Enumerating Floorplans with Columns*

Katsuhisa YAMANAKA†, Member, Md. Saidur RAHMAN††, Nonmember, and Shin-ichi NAKANO†††, Member

SUMMARY Given an axis-aligned rectangle $R$ and a set $P$ of $n$ points in the proper inside of $R$ we wish to partition $R$ into a set $S$ of $n+1$ rectangles so that each point in $P$ is on the common boundary between two rectangles in $S$. We call such a partition of $R$ a feasible floorplan of $R$ with respect to $P$. Intuitively, $P$ is the locations of columns and a feasible floorplan is a floorplan in which no column is in the proper inside of a room, i.e., columns are allowed to be placed only on the partition walls between rooms. In this paper we give an efficient algorithm to enumerate all feasible floorplans of $R$ with respect to $P$. The algorithm is based on the reverse search method, and enumerates all feasible floorplans in $O(|S_P|)$ time using $O(n)$ space, where $S_P$ is the set of the feasible floorplans of $R$ with respect to $P$, while the known algorithms need either $O(n|S_P|)$ time and $O(n)$ space or $O(|S_P|)$ time and $O(n^2)$ space.

key words: enumeration, floorplan, algorithm

1. Introduction

Given an axis-aligned rectangle $R$ and a set $P$ of $n$ points in the proper inside of $R$ we wish to partition $R$ into a set $S$ of $n+1$ rectangles so that each point in $P$ is on the common boundary between two rectangles in $S$. We call such a partition of $R$ a feasible floorplan of $R$ with respect to $P$. Figure 1(b) illustrates the 22 feasible floorplans of $R$ with respect to the point set $P$ in Fig. 1(a). For simplicity we assume no two points have the same $x$-coordinate, and no two points have the same $y$-coordinate. Intuitively, $P$ is the locations of columns and a feasible floorplan is a floorplan in which no column is in the proper inside of a room, i.e., columns are allowed to be placed only on the partition walls between rooms.

In this paper, we consider the problem of enumerating all feasible floorplans. If we can enumerate all the floorplans, the best floorplan for some criteria can be obtained. There are existing results for enumerating floorplans with/without some properties [6], [8], [10].

Ackerman et al. [1], [2] gave an algorithm to enumerate all feasible floorplans with respect to $P$. The algorithm is based on the reverse search method [3], [4] and enumerates all feasible floorplans in either $O(n|S_P|)$ time using $O(n)$ space or $O(|S_P|)$ time using $O(n^2)$ space, where $S_P$ is the set of feasible floorplans with respect to $P$.

In this paper we design a faster algorithm, which is also based on the reverse search method. Our algorithm uses only $O(n)$ space, and enumerates all feasible floorplans in $O(|S_P|)$ time. Using a similar method we have designed efficient enumeration algorithms [5]–[7].

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 defines a tree structure among the feasible floorplans. Section 4 gives our enumeration algorithm. Finally, Sect. 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 $S$ of rectangles. In this paper we have a typical assumption for floorplans. We assume that no four rectangles in $S$ share a common corner point in a floorplan. We call $R$ the outer rectangle and each rectangle in $S$ a face. Given an axis-aligned rectangle $R$ and a set $P$ of $n$ points in $R$, a feasible floorplan of $R$ with respect to $P$ is a partition of $R$ into a set $S$ of $n+1$ rectangles (or faces) so that each point in $P$ is on the common boundary between two rectangles (or faces) in $S$. We assume that no two points have the same location.

---

* A preliminary version of this paper was presented at The 11th Annual International Conference on Combinatorial Optimization and Applications.

a) E-mail: yamanaka@cis.iwate-u.ac.jp
b) E-mail: dmsrahman@gmail.com
c) E-mail: nakano@cs.gunma-u.ac.jp

DOI: 10.1587/transfun.E101.A.1392
x-coordinate, and no two points have the same y-coordinate. Then one can observe that every maximal line segment not on $R$ contains exactly one point in $P$ in any feasible floorplan of $R$ with respect to $P$. Let $S_P$ be the set of all feasible floorplans with respect to $P$. For the rectangle $R$ and the point set $P$ in Fig. 1(a), there are 22 feasible floorplans, as illustrated in Fig. 1(b), and thus $|S_P| = 22$ holds.

