TCG: A Transitive Closure Graph-Based Representation for General Floorplans

Jai-Ming Lin and Yao-Wen Chang, *Member, IEEE*

Abstract—**In this paper, we introduce the concept of the P*-admissible representation and propose a P*-admissible, transitive closure graph-based representation for general floorplans, called *TCG*, and show its superior properties. TCG combines the advantages of popular representations such as sequence pair, BSG, and B*-tree. Like sequence pair and BSG, but unlike O-tree, B*-tree, and CBL, TCG is P*-admissible. Like B*-tree, but unlike sequence pair, BSG, O-tree, and CBL, TCG does not need to construct additional constraint graphs for the cost evaluation during packing, implying faster runtime. Further, TCG supports incremental update during operations and keeps the information of boundary modules as well as the shapes and the relative positions of modules in the representation. More importantly, the geometric relation among modules is transparent not only to the TCG representation but also to its *operations*, facilitating the convergence to a desired solution. All these properties make TCG an effective and flexible representation for handling the general floorplan/placement design problems with various constraints. Experimental results show the promise of TCG.**

Keywords: Floorplanning, Layout, Physical Design, Transitive Closure Graph

I. Introduction

As technology advances, circuit sizes and design complexity in modern VLSI design are increasing rapidly. To handle the design complexity, hierarchical design and reuse of IP modules become popular, which makes floorplanning/placement much more important than ever. The major objective of floorplanning/placement is to allocate the modules of a circuit into a chip to optimize some design metric such as area and timing. The realization of floorplanning/placement relies on a representation which describes geometric relations among modules. The representation has a great impact on the feasibility and complexity of floorplan designs. Thus, it is of particular significance to develop an efficient, effective, and flexible representation for floorplan/placement designs.

A. Previous Work

There exist a few floorplan representations in the literature, e.g., [1], [2], [3], [7], [8], [9], [10], [11], [12], [13], [17], [18]. We shall first review these representations and the types of floorplans that they can represent. A *slicing floorplan* is one of the simplest type of floorplans. A slicing structure can be obtained by recursively cutting rectangles horizontally or vertically into smaller rectangles; it is a non-slicing structure, otherwise. Otten first proposed a binary-tree representation for slicing floorplans [11]. Wong and Liu later in [18] presented a normalized Polish expression (NPE for short) to represent a slicing floorplan. The slicing structure has several advantages such as smaller solution space, implying faster runtime for floorplan design. However, most of real designs are non-slicing. Researchers in [12] and [17] attempted to extend the tree representation to the non-slicing floorplans with special topologies, e.g., the wheel structure.

For the non-slicing floorplan structure, there exist several well-known "old" graph-based representations. Ohtsuki in [9] used a pair of horizontal and vertical directed acyclic graphs, named *polar graphs*, to represent a topological placement. Other representations such as *adjacency graphs*, and *channel intersection graph* are also widely used [14]. Recently, Onodera et al. in [10] used a branch-and-bound method to exhaustively search an optimal solution for module placement. Since the method is quite time-consuming, the size of tractable modules is limited.

The non-slicing floorplan representations have attracted much attention in the literature recently, e.g., *sequence pair* [7], *bounded sliceline grid* [8], *O-tree* [2], *B*-tree* [1], and *corner block list* [3]. Murata et al. in [7] used two sequences (Γ_+, Γ_-) of module names, called sequence pair (SP for short), to represent the geometric relations among modules. They defined the *P-admissible solution space*, which satisfies the following four requirements [7]:

(1) the solution space is finite,

(2) every solution is feasible,

(3) packing and cost evaluation can be performed in polynomial time, and

(4) the best evaluated packing in the space corresponds to an optimal placement.

(By this definition, the slicing tree is not a P-admissible representation since many optimal floorplans are non-slicing.) SP is P-admissible and is flexible for general floorplan/placement design; however, it is harder to handle the floorplan/placement problems with position constraints, e.g., boundary modules, pre-placed modules, range constraints, etc. Further, two constraint graphs need to be constructed for cost evaluation for each perturbation, consuming a significantly larger running time. Tang and Wong [16] recently presented an efficient packing scheme, called *FAST-SP*, to evaluate the cost of a sequence pair by computing its common subsequence. Nakatake et al. in [8] proposed a flexible bounded sliceline grid representation, called BSG. BSG is also P-admissible. However, BSG itself has many redundancies since there could be multiple representations corresponding to one packing, implying a larger solution space and thus longer search time to find an optimal solution.

For tree-based methods, Guo et al. in [2] proposed the O-tree representation for a left and bottom *compacted* placement.

Manuscript received Sep 7, 2002; revised Feb 28, 2003. This paper was recommended by Editor-in-Chief Nagarajan Ranganathan.

This work was supported in part by the National Science Council of Taiwan ROC by Grant No. NSC-89-2215-E-009-117.

Jai-Ming Lin is with Realtek Semiconductor Corp. in Hsinchu Science-Based Industrial Park, Hsinchu 300, Taiwan. E-mail: gis87808@cis.nctu.edu.tw.

Yao-Wen Chang is with the Department of Electrical Engineering & the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan. E-mail: ywchang@cc.ee.ntu.edu.tw.

(A similar idea to the O-tree was independently developed by Takahashi in [15].) In an O-tree, a node denotes a module and an edge denotes the horizontal adjacency relation of two modules. The O-tree can pack modules in linear time, but it needs to perform a sequence of operations to make a placement compacted to the left and the bottom to obtain a feasible O-tree. Such an operation may require an overall quadratic time complexity since it may need a linear number of transformations between O-trees and their corresponding placements (constraint graphs), and each of the transformation takes linear time. Further, the transformation may result in a mismatch between the original representation and its placement, harming the solution structure. Chang et al. recently in [1] presented a binary tree-based representation for a left and bottom *compacted* placement, called B*-tree, and showed its superior properties for operations. In a B*-tree, a node denotes a module, the left child of a node represents the lowest adjacent module on the right, and the right child represents the first module above and with the same x coordinate. Similar to O-tree, the representation could be changed after packing since the space intended for placing a module may be occupied by a previously placed module. Therefore, given an O-tree or a B*-tree, it may not be feasible to find a placement corresponding to its original representation. Since the tree-based representations can represent only compacted floorplans, they induce smaller solution spaces and have lower complexity for a *single* packing operation than SP and BSG. However, they might lead to only suboptimal solutions for some cost metric (such as wirelength) since it is very likely that the optimum ones occur when modules are not compacted. Fig 1(a) shows an uncompacted placement that cannot be represented by any O-tree (since module b is not compacted to module a). Therefore, the O-tree representation will fail to find the optimum solution for wirelength optimization if modules b and c are strongly connected.

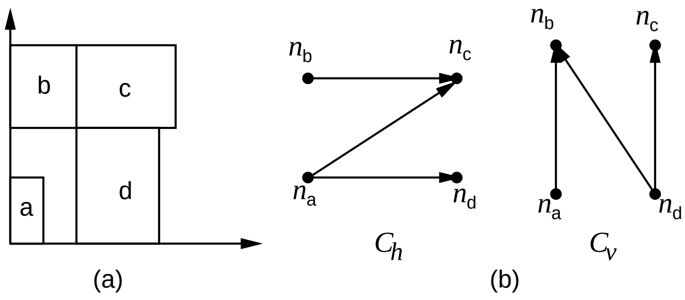

Fig. 1. (a) An uncompacted placement that cannot be represented by any O-tree. (b) The TCG representation for the placement shown in (a).

Recently, Hong et al. in [3] proposed a corner block list (CBL) representation for *mosaic* floorplans. A CBL consists of a 3-tuple (S, L, T), where S is a sequence of corner modules, L is a list of module orientations (0 for a vertical T-junction at the corner, and 1 for a horizontal one), and T is a list of T-junction information. In a mosaic floorplan, each region must contain exactly one module. Obviously, such restriction makes its solution space smaller than SP and BSG. However, CBL is not P-admissible since it cannot guarantee a feasible solution in each perturbation, and many infeasible solutions may be generated before a feasible solution is found.

