Enumerating Floorplans with Some Properties

Shin-ichi NAKANO

Gunma University, Kiryu 376-8515, Japan

Received April 19, 2002; final version accepted November 1, 2002

A plane drawing of a graph is called a floorplan if every face (including the outer face) is a rectangle. A based floorplan is a floorplan with a designated base line segment on the outer face. In this paper we give a simple algorithm to generate all based floorplans with at most \( n \) faces. The algorithm uses \( O(n) \) space and generates such floorplans in \( O(1) \) time per floorplan without duplications. The algorithm does not output entire floorplans but the difference from the previous floorplan. By modifying the algorithm we can generate without duplications all based floorplans having exactly \( n \) faces in \( O(1) \) time per floorplan, and all (non-based) floorplans having exactly \( n \) faces in \( O(n) \) time per floorplan, where an inner room means a face which does not contain a line segment of the contour of the outer face.

**KEYWORDS:** graphs, plane graphs, enumeration, listing

1. Introduction

Generating all graphs with some property without duplications has many applications, including unbiased statistical analysis [6]. A lot of algorithms to solve these problems are already known [2,3,9, etc]. Many nice textbooks have been published on the subject [3,4].

In this paper we wish to generate all “based” floorplans, which will be defined precisely in Sect. 2. All based floorplans with three rooms are shown in Fig. 1. Such floorplans play an important role in many algorithms, including VLSI floor-planning. By checking all (or some of) based floorplans, we can find the best (or possibly nice) floorplan with respect to some given property. Also we obtain a catalog of floorplans.

To solve these all-graph-generating problems some types of algorithms are known.

Classical algorithms [3,p57] first generate all the graphs with given property allowing duplications, but output only if the graph has not been output yet. Thus this method requires quite a huge space to store a list of graphs that have already been output. Furthermore, checking whether each graph has already been output requires a lot of time.

Algorithms based on orderly method [3,p57] need not to store the list, since they output a graph only if it is a “canonical” representative of each isomorphism class.

Algorithms based on reverse search [1] also need not to store the list. The idea is to implicitly define a connected graph \( H \) such that the vertices of \( H \) correspond to the graphs with the given property, and the edges of \( H \) correspond to some relation between the graphs. By traversing an implicitly defined spanning tree of \( H \), one can find all the vertices of \( H \), which correspond to all the graphs with the given property.

The main idea of our algorithm is to define the graph \( H \) in the reverse search as a tree (not a general graph) for some problems [5,7]. Thus our algorithm does not need to find a spanning tree of \( H \), since \( H \) itself is a tree. With some other ideas we give the following simple but efficient algorithms.

Our first algorithm generates all based floorplans with “at most” \( n \) rooms. A **based floorplan** is a floorplan with a designated base line segment on the outer face. For instance, there are six based floorplans with exactly three rooms, as shown in Fig. 1. The base line segments on the outer face are depicted by thick lines. However, there are only two (non-based) floorplans with exactly three rooms. See Fig. 2. The algorithm uses \( O(n) \) space and runs in \( O(f(n)) \) time, where \( f(n) \) is the number of nonisomorphic based floorplans with at most \( n \) rooms. The algorithm generates floorplans without duplications. So the algorithm generates each floorplan in \( O(1) \) time on average. The algorithm does not output entire floorplans but the difference from the previous floorplans.

**Fig. 1. Based floorplans with three rooms.**
By modifying our first algorithm we can generate without duplications all based floorplans having “exactly” \( n \) rooms in \( O(1) \) time per floorplan and in \( O(n) \) space in total. Furthermore, we can generate all (non-based) floorplans having exactly \( n \) rooms in \( O(n) \) time (on average) per floorplan and in \( O(n) \) space in total.

Moreover, a modified version of the algorithm can generate all based floorplans with exactly \( n \) faces containing \( k \) inner rooms with \( k_1 \leq k \leq k_2 \) in \( O(1) \) time per floorplan, where \( n, k_1 \) and \( k_2 \) are given integers, and an inner room means a face which does not contain a line segment of the contour of the outer face.

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 defines a tree graph associated with based floorplans. Section 4 presents our first algorithm. We give two modified algorithms in Sect. 5. Section 6 deals with another enumeration problem. Finally Sect. 7 is devoted to the conclusion.

A preliminary version of the paper is appeared in [8].

