Journal of Information Processing Vol.21 No.4 660–664 (Oct. 2013)

[DOI: 10.2197/ipsjjip.21.660]

Recommended Paper

Two Compact Codes for Rectangular Drawings with Degree Four Vertices

Masashi Saito[1] Shin-ichi Nakano[1,a)]

Received: January 23, 2013, Accepted: July 3, 2013

Abstract: A rectangular drawing is a partition of a rectangle into a set of rectangles. Rectangular drawings have many important applications including VLSI layout. Since the size of rectangular drawings may be huge, compact encodings are desired. Several compact encodings of rectangular drawings without degree four vertices are known. In this paper, we design two compact encodings for rectangular drawings with degree four vertices. We give $5f - B - n_4$ bits and $5f - B - W - 3$ bits encodings for rectangular drawings, where f is the number of inner faces, n_4 is the number of vertices with degree four, and B (resp. W) is the number of inner faces touching the bottommost horizontal (resp. rightmost vertical) line segments.

Keywords: coding, tree, rectangular drawing, depth first search

1. Introduction

Compact encodings of graphs are studied for many classes of graphs [12], for instance, trees [8], and plane graphs [2], [3], [7], [9], [13]. See a nice textbook [12].

The well known naive coding of ordered trees is as follows. Given an ordered tree T we traverse T starting at the root with depth first manner. If we go down an edge then we code it with 0, and if we go up an edge then we code it with 1. Thus any ordered tree T with n vertices has a code with $2(n-1) = 2m$ bits, where n is the number of vertices and m is the number of edges in T. Some examples are shown in **Fig. 1**.

On the other hand, the number of ordered trees with n vertices is known as the Catalan number C_{n-1}, and it is defined as follows [4], [10].

$$\begin{aligned} C_n &= \frac{1}{n+1}\frac{(2n)!}{n!n!} \\ &= \frac{4^n}{(n+1)\sqrt{\pi n}}\left(1 - \frac{1}{8n} + \frac{1}{128n^2} + O(n^{-3})\right) \qquad (1) \end{aligned}$$

For example, the number of ordered trees with four vertices is $C_{4-1} = 5$ as depicted in Fig. 1. We need at least $\log C_{n-1} = 2n - o(n) = 2m - o(n)$ bits to code an ordered tree with n vertices. So the naive coding using $2m$ bits for ordered trees is asymptotically optimal.

A rectangular drawing is a partition of a rectangle into a set of rectangles. Rectangular drawings have many important applications including VLSI layout [5], [6], [11]. Since the size of rectangular drawings may be huge, compact encodings are desired. Several compact encodings of rectangular drawings without degree four vertices are known. See Refs. [5], [6], [14], [15].

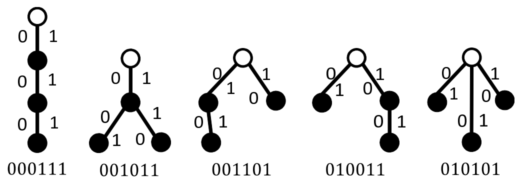


Fig. 1 The naive coding of ordered trees.

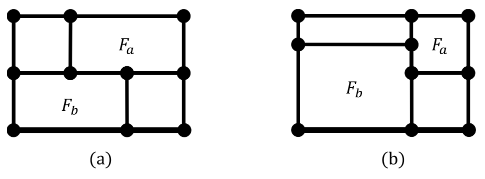


Fig. 2 Two rectangular drawings corresponding to the same plane graph.

In this paper, we design two compact encodings for rectangular drawings with degree four vertices. We give $5f - B - n_4$ bits and $5f - B - W - 3$ bits encodings for rectangular drawings, where f is the number of inner faces, n_4 is the number of vertices with degree four, and B (resp. W) is the number of inner faces touching the bottommost horizontal (resp. rightmost vertical) line segments.