B. Our Contribution

We propose in this paper a transitive closure graph-based representation for general non-slicing floorplans, named *TCG*, and show its superior properties. TCG uses a horizontal and a vertical transitive closure graphs, C_h and C_v, to describe the horizontal and vertical relations for each pair of modules. In the C_h (C_v) of Fig 1(b), for example, an edge from module x to module y represents that x is left to (below) y. (We will give a formal definition on the graphs later.) TCG is the first representation that can perturb on (non-tree) graphs directly and guarantee a feasible solution in each perturbation.

To differentiate the properties of TCG from other existing representations, we extend in this paper the concept of the P-admissible representation to that of the *P*-admissible* one by adding the following condition:
(5) the geometric relation between each pair of modules is defined in the representation.
The fifth condition facilitates the handling of the floorplan/placement design problems with additional requirements such as module sizing and position constraints (e.g., boundary constraints, symmetry constraints, etc). The representation after packing corresponds to the original one if it satisfies the condition. It leads to a better solution (neighborhood) structure, facilitating the search for an optimum solution. The P*-admissible representation corresponds to a general topological modeling of modules, and thus contains a complete structure for searching for an optimum floorplan/placement solution. For example, for the placement of Fig 1(a) that cannot be represented by any O-tree, it can easily be represented by the TCG (a P*-admissible representation) shown in Fig 1(b). Due to the elegant properties, it is desirable to develop an effective and flexible P*-admissible representation.

Among the existing popular representations, SP, BSG, and TCG are P*-admissible while slicing trees, NPE, O-tree, B*-tree, and CBL are not. Slicing trees, NPE, and CBL are not P-admissible and thus non-P*-admissible. The tree-based representations violates both the fifth condition of the P*-admissibility. The insufficiency (due to the oversimplified representations) incurs the following drawbacks:
• Some geometric relations between modules cannot be obtained from the O-tree and the B*-tree representations directly, making O-trees and B*-trees harder to handle the flooprlan design problems with the aforementioned additional requirements. Fig 2(a) shows a compacted placement with five modules a, b, c, d, and e whose widths and heights are (6, 4), (4, 6), (7, 4), (6, 3) and (3, 2), respectively. Fig 2(b) and (c) show the O-tree and the B*-tree corresponding to the placement of Fig 2(a), respectively. As illustrated in the figures, we cannot derive any geometric relation between two modules from the O-tree and B*-tree unless the two corresponding nodes are siblings or on the same path. For example, even though module b is adjacent to module d in the placement, we cannot derive any geometric relation from the representations directly until packing. Further, the geometric relation between two modules for the same O-tree or B*-tree may change if the dimensions of modules are changed. For example, if the dimension of module b is changed to $(1, 6)$, module d is

right to module b *after packing* as shown in Fig 2(d), instead of being above b as in Fig 2(a) (for the same O-tree shown in Fig 2(b)). The mismatch would inevitably complicate the floorplan design process.

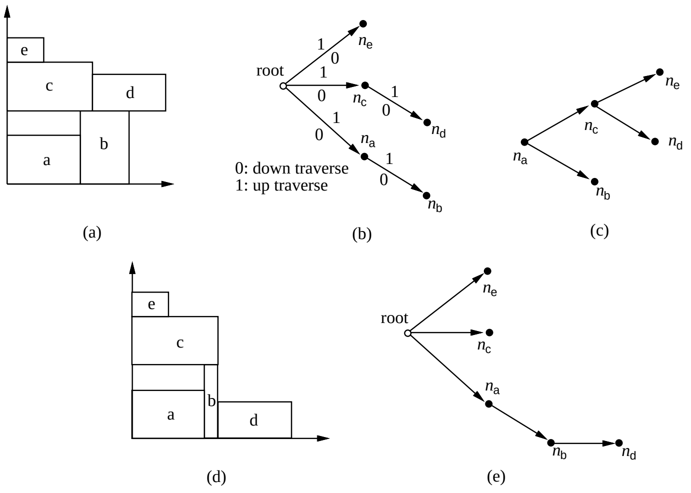

Fig. 2. (a) A placement. (b) The corresponding O-tree. (c) The corresponding B*-tree. (d) The placement after packing if the dimension of module b is changed to (1, 6). (e) The O-tree derived from the placement of (d).

Also, for module sizing, it is better to keep the geometric relation between *each pair* of modules in the representation to prevent a re-sized module from overlapping with other modules. Besides, it is harder to handle boundary and symmetry modules with the tree-based representations. For the O-tree shown in Fig 2(b), n_b seems to denote a right boundary module because there exists no node on its right side. However, module b may not be a right boundary module in the final placement as shown in Fig 2(d). To deal with symmetry constraints, several pairs of modules have to be placed symmetrically with respect to a common axis, and the x or y coordinates of the modules in each pair must be the same. It is desirable to keep the geometric relation between modules in the representation to facilitate the floorplan/placement design with symmetry modules.

• Due to their compaction operations, the O-tree or the B*-tree after packing may not correspond to the original one, which may harm the solution (neighborhood) structure and thus also the convergence to an optimum solution. After packing, for example, the initial O-tree of Fig 2(b) results in the placement of Fig 2(d) which corresponds to a different O-tree shown in Fig 2(e).

Despite of their smaller solution spaces and cheaper *single* packing complexity, the aforementioned drawbacks make non-P*-admissible representations less flexible and effective in handling practical floorplan/placement design problems which need to consider various requirements.

In contrast, TCG combines the advantages of SP, BSG, and B*-tree. Like SP and BSG, but unlike O-tree, B*-tree, and CBL, TCG satisfies the five conditions of P*-admissibility:

(1) its solution space is $(m!)^2$ and thus finite, where m is the number of modules,
(2) every solution is feasible (note that the CBL representation does not guarantee this property),
(3) packing and cost evaluation can be performed in $O(m^2)$ time,
(4) the best evaluated packing in the solution space corresponds to an optimum placement,
(5) the geometric relation between each pair of modules is defined in the TCG representation.

The solution space is the same as SP but the memory usage is smaller since we do not need to maintain a sequence pair. Like B*-tree, but unlike SP, BSG, O-tree, and CBL, TCG does not need to construct additional constraint graphs for the cost evaluation during packing, implying faster running time. Further, TCG supports incremental update during operations, and keeps the information of boundary modules as well as the shapes and the relative positions of modules in the representation. More importantly, the geometric relation among modules is transparent not only to the TCG representation but also to its *operations* (i.e., the effect of an operation on the change of the geometric relation is known *before* packing), facilitating faster convergence to a desired solution and placement with position constraints. For example, as illustrated in Fig 1, the nodes with zero in-degree (out-degree) in the horizontal constraint graph C_h correspond to the left (right) boundary modules in the placement, and the nodes with zero in-degree (out-degree) in the vertical constraint graph C_v correspond to the bottom (top) boundary modules. The transparency of the geometric relation among modules distinguishes TCG from other representations in handling placement with position constraints. All these properties make TCG an effective and flexible representation for handling the general floorplan/placement design problems with various requirements. Experimental results show the promise of TCG. For area optimization, TCG achieved average improvements of 2.22%, 2.04%, 1.18%, and 3.54%, compared to O-tree, enhanced O-tree, B*-tree, and CBL, respectively. Optimizing wirelength, TCG obtained respective average improvements of 3.56% and 3.18%, compared to O-tree and enhanced O-tree. (Note that B*-tree and CBL do not report the results for optimizing wirelength alone.) The runtime requirements of TCG are much smaller than O-tree and B*-tree, and are comparable to enhanced O-tree.

The remainder of this paper is organized as follows. Section II formulates the floorplan/placement design problem. Section III presents the procedures to derive a TCG from a placement and construct a placement from a TCG. Section IV introduces the operations to perturb a TCG. Experimental results are reported in Section V. Finally, we conclude our work and discuss future research directions in Section VI.

II. Problem Definition

Let $B = \{b_1, b_2, ..., b_m\}$ be a set of m rectangular modules whose width, height, and area are denoted by W_i, H_i, and A_i, $1 \leq i \leq m$. Each module is free to rotate. Let (x_i, y_i) denote the coordinate of the bottom-left corner of rectangle b_i, $1 \leq i \leq m$, on a chip. A placement $\mathcal{P}$ is an assignment of (x_i, y_i) for each b_i, $1 \leq i \leq m$, such that no two modules overlap. The goal of floorplanning/placement is to optimize a predefined cost metric such as a combination of the area (i.e., the minimum bounding rectangle of $\mathcal{P}$) and wirelength (i.e., the summation of half bounding box of interconnections) induced by a placement.

