SIAM J. COMPUT.
Vol. 22, No. 3, pp. 500–526, June 1993

004

FLOOR-PLANNING BY GRAPH DUALIZATION: 2-CONCAVE RECTILINEAR MODULES*

KOK-HOO YEAP† AND MAJID SARRAFZADEH†

Abstract. Given a planar triangulated graph (PTG) G, the problem of constructing a floor-plan F such that G is the dual of F and the boundary of F is rectangular is studied. It is shown that if only zero-concave rectilinear modules (CRM) (or rectangular modules) and 1-CRM (i.e., L-shaped) are allowed, there are PTGs that do not admit any floor-plan. However, if 2-bend modules (e.g., T-shaped and Z-shaped) are also allowed, then every biconnected PTG admits a floor-plan. Thus, the employment of 2-bend modules is necessary and sufficient for graph dualization floor-planning. A linear-time algorithm for constructing a 2-CRM floor-plan of an arbitrary PTG is proposed.

Key words. floor-planning, planar graphs, graph dualization, CRM

AMS(MOS) subject classifications. 05C85, 68Q20, 68Q35, 68R10

1. Introduction.

Floor-planning is an early step in VLSI chip design where one decides the relative location of functional entities in a chip. The most immediate representation of a floor-plan is the partition of a rectangular chip area into modules (usually rectilinear polygons) where each module represents a functional entity. Such a partition can be represented by a partition graph where faces of the graph correspond to modules, edges represent the sides of modules, and vertices are junctions. Figure 1(a) shows a floor-plan represented in partition graph.

The dual graph of a partition is a graph where each vertex represents a module and each edge (i, j) represents adjacency of module i and module j. Given a partition, its dual graph is easily and uniquely determined. However, given a dual graph specifying adjacency requirements, it is not readily converted into a partition. In this formulation of floor-planning problem, we are given a graph specifying the connection requirements of circuit modules and wish to find a rectilinear partition on a rectangular chip area.

For simplicity, most floor-planning systems are restricted to 0-concave rectilinear modules (0-CRM), i.e., rectangles. The dual graph of a rectangular floor-plan is a planar triangulated graph (PTG). However, there exist planar triangulated graphs that do not have any dual floor-plans. Kozminski and Kinnen developed the necessary and sufficient conditions for the existence of a 0-CRM floor-plan [2]. They also showed a technique to transform one floor-plan to another where the adjacency requirements were preserved [3]. A linear time algorithm for constructing a rectangular floor-plan, if one exists, was reported in [1]. An algorithm to enumerate all rectangular floor-plans was reported in [13]. An algorithm for sliceable floor-planning was proposed in [14].

The construction of a floor-plan is complicated by the existence of complex triangles (CTs) (a cycle of three edges that is not a face) because the dual of a rectangular floor-plan does not contain any CT. One approach eliminates all CTs to obtain a rectangular floor-plan [12]. This technique introduces new vertices and edges in the original PTG, producing empty spaces in the floor-plan. If the empty spaces are considered as part of some adjacent modules, this approach produces general rectilinear-shaped modules. The weighted CT elimination problem has been shown to be NP-complete [11].

*Received by the editors February 11, 1991; accepted for publication (in revised form) March 24, 1992. This work was supported in part by the National Science Foundation under grant MIP-8921540.

†Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60208.

Adjacency requirements of CTs can be achieved by introducing more complicated shapes instead of rectangular modules. A 1-concave rectilinear module (1-CRM) with 1-bend is required to satisfy the adjacency requirements of a CT. Similarly a 2-concave rectilinear module (2-CRM) can satisfy adjacency requirements of two CTs simultaneously. In some sense, the introduction of more complicated shapes is analogous to eliminating CTs. However, this approach maintains the adjacency requirements without arbitrarily adding new vertices and edges to the input PTG.

In this paper we examine 0-CRM (rectangle), 1-CRM, and 2-CRM. A 1-CRM refers to a rectilinear polygon that consists of six sides, five convex and one concave corners. Similarly, a 2-CRM refers to a rectilinear polygon with eight sides, six convex and two concave corners. A 2-CRM can be viewed as two adjacent 1-CRMs. In actual implementation, a 1- or 2-CRM can be represented by a set of adjacent rectangles.

The necessary and sufficient conditions for the existence of a 1-CRM floor-plan are given in [10]. However, there exist PTGs, as we will show, where 1-CRM floor-plan is not sufficient. Obviously, an arbitrary PTG admits a floor-plan if we allow general rectilinear-shaped modules, i.e., k-CRM for arbitrary large k. It was an open problem to find the least k in which some floor-plans exist for any PTG. We show that any biconnected PTG admits a 2-CRM floor-plan. A linear-time algorithm for constructing one is presented. The biconnected requirement is not a restrictive condition. By adding edges, a 1-connected planar graph can be transformed into a biconnected graph without loosing planarity and adjacency. We thus establish that employment of 2-bend modules is both necessary and sufficient.

The restriction of the dual graphs to PTG is a natural constraint. The dual of a floor-plan is planar since a floor-plan is planar by definition. Also, PTG represents the most dense planar graph. Any planar graph can be triangulated by adding edges. In the floor-planning process, an adjacency graph is first planarized and triangulated to yield a PTG. The PTG then serves as the input to the floor-planning problem.

This paper is organized as follows: Section 2 discusses the major issues arising in the floor-plan construction of a PTG, especially the local adjacency requirements. Section 3 presents a key theorem for the establishment of the sufficient conditions for dualization. An algorithm for floor-plan construction is proposed in §4.

2. Graph dualization. In this section, we discuss major issues arising in graph dualization of a PTG. We examine the local adjacency requirements of a PTG and show that a 1-CRM floor-plan is not sufficient. The notion of sharing, which is crucial to dual construction, is also discussed.

2.1. Floor-plan and its dual. A *floor-plan* F is a plane graph where

1. Each edge is either a horizontal or vertical line segment.
2. The boundary of F is rectangular.
3. Each vertex has either degree 2 or 3.

A degree 4 vertex can be represented by two vertices of degree 3 and an edge e_0 of zero length, as shown in Fig. 1(b). Thus we can assume that each vertex of F has degree 2 or 3 without loss of generality. Also we assume that there is no degree 2 vertex as shown in Fig. 1(c), because it can be represented by a single edge.

A face of a floor-plan is also called a *module*. An angle formed by two neighboring edges e_1 and e_2 incident to a common vertex v is called a *corner*. A corner is denoted by an ordered pair of edges (e_1, e_2) with e_1 preceding e_2 in a clockwise manner centered at v. There are three types of corners in F: *convex* (90°), *aligned* (180°), and *concave* (270°). Because modules are confined by horizontal an vertical line segments, the num-

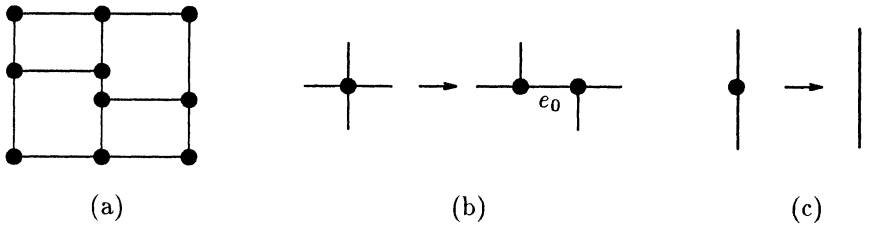

FIG. 1. *A floor-plan and its restrictions.*

ber of convex corners minus that of concave corners is always four. For the purpose of discussion, we always assume that F is confined by four infinite regions r, u, l, and b as shown in Fig. 2.

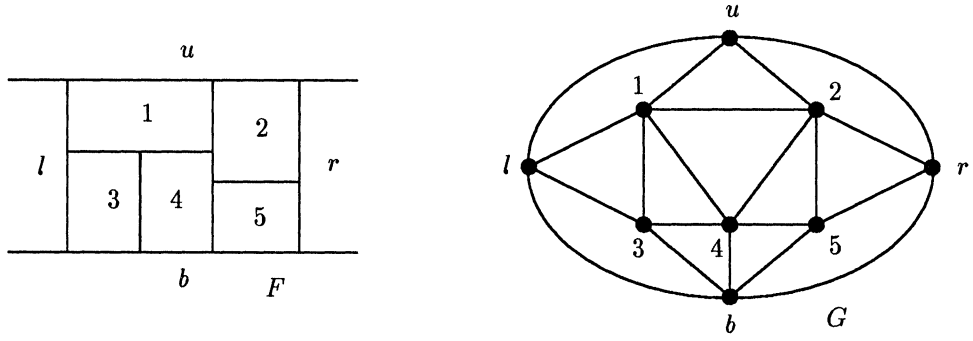

FIG. 2. *A floor-plan F and its dual G.*

The *dual* of a floor-plan F is a plane graph $G = (V, E)$, where each vertex in V corresponds to a face in F and each edge of G corresponds to an edge of F. A face bounded by three edges is also called a *triangle*. A face bounded by two parallel edges is called a *biface*. There is a one to one correspondence between the edges of F and that of G. Given F, the dual G is easily and uniquely determined. Figure 2 shows a floor-plan F and its dual G. If the vertices r, u, l, b (and incident edges) are deleted from G, the remaining subgraph G_c is called the *core* of G and G is called an *extended graph* of G_c. The vertices of G_c are called *core vertices*. Because corners are defined by two edges, there is also a one to one correspondence between corners of F and that of G. The bounded faces of F correspond to the core vertices of G.

In the graph dualization approach to floor-planning, we are given a core G_c and wish to find a floor-plan F where the dual G has core G_c. The very first step in the dualization process is to append the vertices r, u, l, b (and edges) to G_c to obtain G that matches the configuration shown in Fig. 3. This is not a trivial process. An excellent discussion on the details is contained in [2] and we shall not address this problem here. From now on, we will assume that G is given and conforms to the configuration as shown in Fig. 3. If F is restricted to 0-CRM modules, the dual G is a PTG. For a 1-CRM or 2-CRM floor-plan, the dual is also a PTG if we eliminate one parallel edge of each biface. Thus, it is reasonable to restrict our input G to PTGs. A PTG is also the most dense planar graph. Given a core G_c that specifies adjacency requirements of modules, we assume that some preprocessing has been performed to planarize and triangulate G_c to obtain G.

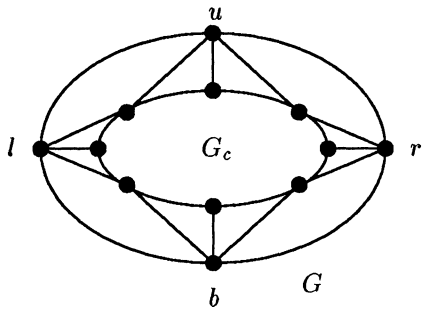

FIG. 3. *Configuration of a dual G.*

2.2. Alternate representation of a floor-plan with its dual. Let G be the dual of a floor-plan F. An *F-labeling* [3] of G is a labeling of edges and corners of G where

1. Dual edges of horizontal line segments are labeled H.
2. Dual edges of vertical line segments are labeled V.
3. Dual corners of G are labeled **convex**, **aligned**, or **concave**.

