

# TCG-S: Orthogonal Coupling of P\*-admissible Representations for General Floorplans \*

Jai-Ming Lin<sup>1</sup> and Yao-Wen Chang<sup>2</sup>

<sup>1</sup>Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

<sup>2</sup>Department of Electrical Engineering & Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan

## Abstract

We extend in this paper the concept of the P-admissible floorplan representation to that of the P\*-admissible one. A P\*-admissible representation can model the most general floorplans. Each of the currently existing P\*-admissible representations, SP, BSG, and TCG, has its strengths as well as weaknesses. We show the equivalence of the two most promising P\*-admissible representations, TCG and SP, and integrate TCG with a packing sequence (part of SP) into a new representation, called TCG-S. TCG-S combines the advantages of SP and TCG and at the same time eliminates their disadvantages. With the property of SP, faster packing and perturbation schemes are possible. Inherited nice properties from TCG, the geometric relations among modules are transparent to TCG-S (implying faster convergence to a desired solution), placement with position constraints becomes much easier, and incremental update for cost evaluation can be realized. These nice properties make TCG-S a superior representation which exhibits an elegant solution structure to facilitate the search for a desired floorplan/placement. Extensive experiments show that TCG-S results in the best area utilization, wirelength optimization, convergence speed, and stability among existing works and is very flexible in handling placement with special constraints.

## 1 Introduction

As technology advances, the circuit size in modern VLSI design increases dramatically. To handle the increasing design complexity, hierarchical designs and IP modules are widely used to optimize area and timing for design convergence. Further, the need to integrate heterogeneous systems or special modules imposes some placement constraints, e.g., the boundary-module constraint which requires some modules to be placed along the chip boundaries for shorter connections to pads, the preplaced-module constraint which pre-assigns modules to specific positions, etc. These trends make floorplanning/placement much more important than ever, and it is of particular significance to consider the floorplanning/placement with various constraints. To cope with these challenges, it is desired to develop an efficient and effective floorplan representation that can model the geometric relations among regular as well as constrained modules.

Many floorplan representations have been proposed in the literature, e.g., slicing tree [13], normalized Polish expression (NPE) [18], Sequence Pair (SP) [10], Bounded-Sliceline Grid (BSG) [12], O-tree [3], B\*-tree [1], Corner Block List (CBL) [4], Transitive Closure Graph (TCG) [8], and Q-sequence [15]. Unlike the traditional classification of the slicing and non-slicing structures, we can alternatively classify them into two categories, *P\*-admissible* and *non-P\*-admissible* representations. A representation is said to be P-admissible if it satisfies the following four conditions [10]: (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. We extend in this paper the concept of the P-admissible representation to that of the P\*-admissible one by

adding the fifth condition: (5) the geometric relation between each pair of modules is defined in the representation. With this condition, general floorplans/placements can be modeled. Therefore, a P\*-admissible representation contains a complete structure for searching for an optimal floorplan/placement solution. Therefore, 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 tree, NPE, O-tree, B\*-tree, CBL, and Q-sequence are not. The slicing tree and NPE are intended for slicing floorplans only. Since an optimal placement could be a non-slicing structure, the two representations are not P-admissible and thus not P\*-admissible (i.e., violation of P\*-admissibility Condition (4)). An O-tree defines only one-dimensional geometrical relation between *compacted* modules and thus can obtain the relation in the other dimension only after packing (i.e., violation of Condition (5)). (Note that O-tree is undefined for some uncompact placements which may correspond to the best solutions for wirelength optimization. Therefore, as far as wirelength optimization is concerned, O-tree is not even P-admissible since Condition (4) is violated.) A B\*-tree requires a placement to be left and/or bottom compacted. However, the space intended for placing a module may be occupied by previously placed modules during packing, resulting in a mismatch between the original representation and its *compacted* placement. Therefore, it may not be feasible to find a *compacted* placement corresponding to the original B\*-tree, and thus it is not P-admissible (i.e., violation of Condition (2)). CBL and Q-sequence can represent only *mosaic* floorplans, in which each region in the floorplan contains exactly one module. CBL and Q-sequence are not P-admissible because it cannot guarantee a feasible solution after a perturbation (i.e., violation of Conditions (2) and (4)). Non-P\*-admissible representations intrinsically have a smaller solution space and lower packing cost since their corresponding floorplanning/placement structures are more restricted (e.g., slicing structures [slicing tree, NPE], mosaic floorplans [CBL, Q-sequence], compacted placements [O-tree, B\*-tree], etc.).<sup>1</sup> However, lack of the definition on the geometric relation between each pair of modules, the guarantee in the feasibility, and/or the optimality of their representations would inevitably complicate floorplan sizing and placement with constraints, lead to longer running times, and/or lower solution quality.

The existing P\*-admissible representations, SP, BSG, and TCG, have their own distinct properties as well as common ones. Nevertheless, researchers tend to favor SP over BSG because BSG incurs many redundancies and thus a much larger solution space, implying a longer running time to search for a good solution. Therefore, we shall focus on SP and TCG. Both SP and TCG are considered very flexible representations and construct constraint graphs to evaluate their packing cost. SP consists of two sequences of modules ( $\Gamma_+$ ,  $\Gamma_-$ ), where  $\Gamma_+$  ( $\Gamma_-$ ) specifies the module ordering from top-left (bottom-left) to bottom-right (top-right). Hence,  $\Gamma_-$  corresponds to the ordering for packing modules to the bottom-left direction and thus can be used to guide module packing. However, like most existing representations (e.g., NPE, BSG, O-tree, B\*-tree, CBL, Q-sequence), the geometric relations between modules are *not transparent* to the operations of SP (i.e., the effect of an operation on the change of module relation is not clear before packing), and thus we need to construct constraint graphs from scratch after each perturbation to evaluate the packing cost; this deficiency makes SP harder to converge to a desired solution and to handle placement with constraints

\*This work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC-89-2215-E-009-117. E-mails: gis87808@cis.nctu.edu.tw; ywchang@cc.ee.ntu.edu.tw.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA. Copyright 2002 ACM 1-58113-461-4/02/0006...\$5.00

<sup>1</sup>Generality from the most general to the least general: general floorplan > compacted floorplan > mosaic floorplan > slicing floorplan.

(e.g., boundary modules, pre-placed modules, etc).