III. Transitive Closure Graph (TCG)

The *transitive closure* of a directed acyclic graph G is defined as the graph $G' = (V, E')$, where $E' = \{(n_i, n_j)$: there is a path from node n_i to node n_j in $G\}$. The *Transitive Closure Graph* (*TCG*) representation describes the geometric relations among modules based on two graphs, namely a *horizontal transitive closure graph* C_h and a *vertical transitive closure graph* C_v. In this section, we first introduce the procedure for constructing C_h and C_v from a placement. Then, we describe how to pack modules from TCG. In the last subsection, we discuss the properties and the solution space of TCG.

A. From a placement to its TCG

For two non-overlapped modules b_i and b_j, b_i is said to be *horizontally* (*vertically*) *related* to b_j, denoted by $b_i \vdash b_j$ ($b_i \perp b_j$), if b_i is on the left (bottom) side of b_j and their projections on the y (x) axis overlap. (Note that two modules cannot have both horizontal and vertical relations unless they overlap.) For two non-overlapped modules b_i and b_j, b_i is said to be *diagonally related* to b_j if b_i is on the left side of b_j and their projections on the x and the y axes do not overlap. In a placement, every two modules must bear one of the three relations: *horizontal relation*, *vertical relation*, and *diagonal relation*. To simplify the operations on geometric relations, we treat a diagonal relation for modules b_i and b_j as a horizontal one, unless there exists a chain of vertical relations from b_i (b_j), followed by the modules enclosed with the rectangle defined by the two closest corners of b_i and b_j, and finally to b_j (b_i), for which we make $b_i \perp b_j$ ($b_j \perp b_i$).

Fig 3(a) shows a placement with five modules a, b, c, d, and e whose widths and heights are (6, 4), (4, 6), (7, 4), (6, 3) and (3, 2), respectively. In Fig 3(a), $a \vdash b$, $a \perp c$, and module e is diagonally related to module b. There exists a chain of vertical relations formed by modules e, c, and b between the two modules e and b (i.e., $b \perp c$ and $c \perp e$). Therefore, we make $b \perp e$. Also, module e is diagonally related to module d. However, there dose not exist a chain of vertical relations between modules e and d, and thus we make $e \vdash d$.

TCG can be derived from a placement as follows. For each module b_i in a placement, we introduce a node n_i with the weight being the width (height) in C_h (C_v). If $b_i \vdash b_j$, we construct a directed edge from node n_i to node n_j (denoted by (n_i, n_j)) in C_h. Similarly, we construct a directed edge (n_i, n_j) in C_v if $b_i \perp b_j$. Given a placement with m modules, we need to perform the above process $m(m-1)/2$ times to capture all the geometric relations among modules (i.e., C_h and C_v have $m(m-1)/2$ edges in total).

As shown in Fig 3(b), for each module b_i, $i \in \{a, b, c, d, e\}$, we introduce a node n_i in C_h and also in C_v. For each node n_i in C_h (C_v), $i \in \{a, b, c, d, e\}$, we associate the node with a weight equal to the width (height) of the corresponding module b_i. Since $b_a \vdash b_b$, we construct a directed edge (n_a, n_b) in C_h. Similarly, we construct a directed edge (n_a, n_c) in C_v since $b_a \perp b_c$. This process is repeated until all geometric relations among modules are defined. As shown in Fig 3(b), each transitive closure graph has five nodes, and there are totally 10 edges in C_h and C_v (four in C_h and six in C_v). From the TCG representation shown in Fig 3(b), we know that module e is above module b because there exists a directed edge (n_b, n_e) in C_v; notice that this relation cannot be directly derived from the O-tree and B*-tree shown in Fig 2. Further, we also know directly from the TCG that module d is right to module a while the relationship is not known to O-tree and B*-tree until modules are placed. For boundary information, we know that modules a, c and e (b and d) are on the left (right) boundary since the in-degrees (out-degrees) of n_a, n_c and n_e (n_b and n_d) are zero in C_h. Similarly, it is easy to know from C_v that modules a and b (e and d) are on the bottom (top) boundary. Therefore, the floorplan/placement design with boundary constraints can be handled easily by checking the degree of a node during each perturbation.

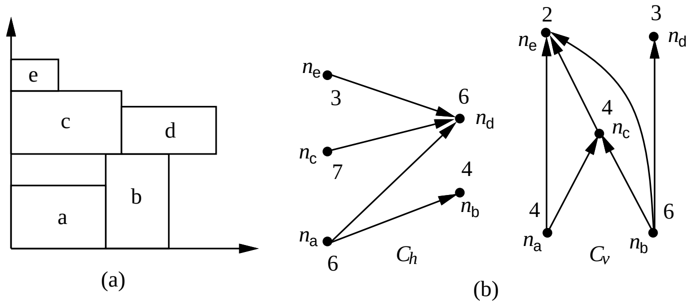

Fig. 3. (a) A placement in a chip. (b) TCG.

B. From a TCG to its placement

We have introduced how to derive a TCG from its placement in the previous section. We now present the packing method for a TCG.

Given a TCG, its corresponding placement can be obtained in $O(m^2)$ time by performing a well-known *longest path algorithm* [5] on the TCG, where m is the number of modules. To facilitate the implementation of the longest path algorithm, we augment the given two closure graphs as follows. (Note that the TCG augmentation is performed only for packing. It will be clear later that such augmentation is not needed for other operations such as solution perturbation.) We introduce two special nodes with zero weights for each closure graph, the source n_s and the sink n_t, and construct an edge from n_s to each node with in-degree equal to zero, and also from each node with out-degree equal to zero to n_t. Fig 4 shows the augmented TCG for the TCG shown in Fig 3(b).

Let $L_h(n_i)$ ($L_v(n_i)$) be the length of the longest path from n_s to n_i in the augmented C_h (C_v). $L_h(n_i)$ ($L_v(n_i)$) can be determined by performing the single source longest path algorithm on the augmented C_h (C_v) in $O(m^2)$ time, where m is number of modules. The coordinate (x_i, y_i) of a module b_i is given by ($L_h(n_i)$, $L_v(n_i)$). Since the respective width and height of the placement for the given TCG are $L_h(n_t)$ and $L_v(n_t)$, the area of the placement is given by $L_h(n_t)L_v(n_t)$.

C. Properties of TCG

Property 1: (TCG Feasibility Conditions) A feasible TCG has the following three properties:
(1) C_h and C_v are acyclic.

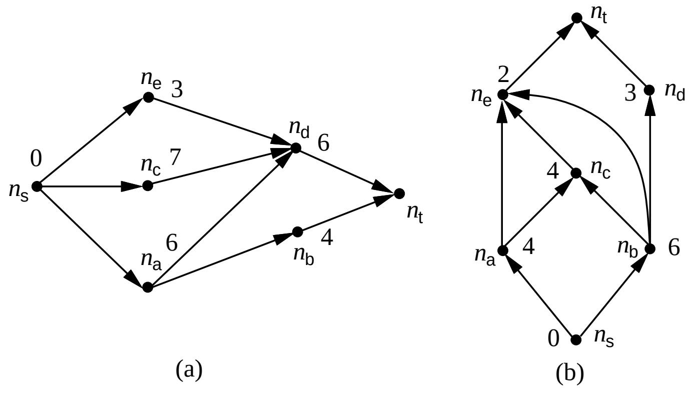

Fig. 4. Augmented TCG. (a) Augmented C_h. (b) Augmented C_v.

(2) Each pair of nodes must be connected by exactly one edge either in C_h or in C_v.

(3) The transitive closure of C_h (C_v) is equal to C_h (C_v) itself.

Proof:

(1) For each pair of nodes, we construct a directed edge according to the geometrical relation of two modules. Since a module cannot be both left and right (below and above) to another module in a placement, the resulting graphs C_h and C_v must be acyclic.

(2) Given a placement with m modules, as mentioned earlier, we construct $m(m-1)/2$ edges to capture all geometric relations among modules. Since there are also $m(m-1)/2$ pairs of nodes and no multiple edges are allowed, each pair of nodes would be connected by exactly one edge either in C_h or in C_v.