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. For each vertex \( v \) in a tree, let \( P(v) \) be the unique path from \( v \) to \( r \). If \( P(v) \) has exactly \( k \) edges then we say the depth of \( v \) is \( k \). The parent of \( v \neq r \) is its neighbor on \( P(v) \), and the ancestors of \( v \neq r \) are the vertices on \( P(v) \) except \( v \). The parent of \( r \) and the ancestors of \( r \) are not defined. We say that \( u \) is a child of \( v \) if \( v \) is the parent of \( u \), and that \( u \) is a descendant of \( v \) if \( v \) is an ancestor of \( u \). A leaf is a vertex having no child.

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. We regard the contour of a face as the clockwise cycle formed by the line segments on the boundary of the face. Two faces \( F_1 \) and \( F_2 \) are ns-adjacent if they share a horizontal line segment. Two faces \( F_1 \) and \( F_2 \) are ew-adjacent if they share a vertical line segment.

A floorplan is a plane drawing in which every face (including the outer face) is a rectangle. In this paper we only consider floorplans which has no vertex shared by four (or more) rectangles. 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, based floorplans with three faces are shown in Fig. 1, in which each base is depicted by a thick line. If two floorplans \( P_1 \) and \( P_2 \) have a one-to-one correspondence (possibly after some rotation) between faces preserving ns- and ew-adjacency, then we say \( P_1 \) and \( P_2 \) are isomorphic. 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.

3. The Removing Sequence and the Genealogical Tree

Let \( S_n \) be the set of all non-isomorphic based floorplans having at most \( n \) faces. In this section we define a tree graph associated with the based floorplans in \( S_n \).

Let \( R_1 \) be the floorplan having exactly one inner face. Assume \( R \) is a based floorplan in \( S_n \) with \( R \neq R_1 \). Let \( F \) be the inner face of \( R \) having the upper-left corner of the outer rectangle of \( R \). We call such a face the first face of the based floorplan \( R \). First faces of based floorplans are shaded in Figs. 3–6. The first face \( F \) is upward removable if \( R \) has a vertical line segment with upper end \( v \) where \( v \) is the lower-right corner of \( F \). See Fig. 3(a). Otherwise, \( R \) has a
Fig. 4. Removing an upward removable face.

Fig. 5. The removing sequence.

Fig. 6. Genealogical tree $T_4$. 
horizontal line segment with left end $v$, and the first face $F$ is leftward removable. See Fig. 3(b). Since $R \neq R_1$, the first face is either upward removable or leftward removable. If $F$ is upward removable, then we can obtain another floorplan with fewer faces by one by continually shrinking the first face into the uppermost horizontal line of $R$ with preserving the width of $F$ and enlarging the faces below $F$, as shown in Fig. 4. Similarly, if $F$ is leftward removable, then we can obtain another floorplan with fewer faces by one by continually shrinking the first face into the leftmost line of $R$ with preserving the height of $F$. If we remove the first face from $R$ then the resulting floorplan is again a based floorplan in $S_n$ with one less faces. We denote such floorplan as $P(R)$. Thus we can define the based floorplan $P(R)$ in $S_n$ for each $R$ in $S_n$ except $R_1$. We say $R$ is a child floorplan of $P(R)$.

Given a floorplan $R$ in $S_n$, by repeatedly removing the first face, we can have the unique sequence $R, P(R), P(P(R)), \ldots$ of floorplans in $S_n$ which eventually ends with $R_1$, which is the floorplan having exactly one inner face. See an example in Fig. 5, in which the first faces are shaded.

By merging those sequences we can have the genealogical tree $T_n$ of $S_n$ such that the vertices of $T_n$ correspond to the floorplans in $S_n$, and each edge corresponds to each relation between some $R$ and $P(R)$. For instance, $T_3$ is shown in Fig. 6, in which the first faces are shaded, each edge corresponds to upward removing is depicted by a solid line, and each edge corresponds to leftward removing is depicted by a dotted line. We call the vertex in $T_n$ corresponding to $R_1$ the root of $T_n$.

4. Algorithm

If the set $S_n$ of all based floorplans is given, we can construct $T_n$ easily according to its definition. However, we want to construct $T_n$ without knowing the set $S_n$ in advance: how can we do that? For this purpose, we use the idea of reversing the removing procedure by iteratively finding all children of a current floorplan.

