

# **IEICE** **TRANSACTIONS**

## **on Information and Systems**

DOI:10.1587/transinf.2024FCP0001

Publicized:2024/05/30

This advance publication article will be replaced by  
the finalized version after proofreading.



**A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY**

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

## PAPER

## Enumerating floorplans with Aligned Columns

Shin-ichi NAKANO<sup>†</sup>, *Member*

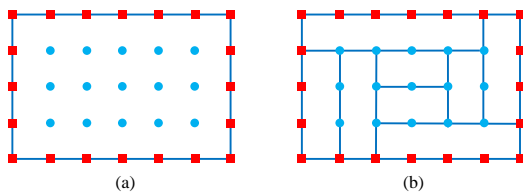
**SUMMARY** A floorplan is a partition of an axis-aligned rectangle into a set of smaller rectangles. Given an axis-aligned rectangle  $R$  and a set  $P$  of points in  $R$  we wish to partition  $R$  into a set  $S$  of rectangles so that each point in  $P$  is on a boundary of a rectangle in  $S$ . We call such a partition of  $R$  a floorplan covering  $P$ . Intuitively  $P$  is the locations of columns and a floorplan covering  $P$  is a floorplan in which no column is in the proper inside of a room and each column is on a wall.

In this paper we design an algorithm to generate all floorplans covering  $P$  when  $P$  is the set of given grid points. The algorithm generates each floorplan in  $O(|P|)$  time.

**key words:** Algorithm, Enumeration, Floorplan

## 1. Introduction

Given an axis-aligned rectangle  $R$  and a set  $P$  of points in  $R$  we wish to partition  $R$  into a set  $S$  of smaller rectangles so that each point in  $P$  is on a boundary of a rectangle in  $S$ . We call such a partition of  $R$  a floorplan covering  $P$ . See an example in Fig.1. Intuitively,  $P$  is the locations of columns and a floorplan covering  $P$  is a floorplan in which no column is in the proper inside of a room, i.e., each column is located on a wall between rooms.



**Fig. 1** (a) An example of the set of given grid points  $P$  and (b) a floorplan covering  $P$ .

Some efficient algorithms to enumerate all floorplans covering  $P$  is known [1], [2], [11], [13]. Let  $S_P$  be the set of the floorplans covering  $P$ . The fastest algorithm is based on the reverse search method [3], [4], and enumerates all floorplans covering  $P$  in  $O(|S_P|)$  time [11], [13]. In those papers  $P$  is in a general position, that is, no two points in  $P$  have the same  $x$ -coordinate, and no two points in  $P$  have the same  $y$ -coordinate.

In this paper we design an enumeration algorithm when  $P$  is the set of given grid points in  $R$ , as shown in Fig. 1(a).

Now  $P$  is not in a general position so we need more cases. However this is more practical situation in modern architecture. See an example in Fig. 1(a). We assume that the boundary of  $R$  also has the grid points. Our algorithm generates all floorplans covering  $P$  in  $O(|P|)$  time for each.

Ackerman et al. [1], [2] gave an algorithm to enumerate all floorplans covering  $P$  in a general position. The algorithm is based on the reverse search method [3], [4] and enumerates all such floorplans in either  $O(n|S_P|)$  time using  $O(n)$  space or  $O(\log n|S_P|)$  time using  $O(n^3)$  space, where  $n = |P|$ . Yamanaka et al. [11], [13] gave a faster algorithm, which is also based on the reverse search method. The algorithm uses  $O(n)$  space, and enumerates all such floorplans in  $(|S_P|)$  time.

Several variants of floorplan enumeration algorithms are also known [5]–[7], [10], [12]. See the brief survey in Section 1 of [5].

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 defines a tree structure among the floorplans covering  $P$ . Section 4 gives our enumeration algorithm using the tree structure. Finally Section 5 is a conclusion.

## 2. Preliminaries

In this section we give some definitions.

A *floorplan* is a partition of an axis-aligned rectangle  $R$  into a set of smaller rectangles. Each line segment of the boundary of  $R$  is called *the outer wall*, each smaller rectangle in the set is called a *room*, and each line segment in a floorplan is called a *wall*. A wall is either horizontal or vertical.