(3) To prove Property 3, we claim that $b_i \vdash b_k$ ($b_i \perp b_k$) if $b_i \vdash b_j$ and $b_j \vdash b_k$ ($b_i \perp b_j$ and $b_j \perp b_k$). Suppose $b_i \vdash b_j$ and $b_j \vdash b_k$, but we make $b_i \perp b_k$. This implies that all modules b_l's overlapped with the rectangle defined by the two closest corners of b_i and b_k have the geometric relations $b_i \perp b_l$ and $b_l \perp b_k$, which is a contradiction to our assumption that $b_i \vdash b_j$ and $b_j \vdash b_k$. Similarly, we claim that $b_i \perp b_k$ if $b_i \perp b_j$ and $b_j \perp b_k$.

■

Property 1 ensures that a module b_i cannot be both left and right to (below and above) another module b_j in a placement. Property 2 guarantees that no two modules overlap since each pair of modules have exactly one of the horizontal or vertical relation. Property 3 is used to eliminate redundant solutions. It guarantees that if there exists a path from n_i to n_j in one closure graph, the edge (n_i, n_j) must also appear in the same closure graph. For example, there exist two edges (n_i, n_j) and (n_j, n_k) in C_h, which means that $b_i \vdash b_j$ and $b_j \vdash b_k$, and thus $b_i \vdash b_k$. If the edge (n_i, n_k) appears in C_v instead of in C_h, b_k is not only left to b_i but also above b_i. The resulting area of the corresponding placement must be larger than or equal to that when the edge (n_i, n_k) appears in C_h. Fig 5 illustrates this phenomenon. In Fig 5(a), there exists a path from n_a to n_e in c_v, which consists of (n_a, n_c) and (n_c, n_e). Thus, the edge (n_a, n_e) must also belong to C_v. If the edge (n_a, n_e) appears in C_h instead of in C_v as shown in Fig 5(c), the resulting area of the placement must be larger than or equal to the configuration of Fig 5(a) and (b). Property 3 eliminates such a redundancy.

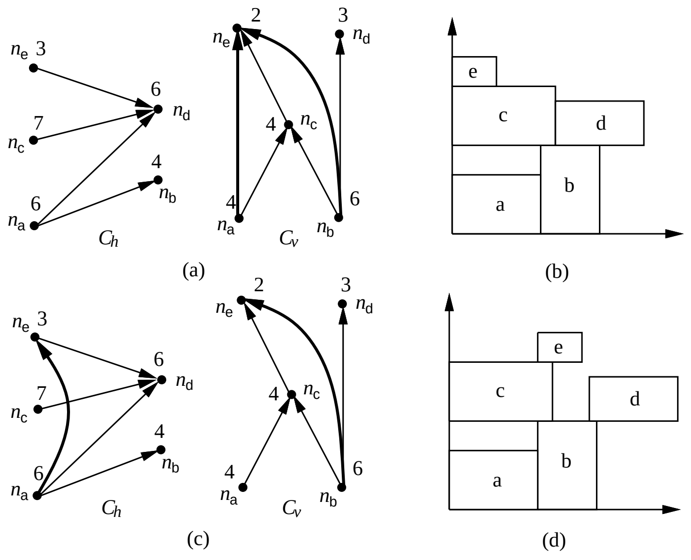

Fig. 5. Examples of the TCG feasibility. (a) A feasible TCG. (b) The placement corresponding to the TCG shown in (a). (c) A non-TCG. (d) The placement corresponding to the non-TCG shown in (c).

Based on the properties of TCG, we have the following theorems.

Theorem 1: There exists a unique placement corresponding to a TCG.

Proof: We first show that each TCG is feasible (i.e., there must exist a placement for each TCG), and then show the uniqueness of the placement.

Property 1 avoids that a module is both left and right to (or below and above) another module in the packing. Property 2 guarantees that no two modules overlap in the packing. Thus, Properties 1 and 2 guarantee that there exists a placement for each TCG. Given a TCG, the x and y coordinates of each module are determined by the respective longest paths in C_h and C_v, which are well-defined values in the TCG. Therefore, the placement is unique. ■

Theorem 2: The size of the solution space for TCG is $(m!)^2$, where m is the number of modules.

Proof: To show the size, we prove that there exists a one-to-one correspondence between a TCG (C_h, C_v) and a sequence pair (Γ_+, Γ_-). Since there are $(m!)^2$ such sequence pairs, the theorem thus follows.

Let the *fan-in* (*fan-out*) of a node n_i, denoted by $F_{in}(n_i)$ ($F_{out}(n_i)$), be the nodes n_j's with edges (n_j, n_i) ((n_i, n_j)). Given a TCG, we can transform it into a sequence Γ_+ by repeatedly extracting a node n_i with $F_{in}(n_i) = \emptyset$ in C_h and $F_{out}(n_i) = \emptyset$ in C_v, and then deleting the edges (n_i, n_j)'s ((n_j, n_i)'s) from C_h (C_v) until no node is left in C_h (C_v). Similarly, we can transform a TCG into another sequence Γ_- by repeatedly extracting the node n_i with $F_{in}(n_i) = \emptyset$ both in C_v and C_h, and then deleting the edges (n_i, n_j)'s from both C_h and C_v until no node is left in C_h and C_v. As the example shown in Fig 3(a), we have $\Gamma_+ = < n_e, n_c, n_a, n_d, n_b >$ and $\Gamma_- = < n_a, n_b, n_c, n_e, n_d >$.

We claim that there exists a unique sequence pair (Γ_+, Γ_-) corresponding to a TCG. Since the node n_i with $F_{in}(n_i) = \emptyset$ in

C_h and $F_{out}(n_i) = \emptyset$ ($F_{in}(n_i) = \emptyset$) in C_v denotes the *unique* module on the left and the top (bottom) boundaries of a placement, such a unique node can be found in each iteration during the transformation. By removing the node n_i and its incident edges, we obtain a new TCG with fewer nodes. It is obvious that there is also a unique module on the left-top (left-bottom) corner in the placement corresponding to the new TCG. Repeating this process, we can transform a TCG into a unique sequence pair (Γ_+, Γ_-).

Given a sequence pair (Γ_+, Γ_-), we can obtain a unique TCG (C_h, C_v) from the two constraint graphs induced from the sequence pair (Γ_+, Γ_-) by removing the source, the sink, and their associated edges in the graphs. To claim that the two resulting constraint graphs form a TCG, we shall prove that they satisfy the three properties of TCG. We shall first review the constraint graph construction for a sequence pair defined in [7]. Module b_a is left (right) to module b_b (i.e, there exists an edge (n_a, n_b) ((n_b, n_a)) in the horizontal constrain graph) if a is before (after) b in both Γ_+ and Γ_-. Module b_a is below (above) module b_b (i.e, there exists an edge (n_a, n_b) ((n_b, n_a)) in the vertical constrain graph) if b is before (after) a in Γ_+, and a is before (after) b in Γ_-.

- Property 1: Suppose there exists a cycle $< n_a, n_b, \ldots, n_a >$ in the horizontal constraint graph of a sequence pair. Then, the corresponding Γ_+ and Γ_- must be both in the sequence $\ldots a \ldots b \ldots a \ldots$, contradicting to the fact that a module cannot appear twice in a sequence. Similarly, there does not exist any cycle in C_v.
- Property 2: For every pair of nodes n_a and n_b, there exists a unique edge (n_a, n_b) or (n_b, n_a) in a horizontal or a vertical constraint graph, depending on the relative positions of a and b in Γ_+ and Γ_-.
- Property 3: To show that the transitive closure of a horizontal (vertical) constraint graph is equal to itself, we shall prove that if there exists a path $< n_i, n_j, n_k >$ in a constraint graph, the edge (n_i, n_k) also exists in the graph. If there exists a path $< n_i, n_j, n_k >$ in the horizontal (vertical) constraint graph of a sequence pair, the sequence pair must be in this form $(\ldots i \ldots j \ldots k \ldots, \ldots i \ldots j \ldots k \ldots)$ $((\ldots k \ldots j \ldots i \ldots, \ldots i \ldots j \ldots k \ldots))$, which implies that the edge (n_i, n_k) also exists in the same graph.

The theorem thus follows. ■

According to the above discussions, we conclude the following theorem.

Theorem 3: TCG is P*-admissible.

