Non Slicing Floorplan Representations in VLSI Floorplanning: A Summary

Leena Jain,Ph.D
Associate Professor & Head-MCA,
Global Institute of Management. &. Emerging Technologies, Amritsar (India)

Amarbir Singh
Ph. D., Research Scholar,
Punjab Technical University Jalandhar (India),
Assistant Professor, Guru Nanak dev University, Amritsar (India)

ABSTRACT
Floorplan representation is a fundamental issue in designing a VLSI floorplanning algorithm as the representation has a great impact on the feasibility and complexity of floorplan designs. This survey paper gives an up-to-date account on various non-slicing floorplan representations in VLSI floorplanning.

General Terms
VLSI floorplanning, non-slicing floorplan, slicing floorplan.

Keywords
VLSI floorplanning, non-slicing floorplan.

1. INTRODUCTION
A floorplan is a rectangular dissection which describes the relative placement of electronic modules on the chip. In the design of VLSI (Very Large-Scale Integrated) circuits floorplanning is an important phase. It determines the topology of layout and this is known to be NP-hard problem, and has received much attention in recent years [1]**.** The major objective of floorplanning is to allocate the modules of a circuit into a chip to optimize some design metric such as area, wire length and timing. During floorplanning the designers have additional flexibility in terms of size shape and orientation of the modules on chip. The shape of the chip and that of the modules is usually a rectangle. Accordingly VLSI floorplanning is the application of Rectangle packing problem. For solving these problems various heuristic, metaheuristic and optimal approaches are available in the literature [2, 3, 4, 5]. The representation has a great impact on the feasibility and complexity of floorplan designs. The redundancy of the representations and the complexity of the transformation between a representation and its corresponding floorplan can determine the execution time and the quality of the results.

In this paper authors have summarized the details about various floorplan representations that have been used for VLSI floorplanning. Results of area-minimization on MCNC benchmarks of different representations have been listed along with summary of search spaces and computational complexity for easy comparison.

2. VLSI FLOORPLAN DESIGN PROBLEM
2.1 Problem Description
VLSI floorplan is to arrange the modules on a chip and the set of modules can be represented as $S = \{M_1; M_2; \ldots; M_N\}$, where N is the number of the modules and M_i $(i = 1,2,\ldots,N)$ represents the ith module. There are two different types of modules:

1. Hard module: - The hard module's shape is fixed, and is denoted as (W, H), where W is the width and H is the height of the module.

2. Soft module: - Area is again fixed in case of Soft module, but the ratio of width/height is included in a given range. It can be denoted as (S, L, U), where S represents the area, L and U the lower and upper boundary of the width/height ratio.

In case that the modules are given, the objective of VLSI floorplanning is to arrange the modules on a chip under the constraints that any two modules are not overlapped, and the area, wire length and other performance indices are optimal [6].

2.2 Floorplan Structure
There are two layout structures in floorplan, namely, slicing and non-slicing floorplan. A slicing floorplan can be obtained by repetitively cutting the floorplan horizontally or vertically, whereas a non-slicing floorplan cannot [7]. The given dimension of each hard module must be kept. All modules are free of rotation; if a module is rotated, its width and height are exchanged. Figure 1 shows a slicing floorplan. A slicing tree is used to represent a slicing floorplan, it is a binary tree with modules at the leaves and cut types at the internal nodes. There are two cut types, V and H. The H cut divides the floorplan horizontally, and the left (right) child represents the bottom (top) sub-floorplan. Similarly, the V cut divides the floorplan vertically, and the left (right) child represents the left (right) sub-floorplan.

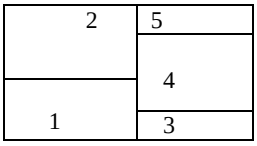


Fig. 1: Slicing floorplan

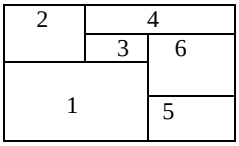


Fig. 2: Non-Slicing floorplan

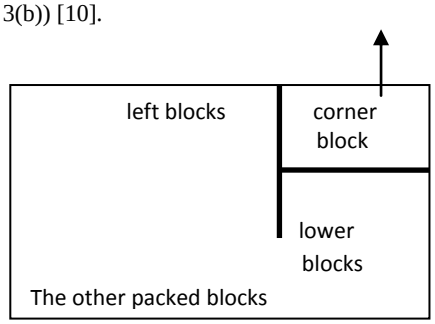
The non-slicing floorplan is more general than the slicing floorplan as shown in Figure 2. However, because of its non-slicing structure, it cannot be modeled using a slicing tree. Instead, we can use a horizontal constraint graph (HCG) and a vertical constraint graph (VCG) to model a non-slicing floorplans. The horizontal constraint graph defines the horizontal relations of modules separately, and the vertical constraint graph defines the vertical ones.

