2005－AL－99　(6)
2005／1／20

フロアプランの圧縮

山中 克久† 中野 眞一†

E-mail: †{yamanaka, nakano}@msc.cs.gunma-u.ac.jp

概 要

m 本の辺からなるフロアプランを, $2m$ ビットを用いてコード化する簡単な方法が知られている. 本文では, よりコンパクトなコード化を提案する. 我々の方法は, 各フロアプランを $5m/3$ ビットでコード化する.

Coding Floorplans with Fewer Bits

(Extended Abstract)

Katsuhisa YAMANAKA†, Shin-ichi NAKANO†

E-mail: †{yamanaka, nakano}@msc.cs.gunma-u.ac.jp

abstract

A naive coding of floorplans needs 2m bits for each floorplan, where m is the number of edges, in this paper we give a new simple coding of floorplans. Our coding needs only 5m/3 bits for each floorplan.

1 Introduction

In this paper we consider the problem of encoding a given floorplan R into a binary string S so that S can be decoded to reconstruct R. Furthermore we wish to minimize the length of S.

Succinct representation of graphs are studied for many classes of graphs, for instance, trees[M97, T84] and plane graphs[C01, C98, K95, P85].

The well known naive coding of ordered trees is as follows. Given a ordered trees 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 n-vertex ordered tree T has a code with $2(n-1) = 2m$ bits, where m is the number of edges in T. Some examples are shown in Fig. 1.

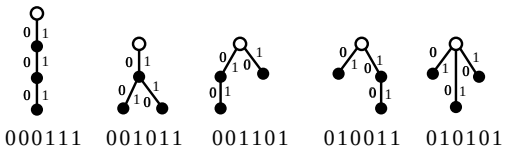

Figure 1: A code for ordered trees.

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[R00, p.145].

$$C_n = \frac{1}{(n+1)}\frac{2n!}{n!n!}$$

† 群馬大学工学部情報工学科 〒 376–8515 群馬県桐生市天神町 1-5-1.
†Department of Computer Science, Gunma University, 1–5–1 Tenjin-Cho, Kiryu, 〒 376–8515

For example, the number of ordered trees with four vertices is $C_{4-1} = 5$, as depicted in Fig. 1. We need at least $\log C_{n-1}$ bits to code those ordered trees with n vertices. Because the Catalan number can be denoted as follows[G94, p495],

$$C_n = \frac{4^n}{(n+1)\sqrt{\pi n}} \left(1 - \frac{1}{8n} + \frac{1}{128n^2} + \frac{5}{1024n^3} - \frac{21}{32768n^4} + O(n^{-5})\right)$$

we need at least $\log C_{n-1} = 2n - o(n) = 2m - o(n)$ bits to code a ordered tree with n vertices. So the naive coding using $2m$ bits for each tree is asymptotically optimal.

In this paper we wish to code floorplans with a small number of bits.

A floorplan is a partition of a rectangle into a set of rectangles. For example, all floorplans with three faces are depicted in Fig. 2.

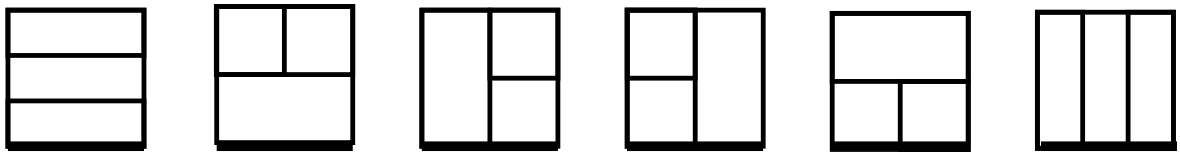

Figure 2: Floorplans with three faces.

A naive coding of floorplans needs $2m$ bits for each floorplan, as we explain in Section 3. In this paper we give a new simple coding of floorplans, which needs only $5m/3$ bits for each floorplan.