An edge label is also called *orientation*. The corners of the infinite face of G (e.g., $((l,u),(u,r))$) are not labeled because they are not interesting in our discussion. Figure 4 shows an example of F-labeling. Vertical edges are denoted by thick lines, whereas horizontal edges are shown with thin lines. A corner formed by two edges of identical orientations is an **aligned** corner. A corner formed by two edges of different orientations is **concave** if there is an arrow pointing to the vertex, otherwise it is **convex**.

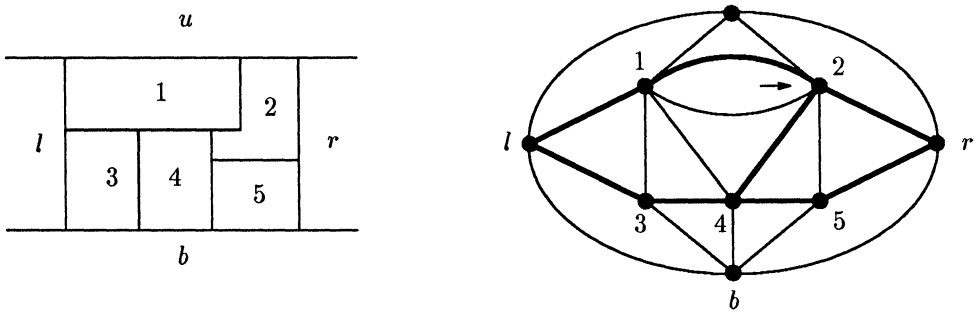

FIG. 4. *An example of F-labeling.*

An arbitrary F-labeling of G may not correspond to a floor-plan. We call an F-labeling *legal* if a corresponding floor-plan exists. There is a one-to-one correspondence between a legal F-labeling of G and a floor-plan F. Given G, the task of finding F can be viewed as finding a legal F-labeling in G.

Consider a legal F-labeling in G. By the definition of floor-plan, a legal F-labeling exhibits the following properties:

P1. All incident edges of u and b have horizontal orientation.

P2. All incident edges of l and r have vertical orientation except (u,l), (u,r), (b,l), and (b,r).

P3. Let the convex, aligned, and concave corners be assigned cost 1, 0, and -1, respectively. The sum of all corner costs incident to a core vertex is exactly 4. The sum

costs of u, b is 0, and l, r is 2. Also, the total corner cost of a triangle of G is 2, and the total cost of a biface is 0. This property is a consequence of rectilinear faces in F.

P4. Each triangle of G has at least one horizontal edge and at least one vertical edge. Each biface has exactly one horizontal and one vertical edge.

P5. No biface is incident to vertices r, u, l, b. All incident corners of r, u, l, b are not concave.

It is also evident that if an F-labeling of G satisfies the above conditions, it is a legal F-labeling. We can reconstruct the horizontal and vertical line segments of floor-plan F with the proper corners specified by the F-labeling. Thus a dualization process from G to F corresponds to finding an F-labeling satisfying the above conditions.

2.3. Complex triangles. A *complex triangle* (CT) is a nonface cycle containing exactly three edges. For our discussion, CTs are only defined on dual G. Figure 5 shows two examples of complex triangles.

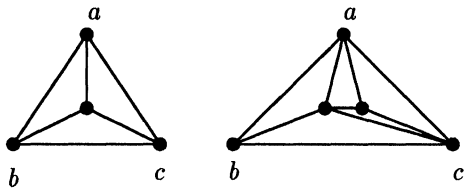

FIG. 5. *CTs.*

Consider the dualization of a CT $\triangle_{abc}$. If only 0-CRM is allowed in the floor-plan, it is impossible to satisfy the adjacency requirements of the edges (a, b), (b, c), and (a, c) simultaneously. However, if 1-CRM is allowed, a floor-plan exists as shown in Fig. 6. A 0-CRM floor-plan does not exist if and only if G contains a CT [2]. Note that the concave corner of module 3 contributes to the existence of a floor-plan. A biface is formed in the dual due to the concave corner of vertex 3. For clarity, some edges of the floor-plans are not shown in the duals.

As far as the CT is concerned, any of the vertices $1, 2, 3$ may be chosen to bear a concave corner so that the adjacency requirements are satisfied. We call such choice the *assignment* of a CT to a vertex. Vertices 1 and 2 are also suitable for assignment.

Intuitively, the assignment of a CT to a vertex v can be seen as adding a concave corner to module v, thus bending it to satisfy the adjacency of the CT. To construct a floor-plan, all CTs in G must be assigned. The number of assignments on a vertex is related to the number of concave corners in the final floor-plan. Since we only allow 2-CRM, the number of assignments on any vertex shall not exceed 2.

A *perfect assignment* of a dual G is a set of assignments of all CTs of G where

1. Every CT is assigned to a core vertex.
2. No vertex carries more than two assignments.
3. Noncore vertices of G (i.e., r, u, l, and b) are not assigned.

In a perfect assignment, a core vertex may be assigned zero, one, or two times.

In our algorithm, if a vertex is not assigned, its corresponding module in F is a rectangle. If a vertex is assigned once, it is a 1-CRM. There is only one type of 1-CRM if we disregard module orientations. A 1-CRM is also called an L-shaped module. If a vertex is assigned twice, a 2-CRM is formed. There are four classes of 2-CRM as shown in Fig. 7.

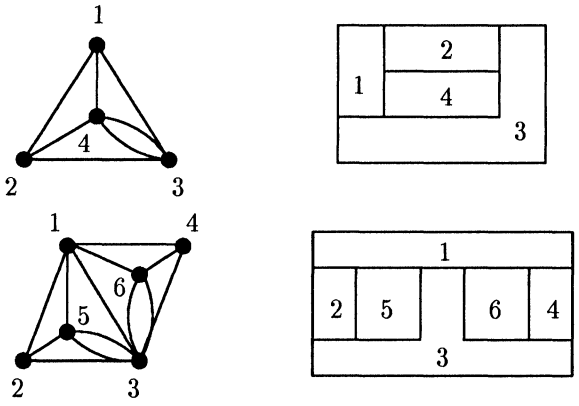

FIG. 6. *Floor-plans of CTs.*

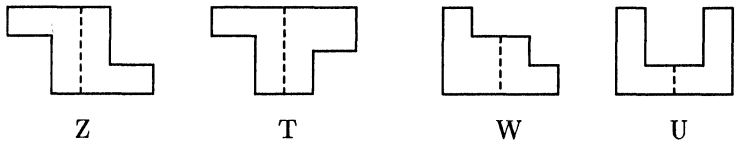

FIG. 7. *Four classes of 2-CRM: Z, T, W, and U.*

2.4. Insufficiency of 1-CRM duals. Given a PTG G, some vertices of a CT must be at least 1-CRM to satisfy the local adjacency requirements. Kozminski and Kinnen proved the necessary and sufficient conditions for a graph to admit a 0-CRM (rectangular) floor-plan [2]. It was further generalized by [10] to prove the necessary and sufficient conditions for the existence of a 1-CRM floor-plan. The primary condition for the existence of a 1-CRM floor-plan is that all nonoverlapping (i.e., one is not contained in the other) CTs can be assigned to some vertices and each vertex is not assigned more than once. If a vertex is assigned, it becomes a 1-CRM in the corresponding floor-plan; otherwise it is a rectangle. However, there exist PTGs that do not admit 1-CRM floor-plans. In particular, consider the graph shown in Fig. 8: The graph contains seven nonoverlapping CTs and only six vertices are available for assignment. There is no assignment that can satisfy the necessary conditions for the existence of a 1-CRM floor-plan. This suggests that the dualization of a PTG requires more complicated shapes. In this paper, we prove that generalization to 2-CRM is necessary and sufficient for the existence of a floor-plan for any biconnected PTG.

2.5. Complex triangle graph and containment tree. A *complex triangle graph* (CTG) of $G = (V, E)$ is $CTG(G) = (V_c, E_c)$, $V_c \subseteq V$ and $E_c \subseteq E$ where

$$V_c = \{\, v \mid v \text{ belongs to some CTs of } G \,\},$$
$$E_c = \{\, e \mid e \text{ belongs to some CTs of } G \,\}.$$

Given G, it is easy to construct the CTG thereof. A CT is always assigned to a vertex in V_c. Therefore, it suffices to consider $CTG(G)$ in search of a perfect assignment. In general, $CTG(G)$ may not be connected. However, it is easy for our assignment algorithm to be

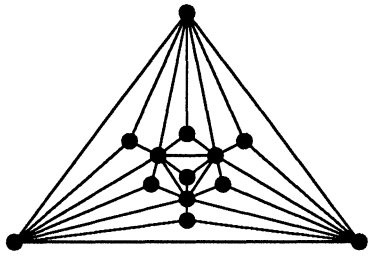

FIG. 8. *A PTG that does not admit any* 1*-CRM floor-plan.*

repeated for each connected component of $CTG(G)$. Therefore, we will assume that $CTG(G)$ is connected.

There is a natural hierarchy among the CTs of G. Formally, we say that CT $\triangle_{abc}$ *contains* CT $\triangle_{def}$ if the vertices d, e, and f lie in the region bounded by the edges (a, b), (a, c), and (b, c). The area of $\triangle_{def}$ must be strictly less than that of $\triangle_{abc}$. We say that $\triangle_{abc}$ *immediately contains* $\triangle_{def}$ if there exist no triangles contained in $\triangle_{abc}$, which contain $\triangle_{def}$. If two complex triangles have no containment relation, we say that they are *independent*. Figure 9 illustrates the containment relationship.

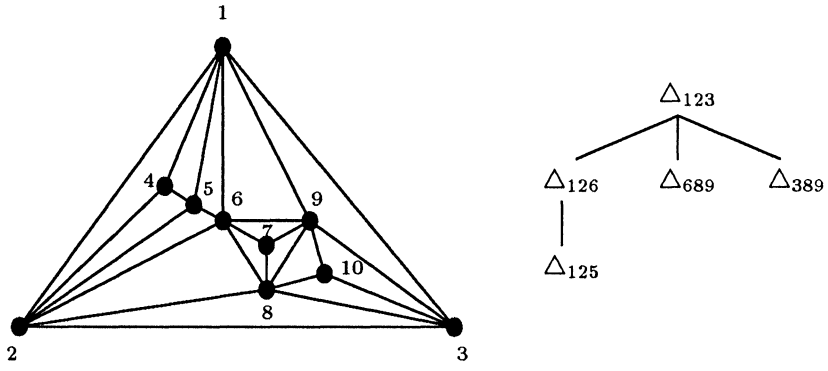

FIG. 9. *A PTG and its corresponding containment tree.*

Given G, we can construct the *containment tree* representing the hierarchy of the CTs. Each tree node represents a CT, and a parent node immediately contains its child nodes. A parent and a child can share at most two vertices. A pseudo-root node is added as the parent of top-level CTs, if needed. Traversing the tree top-down allows us to process all CTs hierarchically.

