Transactions Briefs

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

Jai-Ming Lin and Yao-Wen Chang

***Abstract*—In this brief, we introduce the concept of the P*-admissible representation and propose a P*-admissible, transitive closure graph-based representation for general floorplans, called transitive closure graph (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 a 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 of 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.**

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

For the nonslicing floorplan structure, there exist several well-known "old" graph-based representations, e.g., *polar graphs* in [9] and *adjacency graphs* and *channel intersection graph* in [14]. Recently, the nonslicing floorplan representations have attracted much attention in the literature, 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 the sequence pair (SP), 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;
4) the best evaluated packing in the space corresponds to an optimal placement.

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, preplaced modules, and range constraints. 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 (BSG) representation. 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. (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. 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. Given an O-tree or a B*-tree, it may not be feasible to find a placement corresponding to its original representation.

Recently, Hong *et al.* in [3] proposed a corner block list (CBL) representation for *mosaic* floorplans. In a mosaic floorplan, each region must contain exactly one module. 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 brief a transitive closure graph (TCG)-based representation for general nonslicing floorplans and show its superior properties. To differentiate the properties of TCG from other existing representations, we extend in this brief 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 and symmetry constraints). The representation after packing corresponds to the original

Manuscript received September 7, 2002; revised February 28, 2003. This work was supported in part by the National Science Council of Taiwan ROC under Grant NSC-89-2215-E-009-117.

J.-M. Lin is with Realtek Semiconductor Corporation, Hsinchu 300, Taiwan, R.O.C. (e-mail: gis87808@cis.nctu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: ywchang@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2004.840760

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.

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. All of 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 SP, O-tree, enhanced O-tree, B*-tree, and CBL, respectively. Optimizing wirelength, TCG obtained respective average improvements of 5.04%, 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 brief 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 $B = \{b_1, b_2, \ldots, b_m\}$ be a set of m rectangular modules whose width, height, and area are denoted by W_i, H_i, and A_i, $1 \le i \le m$. Each module is free to rotate. Let (x_i, y_i) denote the coordinate of the bottom-left corner of rectangle b_i, $1 \le i \le m$, on a chip. A placement $\mathcal{P}$ is an assignment of (x_i, y_i) for each b_i, $1 \le i \le m$, 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

The *transitive closure* of a directed acyclic graph G is defined as the graph $G' = (V, E')$, where $E' = \{(n_i, n_j) : \text{there is a path from node } n_i \text{ to node } n_j \text{ in } G\}$. 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.

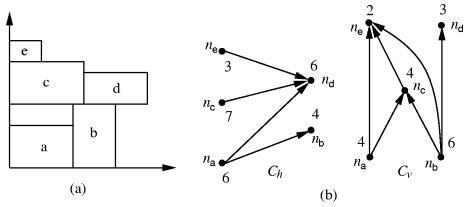


Fig. 1. (a) Placement in a chip. (b) TCG.

A. From a Placement to Its TCG

For two nonoverlapped 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 nonoverlapped 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. 1(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. 1(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 of the geometric relations among modules (i.e., C_h and C_v have $m(m-1)/2$ edges in total).

As shown in Fig. 1(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. 1(b), each transitive closure graph has five nodes, and there are in total ten edges in C_h and C_v (four in C_h and six in C_v).

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.

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 the 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.
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.

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: We can prove there exists a one-to-one correspondence between TCG (C_h, C_v) and a sequence pair (Γ_+, Γ_-). Since there are $(m!)^2$ such sequence pairs, the theorem thus follows. ■

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

Theorem 3: TCG is P^*-admissible.

IV. Floorplanning Algorithm

We develop a simulated annealing-based algorithm [4] using TCG for a nonslicing 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*. 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 $F_{\mathrm{in}}(n_i)$ $(F_{\mathrm{out}}(n_i))$, as the nodes n_j's with edges (n_j, n_i) $((n_i, n_j))$. For nodes n_k, $n_l \in F_{\mathrm{out}}(n_i)$, the edge (n_i, n_k) cannot be a reduction edge if $n_k \in F_{\mathrm{out}}(n_l)$. Hence, we remove those nodes in $F_{\mathrm{out}}(n_i)$ that are fan-outs of others. The edges between n_i and the remaining nodes in $F_{\mathrm{out}}(n_i)$ are reduction edges. For the C_v shown in Fig. 1(a), $F_{\mathrm{out}}(n_a) = \{n_c, n_e\}$. Since n_e belongs to $F_{\mathrm{out}}(n_c)$, 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, $n_l \in F_{\mathrm{out}}(n_i)$, the edge (n_i, n_k) cannot be a reduction edge if $n_k \in F_{\mathrm{out}}(n_l)$.