Let  $P$  be the set of given grid points located in  $R$ . See an example in Fig. 1(a). We assume that the boundary of  $R$  also has the grid points. A floorplan *covers*  $P$  if every point in  $P$  is in a wall. A wall containing exactly two points of  $P$  (on its two end points) is called a *basic wall*. A basic wall  $s$  in a floorplan covering  $P$  is *redundant* if removing  $s$  results in a floorplan covering  $P$ . We assume that a floorplan covering  $P$  has no redundant wall. (Because otherwise we need to treat a huge number of almost similar floorplans.) In a floorplan covering  $P$ , each room has either the width one or the height one, since otherwise the room has a point in  $P$  in the proper inside of the room, a contradiction.

<sup>†</sup>The author is with Gunma University, Maebashi-Shi, 371-8510

### 3. Family Tree

In this section we define a tree structure among the floorplans covering  $P$ , where  $P$  is the set of given grid points.

Let  $f_r$  be the floorplan covering  $P$  by only horizontal walls except the outer walls. See an example in Fig.5, where the top floorplan is  $f_r$ .

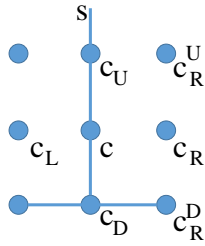


Fig.2 Illustration for  $c, c_D$ , etc.

Given a floorplan  $f \neq f_r$  covering  $P$ , we define the *parent floorplan*  $p(f)$  of  $f$ , which also covers  $P$ , so that the number of horizontal basic walls of  $p(f)$  is always more than that of  $f$ . We need some definitions.

Let  $s$  be the lowest maximal vertical wall among the leftmost maximal vertical walls in  $f$  except the (left) outer wall. We call such  $s$  the *critical wall* of  $f$ . Let  $c_D$  be the lower end point of  $s$ ,  $c$  be the upper neighbor point of  $c_D$ ,  $c_R$  be the right neighbor point of  $c$ ,  $c_R^U$  be the upper neighbor point of  $c_R$ ,  $c_R^D$  be the lower neighbor point of  $c_R$ ,  $c_U$  be the upper neighbor point of  $c$ , and  $c_L$  be the left neighbor point of  $c$ . (See Fig. 2.) Note that, since the face below  $c_D$  is also a rectangle room or the outer face,  $c_D$  has horizontal walls on the both sides. Also note that,  $s$  contains two or more basic walls. (Otherwise if  $s$  contains exactly one basic wall then it is redundant so it has been removed, a contradiction.)

We have the following two cases depending the number of basic walls in  $s$  for the definition of the parent floorplan  $p(f)$  of  $f$ . Recall that  $s$  is the critical wall of a floorplan  $f \neq f_r$ .

**Case 1:**  $s$  contains three or more basic walls.

In this case  $p(f)$  is derived from  $f$  by removing the lowest basic vertical wall from  $s$  with some modification. After the removal of the basic vertical wall between  $c$  and  $c_D$ , we need horizontal walls on the both sides of  $c$ , since the face below  $c$  is a rectangle room. When we append a basic horizontal wall on the right of  $c$  if each basic vertical wall of  $c_R$  becomes redundant then we remove it. On the other hand when we append a basic horizontal wall on the left of  $c$  no redundant vertical wall occurs around  $c_L$  since  $s$  is the leftmost vertical wall.

When we append a basic horizontal wall on the right of  $c$  if  $f$  has basic vertical walls on the both sides of  $c_R$  then either  $c_R$  has no horizontal basic wall on the right or the right neighbor of  $c_R$  is not on the right outer wall, since otherwise