We need some definitions here. Assume $R$ is a floorplan in $S_n$. Let $P_N$ be the uppermost horizontal line segment of $R$, and $u_0, u_1, \ldots, u_6$ be the vertices on $P_N$, each of which is an upper end of a vertical line segment. Assume $u_0, u_1, \ldots, u_6$ appear on $P_N$ from left to right in this order. See an example in Fig. 7(a). Let $F_1$ be the inner face of $R$ with upper-right corner $u_1$, for $1 \leq i \leq x$. We observe that if $F_1$ has $k$ neighbor faces to the right, then $R$ has exactly $k$ child floorplans $R_c$ such that the first face of $R_c$ is upward removable, and the first face has $i$ neighbor faces to the bottom. We denote by $R(U, s, e)$ the child floorplan of $R$ such that (1) the first face of $R(U, s, e)$ is upward removable, and (2) the first face of $R(U, s, e)$ has $s$ neighbor faces to the bottom and $e$ neighbor faces to the right. For instance, face $F_3$ of floorplan $R$ in Fig. 7(a) has three neighbor faces to the right, therefore $R$ has exactly three child floorplans $R(U, 5, 1), R(U, 5, 2), R(U, 5, 3)$ such that the first face has five neighbor faces to the bottom. Similarly we denote by $R(L, s, e)$ the child floorplan of $R$ such that (1) the first face of $R(L, s, e)$ is leftward removable, and (2) the first face of $R(L, s, e)$ has $s$ neighbor faces to the bottom and $e$ neighbor faces to the right. In Fig. 6 the labels are attached to all edges in $T_n$.

![Fig. 7. (a) A floorplan $R$ and (b) some child floorplans of $R$.](image)

Thus, given a based floorplan $R$ in $S_n$ with at most $n - 1$ faces, we can find all child floorplans $R(U, 1, 1), \ldots$ of $R$ in $S_n$. If $R$ has $k$ child floorplans then we can find them in $O(k)$ time, since we can compute $R(U, s + 1, e)$ and $R(U, s, e + 1)$ from $R(U, s, e)$ in $O(1)$ time. This is an intuitive reason why our algorithm generates each floorplan in $O(1)$ time on average.

Recursively repeating this process from the root of $T_n$, corresponding to $R_1$ we can traverse $T_n$ without constructing whole $T_n$. During the traversal of $T_n$, we assign a label either $(U, s, e)$ or $(L, s, e)$ to each edge connecting $R$ and $P(R)$ in $T_n$, as shown in Fig. 6. Each label shows how to generate a child floorplan of $R$, and each sequence of labels on a path starting from the root specifies a floorplan in $S_n$. For instance $(U, 1, 1), (U, 1, 1), (U, 1, 1)$ specify the uppermost floorplan in $T_n$. During our algorithm we maintain these labels only on the path from the root to the “current” vertex, because those are enough information to generate the “current” floorplan. To generate next floorplan, we need to maintain some more information only for the floorplans on the “current” path, which has length at most $n$. This is an intuitive reason why our algorithm uses only $O(n)$ space, while the number of floorplans may not be bounded by a
5. Modification of the Algorithm

Our algorithm is as follows.

**Procedure find-all-child-floorplans** (*R*)

begin
1. Output *R* [Output the difference from the previous tree.]
2. if *R* has exactly *n* faces then return
3. Let \(F_1, F_2, \ldots, F_x\) be the inner face of *R* sharing the uppermost horizontal line segment of *R*, and assume that they appear from left to right in this order.
4. for \(i = 1\) to \(x\)
5. Assume \(F_i\) has \(e(i)\) neighbors to the right.
6. for \(j = 1\) to \(e(i)\)
7. find-all-child-floorplans (*R*[\(U, i, j\)])
8. Let \(F'_1, F'_2, \ldots, F'_y\) be the inner face of *R*
   sharing the leftmost vertical line segment of *R*,
   and assume that they appear from top to bottom in this order.
9. for \(i = 1\) to \(y\)
10. Assume \(F'_i\) has \(s(i)\) neighbors to the bottom.
11. for \(j = 1\) to \(s(i)\)
12. find-all-child-floorplans (*R*[\(L, j, i\)])
end

**Algorithm find-all-floorplans** (*n*)

begin
1. find-all-child-floorplans (*R*[\(1\)])
end

**Theorem 1.** The algorithm uses \(O(n)\) space and runs in \(O(f(n))\) time, where \(f(n)\) is the number of non-isomorphic based floorplan with at most \(n\) faces.