2.6. Sharing assignments. Consider two CTs $\triangle_{abc}$ and $\triangle_{ade}$ where $\triangle_{abc}$ immediately contains $\triangle_{ade}$. Suppose $\triangle_{abc}$ is assigned to vertex a. To satisfy the adjacency requirements of $\triangle_{ade}$, we can naturally assign it to vertex a to reduce the number of assigned vertices (see Fig. 10(b), where $\triangle_{abc} = \triangle_{123}$, $\triangle_{ade} = \triangle_{124}$, $a = 1$). Such assignment of the child $\triangle_{ade}$ is called *sharing assignment* because $\triangle_{ade}$ shares the same concave corner of a to satisfy its adjacency requirements. Certainly, $\triangle_{ade}$ has the free-

dom of being assigned to d or e, but the final floor-plan will have more nonrectangular modules, an undesirable feature (see Fig. 10(c), modules 1 and 4).

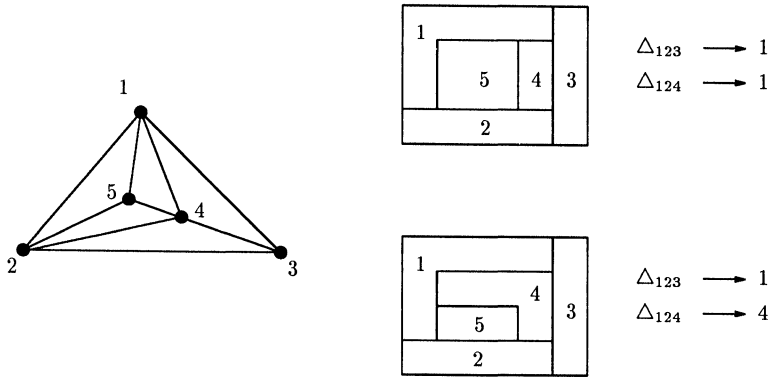

FIG. 10. *Sharing assignment:* (a) *adjacency graph,* (b) $\triangle_{123}$ *and* $\triangle_{124}$ *share vertex* 1; *and* (c) $\triangle_{123}$ *and* $\triangle_{124}$ *have independent assignments.*

When the child triangle $\triangle_{ade}$ shares an assignment with its parent, it does not add more bend to module a. Therefore, we do not increase the assignment count on vertex a when sharing occurs. All descendants of a CT can share its assignment as long as they have a common vertex. We will later see that sharing is necessary for the existence of 2-CRM floor-plans.

3. 2-CRM floor-plans. In this section, we first develop the properties of dual decomposition under a cut (to be defined). We then show that given a graph G with perfect assignment, we can construct a floor-plan where modules are 0-CRM, 1-CRM, or 2-CRM. The floor-plan construction is based on top-down recursive decomposition of G and a case analysis. The problem of floor-planning is thus reduced to finding a perfect assignment on G.

A *path* is an ordered set of vertices $(v_1, \ldots, v_n)$ in $G = (V, E)$, where $v_i \in V$, $i = 1, \ldots, n$, and $(v_i, v_{i+1}) \in E$, $i = 1, \ldots, n-1$. $v_1, \ldots, v_n$ are distinct. We consider the decomposition of G into two subgraphs G_1 and G_2. Let $P = (v_1 = u, \ldots, v_n = b)$ be a path in G that begins on vertex u and ends on vertex b. The path P *decomposes* G into left and right subgraphs G_l and G_r as shown in Fig. 11(a). Note that the vertices of P are duplicated in both subgraphs and vertices r' and l' are appended to G_l and G_r, respectively. G_l and G_r both conform to the configuration of Fig. 3. For a nontrivial decomposition, the path should be chosen such that G_l and G_r each contains at least one core vertex (of G) not in P. An edge (v_i, v_j) where $\mid i - j \mid \geq 2$ is called a *chord* (with respect to P). The existence of a chord will create a new CT in a subgraph when G is decomposed along P. If P does not contain chords and generates nontrivial decomposition of G to left and right subgraphs, we call P a *vertical cut.* A *horizontal cut* is defined similarly where upper and lower subgraphs are decomposed. A *cut* is either a vertical or horizontal cut.

In our algorithm, a floor-plan is constructed by recursive subdivision based on cuts that decompose G into G_1 and G_2. Such a decomposition has the properties stated in Lemma 1.

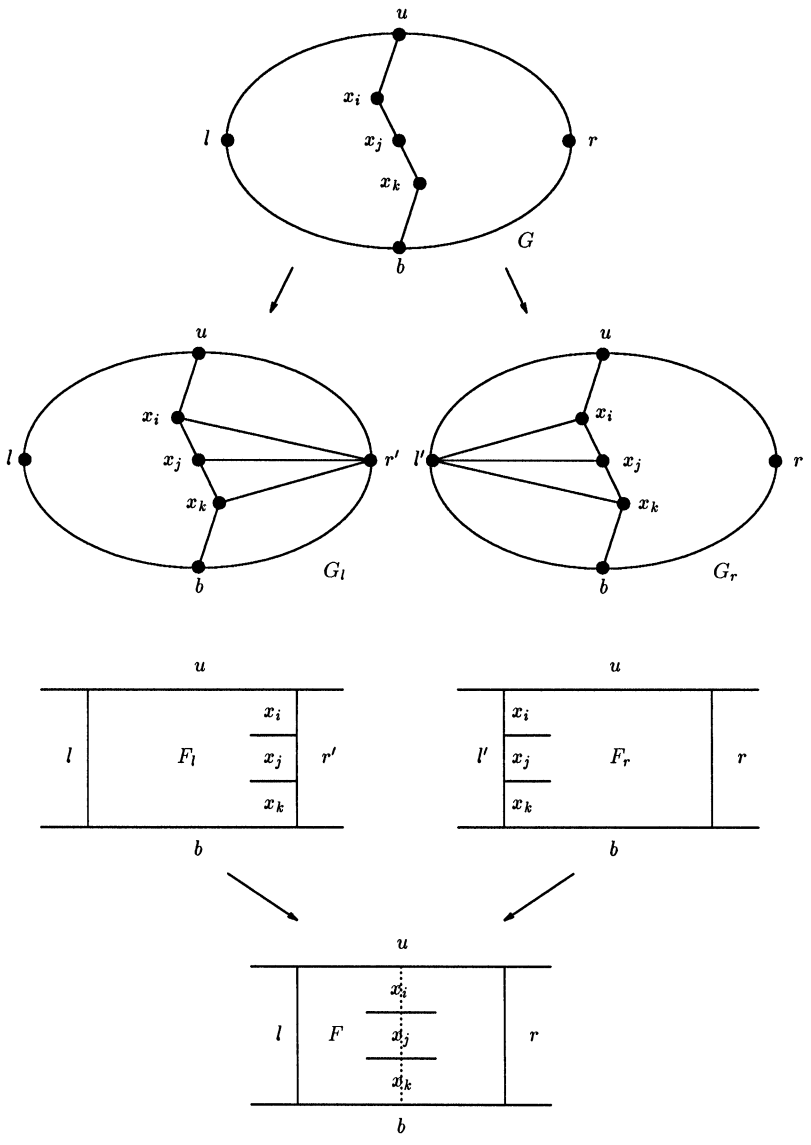

FIG. 11. *Decomposition by a cut:* (a) *decomposition of G to* G_l *and* G_r; *and* (b) *merging floor-plans of* G_l *and* G_r *to obtain the floor-plan of G.*

LEMMA 1. *Let* $P^* = (u = v_1, \ldots, v_n = b)$ *be a vertical cut in G where no edges (chords)* (v_i, v_j), $| i - j | \geq 2$ *exist. Let* G_l *and* G_r *be two subgraphs decomposed by* P^*, *then*

1. $CT(G) = CT(G_l) \cup CT(G_r)$ *and*
2. $CT(G_l) \cap CT(G_r) = \phi$,

where $CT(G)$ *denotes the set of CTs of G.*

Proof. There is no chord among vertices of P^*. Thus the introduction of vertices r' and l' will not create new CTs since r' and l' are only adjacent to vertices of P^*.

1. Since $G_l - r'$ and $G_r - l'$ are subgraphs of G, $CT(G) \supseteq CT(G_l) \cup CT(G_l)$ trivially holds. To show that $CT(G) \subseteq CT(G_l) \cup CT(G_r)$, we consider the converse: If it is true, there must be some CT $\triangle_{abc} \notin CT(G_l) \cup CT(G_l)$ but $\triangle_{abc} \in CT(G)$. The edges (a, b), (b, c), and (c, a) cannot appear simultaneously in G_l or G_r. Assume, without loss of generality, that (a, b) is in G_l and (b, c), (c, a) are in G_r. Since a, b appears in both subgraphs, $a, b \in P^*$. Let $a = v_i$ and $b = v_j$ for some integer i, j. If $\mid i - j \mid = 1$, $\triangle_{abc}$ will be in G_r, a contradiction. Thus $\mid i - j \mid \geq 2$ since a and b are distinct. But this would imply that (a, b) is a chord, another contradiction.

2. If $CT(G_l) \cap CT(G_r)$ is not empty, let $\triangle_{abc}$ be a CT of $CT(G_l) \cap CT(G_r)$. Since the common vertices of G_l and G_r are only the vertices of P^*, a, b, and c must be in P^*. Because P^* contains no chords, edges (a, b), (b, c), and (c, a) cannot form a CT. We reach a contradiction. □

A similar lemma applies to horizontal cuts. Intuitively, Lemma 1 says that the set of CTs in G can be decomposed into two disjoint sets by a cut. Such a decomposition preserves the "identity" of each CT in G, i.e., a complex triangle must be found in either G_1 or G_2, exclusively. Furthermore, no new CT is introduced by the decomposition nor does it destroy existing CTs. If (a, b) is an edge of some CT and $a = v_i$, $b = v_j$ are vertices of the cut, $\mid i - j \mid = 1$ must hold true. The lemma also suggests a method for recursive construction of floor-plans, which leads to our main result.

THEOREM 1. *If G has a perfect assignment, a 2-CRM floor-plan F exists with dual G_a and G is obtained from G_a by eliminating one edge from each biface of G_a.*

Proof. We prove the theorem by constructing an *augmented graph* G_a from the perfect assignment of G. By induction on the number of vertices of G_a, we generate a legal F-labeling on G_a. A 2-CRM floor-plan can thus be constructed.