TCG consists of a horizontal transitive closure graph  $C_h$  to define the horizontal geometric relations between modules and a vertical one  $C_v$  for vertical geometric relations. Contrast to SP, the geometric relations between modules are transparent to TCG as well as its operations, facilitating the convergence to a desired solution. 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. Nevertheless, like SP, constraint graphs are also needed for TCG to evaluate its packing cost, and unlike SP, we need to perform extra operations to obtain the module packing sequence.

Therefore, an interesting question arises: Is it possible to develop a representation that can combine the advantages of SP and TCG and at the same time eliminate their disadvantages? We answer this question in affirmation by showing the equivalence of TCG and SP, and integrating them into TCG-S =  $(C_h, C_v, \Gamma_-)$ . The orthogonal combination leads to a representation with at least the following advantages:

- With the property of SP, faster  $O(m \lg m)$ -time packing and  $O(m)$ -time perturbation schemes are possible for a P\*-admissible representation, where  $m$  is the number of modules. (Note that a linear-time packing scheme is possible for the tree-based representations, but they can represent only more restricted compacted floorplans.)
- Inherited from TCG, the geometric relations among modules are transparent to TCG-S, implying faster convergence to a desired solution.
- Inherited from TCG, placement with position constraints becomes much easier.
- TCG-S can support incremental update for cost evaluation.

These nice properties make TCG-S an effective, efficient, and flexible representation. Extensive experiments based on a set of commonly used MCNC benchmarks show that TCG-S results in the best area utilization, wirelength optimization, convergence speed, and solution stability among existing works.

To show the flexibility of TCG-S, we also consider placement with preplaced and boundary modules. For placement with pre-placed modules, Murata et al. [11] proposed an *adaptation algorithm* to transform an infeasible SP with preplaced modules into a feasible one. However, the process incurs expensive computations. For placement with the boundary-module constraint, Tang and Wong in [16] and Ma et al. in [9] handled this problem using SP and CBL, respectively. However, they cannot guarantee a feasible solution in each perturbation and their final placements. Lai et al. in [6] gave the feasibility conditions for SP with boundary modules and transformed an infeasible solution into a feasible one. However, the method is very complex, and many rules are needed to cope with the constraints.

We present in this paper the methods for handling placement with preplaced and boundary modules. Different from the previous works on boundary modules that cannot guarantee a feasible solution or need to transform an infeasible solution into a feasible one, TCG-S can easily maintain the feasibility during each perturbation. We compared our work with [6] on placement with boundary modules. (Note that there are no common benchmark circuits for this constraint.) Experimental results show that TCG-S results in smaller areas than [6].

The remainder of this paper is organized as follows. Section 2 formulates the floorplan/placement design problem. Section 3 compares SP and TCG. Section 4 presents the procedures to build the TCG-S from a placement and construct the placement from a TCG-S. Section 5 gives the operations to perturb a TCG-S. Section 6 presents our methods to handle placement with boundary and preplaced modules. Experimental results are reported in Section 7. Finally, we give concluding remarks in Section 8.

## 2 Problem Definition

Let  $B = \{b_1, b_2, \dots, b_m\}$  be a set of  $m$  rectangular modules whose width, height, and area are denoted by  $W_i, H_i$ , and  $A_i$ ,  $1 \leq i \leq m$ . Let  $(x_i, y_i)$  ( $(x'_i, y'_i)$ ) denote coordinate of the bottom-left (top-right) corner of module  $b_i$ ,  $1 \leq i \leq m$ , 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/or the wirelength (i.e., the summation of half bounding box of interconnections) induced by the assignment of  $b_i$ 's on the chip.

## 3 P\*-admissible Representations

In this section, we first review the two P\*-admissible representations, TCG and SP, then show their equivalence, and compare their properties.

### 3.1 Review of TCG and SP

TCG describes the geometric relations between modules based on two graphs, namely a *horizontal transitive closure graph*  $C_h$  and a *vertical transitive closure graph*  $C_v$ , in which a node  $n_i$  represents a module  $b_i$  and an edge  $(n_i, n_j)$  in  $C_h$  ( $C_v$ ) denotes that module  $b_i$  is left to (below) module  $b_j$ . TCG has the following three *feasibility properties* [8]:

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.<sup>2</sup>

Figure 1(a) shows a placement with seven modules  $a, b, c, d, e, f$ , and  $g$  whose widths and heights are  $(3.5, 1.5)$ ,  $(2, 2.5)$ ,  $(2, 3.5)$ ,  $(3, 2)$ ,  $(1.5, 1.5)$ ,  $(5, 1.5)$ , and  $(1, 2)$ , respectively. Figure 1(c) shows the  $\text{TCG} = (C_h, C_v)$  corresponding to the placement of Figure 1(a). The value associated with a node in  $C_h$  ( $C_v$ ) gives the width (height) of the corresponding module, and the edge  $(n_i, n_j)$  in  $C_h$  ( $C_v$ ) denotes the horizontal (vertical) relation of  $b_i$  and  $b_j$ . Since there exists an edge  $(n_a, n_g)$  in  $C_h$ , module  $b_a$  is left to  $b_g$ . Similarly,  $b_a$  is below  $b_b$  since there exists an edge  $(n_a, n_b)$  in  $C_v$ .

SP uses a pair of sequences  $(\Gamma_+, \Gamma_-)$  to represent a floorplan/placement, where  $\Gamma_+$  and  $\Gamma_-$  give two permutations of module names. The geometric relation of modules can be derived from an SP as follows. Module  $b_a$  is left (right) to module  $b_b$  if  $a$  appears before (after)  $b$  in both  $\Gamma_+$  and  $\Gamma_-$ . Module  $b_a$  is below (above) module  $b_b$  if  $b$  appears before (after)  $a$  in  $\Gamma_+$  and  $a$  appears before (after)  $b$  in  $\Gamma_-$ . Figure 1(b) shows the corresponding SP. Since  $a$  is before  $g$  in both  $\Gamma_+$  and  $\Gamma_-$ , module  $b_a$  is left to module  $b_g$ . Similarly,  $b_a$  is below  $b_b$  since  $a$  is after  $b$  in  $\Gamma_+$  and before  $b$  in  $\Gamma_-$ .

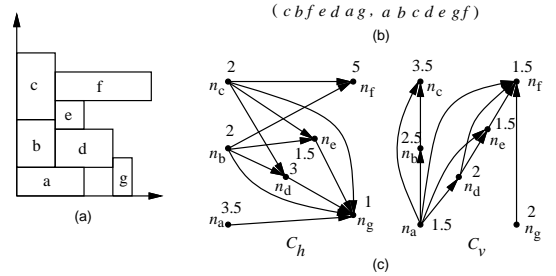


Figure 1: (a) A placement. (b) The corresponding SP of (a). (c) The corresponding TCG of (a).