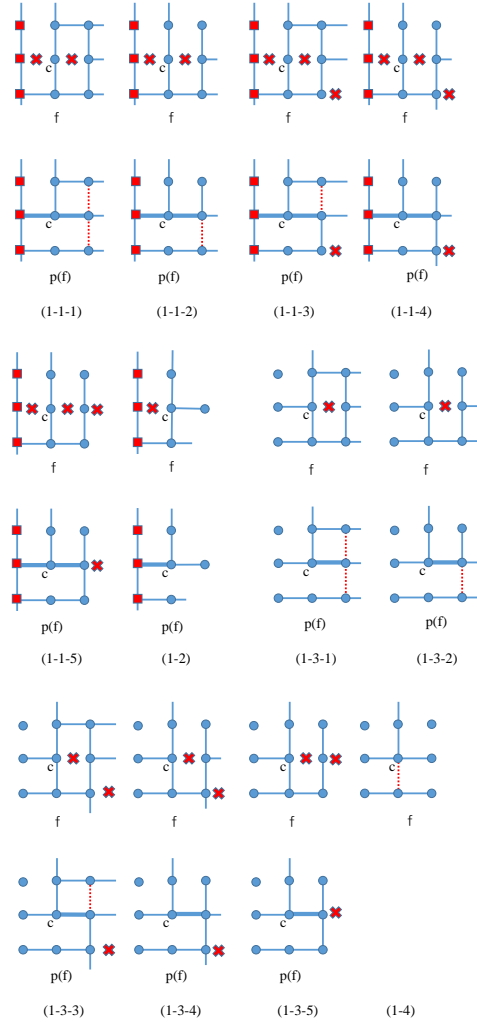


Fig.3 Illustration for the parent floorplan in Case 1.

in  $f$  the horizontal basic wall on the right of  $c_R$  is redundant.

We have the following four subcases depending the horizontal walls at  $c$ .

**Case 1-1:**  $c$  has horizontal wall on neither side.

Note that, since the face on the right of  $c$  is also a rectangle room,  $c_R$  has vertical walls on the both sides. Also note that, since the face on the left of  $c$  is also a rectangle room,  $c_L$  has vertical walls on the both sides, and by the choice of  $s$ ,  $c_L$  is on the left outer wall.

We have the following five subsubcases.

**Case 1-1-1:**  $c_R^U$  has horizontal walls on the both sides,  $c_R^D$  has horizontal walls on the both sides, and  $c_R$  has a horizontal wall on the right (and it is not redundant). (See Fig. 3 (1-1-1).)

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$ , (2) appending basic horizontal walls on the both sides of  $c$ , (3) removing the basic vertical wall above  $c_R$  since it is redundant, and (4) removing the basic vertical wall below  $c_R$  since it is redundant.

**Case 1-1-2:** Otherwise,  $c_R^D$  has horizontal walls on the both sides, and  $c_R$  has horizontal wall on the right (and it is not redundant). (See Fig. 3 (1-1-2).)

Now  $c_R^U$  has at most one horizontal wall since Case (1-1-1) does not occur.

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$ , (2) appending basic horizontal walls on the both sides of  $c$ , and (3) removing the basic vertical wall below  $c_R$  since it is redundant.

**Case 1-1-3:** Otherwise,  $c_R^U$  has horizontal walls on the both sides, and  $c_R$  has horizontal wall on the right (and it is not redundant).

Now  $c_R^D$  has no horizontal wall on the right since Case (1-1-1) does not occur. (See Fig. 3 (1-1-3).)

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$ , (2) appending basic horizontal walls on the both sides of  $c$ , and (3) removing the basic vertical wall above  $c_R$  since it is redundant.

**Case 1-1-4:** Otherwise,  $c_R$  has a horizontal wall on the right (and it is not redundant). (See Fig. 3 (1-1-4).)

Now  $c_R^U$  has at most one horizontal wall (since otherwise Case (1-1-3) occurs) and  $c_R^D$  has no horizontal wall on the right (since otherwise Case (1-1-2) occurs).

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$  and (2) appending basic horizontal walls on the both sides of  $c$ .

**Case 1-1-5:** Otherwise. ( $c_R$  has no horizontal wall on the right.) (See Fig. 3 (1-1-5).)

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$  and (2) appending basic horizontal walls on the both sides of  $c$ . Now each basic vertical wall containing  $c_R$  is not redundant.

**Case 1-2:**  $c$  has a horizontal wall on the right only. (See Fig. 3 (1-2).)

Note that, since the face on the left of  $c$  is also a rectangle room,  $c_L$  has vertical walls on the both sides, and by the choice of  $s$ ,  $c_L$  is on the left outer wall. Since  $c$  has a

basic horizontal wall on the right  $c_R$  has at most one vertical wall. (Otherwise the basic horizontal wall on the right of  $c$  is redundant.)