**Proof.** Given a based floorplan *R*, we can find all \(k\) child floorplans in \(O(k)\) time. Thus our algorithm needs only a constant time of computations for each edge of the tree. Thus the algorithm runs in \(O(f(n))\) time. For each recursive call we need a constant amount of space, and the depth of recursive call is bounded by \(n - 1\). Thus the algorithm uses \(O(n)\) space in total. \(\square\)

5. Modification of the Algorithm

Then we consider our second problem.

Let \(S_{\text{non}}\) be the set of non-isomorphic based floorplans having exactly \(n\) faces. We wish to generate all floorplans in \(S_{\text{non}}\) without duplications. Clearly all such floorplans are in \(S_n\) but with other floorplans. How can we output only floorplans in \(S_{\text{non}}\) efficiently? We have the following lemma.

**Lemma 1.** Let \(g(n)\) be the number of floorplans in \(S_{\text{non}}\). Then \(S_n\) has at most \(2 \cdot g(n)\) floorplans.

**Proof.** Each floorplan *R* with \(n - 1\) or less faces has at least two child floorplans \(*R*[\(U, 1, 1\)]\) and \(*R*[\(L, 1, 1\)]\). By the definition of the genealogical tree each vertex with depth \(n - 1\) in \(T_n\) is a leaf, and the number of those vertices is \(g(n)\). Thus \(T_n\) has at most \(2 \cdot g(n)\) vertices, so \(S_n\) has at most \(2 \cdot g(n)\) floorplans. \(\square\)

Modifying our first algorithm so that it output only based floorplans having exactly \(n\) faces, which corresponds to leaves of \(T_n\), we can have the following theorem.

**Theorem 2.** The modified algorithm uses \(O(n)\) space and outputs based floorplans having exactly \(n\) faces in \(O(1)\) time per floorplan without duplications.

We modify the algorithm further so that it output all (non-based) floorplans having exactly \(n\) faces, as follows.

At each leaf \(v\) of the genealogical tree \(T_n\), the floorplan \(R\) corresponding to \(v\) is checked whether the removing sequence of *R* with the base is the lexicographically first one among the four based floorplans each of which is derived from *R* by rotating *R* and then choosing the bottom line segment as the base, and only if *R* has the lexicographically first one then *R* is output.

For instance, see Fig. 8. Assume that we are about to check whether floorplan (a) is output. First we construct three
floorplans (b)–(d) by rotation. Then we find each removing sequences, as shown in Fig. 8. Since (d) has the lexicographically first one, we do not output (a).

Thus we can output only the canonical representative of each isomorphism class.

**Theorem 3.** The modified algorithm uses $O(n)$ space and runs in $O(n \cdot h(n))$ time, where $h(n)$ is the number of non-isomorphic (non-based) floorplans having exactly $n$ faces.

**Proof.** Given a based floorplan $R$, we can find the removing sequence in $O(n)$ time. For each floorplan corresponding to a leaf of $T_n$, we construct at most four removing sequences associated with isomorphic floorplans, and find the lexicographically first one in $O(4n)$ time, and for each output floorplan our tree contains at most four isomorphic ones corresponding to the four choices of the base. Thus the algorithm runs in $O(n \cdot h(n))$ time. The algorithm clearly uses $O(n)$ space. \qed

6. Rooms with Windows

In this section we consider another similar enumeration problem.

In a floorplan if an inner face contains a line segment of the contour of the outer face, then it is called an outer room, otherwise it is called an inner room. Intuitively, outer rooms are rooms with windows, and inner rooms are rooms without windows.

Given three integers $n$, $k_1$, and $k_2$, we wish to generate all based floorplans with exactly $n$ faces containing at least $k_1$ inner rooms and at most $k_2$ inner rooms. We assume $k_1 \leq k_2$ and $k_2 + 4 \leq n$. Note that to have some inner rooms we need at least four outer rooms surrounding them. Let $S_n(k_1, k_2)$ be the set of such floorplans. We know that every floorplan in $S_n(k_1, k_2)$ corresponds to a leaf of the genealogical tree $T_n$, but other floorplans also correspond to some leaves of $T_n$. So, if we traverse $T_n$ with some screening, then we can generate all floorplans in $S_n(k_1, k_2)$, but such an algorithm may not be efficient. We, however, can solve the problem efficiently by traversing a new tree which is obtained by pruning off unnecessary branches from $T_n$. For this, we need to analyze the structure of $T_n$ further.