### 3.2 Equivalence of SP and TCG

Like the relations between a skewed slicing tree [13] and an NPE [18] for slicing floorplans as well as O-tree and B\*-tree for non-slicing floorplans, TCG and SP are equivalent.

We can transform between TCG and SP as 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 obtain a sequence  $\Gamma_+$  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  $\Gamma_-$  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  $C_v$ . Given an SP =  $(\Gamma_+, \Gamma_-)$ , we can obtain a unique TCG =  $(C_h, C_v)$  from the two constraint graphs of the SP by removing the source, sink, and associated edges. For example, the SP of Figures 1(b) is equivalent to the TCG of Figures 1(c).

<sup>2</sup>The transitive closure of a directed acyclic graph  $G$  is defined as the graph  $G' = (V, E')$ , where  $E' = \{(n_i, n_j) : \text{there is a path from node } n_i \text{ to node } n_j \text{ in } G\}$ .

### 3.3 Comparison between TCG and SP

Although TCG and SP are equivalent, their properties and induced operations are significantly different. (Similar situations are also with skewed slicing trees/NPE's and O-trees/B\*-trees.) Both SP and TCG are considered very flexible representations and construct constraint graphs to evaluate their packing cost.  $\Gamma_-$  of an SP corresponds to the ordering for packing modules to the bottom-left direction and thus can be used for guiding module packing. However, like most existing representations, the geometric relations among modules are not transparent to the operations of SP (i.e., the effect of an operation on the change of module relation is not clear before packing), and thus we need to construct constraint graphs from scratch after each perturbation to evaluate the packing cost; this deficiency makes SP harder to converge to a desired solution and to handle placement with constraints (e.g., boundary modules, pre-placed modules, etc).

Contrast to SP, the geometric relations among modules are transparent to TCG as well as its operations, facilitating the convergence to a desired solution. 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. Unlike SP, nevertheless, we need to perform extra operations to obtain the module packing sequence and an additional  $O(m^2)$  time to find a special type of edges, called *reduction edges*, in  $C_h$  ( $C_v$ ) for some operations. (We will define the edges later.)

For both SP and TCG, the packing scheme by applying the longest path algorithm is time-consuming since all edges in the constraint graphs are processed, even though they are not on the longest path. As shown in  $C_h$  of Figure 1(c), if we add a source with zero weight and connect it to those nodes with zero in-degree, the  $x$  coordinate of each module can be obtained by applying the longest path algorithm on the resulting directed acyclic graph. Therefore, we have  $x_g = \max\{x'_a, x'_b, x'_c, x'_d, x'_e\}$ . However, if we place modules based on the sequence  $\Gamma_-$  and maintain a horizontal and a vertical contours, denoted by  $R_h$  and  $R_v$  respectively, for the placed modules, the number of nodes need to be considered can be reduced. Let  $R_h$  ( $R_v$ ) be a list of modules  $b_i$ 's for which there exists no module  $b_j$  with  $y_j \geq y'_i$  ( $x_j \geq x'_i$ ) and  $x'_j \geq x'_i$  ( $y'_j \geq y'_i$ ). For the placement of Figure 1(a), for example,  $R_h = \langle b_c, b_f \rangle$  and  $R_v = \langle b_g, b_d, b_e, b_f, b_c \rangle$ . Suppose we have packed the modules  $b_a, b_b, b_c, b_d$ , and  $b_e$  based on the sequence  $\Gamma_-$ . Then, the resulting horizontal contour  $R_h = \langle b_c, b_e, b_d \rangle$ . Keeping  $R_h$ , we only need to traverse the contour from  $b_e$ , predecessors of  $b_e$ , to the last module  $b_d$ , which have horizontal relations with  $b_g$  (since there are edges  $(n_e, n_g)$  and  $(n_d, n_g)$  in  $C_h$ ). Thus, we have  $x_g = x'_d$ . Packing modules in this way, we only need to consider  $x_e$  and  $x_d$ , and can get rid of the computation for a maximum value, leading to a faster packing scheme. We will show later how to apply a balanced binary tree to implement the contour operation to get a loglinear-time packing scheme.

## 4 The TCG-S Representation

Combining  $TCG = (C_h, C_v)$  and  $SP = (\Gamma_+, \Gamma_-)$ , we develop a representation, called TCG-S =  $(C_h, C_v, \Gamma_-)$ , which uses a horizontal and a vertical transitive closure graphs as well as a packing sequence  $\Gamma_-$  to represent a placement. In this section, we first introduce how to construct  $\Gamma_-$ ,  $C_h$ , and  $C_v$  from a placement. Then, we propose an  $O(m \lg m)$ -time packing scheme for TCG-S, where  $m$  is the number of modules.

### 4.1 From a placement to TCG-S

In this subsection, we first extract  $\Gamma_-$  from a placement, and then construct  $C_h$  and  $C_v$  according to  $\Gamma_-$ .

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 left to (below)  $b_j$  and their projections on the  $y$  ( $x$ ) axis 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 left to  $b_j$  and their projections on the  $x$  and the  $y$  axes do not overlap. To simplify the operations on geometric relations, a diagonal relation for modules  $b_i$  and  $b_j$  is treated as a horizontal one unless there exists a chain of vertical relations from  $b_i$  ( $b_j$ ), followed by the modules overlapped with the rectangle defined by the two closest corners of  $b_i$  and  $b_j$ , and finally to  $b_j$  ( $b_i$ ), for which it is considered as  $b_i \perp b_j$  ( $b_j \perp b_i$ ).

Given a placement,  $\Gamma_-$  can be extracted based on the procedure described in [10]. For example, the  $\Gamma_-$  for the placement of Figure 2(a) is  $\langle abcdegf \rangle$ . After extracting  $\Gamma_-$ , we can construct  $C_h$  and  $C_v$  based on  $\Gamma_-$ . For each module  $b_i$  in  $\Gamma_-$ , we introduce a node  $n_i$  with the weight being  $b_i$ 's width (height) in  $C_h$  ( $C_v$ ). Also, for each module  $b_i$  before  $b_j$  in  $\Gamma_-$ , we introduce an edge  $(n_i, n_j)$  in  $C_h$  ( $C_v$ ) if  $b_i \vdash b_j$  ( $b_i \perp b_j$ ). As shown in Figures 2(b), for the first two modules  $b_a, b_b$  in  $\Gamma_-$ , we introduce the nodes  $n_a$  and  $n_b$  in  $C_h$  ( $C_v$ ) and assign the weights as their widths (heights). Also, we construct a directed edge  $(n_a, n_b)$  in  $C_v$  since module  $b_a$  is before  $b_b$  and  $b_a \perp b_b$ . The process repeats for all modules in  $\Gamma_-$ , resulting in the TCG-S shown in Figure 2(b). We have the following theorem.