Note that we cannot treat floorplans simply as plane graphs. See two floorplans in Fig. 3. They are identical as plane graphs, however different as floorplans. Because in Fig. 3(a) the two faces F_a and F_b shares a horizontal line, however in Fig. 3(b) they shares a vertical line. We need to store the direction (horizontal or vertical) for each edge in a given floorplan.

It is interesting that we need less bits for floorplans than ordered trees.

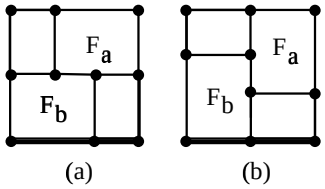

Figure 3: Two floorplans corresponding to the same plane graph.

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 explains the naive coding using $2m$ bits for each floorplan. Section 4 introduces "the removing sequence" of a floorplan, which is the main idea of our coding. Section 5 gives our new coding using $5m/3$ bits for each floorplan. Finally Section 6 is a conclusion.

2 Preliminaries

In this section we give some definitions.

Let G be a connected graph. A *tree* is a connected graph with no cycle. A *rooted* tree is a tree with one vertex r chosen as its *root* . A *ordered* tree is a rooted tree with fixed orderings for siblings.

A drawing of a graph is *plane* if it has no two edges intersect 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*. Two faces F_1 and F_2 are *ns-adjacent* (north-south adjacent) if F_2 is located to the bottom of F_1 and they share a horizontal line segment. Two faces F_1 and F_2 are *ew-adjacent* (east-west adjacent) if F_2 is located to the left of F_1 and they share a vertical line segment.

A *floorplan* is a plane drawing in which every face (including the outer face) is a rectangle. A *based* floorplan is a floorplan with one designated bottom line segment on the contour of the outer face. The designated bottom line segment is called *the base*, and we always draw the base as the lowermost line segment of the drawing. For examples, all based floorplans with three faces are shown in Fig. 2, in which each base is depicted by a thick line. If two based floorplans P_1 and P_2 have a one-to-one correspondence between faces preserving ns- and ew-adjacency, and in which each base corresponding to the other, then we say P_1 and P_2 are isomorphic.

Let n be the number of vertices of a floorplan, m the number of edges, and f the number of faces (including the outer face). In this paper we only consider based floorplans having no vertex shared by four (or more) rectangles. Thus G has $n-4$ vertices with degree three, and 4 vertices with degree two (at the four corner of the outer face), and we have $2m = 3(n-4) + 2 \cdot 4$. The equation and the Euler's formula $n - m + f = 2$ gives $n = 2f$ and $m = 3f - 2$.

A vertex with degree three is *w-missing* (west missing) if it has edges to top, bottom and right. We denote the number of w-missing vertices as n_W. Similarly we define *e-missing* (east missing), *n-missing*, *s-missing*, n_E, n_N and n_S. Note that, since each w-missing vertex is the left end of a maximal horizontal line segment, and each e-missing vertex is the right end of a maximal horizontal line segment, $n_W = n_E$ holds in any floorplan. Similarly $n_N = n_S$ holds. Thus $n_W + n_N = (n-4)/2$.

An inner face F of a floorplan is *U-active* if F shares a line segment with the uppermost horizontal line segment of the contour of the outer face. Intuitively, a face is U-active if it touches the uppermost line segment. For convenience, we regard the outer face also as U-active.

A face F is *Uw-active* if F has a U-active face to the left with sharing a vertical line segment. Intuitively, a face is Uw-active if it touch some U-active face at the left (or west) boundary. Note that "some U-active face" may be the outer face.

3 The Naive Coding

In this section we sketch a naive code for floorplans, based on the depth first search of a tree. The code needs $2m + 3$ bits for each floorplan.

Given a based floorplan R, we replace each lower right corner vertex of each inner face with two vertices as depicted in Fig. 4. See an example in Fig. 6. Note that since we only break each cycle corresponding to an inner face at the lower right corner, the resulting graph has only one face and is still connected. So the resulting graph R' is a tree. Also note that we need some tricky replacement at the two lower corners of the outer face. See Fig. 4.