We summarize in TABLE I the properties of several recently published representations for non-slicing floorplans[1].

IV. Floorplanning Algorithm

We develop a simulated annealing based algorithm [4] using TCG for non-slicing floorplan design. Given an initial solution represented by a TCG, the algorithm perturbs the TCG to obtain a new TCG. To ensure the correctness of the new TCG, as described in the previous section, the new TCG must satisfy the aforementioned three feasibility properties. To identify feasible TCG for perturbation, we introduce the concept of *transitive reduction edges* of TCG in the following section.

A. Transitive Reduction Edges

An edge (n_i, n_j) is said to be a *reduction edge* if there does not exist another path from n_i to n_j, except the edge (n_i, n_j) itself; otherwise, it is a *closure edge*. For example, in the C_v of Fig 5(a), the edges (n_a, n_c), (n_b, n_c), (n_c, n_e), and (n_b, n_d) are reduction edges while (n_a, n_e) and (n_b, n_e) are closure ones since there exist respective paths $< n_a, n_c, n_e >$ and $< n_b, n_c, n_e >$ from n_a to n_e and from n_b to n_e.

Since TCG is formed by directed acyclic transitive closure graphs, given an arbitrary node n_i in one transitive closure graph, there exists at least one reduction edge (n_i, n_j), where $n_j \in F_{out}(n_i)$. Here, we define the fan-in (fan-out) of a node n_i, denoted by $F_{in}(n_i)$ ($F_{out}(n_i)$), as the nodes n_j's with edges (n_j, n_i) ((n_i, n_j)). For nodes $n_k, n_l \in F_{out}(n_i)$, the edge (n_i, n_k) cannot be a reduction edge if $n_k \in F_{out}(n_l)$. Hence, we remove those nodes in $F_{out}(n_i)$ that are fan-outs of others. The edges between n_i and the remaining nodes in $F_{out}(n_i)$ are reduction edges. For the C_v shown in Fig 5(a), $F_{out}(n_a) = \{n_c, n_e\}$. Since n_e belongs to $F_{out}(n_c)$, edge (n_a, n_e) is a closure edge while (n_a, n_c) is a reduction one.

Lemma 1: Given an arbitrary node n_i in one transitive closure graph, for nodes $n_k, n_l \in F_{out}(n_i)$, the edge (n_i, n_k) cannot be a reduction edge if $n_k \in F_{out}(n_l)$.

Proof: For nodes $n_k, n_l \in F_{out}(n_i)$ and $n_k \in F_{out}(n_l)$, the edge (n_i, n_k) cannot be a reduction edge because there exists at least a path $< n_i, n_l, n_k >$ from n_i to n_k except the edge (n_i, n_k). ■

Theorem 4: Given a node n_i in C_h or C_v, it takes $O(m^2)$ time to find a reduction edge (n_i, n_j), where m is the number of modules.

Proof: Given a node n_i, there exist at most $m - 1$ nodes in $F_{out}(n_i)$. For the nodes in $F_{out}(n_i)$, we pick a node n_j from $F_{out}(n_i)$ and remove eack node $n_k \in F_{out}(n_j)$ from $F_{out}(n_i)$. Since n_i has at most $m - 1$ fan-outs, and each of the fan-outs has at most $m - 1$ fan-outs, we need $O(m^2)$ to find a reduction edge (n_i, n_j). ■

B. Solution Perturbation

We apply the following four operations to perturb a TCG:

- *Rotation*: Rotate a module.
- *Swap*: Swap two nodes in both of C_h *and* C_v.
- *Reverse*: Reverse a *reduction edge* in C_h *or* C_v.
- *Move*: Move a *reduction edge* from one transitive closure graph (C_h or C_v) to the other.

Rotation and Swap do not change the topology of a TCG while Reverse and Move do. To maintain the properties of the TCG after performing the Reverse and Move operations, we may need to update the resulting graphs. We detail the four operations as follows.

B.1 Rotation

To rotate a module b_i, we only need to exchange the weights of the corresponding node n_i in C_h and C_v. Fig 6(b) shows the resulting C_h, C_v, and placement after rotating the module d

[1] In [3], the authors claim that the solution space of CBL is $O(m!2^{3m}/m^{1.5})$. However, in the 3-tuple (S, L, T) of CBL, there should be $m!$ combinations for S, 2^{m-1} combinations for L, and 2^{2m-3} combinations for T; there its solution space should be $O(m!2^{3m})$

Representation	SP [7]	FAST-SP [16]	BSG [8]	O-tree [2]	B*-tree [1]	CBL [3]	TCG
Type of floorplans that can be represented	general	general	general	compacted		mosaic	general
Is P*-admissible?	Yes	Yes	Yes	No		No	Yes
Solution space	$(m!)^2$	$(m!)^2$	$m!C(m^2, m)$	$O(m!2^{2m}/m^{1.5})$		$O(m!2^{3m})$	$(m!)^2$
Every solution is feasible?	Yes	Yes	Yes	Resulting packing may be different from the original representation		No	Yes
Runtime for packing	$O(m^2)$	Amortized $O(m \lg\lg m)$	$O(m^2)$	$O(m)$		$O(m)$	$O(m^2)$
Best evaluated packing corresp. to optimal placement?	Yes	Yes	Yes	True for are optimization, but not for wirelength optimization		No	Yes
Geometric relation between any two modules is defined?	Yes	Yes	Yes	No		No	Yes
Need module sequence encoding?	Yes	Yes	No	Yes	No	Yes	No
Construct additional constraint graphs for packing?	Yes	No	Yes	Yes	No	Yes	No
Evaluate cost directly on representation?	No	No	No	No	Yes	No	Yes
Geometric relation is transparent to its operations?	No	No	No	No	No	No	Yes
Boundary information on representation?	No	No	No	No	No	No	Yes

TABLE I
PROPERTIES OF REPRESENTATIONS. HERE, m IS THE NUMBER OF MODULES IN A PLACEMENT.

shown in Fig 6(a). Notice that the weights associated with the node n_d in C_h and C_v have been exchanged.

Theorem 5: TCG is closed under the rotation operation, and such an operation takes $O(1)$ time.

Proof: We do not change a TCG for the Rotation operation, and thus the resulting graphs are still a TCG. It is obvious that exchanging the weights of a node in C_h and C_v takes $O(1)$ time. ■

B.2 Swap

To swap two nodes n_i and n_j, we only need to exchange two nodes in both C_h and C_v. Fig 6(c) shows the resulting C_h, C_v, and placement after swapping the nodes n_a and n_b shown in Fig 6(b). Notice that the nodes n_a and n_b in both C_h and C_v have been exchanged.

Theorem 6: TCG is closed under the swap operation, and such an operation takes O(1) time.

Proof: Since we only exchange two nodes in both C_h and C_v without changing the topology of a TCG for the Swap operation, the resulting graphs are still a TCG. Exchanging the corresponding pointers of two nodes in both C_h and C_v takes $O(1)$ time. ■

B.3 Reverse

The Reverse operation reverses the direction of a *reduction* edge (n_i, n_j) in a transitive closure graph, which corresponds to changing the geometric relation of the two modules b_i and b_j. For two modules b_i and b_j, $b_i \vdash b_j$ ($b_i \perp b_j$) if there exists a reduction edge (n_i, n_j) in C_h (C_v); after reversing the edge (n_i, n_j), we have the new geometric relation $b_j \vdash b_i$ ($b_j \perp b_i$). Therefore, the geometric relation among modules is transparent not only to the TCG representation but also to the Reverse *operation* (i.e., the effect of such an operation on the change of the geometric relation is known *before* packing); this property can facilitate the convergence to a desired solution.

To reverse a reduction edge (n_i, n_j) in a transitive closure graph, we first delete the edge from the graph, and then add the edge (n_j, n_i) to the graph. For each node $n_k \in F_{in}(n_j) \cup \{n_j\}$ and $n_l \in F_{out}(n_i) \cup \{n_i\}$ in the new graph, we shall check whether the edge (n_k, n_l) exists in the new graph. If the graph contains the edge, we do nothing; otherwise, we need to add the edge to the graph and delete the corresponding edges (n_k, n_l) (or (n_l, n_k)) in the other transitive closure graph, if any, to maintain the properties of the TCG.