3. REPRESENTATION SCHEMES FOR NON-SLICING FLOORPLANS
3.1 Bounded Slicing Grid Structure (BSG)
The bounded slicing grid structure (BSG) can be obtained as follows [8]: make a row of non-overlapping horizontal line segments of two unit length and repeat them row by row, shifting by one unit length between the adjacent rows. A set of columns of vertical line segments with two unit length can be constructed in a similar way. Those line segments are called Bounded Slice

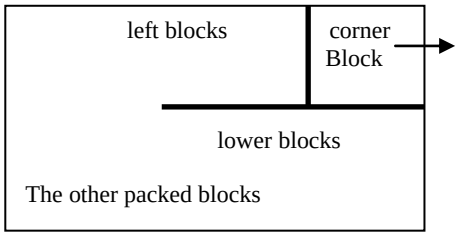
Lines, or BS-lines. None of the BS-lines are intersecting each other. The rectangle region enclosed by four BSlines is called a room. With BSG model, a floorplanning is represented by an assignment of blocks to rooms and this assignment is referred to as a BSG-seed. An empty room contains no block. Otherwise, the room is called occupied room. Given a BSG-seed, a floorplanning can be realized by stretching or shrinking, collectively called sizing, the BS-lines [9].

3.2 Corner Block List (CBL)

Floorplan divides the chip into rectangular rooms with several horizontal and vertical segments and each room is assigned to no more than one module. If the room of a module is along the boundary as required, the module can be moved within the range of the room to reach the boundary without changing the area. In a no empty space floorplan, T-junctions are formed when the internal segments intersect. A T-junction is composed of two segments: a non-crossing segment and a crossing segment. The non-crossing segment has one end touching point in the interval of the crossing segment. The Corner Block is the block packed in the upper right comer room of the floorplan. The joint of the left and bottom segments of the corner block is contained in a T-junction called corner T-junction and the comer block's orientation is defined by the orientation of the corner T-Junction. (Figure3). The T-junction has only two kinds of orientations: T rotated by 90 degrees counterclockwise and by 180 degrees counter clockwise. If T is rotated by 90 degrees counter clockwise, we define the corner block *to* be vertical oriented, and it is denoted by a "0" (Figure 3(a)) [10]. Otherwise, we define the corner block to be horizontal oriented, and it is denoted by a "1" (Figure 3(b)) [10].



(a) Corner block is vertical oriented



(b) Corner block is horizontal oriented

Fig.3 The orientation of the corner block

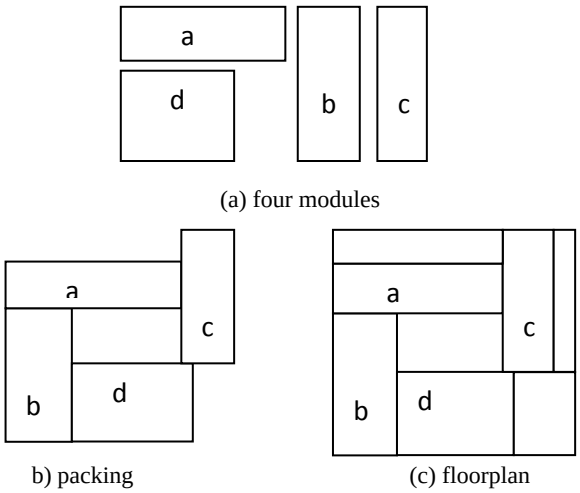
The corner block list is constructed from the record of a recursive corner block deletion. The corner block deletion is based on the constraint graph G = (V,E), in which the nodes in *V* are segments which slice the space and form the rooms of the floorplan with additional nodes used for edges of the placement modules, and the edges in *E* are the rooms of placement modules. The transformation from corner block list to floorplan can be achieved by scanning the CBL in linear time of *O(n)* [10].