Starting at the upper left corner, we traverse the tree R' with depth first manner (with left priority). When we visit a vertex, we have only two choices for the direction of the next vertex, as shown in Fig. 5. Since each edge has traced exactly twice, we need two bits for each edge, except for the first edge, which need only one bit because we always trace the first edge to bottom, and since we have two dummy edges at the two lower corners, we can code the tree R' with $2m + 3$ bits. Given the $2m + 3$ bits code we can easily reconstruct the original floorplan R with a simple algorithm with a stack.

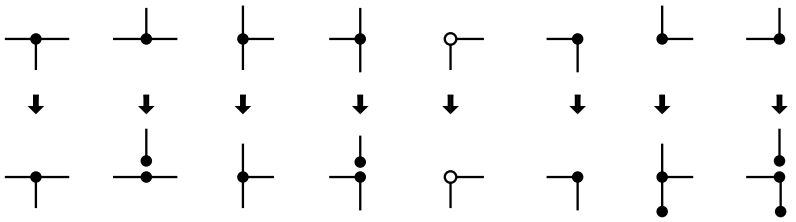

Figure 4: The replacement of vertices.

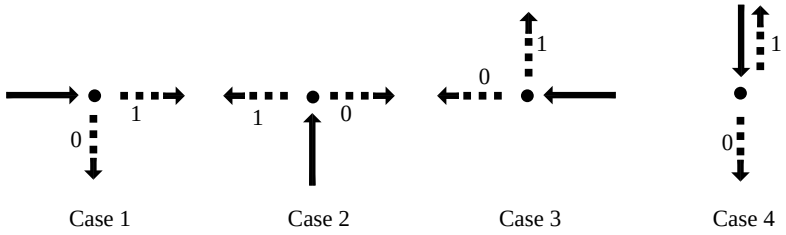

Figure 5: The code for the DFS.

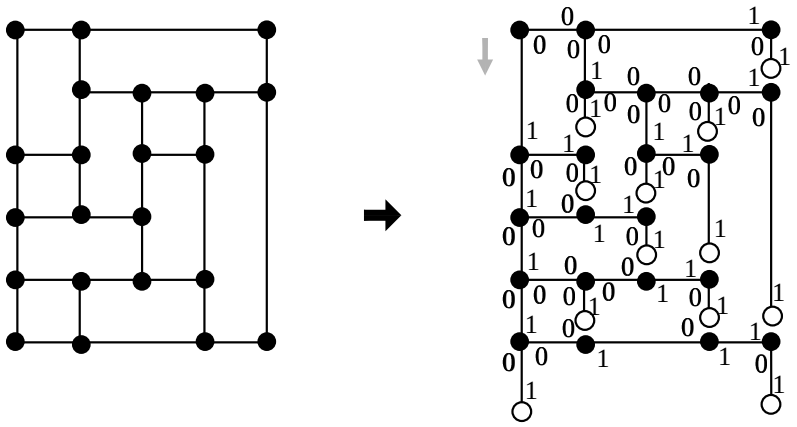

Figure 6: The $2m + 3$ bit code for a floorplan based on the DFS.

4 The Removing Sequence

Let R_1 be the based floorplan having exactly one inner face. Assume that $R(\neq R_1)$ is a based floorplan having $f > 1$ faces.

Let F be the inner face of R having the upper-left corner of the outer face of R. We call such a face *the first* face of the based floorplan R. The first faces of based floorplans are shaded in Figs. 7– 9. Let v be the lower-right corner of F. The first face F is *upward removable* if R has a vertical line segment with upper end v. See Fig. 7 (a). Otherwise, R has a horizontal line segment with left end v, and the first face F is *leftward removable*. See Fig. 7 (b). Note that we have assumed that R has no vertex with degree four.

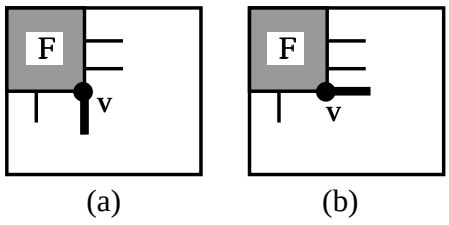

Figure 7: (a) An upward removable face and (b) a leftward removable face.