Proof: For nodes n_k, $n_l \in F_{\mathrm{out}}(n_i)$ and $n_k \in F_{\mathrm{out}}(n_l)$, the edge (n_i, n_k) cannot be a reduction edge because there exists at least a path $\langle n_i, n_l, n_k \rangle$ 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 $F_{\mathrm{out}}(n_i)$. For the nodes in $F_{\mathrm{out}}(n_i)$, we pick a node n_j from $F_{\mathrm{out}}(n_i)$ and remove eack node $n_k \in F_{\mathrm{out}}(n_j)$ from $F_{\mathrm{out}}(n_i)$. 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.

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.

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

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

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%	-

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. ■

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.

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. ■

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.

To maintain the properties of a TCG, we can only reverse a reduction edge. 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 $\langle n_x, \ldots, n_j, n_i, \ldots, n_y \rangle$ 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 $n_k \in F_{\rm in}(n_j) \cup \{n_j\}$ 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 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 ($n_k \in F_{\rm in}(n_j) \cup \{n_j\}$ and $n_l \in F_{\rm out}(n_i) \cup \{n_i\}$) 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. ■

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 $n_k \in F_{\rm in}(n_i) \cup \{n_i\}$ and $n_l \in F_{\rm out}(n_j) \cup \{n_j\}$, 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.

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 exists 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.

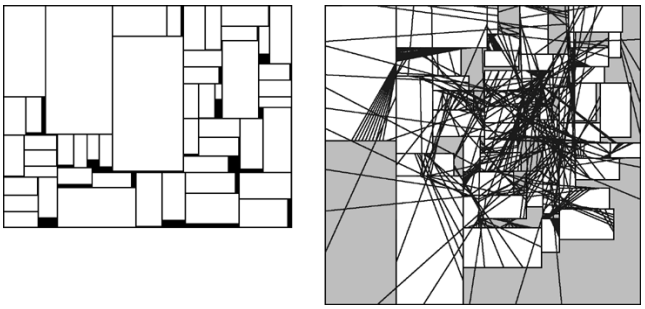
Theorem 8: TCG is closed under the move operation, and such an operation takes $O(m^2)$ time, where m is the number of modules in the placement.

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

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

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%	-



(a) (b)

Fig. 2. Resulting placements of ami49 for (a) optimizing area alone (area = 36.77 mm^2) and (b) optimizing wirelength alone (wire = 604 mm).

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 $\langle n_x, \ldots, n_i, n_j, \ldots, n_y \rangle$ 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 $n_k \in F_{\text{in}}(n_i) \cup \{n_i\}$ and $n_l \in F_{\text{out}}(n_j) \cup \{n_j\}$ 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. ■

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.[1] We compared TCG with O-tree [2], B*-tree [1], enhanced O-tree [13], and CBL [3] based on the five MCNC benchmark circuits.

The experiments consist of two parts: area optimization 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 "SP" [7], O-tree [2], B*-tree [1], enhanced O-tree [13], CBL [3], and TCG are listed in Table I. As shown in Table I, TCG achieves average improvements of 5.04%, 2.22%, 1.18%, 2.04%, and 3.54% in area utilization compared to "SP", O-tree, B*-tree, enhanced O-tree, and CBL, respectively. The runtimes are significantly smaller than that for O-tree and B*-tree and are comparable to that for the enhanced O-tree [13]. Fig. 2(a) 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 II. (Note that B*-tree and CBL did not report the results on optimizing wirelength alone.) As shown in Table II, TCG achieves average reductions of 3.56% and 3.18% in wirelength, compared to those for the O-tree and the enhanced O-tree, respectively. Fig. 2(b) shows the resulting placement for ami49 with wirelength optimization.

[1]The TCG package is available at http://cc.ee.ntu.edu.tw/~ywchang/research.html

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.

REFERENCES

[1] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for nonslicing floorplans," in *Proc. DAC*, 2000, pp. 458–463.

[2] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of nonslicing 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 nonslicing 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*. New York: Holt, Rinehart, and Winston, 1976.

[6] J.-M. Lin and Y.-W. Chang, "TCG: a transitive closure graph-based representation for nonslicing 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] T. Ohtsuki, 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.

[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. 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*. New York: 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.