Note that we cannot treat rectangular drawings simply as plane graphs. See two rectangular drawings in **Fig. 2**. They are isomorphic as plane graphs, however they are different as rectangular drawings. Because in Fig. 2 (a) the two faces F_a and F_b share a horizontal line, however in Fig. 2 (b) they share a vertical line. We store the direction (horizontal or vertical) for each edge in a given rectangular drawing.

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 explains our first encoding of rectangular drawings using $5f - B - n_4$ bits. Section 4 explains our second encoding of rectangular drawings using $5f - B - W - 3$

1 Department of Computer Science, Gunma Univercity, Kiryu, Gunma 376–8515, Japan
a) nakano@cs.gunma-u.ac.jp

The initial version of this paper was presented at FIT2012 held in Sep. 2012, whose program chair then recommended its publication in Journal of Information Processing (JIP).

bits, using the $(4f-4)$ bit encoding in Ref. [14]. Finally Section 5 is a conclusion.

2. Preliminaries

In this section we give some definitions. A *tree* is a connected graph with no cycle. A *rooted* tree is a tree with one vertex chosen as its root. *An ordered tree* is a rooted tree in which children of each vertex are ordered.

A drawing of a graph is *plane* if it has no two edges intersecting geometrically except at a vertex to which they are both incident. A plane drawing divides the plane into connected regions called *faces*. The unbounded face is called *the outer face*, and other faces are called *inner faces*.

A *rectangular drawing* is a partition of a rectangle into a set of rectangles. We regard the four corners of each rectangle as vertices. Let n_d be the number of vertices with degree d. Now $n = 4+n_3+n_4$ holds, since every rectangular drawing has exactly four vertices with degree two at the four corners of the outer face. Thus $2 \cdot 4 + 3 \cdot n_3 + 4 \cdot n_4 = 2m$ holds, where m is the number of egdes. Those equations and the Eular's formula $n - m + f = 1$ give $m = 3f - n_4 + 1$, and $n = 2f - n_4 + 2$.

A degree three vertex is *north missing* if it has a downward edge, a rightward edge, and a leftward edge. Let n_N be the number of north missing vertices. Similarly south missing, east missing, west missing, n_S, n_E, n_W are defined. Thus, $n_N + n_S + n_E + n_W = n_3$ holds. Since each north missing vertex is the top end of some maximal vertical line segment and each south missing vertex is the bottom end of some maximal vertical line segment, $n_N = n_S$ holds. Similarly $n_E = n_W$ holds.

3. Compact code I

In this section we give our first compact encoding for rectangular drawings based on the depth first search of an ordered tree. The encoding is similar to the one in Ref. [15], however which only works for rectangular drawings without degree four. Our encoding needs $5f - B - n_4$ bits for each of the rectangular drawings.

Let B be the number of inner faces touching the bottommost horizontal line segment. Given a rectangular drawing R, we first remove the bottommost horizontal line segment of the outer face as shown in **Fig. 3** (b). Now the resulting graph R' has $m-B$ edges. Then, we replace the lower right corner of each remaining inner face as shown in **Fig. 4**. See a complete example in Fig. 3 (c). Let R'' be the resulting graph.

Lemma 3.1

The resulting graph R'' is a tree with $m - B + n_4$ edges.

Proof. Since we only break each cycle corresponding to an inner face at the lower right corner, the resulting graph has no inner face and is still connected. Thus R'' is a tree. After removing B edges, R' has $m - B$ edges. Since the upward edge of each degree four vertex is replaced by two edges, R'' has $m - B + n_4$ edges.

Q.E.D.

Starting at the upper left corner we traverse the tree R'' with depth first manner (with right priority). See Fig. 3 (d). Each edge is traced exactly twice in both directions. Intuitively we trace counterclockwise around the wall corresponding to the tree R''.