3.3 Corner Sequence (CS)

The CS representation CS= < (S_1,D_1) (S_2,D_2),….,(S_m,D_m) > uses a packing sequence S of the m modules, as well as the corresponding bends D formed by the modules to describe a compacted placement. We refer to each two-tuple (S_i,D_i) $1 \leq i \leq m$ as a term of the CS. Derivation of a CS representation from a compacted placement which can be seen in [11].

3.4 Sequence Pair (SP)

An elegant coding scheme called sequence-pair (SP) has been proposed for RP [12]. A sequence pair is an ordered pair of Γ_+ and Γ_- where each of Γ_+ and Γ_- is a permutation of names of given n modules. For example, $(\Gamma_+;\Gamma_-)$ = (abcd;bdac) is a seq-pair of module set {a, b, c, d}. If module x is the i^{th} module in Γ_+, we denote $\Gamma_+(i) = x$ as well as $\Gamma_+^{-1}(x) = i$. Similar notation is used also for Γ_-. To help intuitive understanding, we use a notation such as $(\Gamma_+; \Gamma_-)$ = (..a..b..;..a..b..) by which we mean $\Gamma_+^{-1}(a) < \Gamma_+^{-1}(b)$ and $\Gamma_-^{-1}(a) < \Gamma_-^{-1}(b)$. A sequence-pair corresponds to a relative position of the module pair as follows [12]. For every module pair {a, b}, a is left of b (equivalently, b is right of a) if $(\Gamma_+; \Gamma_-)$ = (..a..b..;..a..b..). Similarly, a is below b (equivalently, b is above a) if $(\Gamma_+; \Gamma_-)$ = (..b..a..;..a..b..). For example, (abcd ; bdac) corresponds to a packing in Figure 4 [13].



(a) four modules

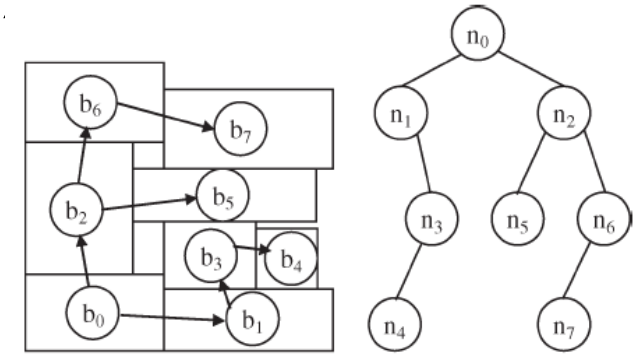
b) packing (c) floorplan

Fig. 4: (Γ_+, Γ_-) = (abcd; bdac) needs empty-room

A sequence pair is easily utilized as the representation of a candidate solution for stochastic algorithms such as genetic algorithm (GA) and simulated annealing (SA). According to some authors Sequence Pair can represent both slicing and non-slicing floorplans using two permutations (Γ_+, Γ_-) of the module indices.

3.5 B*Tree

A B*tree is an ordered binary tree for modeling non-slicing floorplans. Given an admissible placement (in which no blocks can move left or down), one can construct a unique B*tree in linear time to model the placement. Further, given a B*tree, one can also obtain a legal placement by packing the blocks in amortized linear time with a contour structure [14]. Figure 5(a) and 5(b) [15] show an admissible placement and its corresponding B*tree of an example floorplan. A 'B*tree' is an ordered binary tree with its root corresponding to the block at the bottom left corner. Similar to the Depth First Search (DFS) procedure, it is possible to construct a B*tree *T* for an admissible placement in a recursive fashion. Starting from the root, recursively, the left subtree is first constructed and then the right subtree [15].

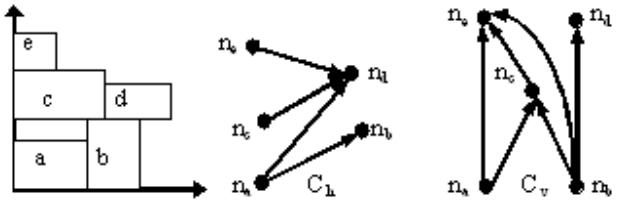


(a) Admissible floorplan (b) B*tree representation

Fig. 5: An example floorplan and its corresponding B*tree representation

3.6 Transitive Closure Graph (TCG)

The transitive closure of a directed acyclic graph G is defined as the graph $G' = (V, E'), \text{where} E' = \{(n_i, n_j) :$ 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, a horizontal transitive closure graph C_h and a vertical transitive closure graph C_v. Figure 6 shows the placement and corresponding TCG [16].