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$  and (2) appending a basic horizontal wall on the left of  $c$ .

**Case 1-3:**  $c$  has a horizontal wall on the left only. (See Fig. 3 (1-3-1)–(1-3-5).)

Note that, since the face on the right of  $c$  is also a rectangle room,  $c_R$  has vertical walls on the both sides. Also  $c_L$  is not on the left outer wall since otherwise the basic horizontal wall on the left of  $c$  is redundant, so it has removed, a contradiction.

Let  $p(f)$  be the floorplan derived from  $f$  by (1) removing the lowest basic vertical wall from  $s$  and (2) appending a basic horizontal wall on the right of  $c$ .

We have five subsubcases, similar to Case 1-1. (See Fig. 3 (1-3-1)–(1-3-5).)

**Case 1-4:**  $c$  has horizontal walls on the both sides. (See Fig. 3 (1-4).)

Then the basic vertical wall below  $c$  is redundant so it has been removed, a contradiction. Thus this case never occurs.

**Case 2:**  $s$  contains exactly two basic wall.

In Case 2,  $c_U$  has horizontal walls on the both sides. In this case  $p(f)$  is derived from  $f$  by removing the lowest basic vertical wall from  $s$  with some modification. After the removal of the basic vertical wall between  $c$  and  $c_D$ , we need horizontal walls on the both sides of  $c$ , since the face below  $c$  is a rectangle room. Additionally, in  $p(f)$  we always need to remove the basic vertical wall above  $c$ , since it is redundant. Also, when we append a basic horizontal wall on the right of  $c$  if each basic vertical wall of  $c_R$  becomes redundant then we remove it.

When we append a basic horizontal wall on the right of  $c$  if  $f$  has basic vertical walls on the both sides of  $c_R$  then either  $c_R$  has no horizontal basic wall on the right or the right neighbor of  $c_R$  is not on the right outer wall, since otherwise in  $f$  the horizontal basic wall on the right of  $c_R$  is redundant.

We have the following four subcases depending the horizontal walls at  $c$ .

**Case 2-1:**  $c$  has horizontal wall on neither side.

We have five subsubcases, similar to Case 1-1. (See Fig. 4 (2-1-1)–(2-1-5).)

**Case 2-2:**  $c$  has a horizontal wall on the right only. (See Fig. 4 (2-2).)

Similar to Case 1-2.

**Case 2-3:**  $c$  has a horizontal wall on the left only.

We have five subsubcases, similar to Case 1-3. (See Fig. 4 (2-3-1)–(2-3-5).)

**Case 2-4:**  $c$  has horizontal walls on the both sides. (See Fig. 4 (2-4).)

This case never occurs. Similar to Case 1-4.

We have defined the parent floorplan. In those cases the number of horizontal basic walls of  $p(f)$  is always