For each assignment of CT $\triangle_{abc}$ to a, we chose an edge (a, x), $x \notin \{b, c\}$ in the CT and duplicate (a, x) to create a biface with an arrow pointing to vertex a. The resulting graph is G_a. The procedure is demonstrated in Fig. 12. There is a one-to-one correspondence between an assignment and a biface.

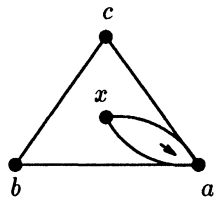

FIG. 12. *Transforming G to augmented graph G_a.*

We now show that there exists a legal F-labeling of G_a, thus implying the existence of a 2-CRM floor-plan.

Induction Basis: This is trivially true for G_a with five vertices.

Induction Hypothesis: There exists a legal F-labeling of G_a with no more than n vertices.

Induction Step: The initial labeling procedure is as follows:

1. All incident edges of u and b are labeled H.
2. All other unlabeled edges incident to l and r are labeled V.
3. If (x, y) is an edge where (u, x, y) or (b, x, y) is a face, label (x, y) as V.
4. If (x, y) is an edge where (l, x, y) or (r, x, y) is a face, label (x, y) as H.

In the labeling process, we only label the edges. The corners are implicitly labeled when their edges are labeled. The implicit rules for corner labeling are as follows:

1. If the two edges of a corner have identical labels, the corner is labeled **aligned** (180°).

2. If the two edges of a corner have distinct labels and there is an arrow pointing to the corner, the corner is labeled **concave** (270°); otherwise the corner is labeled **convex** (90°).

We first try to find a vertical cut P in G. The procedure is rather straightforward:

1. Path search. Find a path $P = (u = v_1, \ldots, v_n = b)$ where there is at least one core vertex on the left and right of P.

2. Chord elimination. If a chord $(v_i, v_j), (i < j)$ exists in P, remove vertices v_k from P, where $i < k < j$. Repeat this procedure until all chords in P are eliminated.

If the resulting path P^* meets the criteria of a vertical cut, we decompose the graph as shown in Fig. 11. Vertices of P^* are duplicated and the vertices r', l' and corresponding edges are added. When the vertices of P^* are duplicated, the assignments are carried along with corresponding CTs that contributed to the assignments. Thus if $\triangle$ was assigned to v and $\triangle$ appears on the left subgraph after decomposition, vertex v on the left subgraph will carry the assignment of $\triangle$. The duplicated vertex v on the right subgraph will not carry the assignment of $\triangle$. However, the vertex v on the right subgraph may carry assignment of other CTs. It can be easily seen that if G has a perfect assignment, the subgraphs also have perfect assignments. Thus the induction hypothesis applies since each subgraph contains one less vertex. A similar procedure is applied to a horizontal cut.

By Lemma 1, CTs are not decomposed by a cut. Thus an edge of a cut cannot be an edge of a biface because a biface only exists in a CT. The decomposition process is depicted in Fig. 11. The resulting floor-plan F is constructed by merging the left and right floor-plans F_l and F_r as shown. Modules (of F_l and F_r) corresponding to vertices in the vertical cut P^* are coalesced when F_l and F_r are merged.

If vertical or horizontal cuts do not exist, we proceed with the following case analysis on G_a:

Case A. At least one of the noncore vertices has degree 3.

Case B. All noncore vertices have degree at least 4.

CASE A. *At least one of the noncore vertices has degree* 3.

Without loss of generality, we assume r has degree 3. The configuration of G_a with its initial labeling is shown in Fig. 13. Consider the distinguished vertex v adjacent to r. Vertex v has at least three edges $(v, r), (v, u), (v, b)$. There are three possibilities for v:

A.1. v is not assigned.

A.2. v is assigned once.

A.3. v is assigned twice.

CASE A.1. *v is not assigned.*

We delete vertex r and its associated edges. We then relabel vertex v as r'. There is no biface incident to v since v is not assigned. All unlabeled incident edges of r' are labeled V except (u, r') and (b, r'), which have been labeled H. The resulting graph has one less vertex and thus the induction hypothesis applies. The F-labeling and corresponding floor-plans are demonstrated in Fig. 14. As before, thick edges have vertical orientation and thin edges have horizontal orientation. The orientations of dotted edges are not determined yet. One can easily verify that the F-labeling of vertex v does not violate the properties stated in §2.2.

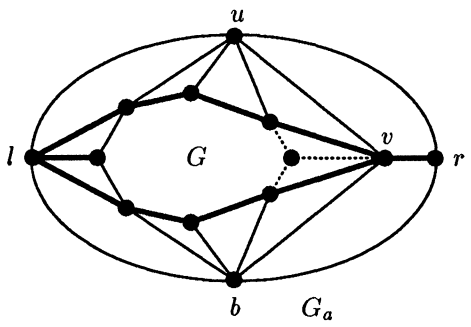

FIG. 13. *Graph of Case* A.

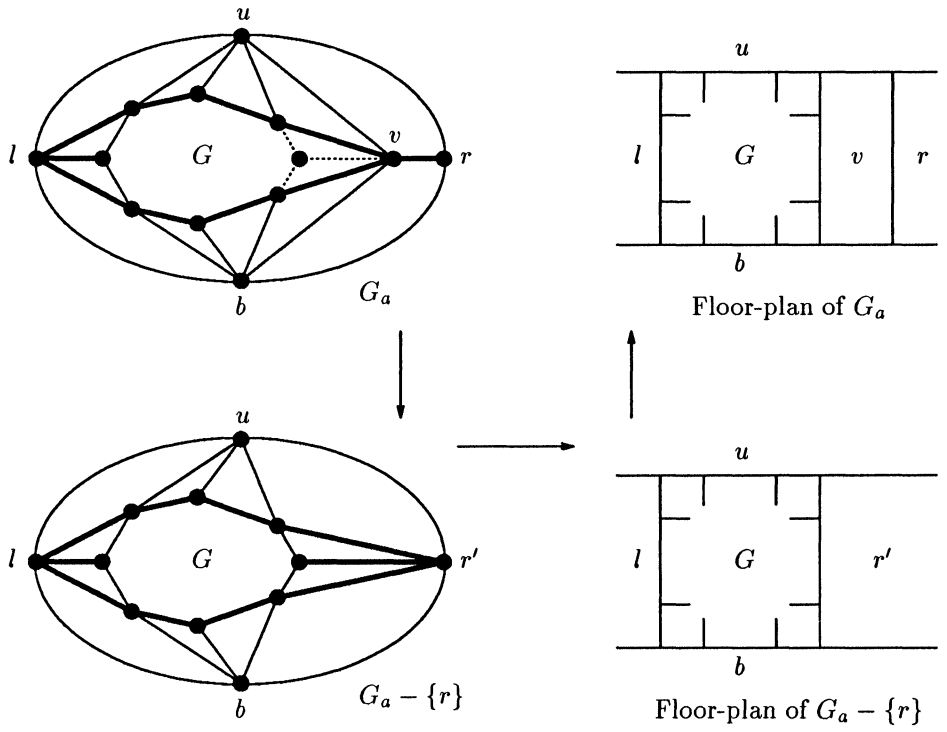

FIG. 14. *Floor-plans and duals in Case* A.1.

CASE A.2. *v is assigned once.*

Let (x_s, v) be the biface of the assignment of v. Refer to the configuration of G_a in Fig. 13. Let $P = \{u = x_1, \ldots, x_s, \ldots, x_p = b\}$ be the path from u to b in counterclockwise order where all vertices of P are adjacent to v. The general configuration of the graph is shown in Fig. 15.

The biface represents a concave corner of module v. Let $\{e_1 = (u, v), \ldots, e_s = (x_s, v), e_{s+1} = (x_s, v), \ldots, e_{p+1} = (b, v)\}$ be the set of edges in counterclockwise order with (e_s, e_{s+1}) as the biface. We claim that the other edges cannot form bifaces. Suppose

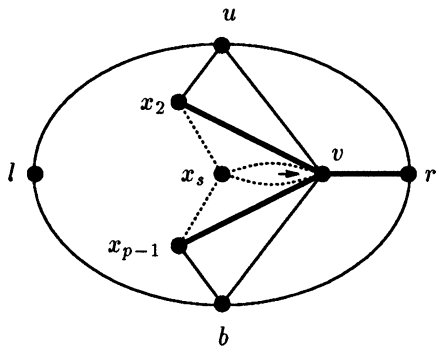

FIG. 15. *Graph of Case* A.2.

for contradiction that e_m, e_{m+1} is also a biface. If the convex corner of the biface is incident to v, v will have an assignment count of more than one. If the convex corner is incident to some other vertex, v must lie strictly inside some CT. From the configuration of Fig. 15, it is impossible for v to lie in any CT.

We perform a local transformation to the graph G_a so that the resulting graph has one less vertex. The transformation procedure is illustrated in Fig. 16. The initial labeling of G_a is shown on Fig. 16(a). We delete vertices r, b and their incident edges. We then split vertex v into two vertices and label them r' and b'. (See Fig. 16(c).) Edges $\{e_1, \ldots, e_s\}$ are now incident to r' and $\{e_{s+1}, \ldots, e_p\}$ are incident to b'. Edges (l, b') and (r', b') are also added. The unlabeled edges $\{e_2, \ldots, e_s\}$ are labeled V, and $\{e_{s+1}, \ldots, e_p\}$ are labeled H, as shown in Fig. 16(d).

Let G'_a be the resulting graph after the transformation. All CTs assigned to v, along with their bifaces, are disintegrated by the transformation process. From the floor-plan of Fig. 16(e), we see that the concave corner (biface) is eliminated by the introduction of edge (r', b'). The merging of the floor-plan of G' with 1-CRM module of v, is illustrated in Fig. 16(b). A degenerate case where $x_{p-1} = l$ is illustrated in Fig. 17. We can verify that the labeling of vertex v and P conforms to the properties stated in §2.2.

CASE A.3. *v is assigned twice.*

Refer to the general configuration of Fig. 13 again. Since v is assigned twice, there must be two bifaces incident to v. Let (x_{s1}, v) and (x_{s2}, v) be the bifaces. As before, we consider the vertices adjacent to v. Let $P = \{u = x_1, \ldots, x_{s1}, \ldots, x_{s2}, \ldots, x_p = b\}$ be the path from u to b in counterclockwise order where all vertices of P are adjacent to v. Also let $\{e_1, \ldots, e_{s1}, e_{s1+1}, \ldots, e_{s2+1}, e_{s2+2}, \ldots, e_{p+2}\}$ be the incident edges with bifaces (e_{s1}, e_{s1+1}) and (e_{s2+1}, e_{s2+2}). Since x_{s1} and x_{s2} are located in two independent CTs, $s2 \geq s1 + 2$ must hold true. Thus there is a vertex $x_s (s1 < s < s2)$ not located in any of the two CTs.