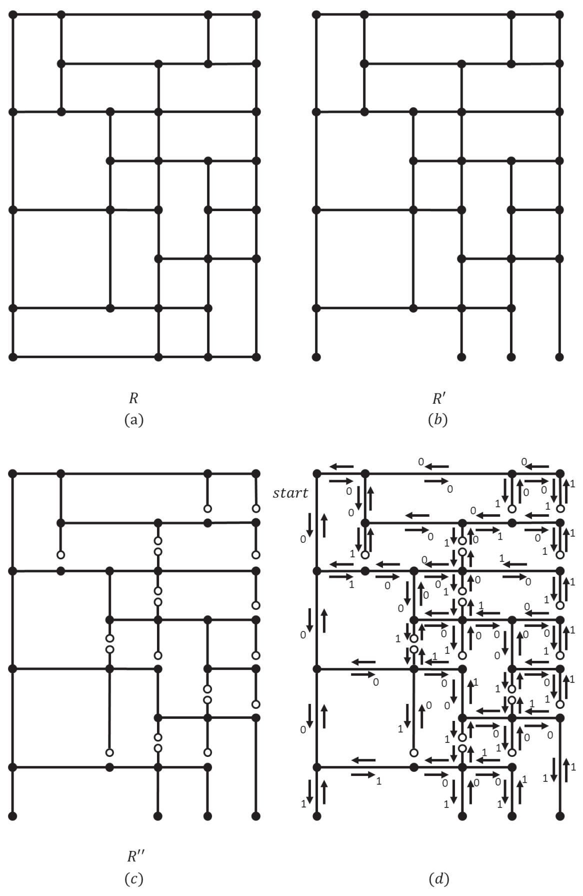


Fig. 3 (a) A rectangular drawing R, (b) removal of the botttommost horizontal line segment, (c) replacement of lower right corners, (d) the trace.

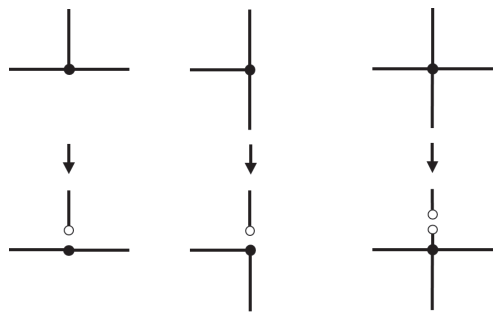


Fig. 4 Replacement of the lower right corner of each inner face.

Whenever we arrive at a vertex, say v, by tracing an edge, say e, we always have only "two" possibilities for the next direction to trace, as shown below, even though there are four possible directions, up, down, left and right. We have four cases.

Case 1: When trace e downward.

If the trace is the first trace of e then the next trace is either downward or upward. See Case 1(a) in **Fig. 5**. If the next trace is leftward then it contradicts the fact that we have cut the lower right corner of each inner face. If the next trace is rightward then this means v has degree two and v has only an upward edge and a rightward edge. In R such vertex exists only at the lower left corner. Since we have removed the bottommost horizontal line segment, R'' has no such vertex. A contradiction.

If the trace is the second trace of e then v has degree four and e is the upward edge of v. Then next trace is always leftward. See