Let $Q$ be a feasible floorplan of $R$ with respect to $P$. A maximal line segment containing no end point of another line segment is called a basic line segment. Each maximal line segment consists of one or more basic line segments. A maximal vertical line segment containing a point $p \in P$ is type$(u,d)$ if it contains $u$ endpoints of maximal horizontal line segments above $p$ and $d$ endpoints of maximal horizontal line segments below $p$. Thus a vertical basic line segment is type$(0,0)$.

3. Family Tree

In this section, we define a rooted tree structure among the feasible floorplans of $R$ with respect to $P$. First, we define a “root” feasible floorplan which corresponds to the root in the tree structure. Next, we define a “parent” feasible floorplan for any feasible floorplan in $S_P$. Finally, using the parent feasible floorplans, we give a rooted tree structure among $S_P$.

A root feasible floorplan of $R$ with respect to $P$, denoted by $Q_r$, is the feasible floorplan such that every point in $P$ is on a horizontal line segment. For instance, if we are given a point set in Fig. 1(a), the root feasible floorplan is the feasible floorplan in the upper-leftmost one in Fig. 1(b).

Now, to define a tree structure among $S_P$, for any feasible floorplan $Q \neq Q_r$ in $S_P$, we define the parent feasible floorplan $P(Q)$, which is also a feasible floorplan in $S_P$, as follows. Let $s$ be the leftmost maximal vertical line segment in $Q$ except the left vertical line segment of $R$, and $p \in P$ be the point on $s$. We have the following two cases to consider.

Case 1: $s$ is type$(0,0)$.

In this case we (1) remove $s$ from $Q$ and then (2) append a horizontal line segment containing $p$ as a horizontal basic line segment, as illustrated in Fig. 2. Intuitively, this operation is a rotation of $s$.

Case 2: Otherwise.

In this case we (1) remove $s$ from $Q$, (2) extend to left each maximal horizontal line segment having its left end on $s$ so that it has the same number of basic line segments as it was (intuitively, we extend the line segment until it touches a vertical one), (3) similarly extend to right each maximal horizontal line segment having its right end on $s$ so that it has the same number of basic line segments as it was, and finally (4) append a horizontal line segment containing $p$ as a horizontal basic line segment. Intuitively, this is a rotation of $s$ after shrinking. See Fig. 3 for an example.

In each case, we define the resulting feasible floorplan as the parent, denoted by $P(Q)$, of $Q$. We say that $Q$ is a child of $P(Q)$. Note that (1) $P(Q)$ is also a feasible floorplan in $S_P$, (2) the number of maximal vertical line segments of $P(Q)$ is decreased by one from the one of $Q$, and (3) the rotated line segment is always a basic one in $P(Q)$.

Now, we are ready to define a tree structure among $S_P$. Given a feasible floorplan $Q$ in $S_P$, by repeatedly computing its parent, we have a sequence of feasible floorplans in $S_P$. We call the sequence the removing sequence of $Q$. The removing sequence of $Q$ starts from $Q$ and eventually ends with $Q_r$, since (1) any feasible floorplan has its parent and (2) the number of vertical maximal line segment decreased by one. See an example of such sequence in Fig. 4.

By merging those sequences we define the family tree $T_P$ of $S_P$ such that the vertices of $T_P$ correspond to the feasible floorplans in $S_P$ and each edge corresponds to each relation between a floorplan in $S_P$ and its parent. See Fig. 5. Note that each vertex of depth $i$ in $T_P$ corresponds to a feasible floorplan with $i$ vertical line segments except the two vertical line segments on $R$, and the height of the family tree is $n$.

4. Algorithm

In this section, we design an algorithm to enumerate all
feasible floorplans of \( R \) with respect to \( P \). If we have an algorithm to compute all child floorplans of a given feasible floorplan in \( S_P \), then by recursively executing the algorithm from the root feasible floorplan \( Q_r \), we can traverse the family tree among \( S_P \), that is, we can compute all feasible floorplans in \( S_P \). We are now going to design such an algorithm.

Let \( Q \) be a feasible floorplan in \( S_P \). To generate a child floorplan of \( Q \), we do the “reverse operation” of the operation to define the parents. More precisely, we increase the number of vertical maximal line segments by one by rotating and extending a horizontal basic line segment.