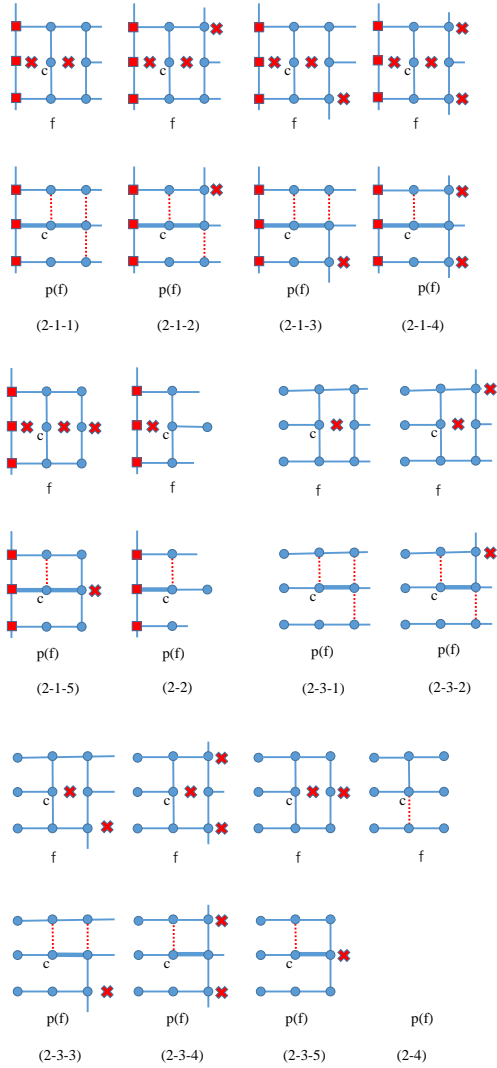


Fig. 4 Illustration for the parent floorplan in Case 2.

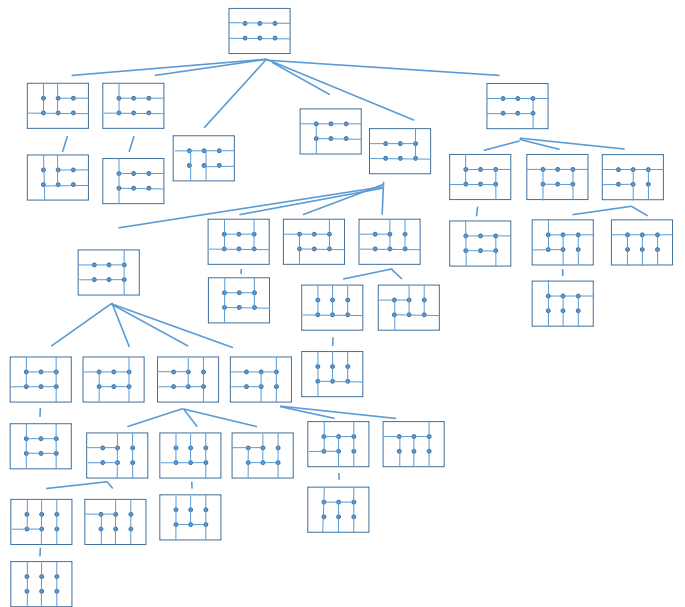


Fig. 5 An example of the family tree.

more than that of  $f$ . Given a floorplan covering  $P$ , by repeatedly computing the parent of the derived floorplan, say  $f, p(f), p(p(f)), \dots$ , we have the unique sequence of floorplans, called *the removing sequence* of  $f$ , and it always ends with  $f_r$ , which is the floorplan covering  $P$  with only horizontal walls.

Given the set  $S_P$  of floorplans covering  $P$ , by merging the removing sequences of the floorplans in  $S_P$ , we have the tree structure  $T_P$  of floorplans in  $S_P$ , in which the root corresponds to  $f_r$ , each vertex corresponds to a floorplan in  $S_P$ , each edge corresponds to each parent-child relation among the floorplans. See an example in Fig.5. The tree structure is called *the family tree* [9] of  $S_P$ .

If  $p(f)$  is the parent of  $f$  then we say  $f$  is a *child* of  $p(f)$ . If  $p(f)$  is derived from  $f$  by Case x above then we say  $f$  is a child floorplan of  $p(f)$  with Case x.

We have the following lemma.

**Lemma 1.** *In Case 1, the critical wall  $s$  of  $f$  remains the*

critical wall of  $p(f)$  after losing the lowest basic vertical wall.

In Case 2, the critical wall  $s$  of  $f$  is completely removed in  $p(f)$  and the critical wall of  $p(f)$  is located either on the right of (removed)  $s$  of  $f$  or on the vertical line containing (removed)  $s$  but above  $s$ . Especially, in Case 2-1-2, 2-1-3, 2-1-4, 2-1-5, 2-3-2, 2-3-3, 2-3-4, 2-3-5, the critical wall of  $p(f)$  is located on the vertical line containing either  $c_R$ , or (removed)  $s$  but above  $s$ .

*Proof.* By case analysis. See Fig.3 and Fig.4.  $\square$

#### 4. Algorithm

In this section we design an algorithm to generate all floorplans covering  $P$ , where  $P$  is the set of given grid points in  $R$ .