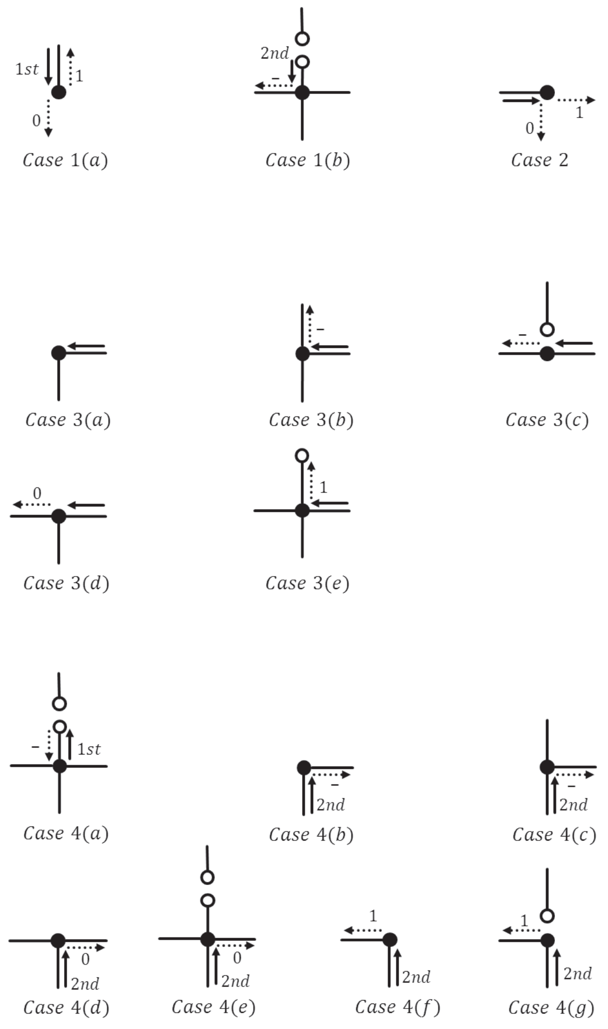


Fig. 5 The possible next traces.

Case 1(b) in Fig. 5.

Case 2: When trace *e* rightward.

The next direction to trace is either downward or rightward. See Case 2 in Fig. 5. If the next trace is upward then which means v has degree two and v has only an upward edge and a leftward edge. A contradiction, as shown in Case 1. If the next trace is leftward then v has degree one and v has only a leftward egde. In R there is no degree one vertex. In R' every degree one vertex has an upward edge. In R'' every degree one vertex has either upward or downward edge. A contradiction.

Case 3: When trace *e* leftward.

We have five subcases, as shown in Fig. 5. Since we have traced e leftward v has a rightward edge.

If v is the upper left corner then this is the last trace so there is no next trace (Case 3(a)). If v has no leftward edge then v has degree three, so the next trace is always upward (Case 3(b)). If v has no downward egde then v has degree two and the next trace is always leftward (Case 3(c)).

Otherwise the next trace is either leftward (Case 3(d)) or upward (Case 3(e)). We need one bit for each occurrence of Case 3(d) or Case 3(e).

Case 4: When trace *e* upward.

We have seven subcases, as shown in Fig. 5. If the trace is the first trace of e then the next trace is always downward (then leftward). See Case 4(a) in Fig. 5. In this case v has degree one.

If the trace is the second trace of e then v has degree either two, three or four. If v is the upper left corner then the next trace is always rightward. See Case 4(b) in Fig. 5. If v has no leftward edge then the next trace is always rightward. See Case 4(c) in Fig. 5.

Otherwise the next trace is either rightward (Case 4(d) and Case 4(e), or leftward (Case 4(f)) and (Case 4(g)). We need one bit for each occurrence of Case 4(d), Case 4(e), Case 4(f) or Case 4(g).

We trace each edge exactly twice in both directions, and we need one bit for each trace to record the direction of the next trace. However the last trace has no next trace (Case 3(a)), so we can save one bit. Thus we need $2m - 1$ bits in total. However we have a chance to save more bits. For Case 1 and Case 2 we need $m - B$ bits in total. For Case 3 we need $n_N + n_4$ bits in total. For Case 4 we need $n_N + n_E + n_4 + 1$ bits in total.

Now we have the following theorem.

Theorem 3.2

There is an encoding of rectangular drawings with degree four vertices using $5f - B - n_4$ bits.

Proof. $(m - B) + (n_N + n_4) + (n_N + n_E + n_4 + 1) = (3f - n_4 + 1 - B) + (n_N + n_S + n_E + n_4) + n_4 + 1 = 3f + 1 - B + (n - n_W - 4) + 1 = 3f + 1 - B + (2f - n_4 + 2) - n_W - 3 = 5f - B - n_4 - n_W$.

The encoding time is linear. With a suitable data structure with a stack, similar to Ref. [15], one can reconstruct the original rectangular drawing from the bitstring in $O(f)$ time. Whenever we find a degree two vertex (in leftward trace) or degree one vertex (in downward trace) we push the starting vertex into the stack, and whenever we find a degree one vertex (in upward trace) we merge it with the vertex at the top of the stack to reconstruct an original degree three or four vertex. See **Fig. 6**.