Since $R \neq R_1$, the first face of R is either upward removable or leftward removable. If F is upward removable, then we can obtain a floorplan with one less faces by continually shrinking the first face F into the uppermost horizontal line of R, with preserving the width of F and enlarging the faces below F, as shown in Fig. 8. Similarly, if F is leftward removable, then we can obtain a floorplan with one less faces by continually shrinking F into the leftmost line of R with preserving the height of F. After we remove the first face from R, the resulting floorplan is again a based floorplan with one less faces. We

denote such a floorplan as $P(R)$. Thus we can define the based floorplan $P(R)$ for each based floorplan $R \neq R_1$.

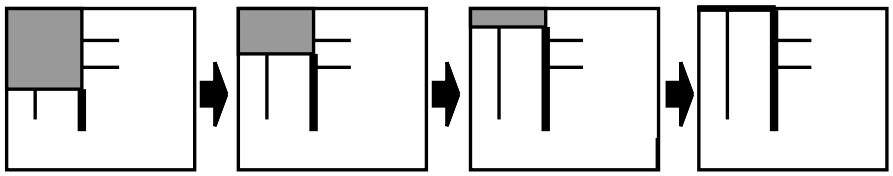

Figure 8: Removing the first face.

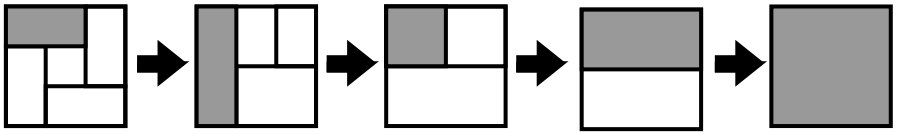

Figure 9: The removing sequence.

Given a based floorplan R having exactly f faces, by repeatedly removing the first face, we can have the unique sequence $R, P(R), P(P(R)), \cdots$ of based floorplans which eventually ends with R_1, which is the based floorplan having exactly one inner face. See an example in Fig. 9, in which the first faces are shaded. We call the sequence $R, P(R), P(P(R)), \cdots$, *the removing sequence* of R.

Let R_k be a floorplan with $k \leq f$ faces. Assume that the first face of R_k has $s(R_k)$ neighbor faces to the bottom and $e(R_k)$ neighbor faces to the right. Given $R_{k-1} = P(R_k)$, if we know (1) whether the first face of R_k is upward removable or leftward removable, and (2) the two values $s(R_k)$ and $e(R_k)$, then we can reconstruct R_k from those information. Thus, for each $k = 2, 3, \cdots, n$, if we store (1) whether the first face of R_k is upward removable or leftward removable, and (2) the two values $s(R_k)$ and $e(R_k)$, then we can reconstruct $R_2, R_3 \cdots, R = R_f$.

A simple coding needs $f - 1$ bits for (1), and $2(f-1)\log f$ bits for (2). In the next section we give more efficient coding for floorplans. The coding needs only $5m/3$ bits for each floorplan.

5 Our Coding

In this section we give a simple coding for based floorplans. The coding needs only $5m/3$ bits for each floorplan.

Basically, given a floorplan R with f faces, we code R as the removing sequence of R.

Let $RS = (\ R_f(=R),\ R_{f-1}(=P(R)),\ R_{f-2}(=P(P(R))), \cdots,\ R_1\ )$ be the removing sequence of R. We construct two new sequences from RS as follows.

By choosing the based floorplans having upward removable first faces from RS, with preserving the order, we derive a new sequence $RS_U = (R_1^U, R_2^U, \cdots, R_{f_U}^U)$. Similarly, by choosing the based floorplans having leftward removable first faces from RS, we derive a new sequence $RS_L = (R_1^L, R_2^L, \cdots, R_{f_L}^L)$. Note that $f_U + f_L + 1 = f$ holds, since R_1 is contained in neither RS_U nor RS_L.

The coding consists of the following five sections.

