only one loop bounding f_∞, the traversal will start in it.

The encoding process requires just a traversal of $\mathcal{G}$. If $\mathcal{G}$ is represented by a half-edge data structure [21], the traversal requires time linear in the number of edges.

Given the arrays of vertices and edges obtained in the encoding process, together with the encoding bit string of length $4E+1$, let us now specify how to build a half-edge data structure representing $\mathcal{G}$ in linear time. Arrays of vertices, half-edges, and faces must be created, each of which stores references to other entities. Incidences among vertices and half-edges can be retrieved in time linear in the number of edges, as well as the assignment of each half-edge to its companion half-edge, by reading the part of the bit string containing symbols +-(). To establish the incidences of faces with half-edges, each half-edge record must be traversed. The first element in the bit strings allows the correct retrieval of f_∞, what uniquely defines the rest of faces.

□

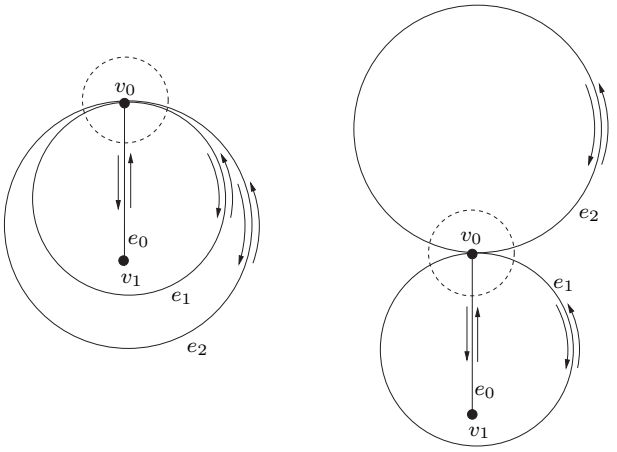

Figure 3: Two different pseudographs with the same cyclic ordering of edges around v_0 and the same half-edge (e_2 oriented counterclockwise) bounding the infinite face.

Assume now the faces in the plane graph are allowed to have holes, i.e. connectedness constraint is removed. The edges of $\mathcal{G}-\mathcal{T}$ are in one to one correspondence with the faces of $\mathcal{G}$. Thus, by adding 1 bit after each edge of $\mathcal{G}-\mathcal{T}$ is first reached we will indicate whether the cor-

responding face has some connected component inside or not. To indicate how many connected components C inside a face there are, $C+1$ bits are required. Finally, an additional bit per connected component is required to know whether the component (or loop in the FE data structure terminology) is an isolated point. Assuming there is only one connected subgraph bounding f_∞, at the end of the encoding of the connected subgraph a) symbol is added to the bit sequence, and the encoding of the next connected component follows.

3 Concluding Remarks

The encoding algorithm developed in this work encodes the connectivity of a plane pseudograph as a bit sequence. This string, together with the arrays of vertices and edges, allows the retrieval of the plane graph. Techniques such as Huffman or arithmetic coding could be applied to obtain better compresion ratios. Compressing the geometry of vertices and edges is another topic of research, out of the scope of this paper.

A problem of practical interest is the comparison of the compression algorithm proposed in this paper with other approaches to compactly represent plane pseudographs, as the multiresolution model developed in [20] to manage plane graphs at different levels of detail.

4 Acknowledgments

Pedro A. Ramos is greatly acknowledged for his helpful comments and the support he has given me when I was writing this paper. This work has been partially supported by the Spanish Ministry of Science and Technology under grant TIC2003-08933-C02-01.

References

[1] Y. T.Chiang, C. C.Lin, H. I.Lu. Orderly spanning trees with applications to graph encoding and graph drawing. In *Symposium of Discrete Algorithms (SODA)*, pages 506–515, 2001.

[2] C. Gotsman, S. Gumhold, L. Kobbelt. Simplification and Compression of 3D Meshes. In *Proceedings of the European Summer School on Principles of Multiresolution in Geometric Modelling (PRIMUS)*, Munich, 2001.

[3] S. Gumhold, W. Straßer. Real Time Compression of Triangle Mesh Connectivity. In *ACM Computer Graphics Proceedings (SIGGRAPH '98)*, pages 133–140, 1998.

[4] X. He, M. Kao, H. Lu. Linear-time succint encodings of planar graphs via canonical orderings. In *SIAM Journal on Discrete Mathematics*, Vol.12, pages 317-325, 1999.

[5] X. He, M. Kao, H. Lu. A fast general methodology for information-theoretically optimal encodings of graphs. In *SIAM Journal of Computing*, Vol. 30, N.3, pages 836–46, 2000.

[6] M. Isenburg, J. Snoeyink. Face Fixer: Compressing Polygon Meshes with Properties. In *ACM Computer Graphics Proceedings (SIGGRAPH 2000)*, 2000.

[7] M. Isenburg, J. Snoeyink. Spirale reversi: Reverse decoding of the Edgebreaker encoding. In *Proceedings of 12th Canadian Conference on Computational Geometry*, pages 247–256, 2000.

[8] M. Isenburg. Compressing Polygon Mesh Connectivity with Degree Duality Prediction. In *Proceedings of Graphics Interface 2002*, pages 161–170, 2002.

[9] K. Keeler, J. Westbrook. Short Encodings of Planar Graphs and Maps. In *Discrete Applied Mathematics*, Vol.58, N.3, pages 239–252, 1995.

[10] A. Khodakovsky, P. Alliez, M. Desbrun, P. Schroeder. Near-optimal connectivity encoding of 2-manifold polygon meshes. In *Graphical Models*, Vol.64, N.3-4, pages 147–168, 2002.

[11] D. King, J. Rossignac. Guaranteed 3.67v bit encoding of planar triangle graphs. In *Proceedings of teh 11th Canadian Conference on Computational Geometry*, pages 146–149, 1999.

[12] C. Gotsman, B. Kronrod. Efficient Coding of Nontriangular Mesh Connectivity. In *Graphical Models*, Vol.63, pages 263–275, 2001.

[13] H. I. Lu. Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits. In *Proceedings of the 13th Symposium on Discrete Algorithms (SODA)*, pages 223–224, 2002.

[14] D. Poulalhon, G. Schaeffer. Optimal Coding and Sampling of Triangulations. In *30th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1080–1094, 2003.

[15] M. Naor. Succinct Representation of General Unlabeled Graphs. In *Discrete Applied Mathematics*, Vol.28, N.3, pages 303–307, 1990.

[16] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. In *IEEE Transactions on Visualization and Computer Graphics*, Vol.5, N.1, 1999.

[17] C. Touma, C. Gotsman, C. Triangle Mesh Compression. In *Proceedings Graphics Interface'98*, pages 26–34, 1998.

[18] G. Turán. On the succinct representation of graphs. In *Discrete Applied Mathematics*, Vol.8, pages 289–294, 1984.

[19] W. T. Tutte. A census of planar triangulations. In *Canadian Journal of Mathematics*, Vol.14, pages 21–38, 1962.

[20] R. Viaña, P. Magillo, E. Puppo, P. A. Ramos. Multi-VMap: a Combinatorially Consistent Muti-Scale Model for Geographic Maps. To appear in *GeoInformatica: An International Journal on Advances of Computer Science for GIS*. Springer.

[21] K. Weiler. Edge-based Data structures for Solid Modeling in a Curved-surface Environment. In *IEEE Computer Graphics and Applications*, 5, 1, pages 21–40, 1985.