(a) (b)

Fig. 6: Placement in a chip TCG

TCG consists of a horizontal transitive closure graph to define the horizontal geometric relations between modules and a vertical one for vertical geometric relations. In Contrast to SP, the geometric relations between modules are transparent to TCG as well as its operations, it helps in the convergence to a desired solution. Apart from this, 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, just like SP, the constraint graphs are also needed for TCG to evaluate its packing cost, but unlike SP, there is 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? The answer of this question is TCG-S, a combination of SP and TCG representation [17].

3.7 Integer Coding

Literature [18] introduces integer coding representation in the format of $< V_1, V_2, \ldots, V_i, \ldots, V_n >$, where $1 \le V_i \le n$, $V_i = j$ and it denotes that the *i*-th module is placed at *j*-th position. Literature [19] adopted the integer coding representation proposed in [18], and modified the heuristic adjustment method.

3.8 O-Tree

An n-node O-tree is a tree with n+l nodes encoded by (E,x), where *E* is a 2n bit string that identifies the branching structure of the tree, and *x* is permutation of the n node labels (excluding the root). In the representation, string *E* gives a traversing sequence. '0' means descending an edge, and a '1' means ascending that edge. There is exactly a '0' and '1' in *E* for each edge. *x* is a permutation of the n node labels (excluding the root). A compaction algorithm is also given in [20]. Since each node has and only has one edge from its parent to it, in the following discussion we will say a 0-1 pair corresponds to a node, which means the 0-1 pair corresponds to the edge lead to the node.

4. BRIEF LITERATURE REVIEW OF NON-SLICING FLOORPLANS

Maggie and Wayne (1997) [9] proposed a new method of non-slicing floorplanning, which is based on the new representation for non-slicing floorplans proposed by [8], called bounded slicing grid (BSG) structure. They developed a new greedy algorithm based on the BSG structure, which runs in linear time, in order to select the alternative shape for each soft block so as to minimize the overall area for general floorplan, including non-slicing structures. Based on BSG structure, they extended SIA-based local search and GA-based global crossover to L-shaped, T-shaped blocks and obtain high density packing of rectilinear blocks.

Yuchun Ma et. al. (2001) [10] implemented the boundary constraint algorithm for general floorplan by extending the Corner Block List (CBL) - a new efficient topology representation for nonslicing floorplan. Their contribution is to find the necessary and sufficient characterization of the modules along the boundary represented by Corner Block List. So that the boundary constraints can be checked by scanning the intermediate solutions in linear time during the simulated annealing process and fix the corner block list in case the constraints are violated. The experimental results show that performance is remarkable.

Liang Huang, Yici Cai, Xianlong Hong (2004) [21] presented a parallel algorithm for non-slicing floorplan using Corner Block List (CBL) topological representation. In this paper, a parallel interconnection cost calculation algorithm with load balancing strategy is initiated in order to speed up the especially time consuming wire length calculation in floorplanning. Multiple Markov chains strategy is also embedded in their algorithm. The experiment results obtained from the tests on MCNC benchmarks indicate considerable speedup and preserved floorplanning quality.

Jai-Ming Lin et. al. (2003) [11] presented a P-admissible representation, called corner sequence (CS), for nonslicing floorplans. It consists of two tuples that denote the packing sequence of modules and the corners to which the modules are placed. It is very effective and simple for implementation. It also supports incremental update during packing. In particular, it induces a generic worst case linear-time packing scheme that can also be applied to other representations.

Hiroshi Murata, Kunihiro Fujiyoshi (1997) [12] proposes such a solution space where each packing is represented by a pair of module name sequences, called a sequence-pair. By searching this space using simulated annealing, large numbers of modules have been packed efficiently as demonstrated by them. For applications to VLSI layout, they used the biggest MCNC benchmark *ami49* with a conventional wiring area estimation method, and obtain a highly promising placement.

Koichi haua et. al. (1999) [22] proposed the adaptive GA for the rectangular packing problem RP and designed new crossover and mutation operators based on sequence-pair representation of individuals. They proposed an adaptive strategy to select appropriate genetic operators during the GA execution. Experimental results showed the effectiveness of their proposed GA in comparison to Simulated Anealing (SA).

Koji kiyota, Kunihiro fuiiyoshi (2000) [13] proposed a novel solution space of floorplans for simulated annealing (SA) which consists of the all general floorplans with exact n rooms, where n is the number of given modules, using sequence-pair. By using ingenious data structure, a feasible adjacent floorplan can be obtained in O(n2) time and the reachability from any floorplan to any other in the proposed solution space will be proved.