Section 1: This section codes whether the first face of each $R_k, k = 2, 3, \cdots, f$ is upward removable or leftward removable. For $k = 1, 2, \cdots, f-1$, the k-th bit is 0 if the first face of R_{k+1} is upward removable, and 1 otherwise. Section 1 has lenght $f - 1$ bits in total.

Section 2: Section 2 and 5 code each $s(R_k),\ k = 2, 3, \cdots, f$. If $R_k \in RS_U$, then we code $s(R_k)$ in Section 2, otherwise $R_k \in RS_L$, and we code $s(R_k)$ in Section 5. Section 2 has lenght $f - 1$ bits in total.

Assume that $R_k = R_j^U \in RS_U$. Now the first face F of R_k is upward removable.

Let $s(R_k)$ be the number of faces which are U-active in R_{k-1}, but not U-active in R_k. That is the number of faces which are located to the bottom of F, and become U-active when we remove F. Note that $s(R_k) \geq 1$, since the first face F always has some face (possibly the outer face) to the bottom.

Also note that $s(R_1^U) + s(R_2^U) + \cdots + s(R_{f_U}^U) \leq f - 1$, since each face becomes U-active exactly once, and R_f has at least one face which is already U-active. We can observe that when we remove a leftward removable face, no face newly becomes U-active.

We code each $s(R_j^U)$, $1 \leq j \leq f_U$, as the consecutive $s(R_j^U) - 1$ copies of "0"s followed by a "1". (Note that, as we mentioned above, $s(R_j^U) \geq 1$ holds for each k.) For example, we code $s(R_j^U) = 5$ as 00001. After we concatenate those codes, we finally append zeroes so that the length of Section 2 becomes $f - 1$ bits in total.

We can easily decode each $s(R_k)$ from the code.

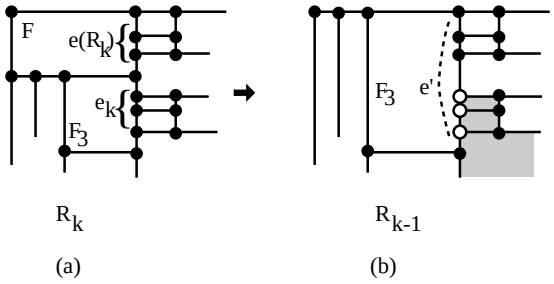

Figure 10: An illustration for Section 3.

Section 3: Section 3 and 4 code each $e(R_k), k = 2, 3, \cdots, f$. If $R_k \in RS_U$, then we code $e(R_k)$ in Section 3, otherwise $R_k \in RS_L$, and we code $e(R_k)$ in Section 4.

We don't directly code each $e(R_k)$, since the sum of them may be large. Our idea is as follows. See Fig. 10(a). The first face of R_k has three $= s(R_k)$ neighbors to the bottom, and three $= e(R_k)$ neighbors to the right. Let $F_1, F_2, \cdots, F_{s(R_k)}$ be the faces located to the bottom of F. We are going to code the number $e(R_k) = 3$.

Given R_{k-1}, we know the number e' of faces located to the right of $F_{s(R_k)}$. In the example of Fig. 10(b), $e' = 6$. Let e_k be the number of faces which are located to the right of $F_{s(R_k)}$, and Uw-active in R_{k-1}, but not Uw-active in R_k. Those faces are shaded in Fig. 10(b). We can observe that each of those faces has a w-missing vertex at the upper left corner in R. (This is our idea to bound the length of Section 3 by n_W.) Those vertices are depicted as white circle in Fig. 10(b). If we have R_{k-1} and $s(R_k)$, then we can count the value of e'. Now we can compute $e(R_k)$ by $e' - e_k$, if we have e_k. Thus we can store e_k instead of $e(R_k)$.

Now we formally explain the code.

Assume that $R_k = R_j^U \in RS_U$. Now the first face F of R_k is upward removable. See Fig. 10(a).

We code each e_k as the consecutive e_k copies of "0"s followed by a "1". By concatenating those codes, we have the code for section 3. We can easily decode each e_k, and then compute $e(R_k)$.