**Theorem 1** *There exists a TCG-S corresponding to a placement.*

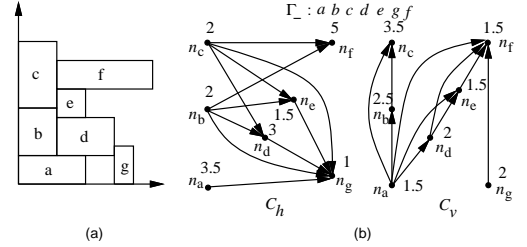


Figure 2: (a) A placement. (b) The corresponding TCG-S of (a).

### 4.2 From TCG-S to a placement

In this subsection, we propose an  $O(m \lg m)$ -time packing scheme based on  $\Gamma_-$  as well as a horizontal and a vertical contours  $R_h$  and  $R_v$ , where  $m$  is the number of modules. The basic idea is to process the modules based on the sequence defined in  $\Gamma_-$ , and then pack the current module to a corner formed by two previously placed modules in  $R_h$  ( $R_v$ ) according to the geometric relations defined in  $C_h$  and  $C_v$ .

We detail the packing scheme as follows. Recall that  $R_h$  ( $R_v$ ) is a list of modules  $b_i$ 's for which there exists no module  $b_j$  with  $y_j \geq y'_i$  ( $x_j \geq x'_i$ ) and  $x'_j \geq x'_i$  ( $y'_j \geq y'_i$ ). ( $R_h$  ( $R_v$ ) consists of modules along the top (left) boundary of a placement.) We can keep the modules  $b_i$ 's in  $R_h$  ( $R_v$ ) in a balanced binary search tree (e.g., the red-black tree [2])  $T_h$  ( $T_v$ ) in the increasing order according to their right (top) boundaries. For easier presentation, we add a *dummy module*  $b_s$  ( $b_t$ ) to  $R_h$  ( $R_v$ ) to denote the left (bottom) boundary module of a placement. We have  $b_s \vdash b_i$  and  $b_t \perp b_i, \forall b_i$ . Let  $(x'_s, y'_s) = (0, \infty)$  and  $(x'_t, y'_t) = (\infty, 0)$ .  $R_h$  ( $R_v$ ) consists of  $b_s$  ( $b_t$ ) initially, and so does the corresponding  $T_h$  ( $T_v$ ). To pack a module  $b_j$  in  $\Gamma_-$ , we traverse the modules  $b_k$ 's in  $T_h$  ( $T_v$ ) from its root, and go to the right child if  $b_k \vdash b_j$  ( $b_k \perp b_j$ ) and the left child if  $b_k \perp b_j$  ( $b_k \vdash b_j$ ). The process is repeated for the newly encountered module until a leaf node is met. Then,  $b_j$  is connected to the leaf node, and  $x_j = x'_p$  ( $y_j = y'_p$ ), where  $b_p$  is the last module with  $b_p \vdash b_j$  ( $b_p \perp b_j$ ) in the path. After  $b_j$  is inserted into  $T_h$  ( $T_v$ ), every predecessor  $b_l$  with  $x'_l \leq x'_j$  ( $y'_l \leq y'_j$ ) in  $T_h$  ( $T_v$ ) is deleted since  $b_l$  is no longer in the contour. (Note that the ordering of nodes in  $T_h$  ( $T_v$ ) can be obtained by depth-first search.) This process repeats for all modules in  $\Gamma_-$ . We have  $W = x'_v$  ( $H = y'_v$ ) if  $b_v$  is the module in the resulting  $T_h$  ( $T_v$ ) with the largest value, where  $W$  ( $H$ ) denotes the width (height) of the placement. We have the following theorems and lemmas.

**Theorem 2** *There exists a unique placement corresponding to a TCG-S.*

**Lemma 1** *For each module  $b_i$  in  $\Gamma_-$ ,  $b_i$  must be placed adjacent to the right (top) boundary of some module  $b_j$  in  $R_h$  ( $R_v$ ) during the packing.*

**Lemma 2** *Given a module  $b_i$  in  $\Gamma_-$  to be placed, if  $b_j \in R_h$  ( $R_v$ ),  $b_j \perp b_i$  ( $b_j \vdash b_i$ ), and  $b_j$  has the largest  $x'_j$  ( $y'_j$ ), any module  $b_k \in R_h$  ( $R_v$ ) with  $x'_k > x'_j$  ( $y'_k > y'_j$ ) cannot have the relation  $b_k \vdash b_i$  ( $b_k \perp b_i$ ).*

**Theorem 3** *The proposed scheme correctly packs all modules in  $O(m \lg m)$  time, where  $m$  is the number of modules.*

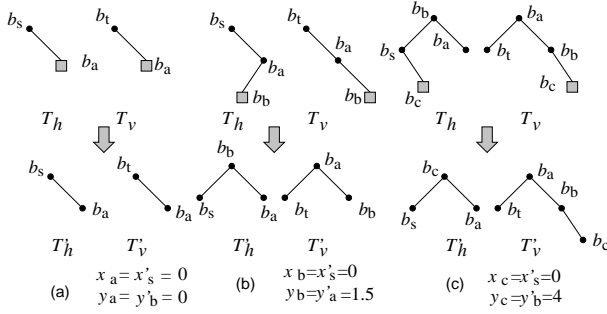


Figure 3: The packing scheme for the TCG-S of Figure 2(b).