4. Compact Code II

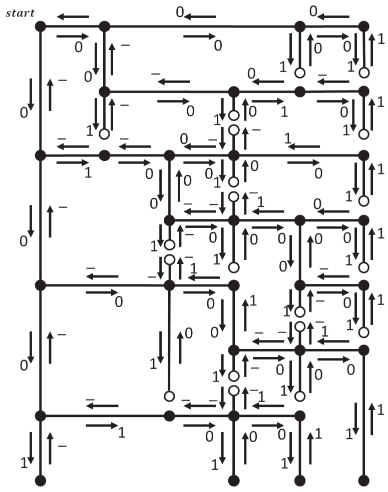
In this section, we give our second encoding for rectangular drawing using the $(4f - 4)$ bits encoding in Ref. [14] for rectangular drawing without degree four vertices. Given a rectangular drawing R, we replace each degree four vertex as shown in **Fig. 7**. See a complete example in **Fig. 8**. Then we encode R into a bitstring S using the method in Ref. [14]. The length of S is $4f - 4$ bits. To reconstruct the original rectangular drawing R, we need to append some information to indicate degree four vertices in the original drawing. So, we encode whether the lower right corner of each rectangle not touching either the bottommost nor rightmost segment has degree four or not, into $f - B - W + 1$ bits. Note that the lower right corner of each face touching either the bottommost or rightmost segment has always degree three or less in the original drawing. Here we use a natural ordering of inner faces defined in Ref. [14].

We have the following theorem.

Theorem 4.1

There is an encoding of rectangular drawing with $5f - B - W - 3$ bits.

The encoding and decoding time is linear with a suitable data structure.



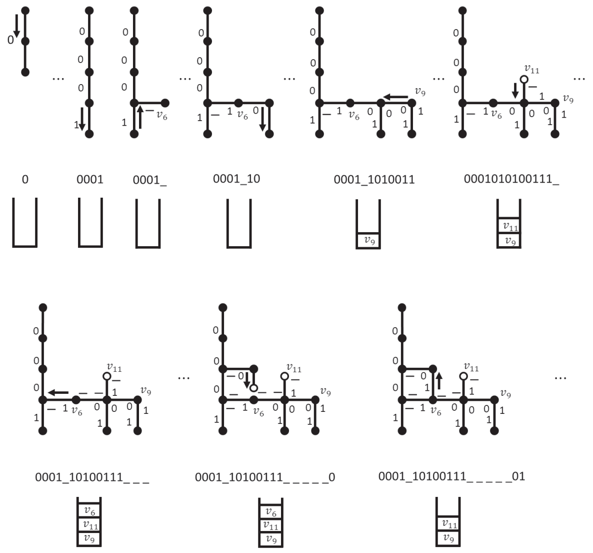


Fig. 6 Decoding of Code I.

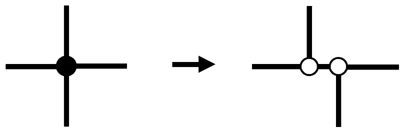


Fig. 7 Replacement of each degree four vertex.

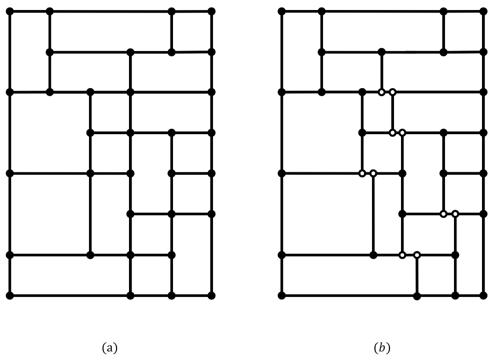


Fig. 8 Replacement of each degree four vertex in a rectangular drawing R.

5. Conclusion

In this paper, we gave two simple compact codings for rectangular drawings with degree four vertices. The coding needs only $5f - B - n_4$ or $5f - B - W - 3$ bits for each rectangular drawings. Code I is more compact for large n_4, and code II is more compact for large W. The running time for encoding and decoding is $O(f) = O(n)$.