We claim that there are no vertices to the left of P. Otherwise, we can apply the chord elimination procedure to P to obtain the vertical cut P^* since vertices of P are adjacent to v. Because of the absence of such vertices, vertex x_{p-1} must be adjacent to l (by triangulation) or $x_{p-1} = l$. (If not, we would have a vertex z with edge (z, b) on the left of (x_{p-1}, b), where z is on the left of P.) We consider the general case where $x_{p-1} \neq l$ (Fig. 18) and treat the case where $x_{p-1} = l$ as a degenerate case (Fig. 19).

Consider the path $Q = (r, v, x_s, x_{s+1}, \ldots, x_{s2}, \ldots, x_{p-1}, l)$. Applying the chord elimination procedure (as described in the beginning of this proof) to the subpath Q'

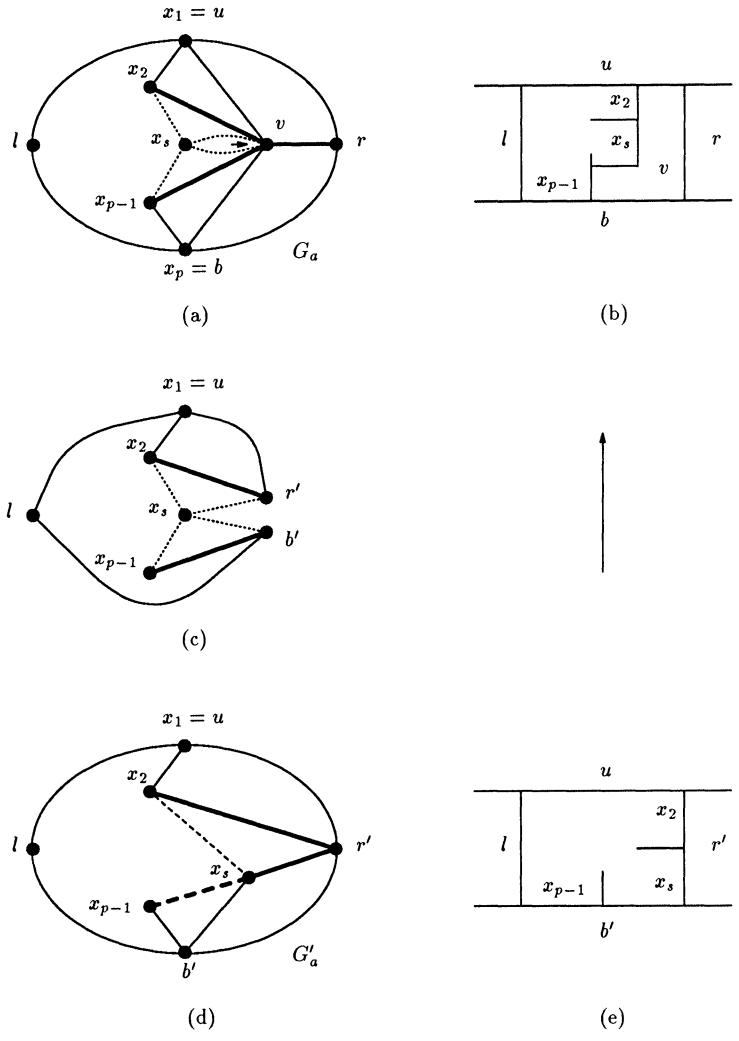

FIG. 16. *Transformation of G_a in Case* A.2.1.

$= (x_s, x_{s+1}, \ldots, x_{s2}, \ldots, x_{p-1}, l)$, we obtain a path $Q^* = (y_1 = r, y_2 = v, y_3 = x_s, \ldots, y_{q-1} = x_t, y_q = l)$. We have assumed that no horizontal cut existed in our case analysis implying Q^* is not a cut. However, all chords of Q^* must be incident to vertex v; otherwise they would have been eliminated by the chord elimination of Q' when constructing Q^*. Thus, the only condition that disqualify Q^* as a cut are the chords (y_j, v) for some j.

Despite the fact that Q^* is not a cut, we will decompose G_a along the horizontal path Q^* and give special treatment to the chords (y_j, v). Let the upper and lower subgraphs be G_u and G_b. The decomposition process is depicted in Fig. 18. Edges (y_i, b') are appended to G_u and edges (y_i, u') are appended to G_b, for $i = 1, \ldots, q$. Because of the chords (y_j, v), the addition of edges (y_j, u') in G_b results in new CTs $\triangle_{y_j v u'}$, which may not exist in G_a. However, from the figures, we can assign the concave corner of these

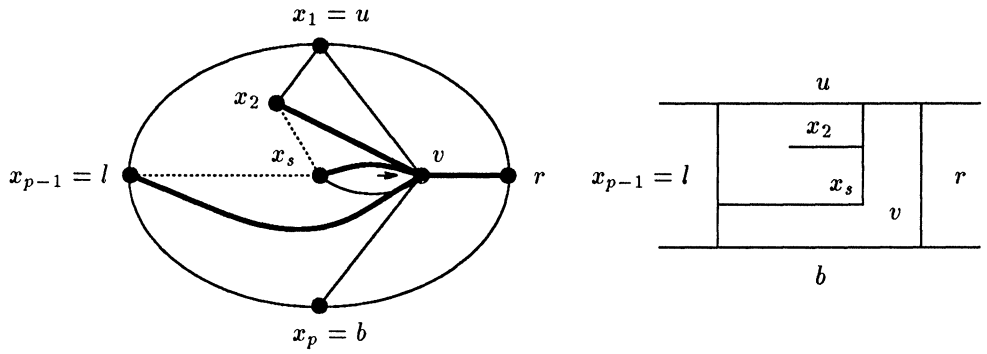

FIG. 17. *A degenerate graph of Case* A.2.

CTs to v. The assignment is a sharing assignment (see §2.6) using the concave corner of the biface (x_{s2}, v). Other CTs of G_a are handled as if Q^* is a cut. We can verify that the vertices along Q^* do not violate the properties given in §2.2.

Note that in some cases we may have $x_t = x_{p-1}$ or $x_t = x_s$ (see Fig. 18 for x_t), however, x_s and x_{p-1} are distinct by our construction. A degenerate example where $x_{p-1} = l$ is illustrated in Fig. 19.

CASE B. *All noncore vertices have degree at least* 4.

This is impossible as we shall show that we can construct a cut on G_a. The general configuration of the graph is shown in Fig. 20. Note that, there are at least four more distinct vertices v_1, v_2, v_3, v_4 in G_a as shown in the figure. We consider two possibilities:

B.1. Some of the edges (v_3, u), (v_3, r), and (v_1, b) exist.

B.2. None of the edges (v_3, u), (v_3, r), and (v_1, b) exist.

CASE B.1. *Some of the edges* (v_3, u), (v_3, r), *and* (v_1, b) *exist.*

Since the edges are topologically identical, we assume that (v_3, r) exists. The path (l, v_3, r) is a horizontal cut. Thus we have a contradiction.

CASE B.2. *None of the edges* (v_3, u), (v_3, r), *and* (v_1, b) *exists.*

We scan adjacent vertices of v_3 clockwise to obtain the path $(l, x_1, \ldots, x_m, b)$. Similarly, we scan adjacent vertices of b clockwise to obtain the path $(v_3, y_1, \ldots, y_n, r)$. Note that $x_m = y_1$ and $y_n = v_2$. We construct the path $P = (l, x_1, \ldots, x_m = y_1, \ldots, y_n, r)$ (see Fig. 21). Vertex u cannot appear in P due to the nonexistence of (v_3, u). Also, v_1 cannot appear in P, otherwise we have a horizontal cut (l, v_3, v_1, r). We apply the chord elimination procedure to P and obtain a horizontal cut P^*. This is another contradiction. □

From Theorem 1, the problem of the existence of a 2-CRM floor-plan is reduced to finding a perfect assignment in G. For a given G, there may be more than one perfect assignment. Different assignments correspond to different floor-plans. In the next section, we present an algorithm to obtain a perfect assignment in G.

4. An algorithm for perfect assignment. In this section, we first demonstrate that assignment sharing is necessary for the existence of a 2-CRM floor-plan. We then propose a hierarchical assignment algorithm to obtain a perfect assignment in a PTG. The algorithm can be easily adapted to the extended dual G, as described in §4.3. Finally, we present an example.

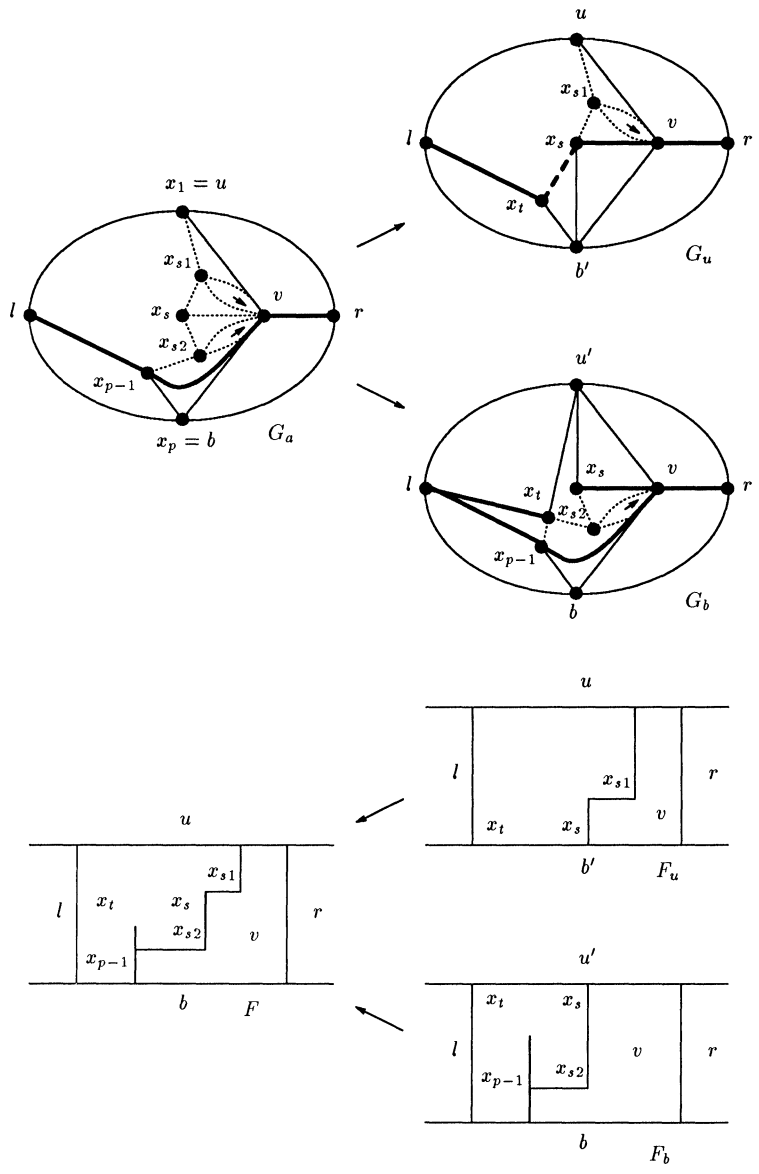

FIG. 18. *Dual decomposition of Case* A.3.