Now, we introduce a notation to represent a “candidate” of a child of \( Q \). Let \( s' \) be the leftmost vertical line segment in \( Q \) except the left vertical line segment of \( R \), and \( p' \in P \) be the point on \( s' \). (Thus if \( Q = Q_r \) then \( s' \) is the right vertical line segment of \( R \), and we regard any point on \( s' \) as \( p' \).) Let \( s \) be a horizontal basic line segment in \( Q \) containing a point \( p \in P \). Let \( u' \) be the number of maximal horizontal line segments above \( p \), and let \( d' \) be the number of maximal horizontal line segments below \( p \). For example for the floorplan in Fig. 6, \( u' = 3 \) and \( d' = 2 \). For two integers \( u < u' \) and \( d < d' \), \( C(s, u, d) \) is the floorplan constructed from \( Q \) by (1) removing \( s \) from \( Q \) then (2) appending a vertical line segment \( s'' \) containing \( p \) as a basic line segment then (3) extending \( s'' \) upward and downward so that it becomes \( \text{type}(u, d) \) then (4) shrinking each maximal horizontal line segment intersecting with \( s'' \) so that it has an endpoint on \( s'' \). See examples in Fig. 6. Intuitively, \( C(s, u, d) \) is a candidate of a child floorplan derived from \( Q \) by a rotation and an extension of \( s \).

Now, let us explain conditions to characterize that \( C(s, u, d) \) is a child of \( Q \). Let \( s \) be a horizontal basic line segment in \( Q \). First, if \( s \) contains a point in \( P \) locating on the right of \( p' \) in \( Q \), \( C(s, u, d) \) is not a child floorplan of \( Q \), since the leftmost vertical line segment of \( C(s, u, d) \) is \( s' \). Thus, \( P(C(s, u, d)) \neq Q \) holds. Hence \( C(s, u, d) \) is not a child floorplan of \( Q \). In what follows, we assume that \( s \) is a horizontal basic line segment containing a point \( p \in P \) locating on the left of \( p' \). Next, we have the following observation from Fig. 7. In the figure, the horizontal line segment \( t \) immediately above \( p \) has a point of \( P \) on the left of \( p \) and consists of two or more (exactly, three in this case) basic line segments in \( Q \). We need to shrink \( t \) in \( C(s, 1, 0) \). For any suitable definition of \( C(s, 1, 0) \) each basic line segments of \( t \) except the leftmost one cannot exist in \( C(s, 1, 0) \) (as depicted as dashed line segments) since we have to cut \( t \). Thus, \( P(C(s, 1, 0)) \) is not \( Q \) and \( C(s, 1, 0) \) is not a child floorplan of \( Q \). The observations above give us the following lemma.

**Lemma 1:** Let \( s' \) be the leftmost vertical line segment in \( Q \) except the left vertical line segment of \( R \), and let \( p' \in P \) be the point on \( s' \). Let \( s \) be a horizontal basic line segment containing a point \( p \in P \) locating on the left of \( p' \). Then, \( C(s, u, d) \) of \( Q \) is a child floorplan of \( Q \) if and only if every shrunk horizontal line segment having a point of \( P \) on the left of \( p \) consists of exactly one basic line segment in \( Q \).

**Proof.** Let us consider the only-if part first. We assume that \( C(s, u, d) \) of \( Q \) is a child floorplan of \( Q \). Then, from the definition of the parent, it is easy to see that every shrunk horizontal line segment in \( C(s, u, d) \) is a basic one in \( Q \).
Algorithm 1: Enum-Floorplan(P)

1 Create the root feasible floorplan Q₀ of a given point set P.
2 Find-All-Children(Q₀)

Algorithm 2: Find-All-Children(Q)

1 Output Q
/* Let H be the set of horizontal basic line segments located in the left of the leftmost vertical line segment except the left vertical line segment of R. Let s be a horizontal basic line segment in H, and suppose that s contains a point p ∈ P. We denote by u' and d' be the numbers of maximal horizontal line segments in Q above p and below p, respectively. */
2 foreach s ∈ H do
3   for d = 0, 1, . . . , d' − 1 do
4     if (d + 1)-th lower horizontal line segment from p has a point in P on the left of s and consists of three or more basic line segments then
5       break; /* No child for the current d or more. */
6     else
7       for u = 0, 1, . . . , u' − 1 do
8         if (u + 1)-th upper horizontal line segment from p has a point in P on the left of s and consists of three or more basic line segments then
9           break; /* No child for the current u or more. */
10          else
11             Find-All-Children(C(s, u, d))
12   end
13 end