Assume a vertex $v$ in $T_n$ corresponds to a floorplan $R$ having $n_a$ rooms including $n_i$ inner rooms. Let $n_r = n - n_a$, which is the number of rooms we are going to introduce to $R$ to construct a floorplan with $n$ faces.

We observe that after introducing a new face to $R$ to construct a child floorplan, (1) the new face is always an outer room, (2) every inner room remains as an inner room, and (3) some outer rooms may become inner rooms since the appearance of the new face may block their windows. Hence, every child floorplan of $R$ has at most one more outer rooms and at least $n_i$ inner rooms. This observation also implies that any descendant floorplan of $R$ has at most $n_r$ more outer rooms than $R$ has.

Also we observe that some descendant floorplans of $R$ can actually have exactly $n_r$ more outer rooms, since $R(U, 1, 1)$ and $R(L, 1, 1)$ of $R$ always has one more outer rooms than $R$ has, and by choosing $R(U, 1, 1)$ or $R(L, 1, 1)$ for each generation we can have at least $2^{n_i}$ descendant floorplans of $R$ having exactly $n_r$ more outer rooms than $R$ has.

Assume $R$ has $n_i$ outer rooms containing a line segment on the lowest horizontal line segment of the outer face of $R$, and $n_o$ outer rooms containing a line segment on the rightmost vertical line segment of the outer face of $R$. Intuitively $n_i$ is the number of rooms having south windows, and $n_o$ is the number of rooms having east windows. We observe that the outer rooms having either south or east windows never become inner rooms, since new rooms always appear on the north or west side. Also we observe that any outer room $F$ having neither south nor east windows can become an inner face, if we appropriately introduce two faces so that they block all north and east windows. Thus if $n_o \geq 2$, then some descendant floorplans of $R$ has $(n_r + 2) + (n_o - n_i - n_r - n_o + 1)$ more inner rooms, since newly introduced $(n_r - 2)$ rooms can finally become inner rooms, and $(n_o - n_i - n_r - n_o + 1)$ outer rooms can become inner rooms. Note that $R$ always has one room having both south and east windows on the lower right corner, and the two rooms lastly added to the floorplan are always outer rooms in the final floorplan. By choosing $R(U, 1, 1)$ or $R(L, 1, 1)$ for each generation, and by appropriately introducing the last two rooms to maximize the number of inner rooms, we can have at least $2^{n_i - 2}$ descendant floorplans of $R$ having exactly $(n_r - 2) + (n_o - n_i - n_r - n_o + 1)$ more inner rooms than $R$ has.

Thus, given a floorplan $R$ in $S_n$, we know the following (1)–(4).

\begin{align*}
\text{(a) } & \text{U11U11L12} \\
\text{(b) } & \text{L11U11U21} \\
\text{(c) } & \text{L11U11L11} \\
\text{(d) } & \text{L11U21L12}
\end{align*}

Fig. 8. Removing sequences.
The maximum number of outer rooms of descendant floorplans is $n_a - n_i + n_r$.

If $n_i \geq 2$ then the maximum number of inner rooms of descendant floorplans is $n_i + (n_r - 2) + (n_a - n_i - n_i - n_r + 1)$.

If $n_i = 1$ then the maximum number of inner rooms of descendant floorplans is $n_i + \max(n_a, n_w)$, where $n_w$ is the number of outer rooms having only north windows, and $n_r$ is the number of outer rooms having only west windows. (We can block only either north or west windows to have the maximum number of inner rooms.)

The minimum number of inner rooms of descendant floorplans is $n_i$.

Based on the analysis above, we can define a new tree from $T_n$ by removing vertices having no descendant floorplan in $S_n(k_1, k_2)$. We denote such tree by $T_n(k_1, k_2)$. The analysis above also shows that we can find all $k$ child floorplans of a given floorplan in $O(k)$ time by using a suitable data structure. We have the following lemma.

**Lemma 2.** Let $v$ be a non-leaf vertex in $T_n(k_1, k_2)$, $n \geq 5$, and $R$ be the floorplan corresponds to $v$. Assume $R$ has $n_a$ faces including $n_i$ inner rooms. Let $n_a$ be the number of outer rooms having only north windows in $R$, and $n_r$ be the number of outer rooms having only west windows in $R$. Then the following (a)–(c) hold.