Fig 6(d) shows the resulting C_h, C_v, and placement after reversing the *reduction edge* (n_c, n_e) of the C_v shown in Fig 6(c). In the C_v of Fig 6(d), $F_{in}(n_e) \cup \{n_e\} = \{n_a, n_b, n_e\}$ and

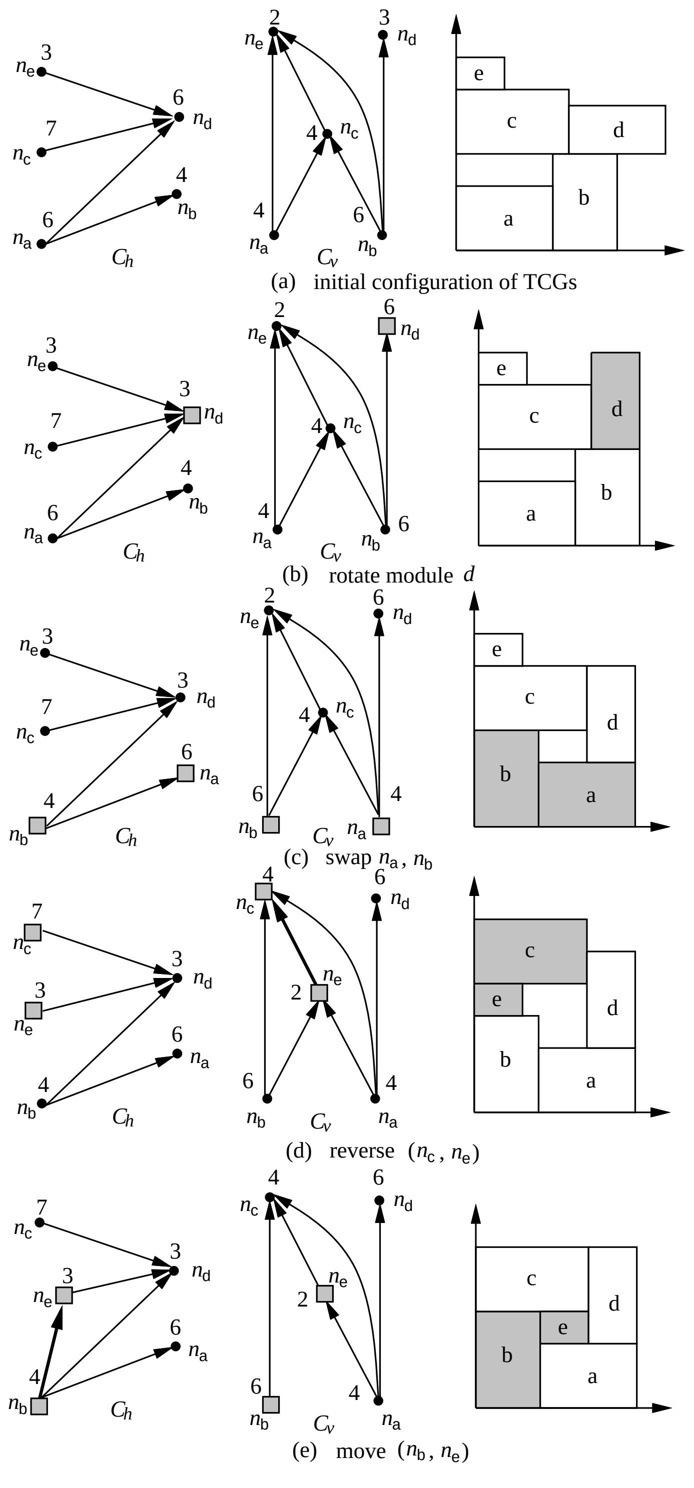

Fig. 6. Four types of perturbation. (a) The initial TCG (C_h and C_v) and placement. (b) The resulting TCG and placement after rotating the module d shown in (a). (c) The resulting TCG and placement after swapping the nodes n_a and n_b shown in (b). (d) The resulting TCG and placement after reversing the reduction edge (n_c, n_e) shown in (c). (e) The resulting TCG and placement after moving the reduction edge (n_b, n_e) from the C_v of (d) to C_h.

$F_{out}(n_c) \cup \{n_c\} = \{n_c\}$. For each node $n_k \in F_{in}(n_e) \cup \{n_e\}$ and $n_l \in F_{out}(n_c) \cup \{n_c\}$, we check whether the edge (n_k, n_l) exists. Since the edge (n_e, n_c) was just added to C_v and the edges (n_b, n_c) and (n_a, n_c) already exist in C_v of Fig 6(c), we do not need to add the three edges to C_v. Neither do we need to update the C_h of Fig 6(c). Notice that by reversing the reduction edge (n_c, n_e) in the C_v, we transform the relation $b_c \perp b_e$ into $b_e \perp b_c$ in the resulting placement of Fig 6(d).

To maintain the properties of a TCG, we can only reverse a reduction edge. For example, if we reverse a closure edge (n_i, n_k) associated with the two reduction edges (n_i, n_j) and (n_j, n_k), a cycle $< n_i, n_j, n_k, n_i >$ is formed, and thus the resulting graphs are no longer a TCG. Further, for each edge introduced in a transitive closure graph, we remove its corresponding edge from the other graph. Therefore, there is always exactly one relation between each pair of modules.

Theorem 7: TCG is closed under the reverse operation, and such an operation takes $O(m^2)$ time, where m is the number of modules in the placement.

Proof: We first show that the resulting graphs C_h and C_v of a TCG satisfy the three properties of TCG after performing the Reverse operation.

Without loss of generality, we focus on the case for reversing a reduction edge (n_i, n_j) in C_h. For Property 1, suppose that the new C_h is not acyclic after the Reverse operation. Then, there must exist a path from n_i to n_j in C_h before the operation, implying that (n_i, n_j) is a closure edge, which is a contradiction. The new C_v must also be acyclic since we do not add any edge into C_v during the operation. For Property 2, each pair of nodes must be connected by exactly one edge either in the new C_h or in the new C_v after the operation because we delete the edge (n_i, n_j) from C_v after adding the edge into C_h. For Property 3, suppose that the new C_h is not a transitive closure of itself. Then, there exists a path $< n_x, \ldots, n_j, n_i, \ldots, n_y >$ in the new C_h, but the C_h does not contain the closure edge (n_x, n_y). During the operation, for each node $n_k \in F_{in}(n_j) \cup \{n_j\}$ and $n_l \in F_{out}(n_i) \cup \{n_i\}$ in C_h, we add the edges (n_k, n_l)'s to the new C_h and delete them from C_v. Therefore, at least one of the edges (n_x, n_j) and (n_i, n_y) does not exist in the original C_h; otherwise, we would have added the closure edge (n_x, n_y) into the new C_h during the Reverse operation. This implies that the original C_h is not a transitive closure graph, contradicting to our assumption. It is clear that the deleted edges in C_v are the closure edges of the new C_h, which cannot be the closure edges in C_v. Therefore, the new C_v is still a transitive closure graph of itself.

The time complexity is dominated by checking whether the edges (n_k, n_l)'s ($n_k \in F_{in}(n_j) \cup \{n_j\}$ and $n_l \in F_{out}(n_i) \cup \{n_i\}$) exist in the new graph and by inserting and deleting the corresponding edges. Since there are at most $O(m)$ n_k's and $O(m)$ n_l's, the operation takes $O(m^2)$ time in total. ■

B.4 Move

The Move operation moves a *reduction* edge (n_i, n_j) in a transitive closure graph to the other, which corresponds to switching the geometric relation of the two modules b_i and b_j between a horizontal relation and a vertical one. For two modules b_i and b_j, $b_i \vdash b_j$ ($b_i \perp b_j$) if there exists a reduction edge

(n_i, n_j) in C_h (C_v); after moving the edge (n_i, n_j) to C_v (C_h), we have the new geometric relation $b_i \perp b_j$ ($b_i \vdash b_j$). Therefore, the geometric relation among modules is also transparent to the Move operation.