Given a floorplan  $f$  covering  $P$ , we design an algorithm to generate all child floorplans of  $f$ . Then, by recursively executing the all child floorplan generation algorithm, we generate all floorplans covering  $P$ . We construct the family tree in the depth first order and generate each floorplan corresponding to each vertex of the family tree.

The algorithm consists of the following three steps. For a point  $c \in P$  we again define  $c_R, c_L, c_U, c_D, c_R^D$  and  $c_R^U$  around  $c$  as in Fig. 2, but this time  $c$  may not be in the critical wall.

**(STEP 0)** A floorplan  $f$  covering  $P$  with  $f \neq f_r$  and the critical wall  $s$  of  $f$  is given. If  $f = f_r$  holds then we regard the right vertical outer wall of  $R$  as  $s$ .

**(STEP 1)(Case 1 child floorplan generation)** Let  $c$  be the lower end point of  $s$ . Now  $c$  has horizontal walls on the both sides. If  $c$  is not on the outer wall then  $c_D$  also has horizontal walls on the both sides.

If  $c$  is not on the outer wall then we generate all possible child floorplans of  $f$  with Case 1 using the reverse operation of Case 1. Each of which is a floorplan derived from  $f$  by extending  $s$  to down by one basic vertical wall with some modification around  $c$  and  $c_R$ . Otherwise ( $c$  is on the outer wall),  $f$  has no child floorplan with Case 1.

For each subcase  $x$  in Case 1, if the child floorplan  $f_c$  with Case  $x$  exists (if  $p(f_c) = f$  occurs) we perform the algorithm recursively for  $f_c$ .

The number of possible child floorplans is a constant and given  $s$  we can check the existence of each possible child floorplan in  $O(1)$  time. Thus we can generate all child floorplans of  $f$  with Case 1 from  $f$  in  $O(1)$  time in total.

**(STEP 2)(Case 2 child floorplan generation)** For each point  $c$  in  $P$  located either on the left of  $s$  and not on the outer wall, or on the line containing  $s$  but below  $s$  with  $c_U$  is not on  $s$ , we generate all possible child floorplans of  $f$  with Case 2 using the reverse operation of Case 2. Each of those is a floorplan derived from  $f$  by appending the basic vertical walls on the both sides of  $c$  with some modification around

$c$  and  $c_R$ . By the choice of  $c$ ,  $c$  has horizontal walls on the both sides,  $c_U$  has horizontal walls on the both sides,  $c_D$  has horizontal walls on the both sides and  $c$  has no vertical walls. For other  $c$ ,  $f$  has no child floorplan with Case 2.

For each  $c$  located either (1) on the left of  $s$  or (2) on the line containing  $s$  but below  $s$  with  $c_U$  is not on  $s$ , for each subcase  $x$  in Case 2, if the child floorplan  $f_c$  with Case  $x$  with respect to  $c$  exists (if  $p(f_c) = f$  occurs) we perform the algorithm recursively for  $f_c$ .

The number of possible child floorplans is a constant and given  $s$  and  $c$  we can check the existence of each possible child floorplan in  $O(1)$  time. Thus we can generate all child floorplans of  $f$  with Case 2 from  $f$  in  $O(|P|)$  time in total.

In this way after generating  $f_r$  in  $O(|P|)$  time we can generate all floorplans covering  $P$  in  $O(|S_P||P|)$  time in total.

If we generate each floorplan in the depth first order in the family tree we may need much time to generate the next floorplan of the last floorplan in a large subtree. However, by the prepost order method [8], which generates each floorplan (Case ODD) before its child floorplans if the level of recursive call is odd and (Case EVEN) after its child floorplans otherwise, one can generate each floorplan in  $O(|P|)$  time, as the difference from the preceding floorplan. Note that by tracing at most three edges of the family tree we can output the next floorplan.