Ning Xu' et. al. (2003) [23] applied Tabu search algorithm to solve module placement problem. Firstly, all modules are merged into some clusters according to the ratio-connectivity of circuit modules, the placement *of* the large modules (are included by some modules) **is** then represented by sequence-pairs. The searching of optimal solution of placement is performed by the tabu search algorithm.

Chikaaki kodama et. al. (2004) [24] proposed a novel method to encode a given rectangle packing into a sequence-pair in O(n log n) time, as encoding methods are not found except the original one called "gridding". The gridding requires almost $O(n^3)$ time for a packing of n rectangular modules and it is hard to implement. Apart from it they also proposed a linear time method to obtain a sequence-pair from a given rectangular dissection represented by a Q-sequence. The proposed methods can be used for the compaction keeping topology, for example, in the post-process of the Force Directed Relaxation, a method used in module placement.

Pradeep Fernando and Srinivas Katkoori (2008) [25] proposed a multi-objective genetic algorithm for floorplanning that simultaneously minimizes area and total wirelength. The proposed genetic floorplanner is the first to use non-domination concepts to rank solutions. In this paper two novel crossover operators are presented that build floorplans using good sub-floorplans. Efficiency of the proposed approach is illustrated by the 18% wirelength savings and 4.6% area savings obtained for the GSRC benchmarks and 26% wirelength savings for the MCNC benchmarks for a marginal 1.3% increase in area when compared to previous floorplanners that perform simultaneous area and wirelength minimization.

Dipanjan Sengupta et. al. (2011) [26] presented a new floorplanning algorithm based on the sequence pair representation that can floorplan blocks in the form of islands. When the possible supply voltage choices are given for each block, the floorplanner simultaneously attempts to reduce power and area of the chip. Their floorplanner integrates the tasks of assigning blocks to different supply voltages and the placing of the blocks in the chip. In comparison to previous work, the proposed floorplanner on average reduces the area overhead of the chip by 13.5% with 34% runtime improvement.

Zhen Chen et. al. (2012) [27] proposed a co evolutionary multi objective particle swarm optimization (CMOPSO) algorithm to solve a VLSI (Very Large Scale Integrated) Floorplanning problem which is a multi objective combinatorial optimization and has been proved to be a NP-hard problem. The algorithm imports the concept of co evolutionary algorithm and elitist strategy into basic PSO algorithm, It takes both the layout area and total interconnection wire length into consideration simultaneously.

Samsuddin et. al. (2008) [28] proposes an optimization approach for macro-cell placement which minimizes the chip area size. The binary tree method for non-slicing tree construction process is utilized for the placement and area optimization of macro-cell layout in very large scaled integrated (VLSI) design. Different types of genetic algorithms: simple genetic algorithm (SGA), steady-state algorithm (SSGA) and adaptive genetic algorithm (AGA) are employed in order to examine their performances in converging to their global minimums. Apart from it, the robustness of genetic algorithm also has been investigated in order to validate the performance stability in achieving the optimal solution for every runtime.

Yun-Chih Chang et. al. (2000) [14] presented an efficient, flexible, and effective data structure, B*-trees, for non-slicing floorplans. Inheriting from the nice properties of ordered binary trees, B*-trees are very easy for implementation and can perform the respective primitive tree operations search, insertion, and deletion in only *O(1), O(1)*and *O(n)* times while existing representations for non-slicing floorplans need at least *O(n)* time for each of these operations, where *n* is the number of modules. They further show the flexibility of B*-trees by exploring how to handle rotated, pre-placed, soft, and rectilinear modules. The Experimental results on MCNC benchmarks show that the B*-tree representation runs about 4.5times faster, consumes about 60% less memory. They also develop a B*-tree based simulated annealing scheme for floorplan design; the scheme achieves near optimum area utilization even for rectilinear modules.

Tung-Chieh Chen, and Yao-Wen Chang (2006) [15] studied two types of modern floorplanning problems: 1) fixed-outline floorplanning and 2) bus-driven floorplanning (BDF). This floorplanner uses B*-tree floorplan representation based on fast three-stage simulated annealing (SA) scheme called Fast-SA. The authors proposed an adaptive Fast-SA for fixed-outline floorplanning that can dynamically change the weights in the cost function to optimize the wire length under the outline constraint. For the BDF, the authors explore the feasibility conditions of the B*-tree with the bus constraints, and developed a BDF algorithm based on the conditions and Fast-SA. The experimental results show that this floorplanner obtains much smaller dead space for the floorplanning with hard/soft macro blocks, compared with the most recent work.