Figure 3 shows the packing scheme for the TCG-S of Figure 2(b).  $T_h$  ( $T_v$ ) consists of  $b_s$  ( $b_t$ ) initially. To pack the first module  $b_a$  in  $\Gamma_-$ , we traverse  $T_h$  ( $T_v$ ) from the root  $b_s$  ( $b_t$ ) and insert it to the right child of  $b_s$  ( $b_t$ ) since  $b_s \vdash b_a$  ( $b_t \perp b_a$ ). Therefore, the first module  $b_a$  in  $\Gamma_-$  is placed at the bottom-left corner (i.e.,  $(x_a, y_a) = (0, 0)$ ) since  $b_s$  ( $b_t$ ) is the last module that is horizontally (vertically) related to  $b_a$  and  $x'_s = 0$  ( $y'_t = 0$ ). (Note that  $T'_h$  ( $T'_v$ ) in Figures 3(a) denotes a balanced binary search tree after  $b_a$  is inserted into  $T_h$  ( $T_v$ ).) Similarly, to pack the second module  $b_b$  in  $\Gamma_-$ , we traverse  $T_h$  from the root  $b_s$  and then its right child since  $b_s \vdash b_b$ . Then,  $b_b$  is inserted to the left child of  $b_a$  since  $b_s \vdash b_a$ . Because  $b_s$  is the last module with  $b_s \vdash b_b$  in the path,  $x_b = x'_s = 0$ . Similarly, we traverse  $T_v$  from the root  $b_t$  and then its right child  $b_a$  since  $b_a \perp b_t$ . Then,  $b_b$  is inserted to the right child of  $b_a$  in  $T_v$  since  $b_a \perp b_b$ . Therefore,  $y_b = y'_a = 1.5$  because  $b_a$  is the last module with  $b_a \perp b_b$  in the path. The resulting balanced binary search trees after performing tree rotations  $T'_h$ ,  $T'_v$  are shown in Figure 3(b) (see [2] for the rotation operations for keeping a tree balanced). As shown in Figure 3(c), after  $b_c$  is inserted,  $b_b$  in  $T_h$  is deleted since  $b_b$  is a predecessor of  $b_c$  and  $x'_b \leq x'_c$  (i.e.,  $b_b$  is no longer in the contour). The resulting  $T'_h$  and  $T'_v$  are shown in Figure 3(c). The process is repeated for all modules in  $\Gamma_-$ .

According to this packing scheme, if the coordinate of a module  $b_i$  in  $\Gamma_-$  is changed, we only need to recompute the coordinates of modules after  $b_i$  in  $\Gamma_-$  since the coordinates of modules before  $b_i$  do not change.

## 5 Floorplanning Algorithm

We develop a simulated annealing based algorithm [5] by using TCG-S for general floorplan design. Given an initial TCG-S, the algorithm perturbs the TCG-S into a new TCG-S to find a desired solution. To ensure the correctness of the new  $C_h$  and  $C_v$ , they must satisfy the three feasibility conditions given in Section 3.1. To identify feasible  $C_h$  and  $C_v$  for perturbation, we describe the concept of *reduction edges* in the following subsection.

### 5.1 Reduction Edge

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 some operations. In Figure 2(b), for example, edges  $(n_a, n_g)$ ,  $(n_d, n_g)$ , and  $(n_e, n_g)$  are reduction edges while  $(n_b, n_g)$  and  $(n_c, n_g)$  are closure ones. With  $\Gamma_-$ , we can find a set of reduction edges in  $O(m)$  time (where  $m$  is the number of modules), a significant improvement from  $O(m^2)$  time using TCG alone [8].

Given an arbitrary node  $n_i$  in a transitive closure graph  $C_h$  ( $C_v$ ), we can find all the nodes  $n_j$ 's that form reduction edges  $(n_i, n_j)$ 's using a *Linear Scan* method as follows. First, we extract from  $\Gamma_-$  those nodes  $n_j$ 's in  $F_{out}(n_i)$  of  $C_h$  ( $C_v$ ) and keep their original ordering in  $\Gamma_-$ . Let the resulting sequence be  $S$ . The first node  $n_k$  in  $S$  and  $n_i$  must form a reduction edge  $(n_i, n_k)$ . Then, we continue to traverse  $S$  until a node  $n_l$  with  $(n_k, n_l)$  not in  $C_h$  ( $C_v$ ) is encountered.  $(n_i, n_l)$  must also be a reduction edge. Starting from  $n_l$ , we continue the same process until no node is left in  $S$ .

As an example shown in  $C_h$  of Figure 2(b), we are to extract all reduction edges emanating from  $n_e$ . We first find  $S = \langle n_d, n_e, n_g, n_f \rangle$  by extracting nodes in  $F_{out}(n_e)$  based on the sequence in  $\Gamma_-$ .  $n_e$  and the first node  $n_d$  in  $S$  form a reduction edge  $(n_e, n_d)$ . Traversing  $S$ , we have another reduction edge  $(n_e, n_g)$  since edge  $(n_d, n_g)$  is not in  $C_h$ .

Starting from  $n_e$ , we search the next node  $n$  with  $(n_e, n)$  not in  $C_h$ . We find node  $n_f$ , implying that  $(n_e, n_f)$  is also a reduction edge. Therefore, we have found all reduction edges emanating from  $n_e$ :  $(n_e, n_d)$ ,  $(n_e, n_g)$ , and  $(n_e, n_f)$ . (Note that  $(n_e, n_g)$  is not a reduction edge because we found  $(n_e, n_g)$  in  $C_h$  during the processing.)

**Theorem 4** Given a node  $n_i$  in  $C_h$  ( $C_v$ ), the *Linear Scan* method finds all reduction edges emanating from  $n_i$  in  $O(m)$  time, where  $m$  is the number of modules.

## 5.2 Solution Perturbation

We extend the four operations *Rotation*, *Swap*, *Reverse*, and *Move* presented in [8] to perturb  $C_h$  and  $C_v$ . During each perturbation, we must maintain the three feasibility properties for  $C_h$  and  $C_v$ . Unlike the *Rotation* operation, *Swap*, *Reverse*, and *Move* may change the configurations of  $C_h$  and  $C_v$  and thus their properties. Further, we also need to maintain  $\Gamma_-$  to conform to the topological ordering of the new  $C_h$  and  $C_v$ .

### 5.2.1 Rotation

To Rotate a module  $b_i$ , we exchange the weights of the corresponding node  $n_i$  in  $C_h$  and  $C_v$ . Since the configurations of  $C_h$  and  $C_v$  do not change, so does  $\Gamma_-$ . Figure 4(a) shows the resulting TCG-S after rotating the module  $g$  shown in Figure 2(b). Notice that the new  $\Gamma_-$  is the same as that in Figure 4(a).