To move a reduction edge (n_i, n_j) from a transitive closure graph G to the other G' in a TCG, we first delete the edge from G and add it to G'. Similar to the Reverse operation, for each node $n_k \in F_{in}(n_i) \cup \{n_i\}$ and $n_l \in F_{out}(n_j) \cup \{n_j\}$, we shall check whether the edge (n_k, n_l) exists in G'. If G' contains the edge, we do nothing; otherwise, we need to add the edge to G' and delete the corresponding edge (n_k, n_l) (or (n_l, n_k)) in G, if any, to maintain the properties of the TCG.

Fig 6(e) shows the resulting C_h, C_v, and placement after moving the *reduction edge* (n_b, n_e) in the C_v of Fig 6(d) to C_h. In the C_h shown in Fig 6(e), $F_{in}(n_b) \cup \{n_b\} = \{n_b\}$ and $F_{out}(n_e) \cup \{n_e\} = \{n_d, n_e\}$. For each node $n_k \in F_{in}(n_b) \cup \{n_b\}$ and $n_l \in F_{out}(n_c) \cup \{n_c\}$, we check whether the edge (n_k, n_l) exists in G_h. Since the edge (n_b, n_e) was just added to C_h and the edge (n_b, n_d) already exists in C_h of Fig 6(d), we need to do nothing (except moving the reduction edge (n_b, n_e) from C_v to C_h). Neither do we need to update C_v (except removing the edge (n_b, n_e)). Notice that by moving the reduction edge (n_b, n_e) from C_v to C_h, we transform the relation $b_b \perp b_e$ into $b_b \vdash b_e$ in the resulting placement shown in Fig 6(e).

To maintain the properties of a TCG, we can only move a reduction edge. If we move a closure edge (n_i, n_k) associated with the two reduction edges (n_i, n_j) and (n_j, n_k) in one transitive closure graph to the other, then there exist a path from n_i to n_k in the two graphs, implying that $b_i \vdash b_k$ and $b_i \perp b_k$, which gives a redundant solution. Further, for each edge introduced in a transitive closure graph, we remove its corresponding edge from the other graph. Therefore, there is always exactly one relation between each pair of modules.

Theorem 8: TCG is closed under the move operation, and such an operation takes $O(m^2)$ time, where m is the number of modules in the placement.

Proof: We first show that the resulting graphs C_h and C_v of a TCG satisfy the three properties of TCG after performing the Move operation.

Without loss of generality, we focus on the case for moving a reduction edge (n_i, n_j) from C_h to C_v. For Property 1, suppose that the resulting C_v is not acyclic after we move a reduction edge (n_i, n_j) from C_h to C_v. There must exist a path from n_j to n_i in the original C_v. This implies that the edge (n_j, n_i) is also in the original C_v since C_v is a transitive closure graph. This is a contradiction since (n_i, n_j) and (n_j, n_i) cannot both exist in the original TCG (Property 2). Therefore, the new C_v must be acyclic. The new C_h must also be acyclic since we do not add any edge into the original C_h. For Property 2, each pair of nodes must be connected by exactly one edge either in the new C_h or in the new C_v after the operation because the corresponding edge will be deleted from C_h after the edge (n_i, n_j) is added to C_v. For Property 3, suppose that the new C_v is not a transitive closure of itself. Then, there exists a path $< n_x, \ldots, n_i, n_j, \ldots, n_y >$ in the new C_v, but the C_v does not contain the closure edge (n_x, n_y). During the operation, for each node $n_k \in F_{in}(n_i) \cup \{n_i\}$ and $n_l \in F_{out}(n_j) \cup \{n_j\}$ in C_v, we add the edges (n_k, n_l)'s to the new C_v and delete them from C_h. Therefore, at least one of the edges (n_x, n_i) and (n_j, n_y) does not exist in the original C_h; otherwise, we would have added the closure edge (n_x, n_y) into the new C_v during the Move operation. This implies that the original C_v is not a transitive closure graph, contradicting to our assumption. It is clear that the deleted edges of C_h are the closure edges of the new C_v, which cannot be the closure edges in C_h. Therefore, the new C_h is still a transitive closure graph of itself.

Similar to the arguments in the proof of Theorem 7, the operation takes $O(m^2)$ time in total. ■

Circuit	#Modules	#I/O pads	#Nets	#Pins
apte	9	73	97	214
xerox	10	107	203	696
hp	11	43	83	264
ami33	33	42	123	480
ami49	49	24	408	931

TABLE II
THE FIVE MCNC BENCHMARK CIRCUITS.

V. EXPERIMENTAL RESULTS

Based on a simulated annealing method [4], we implemented the TCG representation in the C++ programming language on a 433 MHz SUN Sparc Ultra-60 workstation with 1 GB memory. The TCG package is available at http://cc.ee.ntu.edu.tw/∼ywchang/research.html. We compared TCG with O-tree [2], B*-tree [1], enhanced O-tree [13], and CBL [3] based on the five MCNC benchmark circuits listed in TABLE II. Columns 2, 3, 4, and 5 of TABLE II list the respective numbers of modules, I/O pads, nets, and pins of the five circuits.

The experiments consist of three parts: area optimization, wirelength optimization, and simultaneous area and wirelength optimization. The area of a placement is measured by that of the minimum bounding box enclosing the placement. The area and runtime comparisons among O-tree [2], B*-tree [1], enhanced O-tree [13], CBL [3], and TCG are listed in TABLE III. As shown in TABLE III, TCG achieves average improvements of 2.22%, 1.18%, 2.04%, and 3.54% in area utilization compared to O-tree, B*-tree, enhanced O-tree, and CBL, respectively. The runtimes are significantly smaller than O-tree and B*-tree, and comparable to the enhanced O-tree [13]. Fig 7 (left) shows the resulting placement for ami49 with area optimization.

For wirelength optimization, we estimated the wirelength of a net by half the perimeter of the minimum bounding box enclosing the net. The wirelength of a placement is given by the summation of the wirelengths of all nets. The comparisons with the previous works are listed in TABLE IV. (Note that B*-tree and CBL did not report the results on optimizing wirelength alone.) As shown in TABLE IV, TCG achieves average reductions of 3.56% and 3.18% in wirelength, compared to the O-tree and the enhanced O-tree, respectively. Fig 7 (right) shows the resulting placement for ami49 with wirelength optimization. For simultaneous area and wirelength optimization, we assigned the same weight for area and wirelength in the cost function. The results

Circuit	SP		O-tree		B*-tree		enhanced O-tree		CBL		TCG	
	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)
apte	48.12	13	47.1	38	46.92	7	46.92	11	NA	NA	46.92	1
xerox	20.69	15	20.1	118	19.83	25	20.21	38	20.96	30	19.83	18
hp	9.93	5	9.21	57	8.947	55	9.16	19	-	-	8.947	20
ami33	1.22	676	1.25	1430	1.27	3417	1.24	118	1.20	36	1.20	306
ami49	38.84	1580	37.6	7428	36.80	4752	37.73	406	38.58	65	36.77	434
Comp.	+5.04%	-	+2.22%	-	+1.18%	-	+2.04%	-	+3.54%	-	0.00%	-

TABLE III

AREA AND RUNTIME COMPARISONS AMONG SP (ON SUN ULTRA60), O-TREE (ON SUN ULTRA60), B*-TREE (ON SUN ULTRA-I), ENHANCED O-TREE (ON SUN ULTRA60), CBL (ON SUN SPARC 20), AND TCG (ON SUN ULTRA60) FOR AREA OPTIMIZATION. (NA: NOT AVAILABLE.)

Circuit	O-tree		enhanced O-tree		TCG	
	Wire (mm)	Time (sec)	Wire (mm)	Time (sec)	Wire (mm)	Time (sec)
apte	317	47	317	15	363	2
xerox	368	160	372	39	366	15
hp	153	90	150	19	143	10
ami33	52	2251	52	177	44	52
ami49	636	14112	629	688	604	767
Comp.	+3.56%	-	+3.18%	-	0.00%	-

TABLE IV

WIRELENGTH AND RUNTIME COMPARISONS AMONG O-TREE (ON SUN ULTRA60), ENHANCED O-TREE (ON SUN ULTRA60), AND TCG (ON SUN ULTRA60) FOR WIRELENGTH OPTIMIZATION.

are listed in TABLE V, which shows that ours are slightly better than previous works.[2]

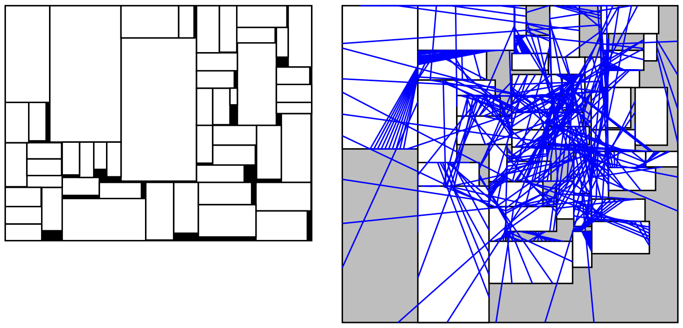

Fig. 7. Resulting placements of ami49 for (1) left: optimizing area alone (area $= 36.77mm^2$); (2) right: optimizing wirelngth alone (wire $= 604mm$).