Note that $e_2 + e_3 + \cdots + e_f \leq n_W$, since each face becomes Uw-active exactly once, and R_f has at least one face which is already Uw-active. Similar to Section 2, we finally append zeroes so that the length of Section 3 becomes $n_W + f_U$ bits in total

Section 4: This section codes each $e(R_k), k = 2, 3, \cdots, f$ only for each $R_k \in RS_L$. (We code each $R_k \in RS_U$ in Section 3.)

Omitted. Similar to Section 2. Section 4 has length $f - 1$ bits in total.

Section 5: This section codes each $s(R_k), k = 2, 3, \cdots, n$ only for each $R_k \in RS_L$. (For $R_k \in RS_U$ we code each $s(R_k)$ in Section 2.)

Omitted. Similar to Section 3.

Section 5 has length $n_N + f_L$ bits in total.

Note that $n_W + n_N = (n-4)/2 = (f-2)$ holds. Thus the total length of the code consisting of the five sections above is

$$\begin{aligned}(f-1) + (f-1) + (n_W + f_U) + (f-1) + (n_N + f_L) \\ = (4f-4) + (f-2) = 5f - 6 \\ = \frac{5(m+2)}{3} - 6 = \frac{5m-8}{3}\end{aligned}$$

We have the following theorem.

Theorem 5.1 *One can encode a floorplan with $5m/3$ bits.*

6 Conclusion

In this paper we gave a simple and short coding for floorplans. The coding needs only $5m/3$ bits for each floorplan.

An efficient enumeration algorithm for based floorplans is known[N02a, N02b]. Let N_k be the number of based floorplans with k faces. By implementing the algorithm we have counted $N_{11} = 10948768$. Thus we need at least $24 > \log N_{11} = 23.5$ bits to code based floorplans with 11 faces, while the total length of our coding is $5 \cdot 11 - 6 = 49$ bits. Thus there are still many chances to reduce the length of the code.

For $N_{12} = 89346128$ we need at least $27 > \log N_{12} = 26.4$ bits to code based floorplans with 12 faces, while our coding needs $5 \cdot 12 - 6 = 54$ bits.

References

[G94] R. L. Graham, D. E. Knuth and O. Patashnik, Exercise 9.60 in *Concrete Mathematics, 2nd ed.*, Addison-Wesley, (1994).

[C01] Y.-T. Chiang, C.-C. Lin and H.-I Lu, *Orderly Spanning Trees with Applications to Graph Encoding and Graph Drawing*, Proc. of 12th Annual ACM-SIAM Symposium on Discrete Algorithms, (2001), pp.506–515.

[C98] R. C.-N Chuang, A. Garg, X. He, M.-Y. Kao, H.-I Lu, *Compact Encodings of Planar Graphs via Canonical Orderings and Multiple Parentheses*, Proc. of 25th International Colloquium on Automata, Languages, and Programming, LNCS 1443, (1998), pp.118–129.

[K95] K. Keeler and J. Westbrook, *Short Encodings of Planar Graphs and Maps*, Discrete Applied Mathematics, 58, No 3, (1995), pp.239–252.

[M97] J. Ian Munro and Venkatesh Raman, *Succinct Representation of Balanced Parentheses, Static Trees and Planar Graphs*, Proc. of 38th IEEE Symposium on Foundations of Computer Science, (1997), pp.118–126.

[N02a] S. Nakano *Enumerating Floorplans with n Rooms*, IEICE TRANS. FUNDAMENTALS, Vol.E85-A, No. 7, (2002), pp.1746–1750.

[N02b] S. Nakano, *Enumerating Floorplans with Some Properties*, Interdisciplinary Information Sciences, Vol. 8, No. 2, (2002), pp.199-206.

[P85] C. Papadimitriou and M. Yannakakis, *A Note on Succinct Representations of Graphs*, Information and Computation, Vol. 71, (1985), pp.181–185.

[R00] K. H. Rosen (Eds.), *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, Boca Raton, (2000).

[T84] G. Turan, *Succinct Representations of Graphs*, Discrete Applied Math, Vol. 8, (1984), pp.289–294.