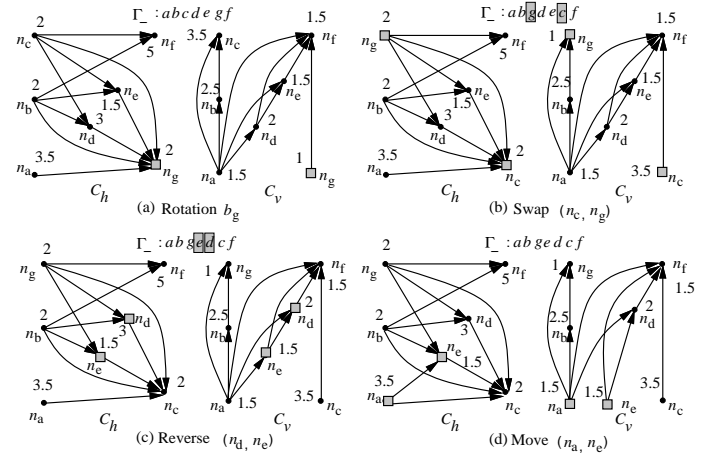


Figure 4: Four types of perturbation. (a) The resulting TCG-S after rotating the module  $g$  shown in Figure 2(b). (b) The resulting TCG-S after swapping the nodes  $n_c$  and  $n_g$  shown in (a). (c) The resulting TCG-S after reversing the reduction edge  $(n_d, n_e)$  shown in (b). (d) The resulting TCG-S after moving the reduction edge  $(n_a, n_e)$  from the  $C_v$  of (c) to  $C_h$ .

### 5.2.2 Swap

Swapping  $n_i$  and  $n_j$  does not change the topologies of  $C_h$  and  $C_v$ , except that nodes  $n_i$  and  $n_j$  in both  $C_h$  and  $C_v$  are exchanged. Therefore, we only need to exchange  $b_i$  and  $b_j$  in  $\Gamma_-$ . Figure 4(b) shows the resulting TCG-S after swapping the nodes  $n_c$  and  $n_g$  shown in Figure 4(a). Notice that the modules  $b_c$  and  $b_g$  in  $\Gamma_-$  in Figure 4(b) are exchanged.

### 5.2.3 Reverse

Reverse changes the geometric relation between  $b_i$  and  $b_j$  from  $b_i \vdash b_j$  ( $b_i \perp b_j$ ) to  $b_j \vdash b_i$  ( $b_j \perp b_i$ ). To reverse a reduction edge  $(n_i, n_j)$  in one transitive closure graph, we first delete the edge  $(n_i, n_j)$  from the graph, and then add the edge  $(n_j, n_i)$  to the graph. To keep  $C_h$  and  $C_v$  feasible, 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 have to keep the edge  $(n_k, n_l)$  in the new graph. If the edge does not exist in the graph, we add the edge to the graph and delete the corresponding edge  $(n_k, n_l)$  (or  $(n_l, n_k)$ ) in the other graph. To make  $\Gamma_-$  conform to the topological ordering of the new  $C_h$  and  $C_v$ , we delete  $b_i$  from  $\Gamma_-$  and insert  $b_i$  after  $b_j$ . For each module  $b_k$  between  $b_i$  and  $b_j$  in  $\Gamma_-$ , we shall check whether the edge  $(n_i, n_k)$  exists in the same graph. We do nothing if the edge  $(n_i, n_k)$  does not exist in the same graph; otherwise, we delete  $b_k$  from  $\Gamma_-$  and insert it after the most recently inserted module.

Figure 4(c) shows the resulting TCG-S after reversing the *reduction edge*  $(n_d, n_e)$  of the  $C_v$  in Figure 4(b). Since there exists no module between  $b_d$  and  $b_e$  in  $\Gamma_-$ , we only need to delete  $b_d$  from  $\Gamma_-$  and insert it after  $b_e$ , and the resulting  $\Gamma_-$  is shown in Figure 4(c).

#### 5.2.4 Move

Move changes the geometric relation between  $b_i$  and  $b_j$  from  $b_i \vdash b_j$  ( $b_i \perp b_j$ ) to  $b_i \perp b_j$  ( $b_i \vdash b_j$ ). To move a reduction edge  $(n_i, n_j)$  from a transitive closure graph  $G$  to the other  $G'$ , we delete the edge from  $G$  and then add it to  $G'$ . Similar to Reverse, 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  $G'$ , we must move the edge  $(n_k, n_l)$  to  $G'$  if the corresponding edge  $(n_k, n_l)$  (or  $(n_l, n_k)$ ) is in  $G$ . Since the operation changes only the edges in  $C_h$  or  $C_v$  but not the topological ordering among nodes,  $\Gamma_-$  remains unchanged.

Figure 4(d) shows the resulting TCG-S after moving the *reduction edge*  $(n_a, n_e)$  from  $C_v$  to  $C_h$  in Figure 4(c). Notice that the resulting  $\Gamma_-$  is the same as that in Figure 4(c).

For the above operations, Rotation and Move do not change the topological ordering of  $\Gamma_-$  while Swap and Reverse need respective  $O(1)$  and  $O(m)$  times to maintain the topological ordering of  $\Gamma_-$ , where  $m$  is the number of modules. We have the following theorem.

**Theorem 5** *TCG-S is closed under the Rotation, Swap, Reverse, and Move operations.*

In particular, it suffices to apply the four perturbations to explore the whole solution space. Extending the similar work by [17] for TCG, we have the following theorem.

**Theorem 6** *Given two arbitrary TCG-S's  $S_1$  and  $S_2$ , we can obtain  $S_2$  from  $S_1$  by applying a finite number of Rotation, Swap, Reverse, and Move operations, and vice versa.*

## 6 Placement with Constraints

In this section, we demonstrate the flexibility of TCG-S by extending it to handle placement with boundary and pre-placed modules.

### 6.1 TCG-S with Boundary Modules

The placement with boundary constraints is to place a set of prespecified modules along the designated boundaries of a chip, which can be formulated as follows:

**Definition 1** *Boundary Constraint: Given a boundary module  $b$ , it must be placed in one of the four sides: on the left, on the right, at the bottom or at the top in a chip in the final packing.*

TCG-S keeps the following properties that make placement with boundary constraints much easier than other representations.

**Theorem 7** *If a module  $b_i$  is placed along the left (right) boundary, the in-degree (out-degree) of the node  $n_i$  in  $C_h$  equals zero. If a module  $b_i$  is placed along the bottom (top) boundary, the in-degree (out-degree) of node  $n_i$  in  $C_v$  equals zero.*

For each perturbation, we can guarantee a feasible placement by checking whether the conditions of boundary modules are satisfied. We discuss the modifications for the four perturbation operations as follows.

#### 6.1.1 Rotation

Since Rotation does not change module location, the operation remains the same as before.