---

#### Algorithm 1 Enumerate-All-Child-Floorplans( $f,s,P$ )

---

```

1: if The level of recursive call is odd then
2:   Output  $f$ 
3: end if
4: /* (STEP 1) */
5: /* Let  $c$  be the lower end point of the critical wall  $s$  of  $f$  */
6: if  $c$  is not on the outer wall then
7:   for each subcase  $x$  in Case 1 do
8:     if The child floorplan  $f_c$  with Case  $x$  exists then
9:       Enumerate-All-Child-Floorplans( $f_c,s,P$ )
10:      /*  $s$  remains the critical wall of  $f_c$  */
11:     end if
12:   end for
13: end if
14: /* (STEP 2) */
15: for each  $c$  located either (1) on the left of  $s$  or (2) on the line containing
     $s$  but below  $s$  and  $c_U$  not on  $s$  do
16:   for each subcase  $x$  in Case 2 do
17:     if The child floorplan  $f_c$  with Case  $x$  with respect to  $c$  exists
       then
18:       Enumerate-All-Child-Floorplans( $f_c,s',P$ )
19:       /*  $s'$  is the critical wall of  $f_c$  consisting of the two vertical
         basic walls containing  $c$  */
20:     end if
21:   end for
22: end for
23: if The level of recursive call is even then
24:   Output  $f$ 
25: end if

```

---

We have the following theorem.

**Theorem 1.** One can generate all floorplans covering  $P$  in  $O(|P|)$  time for each, where  $P$  is the set of given grid points.

## 5. Conclusion

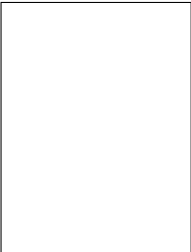
In this paper we have designed an algorithm to generate all floorplans covering  $P$  when  $P$  is the set of given grid points. The algorithm generates each floorplan covering  $P$  in  $O(|P|)$  time for each.

Can we design more efficient algorithms?

Can we efficiently generate all floorplans covering  $P$ , when  $P$  is any set of points in  $R$ ?

## References

- [1] E. Ackerman, G. Barequet and R. Y. Pinter, On the number of rectangulations, Proc. of SODA 2004, pp.729–738 (2004).
- [2] E. Ackerman, G. Barequet and R. Y. Pinter, On the number of rectangulations of a planar point set, Journal of Combinatorial Theory, Series A, 113, pp.1072-1091 (2006).
- [3] D. Avis, Generating rooted triangulations without repetitions, Algorithmica, 16, pp.618-632 (1996).
- [4] D. Avis and K. Fukuda, Reverse search for enumeration, Discrete Applied Mathematics, 65, pp.21-46 (1996).
- [5] A. I. Merino and T. Mütze: Efficient Generation of Rectangulations via permutation languages, Proc. of SoCG 2021, 54:1-18 (2021).
- [6] S. Nakano, Enumerating floorplans with  $n$  rooms, Proc. of ISAAC 2001, LNCS 2223, pp.107-115 (2001).
- [7] S. Nakano, Enumerating Floorplans with  $n$  Rooms. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E85-A, pp.1746-1750 (2002).
- [8] S. Nakano and T. Uno, Constant time generation of trees with specified diameter, Proc. of WG04, LNCS 3353, pp.33-45 (2004).
- [9] S. Nakano, Family trees for enumeration, International Journal of Foundations of Computer Science Accepted 2023.5.16
- [10] M. Takagi and S. Nakano, Listing all rectangular drawings with certain properties, Systems and Computers in Japan, 35, pp.1–8 (2004).
- [11] K. Yamanaka, Md. S. Rahman and S. Nakano, Floorplans with columns, Proc. of COCOA 2017, LNCS 10627, pp.33-40 (2017).
- [12] K. Yamanaka and S. Nakano, Floorplans with walls, Proc. of TAMC 2020, LNCS 12337, pp. 50-59 (2020).
- [13] K. Yamanaka, Md. S. Rahman and S. Nakano, Enumerating floorplans with columns, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E101-A, pp.1392-1397 (2018).



**Shin-ichi Nakano** received his B.E. and M.E. degrees from Tohoku University, Sendai, Japan, in 1985 and 1987, respectively. In 1987 he joined Seiko Epson Corp. and in 1990 he joined Tohoku University. In 1992, he received Dr. Eng. degree from Tohoku University. Since 1999 he has been a faculty member of Gunma University. His research interests are algorithms. He is a member of IPSJ and ACM.