Fubing Mao et. al. (2009) [29] proposed hybrid algorithm which based on B*-tree representation to improve the area utilization. The simulated annealing was embedded into tabu search for floorplanning. Experimental results show that their approach can improve the area utilization in shorter time. It shows that the method they proposed is effective and efficient.

Jiarui Chen, Jianli Chen (2010) [30] presented a hybrid evolution algorithm for VLSI floorplanning based on B*-tree. In this method, BFS sequence of B*-tree is adopted as the individual encoding, and the crossover is constructed. Based on the concept of evolutionary algorithm and simulated annealing, a hybrid evolutionary algorithm (ESA) is proposed. Furthermore, A fast SA is embedded into the evolution iteration for more accurate search and faster convergence. Experimental results show that our algorithm is efficient and effective. To further study the method presented in this paper, we will apply it to a multilevel floorplanning framework for larger scale circuit.

Jianli Chen, Wenxing Zhu (2010) [31] described that HGA uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region. Experimental results on MCNC benchmarks show that the HGA is effective and promising in building block layout application.

S. Anand et. al. (2010) [32] developed Simulated Spheroidizing Annealing Algorithm (SSAA) based on a Simulated Annealing Algorithm (SAA) heuristic and improvements in the proposed heuristic algorithm are also suggested to improve its performance. Exploration capability of the proposed algorithm is due to the mechanism of reducing the uphill moves made during the initial stage of the algorithm, extended search at each temperature and the improved neighborhood search procedure. The proposed SSAA algorithm is also found more efficient for problems of larger sizes.

Jianli Chen et. al. (2011) [1] presented a hybrid simulated annealing algorithm (HSA) for non slicing VLSI floorplanning. The HSA uses a new greedy method to construct

an initial B*-tree, a new operation on the B*-tree to explore the search space, and a novel bias search strategy to balance global exploration and local exploitation. Experimental results show that the HSA can quickly produce optimal or nearly optimal solutions for all the tested problems.

Yiding Han et. al. (2011) [33] proposed a novel floorplanning algorithm for GPUs. Floorplanning is an inherently sequential algorithm, far from the typical programs suitable for Single Instruction Multiple Thread (SIMT) style concurrency in a GPU. They propose a fundamentally different approach of exploring the floorplan solution space, where they evaluate concurrent moves on a given floorplan. Compared to the sequential algorithm, their techniques achieve 4-30X speedup for a range of MCNC benchmarks.

Jai-Ming Lin and Yao-Wen Chang (2001) [34] proposed a 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. 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.

Jai-Ming Lin and Yao-Wen Chang (2002) [17] proposed the equivalence of the two most promising P*-admissible representations, SP and TCG, and integrated TCG with a packing sequence (part of SP) into a new representation, called TCG-S. It combines the advantages of SP and TCG and at the same time eliminates their disadvantages. By using TCG-S placement with position constraints becomes much easier, and incremental update for cost evaluation can be realized. All these nice properties make TCG-S a superior representation which exhibits an elegant solution structure to facilitate the search for a desired floorplan/placement.

Jai-Ming Lin and Yao-Wen Chang (2005) [16] 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.

Guolong Chen et. al. (2008) [35] proposed a novel floorplanning algorithm based on Discrete PSO (DPSO) algorithm, in which integer coding based on module number was adopted. The principles of mutation and crossover operator in the Genetic Algorithm (GA) are also incorporated into the proposed PSO algorithm to achieve better diversity and break away from local optima. The proposed algorithm can avoid the solution from falling into local minimum and have good convergence performance.

Guolong Chen et. al. (2009) [36] proposed a novel intelligent decision algorithm based on the particle swarm optimization (PSO) technique to obtain a feasible floorplanning in VLSI circuit physical placement. The PSO was applied with integer coding based on module number and a new recommended value of acceleration coefficients for optimal placement solution. Inspired by the physics of genetic algorithm (GA), the principles of mutation and crossover operator in GA are incorporated into the proposed PSO algorithm to make this algorithm to break away from local optima and achieve a better diversity. Experiments employing MCNC and GSRC benchmarks show that the proposed algorithm is effective.