#### 6.1.2 Swap

We can swap two nodes  $n_a$  and  $n_b$  if

1.  $b_a$  and  $b_b$  are not boundary modules,
2.  $b_a$  and  $b_b$  are boundary modules of the same type, or
3.  $b_a$  is a boundary module and  $b_b$  is not, and  $n_b$  satisfies the boundary constraint of  $b_a$ .

#### 6.1.3 Reverse

If  $b_a$  is a left boundary module or  $b_b$  is a right boundary module, then the reduction edge  $(n_a, n_b)$  in  $C_h$  cannot be reversed. Similarly, we cannot reverse the reduction edge  $(n_a, n_b)$  in  $C_v$  if  $b_a$  is a bottom boundary module or  $b_b$  is a top boundary module.

#### 6.1.4 Move

If  $b_a$  is a top boundary module or  $b_b$  is a bottom boundary module, we cannot move the reduction edge  $(n_a, n_b)$  from  $C_h$  to  $C_v$ . Similarly, we cannot move the reduction edge  $(n_a, n_b)$  from  $C_v$  to  $C_h$  if  $b_a$  is a right boundary module or  $b_b$  is a left boundary module.

We have the following theorem.

**Theorem 8** *TCG-S is closed under the Rotation, Swap, Reverse, and Move operations with boundary constraints.*

## 6.2 TCG-S with Pre-placed Modules

The placement with pre-placed modules is to place a set of prespecified modules at the designated locations of a chip, which can be formulated as follows:

**Definition 2** *Pre-placed Constraint: Given a module  $b_i$  with a fixed coordinate  $(x_i, y_i)$  and an orientation,  $b_i$  must be placed at the designated location with the same orientation in the final packing.*

Whether a pre-placed module is packed at a correct location is not known until packing. Also, changing the coordinate of a module  $b_i$  may affect the packing for other modules after  $b_i$  in  $\Gamma_-$ . Therefore, we may need to modify a TCG-S to guarantee a feasible placement with the pre-placed constraint after each perturbation.

Given a TCG-S, modules are packed one by one based on the sequence of  $\Gamma_-$ . A module  $b_i$  interacts with another module  $b_j$  if (1)  $b_i$  overlaps  $b_j$ , (2)  $b_j \vdash b_i$  and their projections on the  $y$  axis overlap, or (3)  $b_j \perp b_i$  and their projections on the  $x$  axis overlap. If  $b_i$  interacts with a pre-placed module  $b_j$  and  $b_j$  was not placed,  $n_i$  and  $n_j$  are swapped in the TCG-S to make  $b_j$  placed at the designated location. If a pre-placed module  $b_i$  was placed and the resulting placement of  $b_i$  does not interact with itself at the designated location, we swap  $b_i$  with the node  $b_j$  right after  $b_i$  in  $\Gamma_-$ ; otherwise,  $b_i$  is placed at the designated location if there exists no module behind  $b_i$  in  $\Gamma_-$ .

## 7 Experimental Results

Based on a simulated annealing method [5], we implemented the TCG-S representation in the C++ programming language on a 433 MHz SUN Sparc Ultra-60 workstation with 1 GB memory. The source code is available at <http://cc.ee.ntu.edu.tw/~ywchang/research.html>. Based on the five commonly used MCNC benchmark circuits, we conducted four experiments: (1) area optimization, (2) wirelength optimization, (3) solution convergence speed and stability, and (4) placement with boundary constraints. In Table 1, Columns 2 and 3 list the respective numbers of modules and nets of the five circuits.

For Experiment (1), the area and runtime comparisons among SP, O-tree, B\*-tree, enhanced O-tree, CBL, and TCG are listed in Table 1. As shown in Table 1, TCG-S achieves the best area utilization for the benchmark circuits in very efficient running times. Figure 6 (left) shows the resulting placement for ami49 with area optimization.

For Experiment (2), 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. As shown in Table 2, TCG-S achieves better average results in wirelength than O-tree, enhanced O-tree, and TCG in smaller running times. (Note that we did not compare with B\*-tree and CBL here since they did not report the results on optimizing wirelength alone.)

In addition to the area and timing optimization, in Experiment(3), we also compared the solution convergence speed and stability among SP, TCG, and TCG-S to eliminate the possible unfairness due to the non-deterministic behavior of simulated annealing, which were neglected in most previous works. (Note that other tools are not available to us for the comparative study.) We randomly ran SP, TCG, and TCG-S on ami49 ten times based on the same initial placement whose area is  $102mm^2$ . The resulting areas are plotted as functions of the running times (sec). Figures 5(a), (b), and (c) show the resulting curves of SP, TCG, and TCG-S, respectively. To see the detailed convergence rates, we show in Figures 5 only the portions whose areas are smaller than  $47mm^2$ . As illustrated in Figure 5(c), TCG-S converges very fast to desired solutions, and the results are very stable ( $\leq 37.5mm^2$  for all runs). In contrast, the convergence speed of SP is much slower than TCG-S and TCG, and the resulting areas are often larger than  $39mm^2$ . Further, there

Circuit	# of modules	# of nets	SP		O-tree		B*-tree		Enhanced O-tree		CBL		TCG		TCG-S	
			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	Area $mm^2$	Time sec
apte	9	97	48.12	13	47.1	38	<b>46.92</b>	7	<b>46.92</b>	11	NA	NA	<b>46.92</b>	1	<b>46.92</b>	1
xerox	10	203	20.69	15	20.1	118	19.83	25	20.21	38	20.96	30	19.83	18	<b>19.796</b>	5
hp	11	83	9.93	5	9.21	57	<b>8.947</b>	55	9.16	19	(66.14)	(32)	<b>8.947</b>	20	<b>8.947</b>	7
ami33	33	123	1.22	676	1.25	1430	1.27	3417	1.24	118	1.20	36	1.20	306	<b>1.185</b>	84
ami49	49	408	38.84	1580	37.6	7428	36.80	4752	37.73	406	38.58	65	36.77	434	<b>36.40</b>	369

Table 1: Area and runtime comparisons among SP (on Sun Sparc Ultra60), O-tree (Sparc Ultra 60), B\*-tree (Sparc Ultra-I), enhanced O-tree (Sparc Ultra60), CBL (Sparc 20), TCG (Sparc Ultra60) and TCG-S (Sparc Ultra60) for area optimization. The best areas are in boldface.

is a large variance in its final solutions. Based on the experimental results, we rank the convergence speed from the fastest to the slowest and the solution stability from the most stable to the least stable as follows: TCG-S  $\succ$  TCG  $\succ$  SP. We note that the stability and convergence speed should be very important metrics to evaluate the quality of a floorplan representation because they reveal the corresponding solution structure for optimization. However, they were often ignored in previous works. (Most previous works focus on the comparison of solution space and packing complexity. Nevertheless, we find that the solution structure induced by a representation plays an even more important role in floorplan optimization.)