4.1. Necessity of sharing assignments. Sharing assignment refers to the assignment of a child CT that uses the concave corner of its parent CT to satisfy its adjacency requirements. The notion of sharing has been discussed in §2.6, and Fig. 10 illustrates an example of a sharing assignment. An assignment can be shared by more than one generation of CTs. Sharing reduces the number of nonrectangular modules in a floor-plan. In some graphs, sharing is even necessary to achieve perfect assignments.

Consider a series of PTGs G_i as shown in Fig. 22. G_n is obtained by adding a vertex and three edges to each bounded face of G_{n-1}. The containment tree of G_n is a full

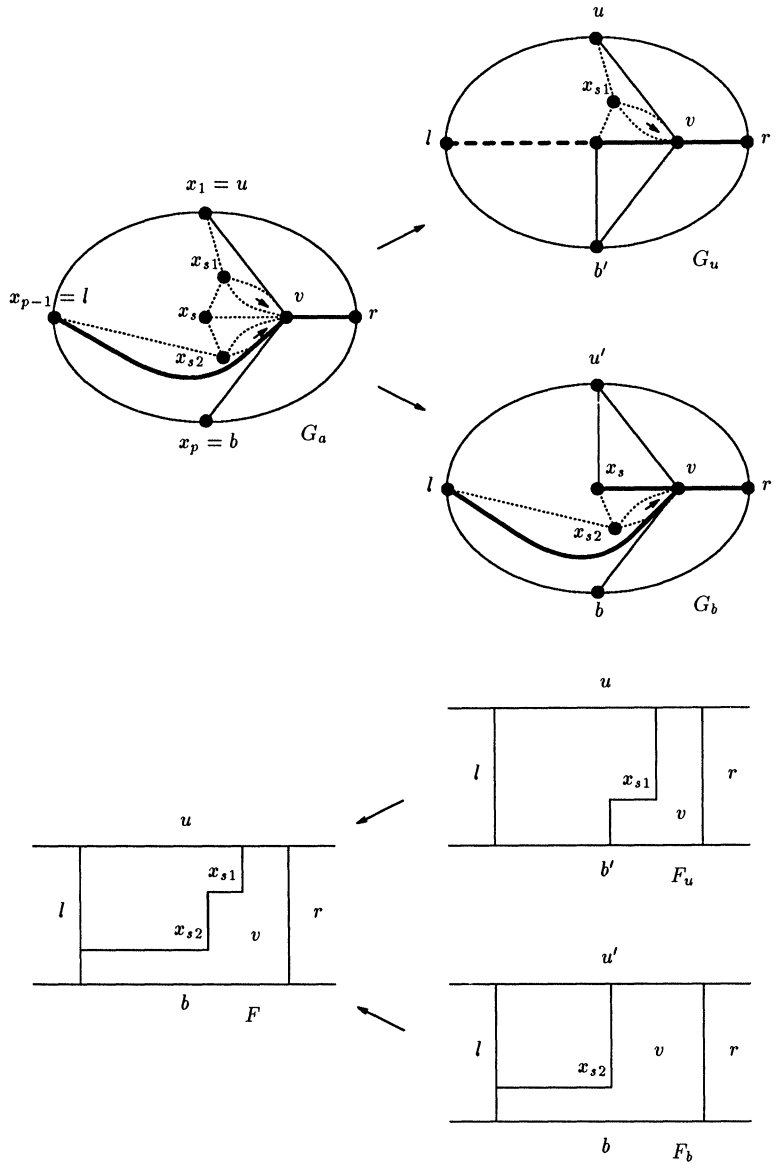

FIG. 19. *A degenerate example of Case* A.3.

ternary tree. Let $f(n)$ denote the number of bounded faces in G_n and $CT(G_n)$ denote the number of CTs in G_n. From G_n to G_{n+1}, each bounded face of G_n becomes a CT of G_{n+1}. Because $f(n) = 3^n$, we have

$$CT(G_{n+1}) = CT(G_n) + 3^n, \qquad n \geq 0,$$
$$CT(G_0) = 0.$$

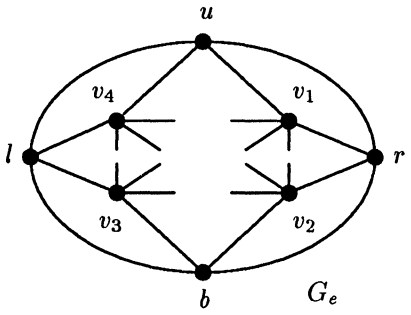

FIG. 20. *Graph of Case* B.

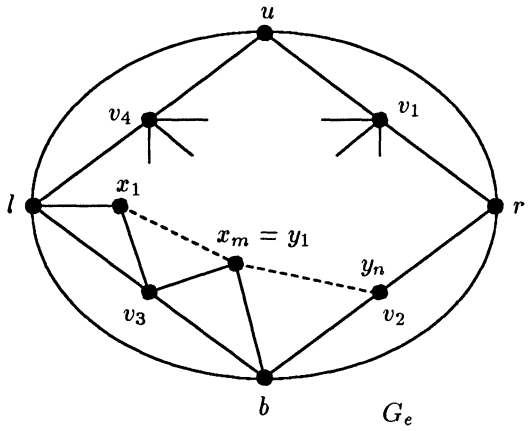

FIG. 21. *Graph of Case* B.2.

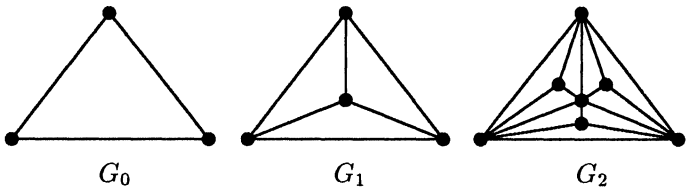

FIG. 22. *Family of graphs to show the necessity of sharing assignment.*

Solving the recurrence equations gives

$$CT(G_n) = (3^n - 1)/2,$$

which is also the number of nodes in the containment tree of G_n. Let $V(G_n)$ denote the number of vertices available for assignment in G_n (this number is not equal to the number of vertices in G_n). For $n \geq 1$, we have

$$V(G_{n+1}) = V(G_n) + 3^{n-1}, \qquad n \geq 1,$$
$$V(G_1) = 3.$$

Solving the recurrence equations gives

$$V(G_n) = (3^{n-1} + 5)/2.$$

Since each vertex can carry at most two assignments, the maximum number of assignments $A(G_n)$ in G_n is

$$A(G_n) = 2V(G_n) = 3^{n-1} + 5.$$

For $n \geq 4$, $CT(G_n) > A(G_n)$. If each assignment of a CT is counted once, it is impossible to obtain a perfect assignment since the number of CTs outnumber that of available assignments. This is also intuitively evident from the fact that each new vertex introduces three CTs but only two new assignments.

We thus see that sharing is necessary to achieve a perfect assignment. When a CT shares its assignment with its parent, we do not increase the assignment count on the vertex because sharing does not result in more complicated shapes. The assignment of a vertex is counted twice only if it is assigned by two independent CTs (i.e., one does not contain the other and vice versa). From the practical point of view, sharing should be encouraged whenever possible since we prefer simpler shaped modules.

4.2. Hierarchical assignment algorithm for PTG. The hierarchy of CTs suggests a recursive assignment procedure. We proceed in breadth-first manner in the containment tree. At each step of the recursion, we only consider the CTs that have identical parent. Let G_i be the graph induced by the CTs in step i, i.e., the CTG of siblings having identical parent $\triangle$. We also call $\triangle$ the parent of G_i. We assume that G_i is connected. If not, our algorithm will be applied to each connected component of G_i. After G_i is assigned, the algorithm is recursively applied to the children of each CT of G_i in the containment tree.

We define the following terms on G_i: An edge (a, b) is called a *boundary edge* if it is an edge of the infinite face of G_i; otherwise it is called an *internal edge*. A vertex is called a *boundary vertex* if there are some boundary edges incident to it; otherwise, it is called an *internal vertex*. From the definition, the following properties hold on graph G_i:

P1. A boundary edge has exactly one triangle on one side.

P2. A boundary vertex has at least two boundary edges incident to it.

P3. A boundary vertex is adjacent to at least two other boundary vertices.

We call a vertex *saturated* if it has already been assigned twice. Consider vertices of G_i. Let $\triangle_{abc}$ be the the parent of G_i. Vertices a, b, c may appear in G_i. In the worst case, all of them may be saturated due to previous recursive assignment steps. Suppose $\triangle_{abc}$ is assigned to vertex a in the previous recursive step. Because of the sharing scheme, vertex a is allowed to carry one more assignment in G_i. Therefore, even if the vertex is already saturated from its previous assignments, we can *force* (artificially set the assignment count) it to take one more assignment in G_i. Thus at most two vertices (b and c) are saturated in G_i, and both of them are boundary vertices. In fact, the vertex a is also allowed to carry assignments of any descendant of $\triangle_{abc}$ as long as the assignments are shared.

As the algorithm proceeds, we delete edges and vertices of G_i that do not have unassigned triangles incident to them. We always maintain the following invariant properties:

Q1. At most two boundary vertices are saturated. The other boundary vertices may have zero or one assignment.

Q2. Internal vertices have zero assignment.

We choose an unsaturated vertex v from the set of boundary vertices. Such a vertex always exists since a nontrivial G_i has at least three boundary vertices, of which at most two are saturated. We scan the vertices adjacent to v counterclockwise. Let $v_0, \ldots, v_n$, $n \geq 1$ be the vertices, where (v, v_0) and (v, v_n) are boundary edges. (Note that v_0 and v_n are boundary vertices.) There are at most n triangles $\triangle_1, \ldots, \triangle_n$ incident to v, where $\triangle_i$ is $\triangle_{v_{i-1}\, v_i\, v}$. There are two possible cases:

Case 1. $v_1, \ldots, v_{n-1}$ are all internal vertices.

Case 2. Some of $v_1, \ldots, v_{n-1}$ are boundary vertices.

CASE 1. *$v_1, \ldots, v_{n-1}$ are all internal vertices.*

If $v_1, \ldots, v_{n-1}$ are all internal vertices, we will assign $\triangle_i$ to v_i for $i = 1, \ldots, n-1$ and assign $\triangle_n$ to v. If a $\triangle_i$ does not exist, we simply ignore its assignment. We delete vertex v and edges $(v, v_0), \ldots, (v, v_n)$ since all incident triangles have been assigned. Each of the edges (v_i, v_{i+1}), $i = 0, \ldots, n-1$ and vertices v_i, $i = 0, \ldots, n$ is deleted if it has no more incident triangles. The remaining vertices in $\{v_1, \ldots, v_{n-1}\}$ are added to the set of boundary vertices. The resulting graph is smaller and the invariant properties Q1, Q2 are maintained. If G_i is not connected, the assignment algorithm is applied to each connected component. The vertex v, which has been deleted, is assigned at most twice. The procedure is shown in Fig. 23.