For the if-part, we show that P(C(s, u, d)) = Q holds. Let sₓ be a horizontal maximal line segment having its right endpoint on sₓ′, which is the vertical line segment obtained by rotating and extending sₓ in C(s, u, d). Note that the point on sₓ is located in the left of p on s. Then, sₓ consists of only a horizontal basic line segment in P(C(s, u, d)). From the assumption, sₓ also consists of only a horizontal basic line segment in Q. Thus, the two line segments in P(C(s, u, d)) and Q are coincident. Let sᵧ be a horizontal maximal line segment having its left endpoint on s in C(s, u, d). Note that the point on sᵧ is located in the right of p on s. Since the numbers of basic line segments of sᵧ in P(C(s, u, d)) is equal to the one of sᵧ in Q, it is easy to see that sᵧ in P(C(s, u, d)) coincides with the sᵧ in Q. Therefore, P(C(s, u, d)) = Q holds.

Based on the above lemma, we can enumerate all child floorplans of Q. The algorithm is shown in Algorithm 1 and Algorithm 2.

We now explain data structures required for our algorithm above. We regard each corner of a rectangle as a vertex and each basic line segment as an edge and a floorplan as a graph. We store and maintain the current floorplan using some standard data structure for planar graphs during the execution of our enumeration algorithm. For example, for each endpoint (vertex) v, we store the endpoints adjacent to v to the upper, lower, right, or left direction with information about directions. A basic line segment is represented as its two endpoints and the information representing whether it is vertical or horizontal. To store the current floorplan and its data structure, we use O(n) space in global memory space. We can efficiently trace the basic line segments on the boundary of each face. Also given an endpoint and a direction (up/down/left/right) we can find the neighbour endpoint in constant time.

We also maintain the list of the horizontal basic line segments located in the left of the leftmost vertical line segment except the left vertical line segment of R. We assume the horizontal basic line segments are sorted in the list by the x-coordinates of the points in P on the horizontal basic line segments. For Q, such list can be constructed in O(n log n) time. For any feasible floorplan with respect to P, the list can be obtained from the list of its parent by taking a prefix of the list for its parent. Thus we use O(n) space for the list and can update it efficiently.

For each recursive call, we use a constant amount of memory and the depth of the call is at most n so this part uses O(n) space in total. Thus we use O(n) space in total.

Now, let us estimate the running time of our algorithm. First, one can generate a child floorplan C(s, 0, 0) from Q in constant time, since we maintain the list of the horizontal basic line segments located in the left of the leftmost vertical line segment. We have the following lemma.

Lemma 2: Given a child floorplan C(s, u, d) of Q one can check if C(s, u + 1, d) is a child floorplan of Q or not in constant time, and if it is a child floorplan of Q one can generate C(s, u + 1, d) in constant time.

Proof. Let r be the maximal horizontal line segment containing the upper endpoint of sₓ′ in C(s, u, d). Recall that sₓ′ is the vertical line segment in C(s, u, d) obtained by rotating and extending sₓ in Q. Now t consists of two or more basic line segments in C(s, u, d). Note that the leftmost endpoint of t is on the left vertical line segment of R. It can be observed that, in C(s, u, d), the upper endpoint of sₓ′ is the second leftmost endpoint (a junction between t and sₓ′) on t. We have the following three cases.

If t has a point in P on the left of sₓ′ and t consists of exactly two basic line segments in C(s, u, d), then removing the right basic line segment of t from C(s, u, d) then extending sₓ′ upward so that it has one more basic line segment results in C(s, u + 1, d) and it is a child of Q. The number of different segments between them is clearly a constant. This check and this transformation can be done in constant time.

If t has a point in P on the left of sₓ′ and t consists of three or more basic line segments in C(s, u, d), then C(s, u + 1, d) is not a child of Q, as explained immediately before Lemma 1. This check can be done in constant time.
If \( t \) has a point in \( P \) on the right of \( s'' \) then removing the leftmost basic line segment of \( t \) from \( C(s, u, d) \) then extending \( s'' \) upward so that it has one more basic line segment results in \( C(s, u + 1, d) \) and it is a child of \( Q \). The number of different segments between them is also a constant.