(a) If $n_a \leq 2$, then $v$ has one or more child vertices in $T_n(k_1, k_2)$.

(b) If $3 \leq n_a \leq n - 2$, then $v$ has two or more child vertices in $T_n(k_1, k_2)$.

(c) If $n_a = n - 1$, then $v$ has one or more child vertices in $T_n(k_1, k_2)$.

**Proof.** By $v \in T_n(k_1, k_2)$, $n_i \leq k_2$ holds. Also by $v \in T_n(k_1, k_2)$, if $n_i \geq 2$ then $k_1 \leq n_i + (n_r - 2) + (n_a - n_i - n_i - n_r + 1)$, and if $n_i = 1$ then $k_1 \leq n_i + \max(n_a, n_w)$ hold.

We first prove (a) and (c). By definition of $T_n(k_1, k_2)$, $v$ has one or more descendant in $S_n(k_1, k_2)$. This means $v$ has at least one child $v'$ having one or more descendant in $T_n(k_1, k_2)$.

Then we prove (b). This time we need to prove $v$ has two child vertices, or equivalently $R$ has two child floorplans.

We consider the following two cases.

**Case 1:** $k_1 = n_i + (n_r - 2) + (n_a - n_i - n_i - n_r + 1)$.

If we introduce a face having either south or east windows, then, in the derived floorplan $R'$, $n_i$ is unchanged or increased, $n_a$ is decreased by one, and $n_a$ or $n_r$ is increased by one. So any descendant floorplan of $R$ has at most $k_1 - 1$ inner rooms. Thus $R \notin S_n(k_1, k_2)$.

If we introduce a face having neither south nor east windows, then, in the derived floorplan $R'$, $n_a$ is unchanged or increased, $n_i$ is decreased by one, and $n_a$ or $n_r$ is increased by one, and $n_i$ or $n_a$ is changed. So some descendant floorplan of $R$ has $k_1$ inner rooms. Thus we need two child floorplan such that the first faces have neither south nor east windows. We consider three cases.

If the first face $F_1$ of $R$ has neither south nor east window, then $R(U, 1, 1)$ and $R(L, 1, 1)$ are such child floorplans. Then, $R(U, 1, 1)$ and $R(L, 1, 2)$ are such child floorplans. Otherwise $F_1$ has exactly one neighbor face to the right, then $R(U, 1, 1)$ and $R(U, 1, 2)$ are such child floorplans.

If the first face $F_1$ of $R$ has east window, then we consider two subcases. If $F_1$ has two or more neighbor face to the right, then $R(U, 1, 1)$ and $R(U, 1, 2)$ are such child floorplans. Otherwise $F_1$ has exactly one neighbor face to the right, then $R(U, 1, 1)$ and $R(U, 1, 2)$ are such child floorplans.

**Case 2:** $k_1 < n_i + (n_r - 2) + (n_a - n_i - n_i - n_r + 1)$.

Both $R(U, 1, 1)$ and $R(L, 1, 1)$ are such child floorplans.

The lemma above shows the number of vertices in $T_n(k_1, k_2)$ is bounded by three times the number of leaves plus a constant, where the constant is the number of floorplans with $n_a \leq 2$, that is at most 3. (See Fig. 6.)

We have the following lemma.

**Lemma 3.** Let $|V|$ be the number of vertices in $T_n(k_1, k_2)$, and $|L|$ be the number of leaves in $T_n(k_1, k_2)$. Then, $|V| \leq 3|L| + 3$ holds.

By traversing the tree $T_n(k_1, k_2)$ as before we can generate all such based floorplans efficiently. Now we have the following theorem.

**Theorem 4.** We can enumerate all based floorplans with $n$ faces containing at least $k_1$ inner rooms and at most $k_2$ inner rooms. The algorithm uses $O(n)$ space in total and runs in $O(1)$ time per such floorplan.

By a similar method to the one explained in the proof of Theorem 3, we can also solve the non-based version of the problem in $O(n)$ space in total and in $O(n)$ time per floorplan.

7. Conclusion

In this paper we have given four simple algorithms to generate all graphs with some properties. Each of our algorithms finds all graphs with some properties by traversing an implicitly defined genealogical tree such that the
vertices correspond to graphs to be enumerated.

It is remained as an open problem whether our method can be applied to other enumeration problems.

REFERENCES