Pei-Ning Guo et. al. ((2001) [37] presented an ordered tree (O tree) structure to represent non slicing floorplans. The O tree representation uses only n (2 + log (n)) bits for a floorplan of n rectangular blocks. Given an O tree, it takes only linear time to construct the placement and its constraint graph. They have developed a deterministic floorplanning algorithm utilizing the structure of O tree. Empirical results on MCNC (www.mcnc.org) benchmarks show promising performance with average 16% improvement in wire length and 1% less dead space over previous central processing unit (CPU) intensive cluster refinement method.

Hiroshi ninomiya et. al. (2006) [38] described the two-staged Tabu search for the non-slicing floorplan problem using the ordered tree representation called O-tree. The floorplan problem is a part of VLSI layout design problem. Furthermore, they combine ideas from the simulated annealing into the two-staged Tabu search and proposed a novel hybrid algorithm for floorplan represented by O-tree. Finally, they demonstrated the validity of two-staged search and hybrid method for MCNC benchmark tests through the computer simulations.

Maolin Tang and Xin Yao (2007) [39] proposed a memetic algorithm (MA) for a nonslicing and hard-module VLSI floorplanning problem. This MA is a hybrid genetic algorithm that uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region. The exploration and exploitation are balanced by a novel bias search strategy. The MA has been implemented and tested on popular benchmark problems. In addition, it only takes $O(n)$ to transform between an O-tree representation and its corresponding floorplan.

Maolin Tang, Raymond Y.K. Lau (2007) [40] presented a parallel genetic algorithm (GA) for floorplan area optimization. This parallel GA is based an island model with an asynchronous migration mechanism, and is implemented using Web services and multithreading technologies. Furthermore, parallel GA is compared with a sequential GA and experimental results show that the parallel GA can produce better results than the sequential GA when they use the same amount of computing resources.

Table 1: Summary of search spaces and computational complexity

Floorplan representations	Search space	Computational complexity
B* Tree	$O[(n!2^{2n-1})/n^{1.5}]$	O(n)
O-Tree	$O[(n!2^{2n-1})/n^{1.5}]$	O(n)
TCG	$O((n!)^2)$	$O(n^2)$
TCG-S	$O((n!)^2)$	$O(n \cdot \log n)$
SP	$O((n!)^2)$	$O(n^2)$
Fast-SP	$O((n!)^2)$	$O(n \cdot \log(\log n))$
CBL	$O[(n!2^{3n-3})/n^{1.5}]$	O(n)
CS	$O((n!)^2)$	O(n)

5. CONCLUSION

In this paper authors have presented a detailed study on representations for non-slicing floorplans and these representations are much harder for implementation and operation and incur more restrictions in comparison with representations for slicing floorplans. It is clear from Table 2, that B*-Tree representation has more advantages over other types and results of area-minimization on MCNC benchmarks for this representation are also very competitive as displayed in Table 3. The summary of search spaces and computational complexity is given in Table 1, to help in making selection of representation scheme for a floorplanning problem.

Table 2: Representation Comparison

Floorplan Representation	Advantages	Dis-Advantages
TCG	➢ No need to construct additional constraint graphs for the cost evaluation during packing ➢ Implies faster runtime ➢ Supports incremental update during operations ➢ Memory usage is smaller	➢ Cannot handle the slicing structure
TCG-S	➢ Implies faster convergence to a desired solution. ➢ The placement with position constraints becomes much easier. ➢ Can support incremental update for cost evaluation.	➢ Cannot handle the slicing structure
CS	➢ Effective and simple for implementation ➢ It supports incremental update during packing	➢ Cannot handle the slicing structure
Sequence Pair	➢ Can handle both slicing and non-slicing structure ➢ Very flexible in representation	➢ Time-consuming ➢ The solution space is large ➢ Sequence encoding cost is high ➢ Difficult to transform between a sp and a placement ➢ Cannot handle soft modules directly
CBL	➢ Can handle non-slicing structure ➢ Very flexible in representation	➢ Many infeasible solutions may be generated before a feasible solution is found ➢ It is not p-admissible
BSG	➢ Can handle non-slicing structure ➢ Placement becomes easy ➢ Very flexible in representation	➢ Time-consuming ➢ The solution space is large ➢ Incurs redundancies
O-Tree	➢ Can handle non-slicing structure ➢ The solution space is smaller ➢ Transformation between representation and placement takes only linear time ➢ Encoded by fewer bits than sequence pair and BSG	➢ Less flexible than sequence pair in representation ➢ Tree structure is irregular and harder for implementation ➢ Required to encode and operate on module sequence ➢ Inserting positions are limited and might deviate from the optimal during solution perturbation
B*-tree	➢ Efficient and flexible to deal with hard, pre-placed, soft, and rectilinear modules, etc ➢ Smaller encoding cost ➢ Takes only linear time ➢ Can evaluate area cost incrementally ➢ The solution space is smaller ➢ Compact placement	➢ Lesser flexible than sequence pair in representation ➢ It may not be feasible to find a placement corresponding to its original representation