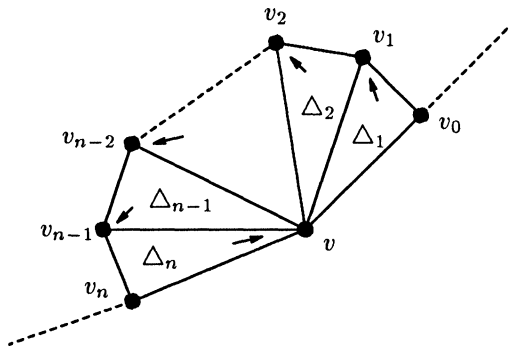

FIG. 23. *Assignment when $v_1, \ldots, v_{n-1}$ are internal vertices.*

CASE 2. *Some of $v_1, \ldots, v_{n-1}$ are boundary vertices.*

In this case, we will not make any assignment. Instead we will decompose G_i into two smaller subgraphs where the invariant properties Q1, Q2 are maintained in both subgraphs. Let v_m $(1 \leq m \leq n-1)$ be the boundary vertex with the smallest index m. Consider the edge (v, v_m) where v and v_m are boundary vertices. The edge (v, v_m) decomposes G_i into two subgraphs G_{ir} and G_{il} with (v, v_m) appearing in both subgraphs. There are two cases:

Case 2.1. v_m is saturated.

Case 2.2. v_m is not saturated.

CASE 2.1. *v_m is saturated.*

There are only two saturated vertices in G_i and by our selection, v is not saturated. Without loss of generality, we can assume that the other saturated vertex v_s is in G_{il}. We *force* vertex v in G_{ir} to be saturated, thus not allowing triangles in G_{ir} to be assigned

to v. G_{ir} has exactly two saturated vertices: v_m and v (forced saturation), so does G_{il}: v_m and v_s. Vertex v in G_{il} will keep whatever assignment it carries in G_i but it will not be assigned by triangles in G_{ir}. Thus vertex v in G_i will not be oversaturated due to the forced saturation of v in G_{ir}. The decomposition is depicted in Fig. 24.

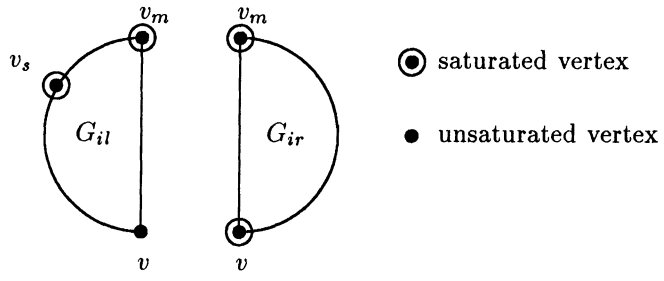

FIG. 24. *Decomposition when v_m is saturated.*

CASE 2.2. *v_m is not saturated.*

Let the two saturated vertices be v_{s1} and v_{s2}. v_{s1} (v_{s2}) appears in exactly one of the subgraphs after decomposition. If $v_{s1} \in G_{ir}$ and $v_{s2} \in G_{ir}$, we force v and v_m of G_{il} to be saturated. If $v_{s1} \in G_{ir}$ and $v_{s2} \in G_{il}$, we force v of G_{il} and v_m of G_{ir} to be saturated. If $v_{s1} \notin G_{ir}$ and $v_{s2} \notin G_{ir}$, we force v and v_m of G_{ir} to be saturated. In any case, G_{ir} and G_{il} have exactly two saturated vertices. v (v_m) of G_{il} is saturated if and only if v (v_m) of G_{ir} is not saturated. Thus v and v_m will not be oversaturated in G_i. The decomposition is shown in Fig. 25.

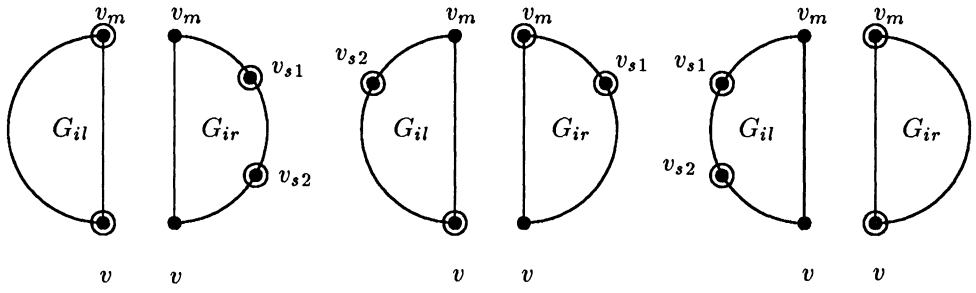

FIG. 25. *Decomposition when v_m is not saturated.* (a) $v_{s1}, v_{s2} \in G_{ir}$, (b) $v_{s1} \in G_{ir}$, $v_{s2} \in G_{il}$, (c) $v_{s1}, v_{s2} \notin G_{ir}$.

From the above discussion, we either eliminate some triangles or decompose G_i into smaller subgraphs. Thus, we have an assignment algorithm on G_i with the invariant properties as induction hypothesis. If G_i has only one triangle, the assignment problem is trivial.

A formal description of the algorithm is as follows:

ALGORITHM ASSIGN($CTG, TREE, root$)
INPUT: CTG is a complex triangle graph of a planar triangulated graph and $TREE$ is the corresponding containment tree. $root$ is the parent node of the set of complex triangles to be assigned in the current recursive step.
OUTPUT: A perfect assignment $ASSIGN = \{(\triangle_1, v_1), \ldots, (\triangle_n, v_n)\}$, where $\triangle_i = \triangle_{a_i b_i c_i}$ is a descendant complex triangle of $root$ and $v_i \in \{a_i, b_i, c_i\}$.

BEGIN
Let $root$ be $\triangle_{abc}$ and assume it was assigned to a.
Find the subgraph $G_i = (V_i, E_i)$ induced by the children of $root$.
Force $v_{s1} = b$ and $v_{s2} = c$ to be saturated.
If a is saturated, force a to have an assignment count of 1.
$ASSIGN = \phi$.
For each connected component G_{ic} of G_i **do**
Let $G_{ic} = (V_{ic}, E_{ic})$.
$B =$ **CONNECTED_ASSIGN**$(G_{ic}, \{b, c\} \cap V_{ic})$.
$ASSIGN = ASSIGN \cup B$.
End For.
For each $\triangle_{xyz} \in G_i$ **do**
Let G_{is} be the subgraphs induced by children of $\triangle_{xyz}$.
$C =$ **ASSIGN**$(G_{is}, TREE, \triangle_{xyz})$.
$ASSIGN = ASSIGN \cup C$.
End For.
Return ($ASSIGN$).
END algorithm.

PROCEDURE CONNECTED_ASSIGN(G, $SATURATED$)
INPUT: $G = (V, E)$ is a connected planar graph induced by children of a node in the containment tree. $SATURATED \subset V$ is the set of saturated boundary vertices of G, $| SATURATED | \leq 2$.
OUTPUT: A perfect assignment $A = \{(\triangle_1, v_1), \ldots, (\triangle_n, v_n)\}$, where $\triangle_i = \triangle_{a_i b_i c_i}$ is a complex triangle of G and $v_i \in \{a_i, b_i, c_i\}$.

BEGIN
Find the set of boundary edges $E_b \subseteq E$.
Find the set of boundary vertices $V_b \subseteq V$.
$A = \phi$.
Find a vertex $v \in (V_b - SATURATED)$.
Let $v_0, \ldots, v_n$, $n \geq 1$ be the vertices where $(v, v_0), (v, v_n) \in E_b$ and $v_0, v_n \in V_b$.
If $\{v_1, \ldots, v_{n-1}\}$ are all internal vertices **then**
/* CASE 1, assign $\triangle_i$ to v_i and $\triangle_n$ to v */
Let $\triangle_i = \triangle_{v_{i-1}\, v_i\, v}$, $i = 1, \ldots, n$.
For $i = 1$ to $n - 1$ **do**
$A = A \cup \{(\triangle_i, v_i)\}$ if $\triangle_i$ exists.
$A = A \cup \{(\triangle_n, v)\}$ if $\triangle_n$ exists. /* last triangle */
Delete vertex v and incident edges.
Delete vertex v_i, $i = 0, \ldots, n$ from G if it has no more incident triangle.
Delete edge (v_i, v_{i+1}), $i = 0, \ldots, n - 1$ from G if it has no more incident triangle.
For each connected component $G_c = (V_c, E_c)$ of the remaining graph **do**

$A = A \cup$ **CONNECTED_ASSIGN**$(G_c, SATURATED \cap V_c)$.

Else

/* CASE 2, decompose G into G_l and G_r */
Let v_{s1} and v_{s2} be two vertices of $SATURATED$.
Let $v_m \in \{v_1, \ldots, v_{n-1}\} \cap V_b$ where $v_1, \ldots, v_{m-1} \notin V_b$.
$SATURATED\_L = \phi$.
$SATURATED\_R = \phi$.
Decompose G into G_l and G_r with $(v, v_m) = G_l \cap G_r$.
If $v_m \in SATURATED$ **then**
/* CASE 2.1 */
Let $v_{s1} = v_m$, without loss of generality.
If v_{s2} in G_l **then**
$SATURATED\_L = \{v_{s2}, v_m\}$.
$SATURATED\_R = \{v, v_m\}$.
Else
$SATURATED\_L = \{v, v_m\}$.
$SATURATED\_R = \{v_{s2}, v_m\}$.
End If.
Else /* $v_m \notin SATURATED$ */
/* CASE 2.2 */
If $v_{s1} \in G_r$ **then**
$SATURATED\_R = SATURATED\_R \cup \{v_{s1}\}$.
$SATURATED\_L = SATURATED\_L \cup \{v\}$.
Else
$SATURATED\_R = SATURATED\_R \cup \{v\}$.
$SATURATED\_L = SATURATED\_L \cup \{v_{s1}\}$.
End If.
If $v_{s2} \in G_l$ **then**
$SATURATED\_R = SATURATED\_R \cup \{v_m\}$.
$SATURATED\_L = SATURATED\_L \cup \{v_{s2}\}$.
Else
$SATURATED\_R = SATURATED\_R \cup \{v_m\}$.
$SATURATED\_L = SATURATED\_L \cup \{v_{s2}\}$.
End If.
End If.
$B =$ **CONNECTED_ASSIGN**$(G_l, SATURATED\_L)$.
$C =$ **CONNECTED_ASSIGN**$(G_r, SATURATED\_R)$.
$A = A \cup B \cup C$.
End if.
Return (A).
END procedure.