The number of rectangular drawings with no degree four vertices is $\Omega(11.56^f) = \Omega(2^{3.53f})$ [1]. We need at least $\log |C|$ bits on average to encode an object in a set C. So we need at least $3.53f + c$ bits to encode a rectangular drawing for some constant c.

References

[1] Amano, K., Nakano, S. and Yamanaka, K.: On the number of rectanglar drawings: Exact couting and lower and upper bounds, IPSJ SIG Notes, Vol.AL-115, No.5, pp.33–40 (2007).

[2] Chuang, R.C.-N., Garg, A., He, X., Kao, M.-Y. and Lu, H.-I.: Compact encodings of planar graphs via cannical ordering and multiple parentheses, *Proc. 25th International Colloquium on Automata, Languages, and Programming*, LNCS 1443, pp.118–129 (1998).

[3] Chiang, Y.T., Lin, C.-C. and Lu, H.-I.: Orderly spanning trees with applications to graph encoding and graph drawing, *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp.506–515 (2001).

[4] Graham, R.L., Knuth, D.E. and Patashinik, O.: *Exercise 9.60 in Concrete Mathematics,* 2nd ed., Addison-Wesley (1994).

[5] Kant, G. and He, X.: Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems, *Thero. Comput. Sci.*, Vol.172, pp.175–193 (1997).

[6] Kozminiski, K. and Kinnen, E.: An algorithm for finding a rectangular dual of a planar graph for use in area planning for VLSI integrated Circuits, *Proc. 21st DAC*, pp.655–656 (1984).

[7] Keeler, K. and Westbrook, J.: Short encodings of planar graphs and maps, *Discrete Appl. Math.*, Vol.58, No.3, pp.239–252 (1995).

[8] Minro, J.I. and Raman, V.: Succinct representations of balanced parentheses, static trees and planar graphs, *Proc. 38th IEEE symposium on Foundations of Computer Science*, pp.118–126 (1997).

[9] Papadimitriou, C. and Yannakakis, M.: A note on succinct representations of graphs, *Inf. Comput.*, Vol.71, pp.181–185 (1985).

[10] Rosen, K.H. (Ed.): *Handbook of discrete and combinatorial mathematics*, CRC Press, Boca Raton (2000).

[11] Rahman, S., Nishizeki, T. and Ghosh, S.: Rectangular drawings of plannar graphs, *J. Algorithms*, Vol.50, pp.62–78 (2004).

[12] Spinard, J.P.: Efficient graph representations, AMS (2003).

[13] Turan, G.: Succinct representations of graphs, *Discrete Appl. Math.*, Vol.8, pp.289–294 (1984).

[14] Takahashi, T., Fujimaki, R. and Inoue, Y.: A (4n-4) bit respresentations of rectangular drawing of floorplan, *Proc. COCOON 2009*, LNCS 5609, pp.44–55 (2009).

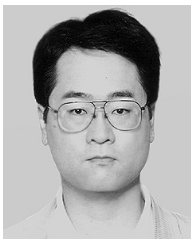
[15] Yamanaka, K. and Nakano, S.: Coding floorplans with fewer bits, *IEICE Trans. Fundamentals*, Vol.E89-A, No.5, pp.1181–1185 (2006).

Editor's Recommendation

The authors propose an efficient technology to encode rectangle-divided rectangles. Addressing a previously unresolved problem to deal with vetices of degree four, the paper was highly evaluated in terms of both novelty and interest.

(Koiti Hasida, FIT2012 program chair)

Masashi Saito received his M.E. degree from Gunma University in 2013. His research interests include graph algorithms.



Shin-ichi Nakano received his M.E. degree from Tohoku University in 1987. In 1987 he joined Seiko Epson Corp. and in 1990 he joined Tohoku University. Since 1999 he has been a faculty member of Department of Computer Science, Gunma University. His research interests are graph algorithms and graph theory.