Thus in constant time we can check if \( C(s, u + 1, d) \) is a child of \( Q \) or not, and if it is a child one can generate \( C(s, u + 1, d) \) from \( C(s, u, d) \).

Intuitively, we can generate \( C(s, u + 1, d) \) from \( C(s, u, d) \) by removing a suitable horizontal basic line segment having an endpoint at the upper endpoint of \( s \) then appending the vertical basic line segment having a lower endpoint at the upper endpoint of \( s \). See Fig. 6.

Similarly, given \( C(s, u, d) \) one can check if \( C(s, u, d + 1) \) is a child or not, and if it is a child one can generate \( C(s, u, d + 1) \) from \( C(s, u, d) \) in constant time. We can observe that \( C(s, u, d) \) is not a child, then \( C(s, u', d') \) for all \( u < u' \) and \( d < d' \) is not a child either. Thus we have the following lemma.

\textbf{Lemma 3:} One can enumerate all child floorplans of a given feasible floorplan \( Q \) with respect to \( P \) in \( O(k) \) time, where \( k \) is the number of the child floorplans of \( Q \).

Since the algorithm takes \( O(k) \) time for each vertex of the family tree, where \( k \) is the number of the children of the floorplan corresponding to the vertex, the algorithm above runs in \( O(|S_P|) \) time, where \( S_P \) is the set of feasible floorplans of \( R \) with respect to \( P \). We have the following theorem.

\textbf{Theorem 1:} Let \( R \) be an axis-aligned rectangle, and let \( P \) be a set of \( n \) points in the proper inside of \( R \). Then, after \( O(n \log n) \) time preprocessing, one can enumerate all feasible floorplans of \( R \) with respect to \( P \) in \( O(|S_P|) \) time and \( O(n) \) space.

Using the alternative output method [9], we have the following corollary.

\textbf{Corollary 1:} Let \( R \) be an axis-aligned rectangle, and let \( P \) be a set of \( n \) points in the proper inside of \( R \). Then, after \( O(n \log n) \) time preprocessing, one can enumerate all the feasible floorplans of \( R \) with respect to \( P \) in constant time for each and in \( O(n) \) space.

5. Conclusion

In this paper we have designed an efficient algorithm to enumerate all feasible floorplans of an axis-aligned rectangle \( R \) with respect to a given point set \( P \). Our algorithm enumerates all such floorplans in \( O(|S_P|) \) time and \( O(n) \) space after \( O(n \log n) \) time preprocessing, where \( S_P \) is the set of floorplans with respect to \( P \), and \( n = |P| \).

Can we enumerate all feasible floorplans with respect to \( P \) when some walls are fixed? Can we enumerate all feasible floorplans with respect to \( P \) when some rooms are fixed? Can we enumerate all feasible floorplans with respect to \( P \) so that each room has at least one window, which means each room must share some part of the boundary of \( R \)?

\textbf{References}


Katsuhisa Yamanaka is an assistant professor of Department of Systems Innovation Engineering, Faculty of Science and Engineering, Iwate University. He received B.E., M.E., and Dr. Eng. degrees from Gunma University in 2003, 2005 and 2007, respectively. His research interests include combinatorial algorithms and graph algorithms.

Md. Saidur Rahman was born in Salthia, Pabna, Bangladesh in 1966. He received B.Sc. Engg. degree in Electrical and Electronic Engineering in 1989 and M.Sc. Engg. degree in Computer Science and Engineering in 1992 from Bangladesh University of Engineering and Technology (BUET). He received M.Sc. and Ph.D. degrees in Information Sciences from Tohoku University, Japan in 1996 and 1999 respectively. He joined the Department of Computer Science and Engineering, BUET, as a lecturer in 1991. He is currently serving the department as a Professor. He also worked as an Associate Professor at Graduate School of Information Sciences, Tohoku University, Japan during 2003–2004. His research interests are in the fields of graph algorithms and graph theory. He has more than 100 publications in internationally reputed journals and conferences.
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 Department of Computer Science, Faculty of Engineering, Gunma University. His research interests are graph algorithms and graph theory. He is a member of IPSJ, JSIAM, and ACM.