Fig 8 show the stability and convergence-rate comparison between SP and TCG based on the circuit ami33. We randomly ran the programs for SP and TCG on ami33 ten times, based on the same initial solution each time. In Fig 8(a) and (b), the resulting areas are plotted as functions of the running times for SP and TCG using the same simulated annealing procedure. (Note that the parts with areas above 1.7 mm^2 are not shown in the curves for clarity of the comparison.) As illustrated in Fig 8, TCG converges much faster to a desired solution and the results are much more stable than SP. For TCG, all of the ten runs converged in about 15 sec and terminated in about 120 sec. We note that the stability and convergence rate should be very important metrics to evaluate the quality of a floorplan representation. However, they were often ignored in previous works.

[2]We excluded the CBL results for hp in TABLE III and for apte in TABLE V in the comparisons since the CBL test cases may not be the same as others. For example, CBL reports an area of 66.14 mm^2 for hp, which is about seven times larger than others.

VI. CONCLUDING REMARKS

We have introduced the concept of the P*-admissible representation, presented the P*-admissible TCG representation for general floorplans, and shown its superior properties. Experimental results have shown that TCG is very efficient, effective, and stable in floorplan optimization. As revealed in the representation, TCG keeps the information of boundary modules as well as the shapes and the relative positions of modules. These properties make TCG a promising choice for dealing with the general floorplan/placement problems with various requirements. Research along this direction is ongoing.

REFERENCES

[1] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A New Representation for Non-Slicing Floorplans," in *Proc. DAC*, 2000, pp. 458–463.

[2] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications," in *Proc. DAC*, 1999, pp. 268–273.

[3] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner Block List: An Effective and Efficient Topological Representation of Non-slicing Floorplan," in *Proc. ICCAD*, 2000, pp. 8–12.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp.671–680, May 13, 1983.

[5] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, 1976.

[6] J.-M. Lin and Y.-W. Chang, "TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans," in *Proc. DAC*, 2001, pp. 764–769.

[7] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing Based Module Placement," in *Proc. ICCAD*, 1995, pp. 472–479.

[8] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module Placement on BSG-Structure and IC Layout Applications," in *Proc. ICCAD*, 1996, pp. 484–491.

[9] Ohtsuki, T., N. Suzigama and H. Hawanishi, "An Optimization Technique for Intergrated Circuit Layout Design," in *Proc. ICCST*, 1970, pp. 67–68.

[10] H. Onodera, Y. Taniquchi, and K. Tamaru, "Branch-and-bound Placement for Building Block Layout," in *Proc. DAC*, 1991, pp. 433–439.

Circuit	O-tree			enhanced O-tree			CBL			TCG		
	Area (mm^2)	Wire (mm)	Time (sec)	Area (mm^2)	Wire (mm)	Time (sec)	Area (mm^2)	Wire (mm)	Time (sec)	Area (mm^2)	Wire (mm)	Time (sec)
apte	51.92	320.7	47	51.95	320.7	14	-	-	NA	48.48	378.0	50
xerox	20.42	380.6	142	20.42	380.6	41	20.233	403.47	NA	20.42	385.0	114
hp	9.490	152.6	84	9.384	151.9	21	NA	NA	NA	9.490	151.8	59
ami33	1.283	51.31	2349	1.299	52.13	205	1.226	51.67	NA	1.237	50.29	939
ami49	39.55	688.7	15318	39.92	702.8	700	38.378	732.84	NA	38.20	663.1	3613
Comp.	+2.87%	-2.01%	-	+3.33%	-1.34%	-	-0.45%	+5.98%	-	0.00%	0.00%	-

TABLE V

AREA, WIRELENGTH, AND RUNTIME COMPARISONS AMONG O-TREE (ON SUN ULTRA60), ENHANCED O-TREE (ON SUN ULTRA60), CBL (ON SUN SPARC 20), AND TCG (ON SUN ULTRA60) FOR SIMULTANEOUS AREA AND WIRELENGTH OPTIMIZATION.

[11] R. H. J. M. Otten, “Automatic Floorplan Design,” in *Proc. DAC*, 1982, pp.261–267.

[12] P. Pan and C.-L. Liu, “Area minimization for floorplans,” *IEEE Trans. on Computer-Aided Design*, Vol. 14 no. 1, pp. 123–132, Jan. 1995.

[13] Y.-Pang, C.-K. Cheng, and T. Yoshimura, “An Enhanced Perturbing Algorithm for Floorplan Design using the O-tree Representation,” in *Proc. ISPD*, 2000, pp. 168–173.

[14] S. M. Sait and H. Youssef, *VLSI Physical Design Automation.* McGraw-Hill, 1995.

[15] T. Takahashi, “A New Encoding Scheme for Rectangle Packing Problem,” in *Proc. ASP-DAC*, 2000, pp. 175–178.

[16] X. Tang and D. F. Wong, “FAST-SP: A Fast Algorithm for Block Placement Based on Sequence Pair,” in *Proc. ASP-DAC*, 2001, pp. 521–526.

[17] T.-C. Wang, and D. F. Wong, “An Optimal Algorithm for Floorplan and Area Optimization,” in *Proc DAC*, 1990, pp.180–186.

[18] D. F. Wong, and C.-L. Liu, “A New Algorithm for Floorplan Design,” in *Proc DAC*, 1986, pp. 101–107.

1

1

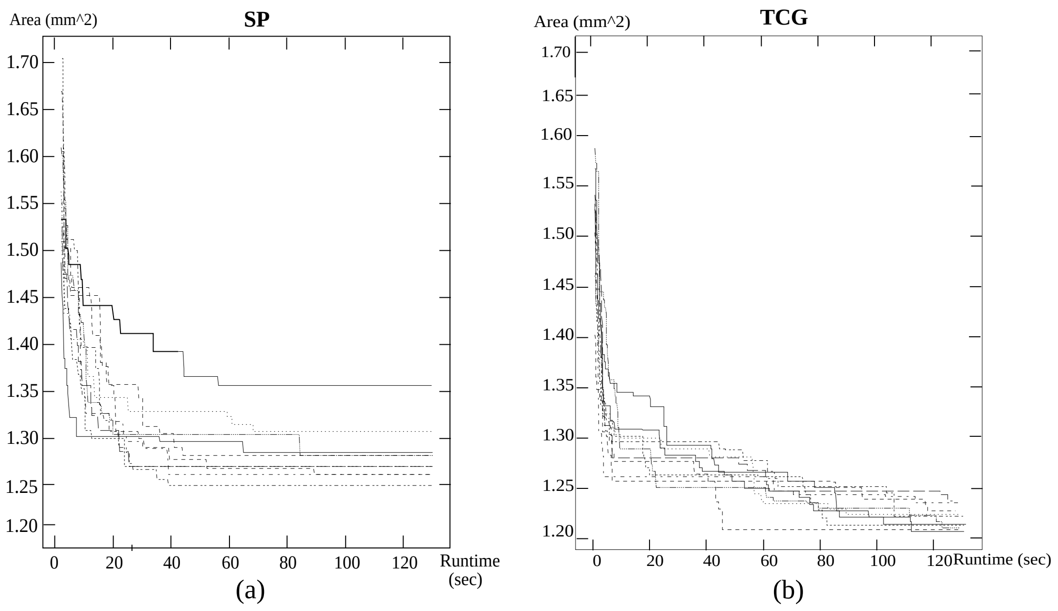

Fig. 8. Stability and convergence comparison between SP and TCG for ami33. (a) SP. (b) TCG.