Notice that the sharing assignment scheme is implicitly incorporated by forcing vertex a of $\triangle_{abc}$ in **ASSIGN()** to reduce its assignment count. Data structures that support **ASSIGN()** and **CONNECTED_ASSIGN()** are relatively simple. A doubly connected edge list [8] can be used to find CTs incident to a vertex. The algorithm visits each CT without backtracking. Thus the complexity of the assignment algorithm is $O(c)$, where c is the number of CTs. c is $O(|V|)$ in a planar graph. Therefore the overall time complexity is $O(|V|)$ and we have the following theorem:

THEOREM 2. *Given a PTG, a perfect assignment can be constructed in $O(|V|)$ time, where $|V|$ is the number of vertices of the PTG.*

4.3. Perfect assignment for extended graph. In general the algorithm will make an assignment to any nonsaturated vertex of G_i. We have pointed out that in a perfect assignment of G, the four noncore vertices r, u, l, b are not allowed to carry assignment. This may violate the invariant properties because in the worst case, we may have four saturated vertices to begin with. However, we know that all core vertices have an assignment count of zero initially. We can exploit this fact to decompose G into two subgraphs satisfying the invariant properties.

Given a biconnected PTG G_c, we construct an extended graph G by adding the vertices r, u, l, b. We force the vertices r, u, l, b to be saturated. Consider the graph $G' = G - \{(r,u),(u,l),(l,b),(b,r)\}$. Since G_c is biconnected, the edges (r,u), (u,l), (l,b), and (b,r) do not appear in any CT of G. Therefore the CTG of G' and that of G are identical, and we will only consider decomposition of G'.

If each connected component of CTG $CTG(G')$ contains no more than two saturated vertices, we are done. Suppose a connected component of $CTG(G')$ contains three or more saturated vertices. We identify the first boundary vertex v_r of G', which is clockwise adjacent to r. Similarly we find v_l. (See Fig. 26.) Since G_c is biconnected, v_r and v_l must be distinct. We find a path $P = (v_r, \ldots, v_l)$ in G' and construct a path P^* by applying the chord elimination procedure. By our selection, v_l and v_r are located at the infinite face of G'. We cut G' into G'_1 and G'_2 along the path P^*, as shown in Fig. 26. $CTG(G'_1)$ ($CTG(G'_2)$) has at most two saturated vertices since G'_1 (G'_2) has at most two saturated vertices. Vertices of P^* in $CTG(G'_1)$ and $CTG(G'_2)$ are forced to have assignment count of one. Therefore vertices of P^* in $CTG(G')$ will not be oversaturated after we apply the assignment algorithm to $CTG(G'_1)$ and $CTG(G'_2)$.

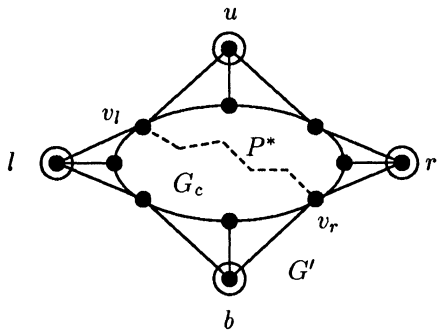

FIG. 26. *Initial decomposition of G.*

THEOREM 3. *Given any extended graph G constructed from a biconnected PTG G_c, a perfect assignment can be constructed in $O(|V|)$ time, where $|V|$ is the number of vertices of G_c.*

Note that the biconnected condition of G is needed only for the initial decomposition of G'. The assignment algorithm is still valid for some classes of weakly connected core graphs as long as the initial decomposition exists. However, not all extended graphs of planar graphs can be decomposed in this manner. Figure 27 is an example of such a graph.

As a direct result of Theorems 1 and 3, we have the follwing theorem:

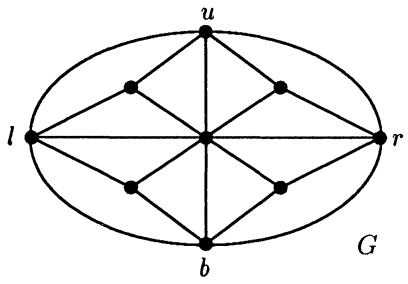

FIG. 27. *An example of G that does not have a perfect assignment.*

THEOREM 4. *Given any biconnected PTG G_c with $|V|$ vertices, a 2-CRM floor-plan F can be constructed in $O(|V|)$ time, where the dual of F has core G_c when parallel edges are eliminated.*

Example. We demonstrate an example of the construction process using the graph of Fig. 8, which does not admit any 1-CRM floor-plan. The results are shown in Fig. 28: (a) is the core graph and (b) shows an extended graph. (c) is a perfect assignment and (d) gives the 2-CRM dual floor-plan constructed with the perfect assignment in (c).

5. Conclusion. We have presented a linear time algorithm for floor-plan construction using a graph dualization technique. By allowing more general shapes, 1-CRM and 2-CRM, we have shown that all adjacency requirements of a PTG can be satisfied on a rectangular chip. The construction ensures that no unnecessary adjacency is added in the final floor-plan, thus paves the way for a more compact layout. If the input graph is planar but not triangulated, a floor-plan is still possible by introducing trivial adjacency. The most complicated shape created is 2-CRM but most modules are rectangular. The 1-CRM and 2-CRM can be easily incorporated in the widely used rectilinear systems. The expected number of CTs for a randomly generated PTG is approximately 16% of the number of vertices [14].

The algorithm presented serves as a theoretical basis for floor-planning based on the graph dualization approach. For practical applications, other requirements such as area, aspect ratio, and perimeter constraints should be considered. We are currently looking at the applications of the results obtained from this study. Sizing and aspect ratio is one of our major focuses. Only when such issues are resolved could one propose dualization-based floor-planning as a practical alternative. The floor-plan sizing problem has been shown to be NP-complete for general rectangular floor-plans [9]. Future research is focused on approximate solutions or restriction to special classes of floor-plans, for example, sliceable floor-plans, which are more promising in practice. More information can be found in [7] and in Chapter 7 of [5].

One interesting problem is to enumerate all floor-plans with identical duals. For rectangular floor-plans, the problem have been studied [3], [13]. For 1- and 2-CRM floor-plans, the problem has yet to be studied. Planarization of the circuit is especially not well understood. Current solutions are based on heuristic approaches [6]. Many problems related to graph dualization need to be investigated even for rectangular floor-plans.

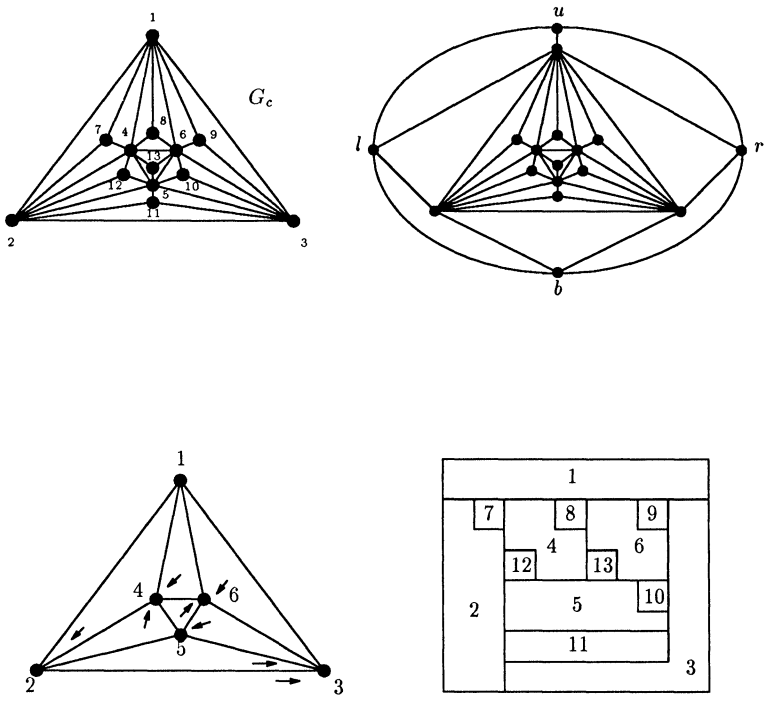

FIG. 28. *An example of dual construction:* (a) *core graph* G_c; (b) *an extended graph constructed from* G_c; (c) *a perfect assignment; and* (d) *a floor-plan constructed from the perfect assignment in* (c).

REFERENCES

[1] J. BHASKER AND S. SAHNI, *A linear algorithm to find a rectangular dual of a planar triangulated graph*, Algorithmica, 3 (1988), pp. 247–278.

[2] K. KOZMINSKI AND E. KINNEN, *Rectangular dual of planar graphs*, Networks, 15 (1985), pp. 145–157.

[3] ———, *Rectangular dualization and rectangular dissection*, IEEE Trans. Circuits and Systems, 35 (1988), pp. 1401–1416.

[4] Y. T. LAI AND S. M. LEINWAND, *Algorithms for floor-plan design via rectangular dualization*, IEEE Trans. Computer-Aided Design, 7 (1988), pp. 1278–1289.

[5] T. LENGAUER, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, New York, 1990.

[6] B. LOKANATHAN AND E. KINNEN, *Performance optimized floor planning by graph planarization*, Proc. of 26th Design Automation Conference, 1989, pp. 116–121.

[7] R. H. J. M. OTTEN, *Automatic floorplan design*, Proc. of 19th Design Automation Conference, 1982, pp. 261–267.

[8] F. PREPARATA AND M. SHAMOS, *Computational Geometry—An Introduction*, Springer-Verlag, Berlin, New York, 1985.

[9] L. STOCKMAYER, *Optimal orientation of cells in slicing floorplan designs*, Inform. and Control, 57 (1983), pp. 91–101.

[10] Y. SUN AND M. SARRAFZADEH, *Floorplanning by graph dualization: L-shaped models*, Proc. of IEEE International Symposium on Circuits and Systems, 1990, pp. 2845–2848; also to appear in Algorithmica.

[11] Y. SUN AND K. H. YEAP, *Edge covering of complex triangles in rectangular dual floorplanning*, J. Circuits Systems Comput., to appear.

[12] S. TSUKIYAMA, K. KOIKE, AND I. SHIRAKAWA, *An algorithm to eliminate all complex triangles in a maximal planar graph for use in VLSI floor-plan*, Proc. of IEEE International Symposium on Circuits and Systems, 1986, pp. 321–324.

[13] S. TSUKIYAMA, K. TANI, AND T. MARUYAMA, *A condition for a maximal planar graph to have a unique rectangular dual and its application to VLSI floor-plan*, Proc. of IEEE International Symposium on Circuits and Systems, 1989, pp. 931–934.

[14] K. H. YEAP AND M. SARRAFZADEH, *A theorem of sliceability*, 2nd Great Lakes Computer Science Conference, Western Michigan University, Kalamazoo, MI, 1991.