Figure 6: Resulting placements of ami49 (1) left: area optimization (area =  $36.398mm^2$ ); (2) right: placement with boundary modules being heavily shaded ( $(|T|, |B|, |L|, |R|) = (3, 3, 2, 3)$ , area =  $36.765mm^2$ ). Note that the lightly shaded regions denote dead spaces.

Circuit	O-tree		enhanced O-tree		TCG		TCG-S	
	Wire $mm$	Time sec	Wire $mm$	Time sec	Wire $mm$	Time sec	Wire $mm$	Time sec
apte	<b>317</b>	47	<b>317</b>	15	363	2	363	2
xerox	368	160	372	39	<b>366</b>	15	<b>366</b>	6
hp	153	90	150	19	<b>143</b>	10	<b>143</b>	4
ami33	52	2251	52	177	44	52	<b>43</b>	89
ami49	636	14112	629	688	604	767	<b>579</b>	570

Table 2: Wirelength and runtime comparisons among O-tree, enhanced O-tree, TCG, and TCG-S for wirelength optimization. All ran on Sun Sparc Ultra60.

Circuit	$ T ,  B ,  L ,  R $	Lai et al. [6]		TCG-S	
		Area $mm^2$	Time sec	Area $mm^2$	Time sec
apte	1, 1, 1, 1	<b>46.92</b>	15	<b>46.92</b>	3
xerox	1, 1, 1, 1	20.4	19	<b>19.977</b>	33
hp	1, 1, 1, 1	9.24	23	<b>9.158</b>	26
ami33	2, 2, 2, 2	1.21	290	<b>1.190</b>	238
ami49	3, 3, 2, 3	36.84	601	<b>36.765</b>	584

Table 3: Area and runtime comparisons between [6] (on Pentium-II 350) and TCG-S (on Sun Sparc Ultra60) for placement with boundary modules.

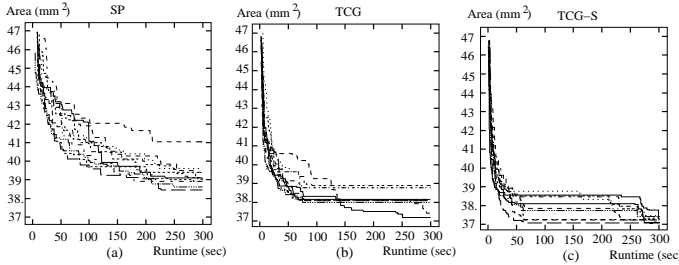


Figure 5: Comparison for stability and convergence among SP, TCG, and TCG-S for ami49.

For the experiments with boundary modules, we compared TCG-S with the SP-based method in [6] using the same data generated by [6]. The second column of Table 3 shows the numbers of the top, bottom, left, and right modules, denoted by  $|T|$ ,  $|B|$ ,  $|L|$ , and  $|R|$ , respectively. As shown in Table 3, TCG-S obtains smaller silicon areas than [6]. Figure 6 (right) shows the resulting placement for ami49 with boundary modules.

## 8 Concluding Remarks

We have presented the TCG-S representation for general floorplans by combining the advantages of two most promising P\*-admissible representations TCG and SP, and at the same time eliminating their disadvantages. We also have proposed a loglinear-time packing scheme and a linear-time perturbation scheme for TCG-S. We note that these schemes can also be applied to most existing representations such as SP and BSG. Based on our theoretical study and extensive experiments, we also have shown the superior capability, efficiency, stability, and flexibility of TCG-S for floorplan design.

## References

- [1] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B\*-trees: A new representation for non-slicing floorplans," *Proc. DAC*, pp. 458–463, 2000.
- [2] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill Book Company, 1990.
- [3] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-Tree representation of non-slicing floorplan and its applications," *Proc. DAC*, pp. 268–273, 1999.
- [4] X. Hong, G. Huang, T. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner Block List: An effective and efficient topological representation of non-slicing floorplan," *Proc. ICCAD*, pp. 8–12, 2000.
- [5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, May 13, 1983, pp. 671–680.
- [6] J.-B. Lai, M.-S. Lin, T.-C. Wang, and L.-C. Wang, "Module placement with boundary constraints using the sequence-pair," *ASP-DAC*, pp. 515–520, 2001.
- [7] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, 1976.
- [8] J.-M. Lin and Y.-W. Chang, "TCG: A transitive closure graph-based representation for non-slicing floorplans," *Proc. DAC*, pp. 764–769, 2001.
- [9] Y. Ma, S. Dong, X. Hong, Y. Cai, C.-K. Cheng, and J. Gu, "VLSI floorplanning with boundary constraints based on corner block list," *Proc. ASP-DAC*, pp. 509–514, 2001.
- [10] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing based module placement," *Proc. ICCAD*, pp. 472–479, 1995.
- [11] H. Murata, K. Fujiyoshi, and M. Kaneko, "VLSI/PCB placement with obstacles based on sequence pair," *Proc. ISPD*, pp. 26–31, 1997.
- [12] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-Structure and IC layout applications," *Proc. ICCAD*, pp. 484–491, 1996.
- [13] R. H. J. M. Otten, "Automatic floorplan design," *Proc. DAC*, pp. 261–267, 1982.
- [14] Y. Pang, C.-K. Cheng, and T. Yoshimura, "An enhanced perturbing algorithm for floorplan design using the O-tree representation," *Proc. ISPD*, pp. 168–173, 2000.
- [15] K. Sakanushi and Y. Kajitani, "The quarter-state sequence (Q-sequence) to represent the floorplan and applications to layout optimization," *Proc. Asia Pacific Conf. Circuits and Systems*, pp. 829–832, 2000.
- [16] X. Tang and D. F. Wong, "FAST-SP: A fast algorithm for block placement based on sequence pair," *Proc. ASP-DAC*, pp. 521–526, 2001.
- [17] T.-C. Wang, Personal communication, 2001.
- [18] D. F. Wong, and C. L. Liu, "A new algorithm for floorplan design," *Proc. DAC*, pp. 101–107, 1986.