Table 3: Published results of area-minimization on MCNC benchmarks of different representations

Floorplan Representations	Published Results					
	Publishing Details	apte	xerox	hp	ami33	ami49
Optimal	[41]	46.9	19.8	8.95	time-out	time-out
CBL	[21]	47.614	20.641	NA	1.2581	38.507
TCG	[16]	46.92	19.83	8.947	1.20	36.77
TCG-S	[17]	46.9	19.796	8.947	1.185	36.40
CS	[11]	46.92	19.83	8.947	1.18	36.28

B*-Tree	[14]	46.92	19.83	8.95	1.27	36.80
	[31]	47.01	20.14	9.13	1.19	37.49
	[32]	48.47	20.42	9.48	1.23	38.10
	[1]	48.12	21.86	9.43	1.25	40.01
O-tree	[37]	48.3	20.4	9.71	1.26	41.3
Integer Coding	[35]	46.92	20.44	NA	1.29	39.27
Fast-SP	[42]	46.92	19.80	8.94	1.20	36.50
GPE	[43]	45.9	20.14	9.12	1.18	36.45
Sequence Pair (SP)	[44]	47.07	19.83	9.14	1.19	37.27

6. REFERENCES

[1] Jianli Chen, Wenxing Zhu, and M. M. Ali, "A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning", IEEE transactions on systems, man and cybernetics—part c: applications and reviews, VOL. 41, NO. 4, 2011, pp. 544-553.

[2] Jain L. and Singh G., "A Review: Meta- heuristic Approaches for solving Rectangle Packing Problem", International Journal of Computer Engineering and Technology, Volume 4, Issue 2, 2013, pp. 410- 424.

[3] Singh K. and Jain L., "Experimenting Genetic approach to extend rectangular packing heuristic solutions", International Journal of Computer Applications. Special Issue on "Evolutionary Computation for Optimization Techniques", 2010, pp. 1-7.

[4] Singh K. and Jain L., "An improved Heuristic for 2D Rectangular packing problem", Proceeding of IEEE International Advance Computing Conference, March-2009, Thaper University, Patiala, IEEE Delhi Section, 2009, pp. 1185-1190.

[5] Singh K. and Jain L., "Optimal Solution for 2-D Rectangle Packing Problem", International Journal of Applied Engineering Research. VOL. 4, NO. 11, 2009, pp. 2203–2222.

[6] Guolong Chen, Wenzhong Guo, Yuzhong Chen, "A PSO-based intelligent decision algorithm for VLSI floorplanning", Springer, Soft Computing, 2010, pp. 1329–1337.

[7] D. F. Wong, C. L. Liu, "A new algorithm for floorplan design." Proceeding of the ACWIEEE Design Antomation Conference, 1986, pp. 101-107.

[8] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, "Module placement on BSG-structure and IC layout applications", Proceedings of 1996 IEEE/ACM, International Conf. on Computer Aided Design, 1996, pp. 484-491.

[9] Maggie Kang, Wayne W. M. Dai, "General floorplanning with L-shaped, T-shaped and Soft Blocks Based on Bounded Slicing Grid Structure" In proceedings of Design Automation Conference 1997 Asia and South Pacific, IEEE, 1997, pp. 265 – 270.

[10] Yuchun Ma', Sheqin Dong', Xianlong Hong', yici Cai', Chung-Kuan Cheng2, Jun Gu3," VLSI Floorplanning with Boundary Constraints Based on Corner Block List", IEEE, 2001, pp 509-514.

[11] J.-M. Lin, Y.-W. Chang, S.-P. Lin,"Corner sequence: a P-admissible floorplan representation with a worst case linear-time packing scheme", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2003, pp. 679–686.

[12] H. Murata, K. Fujiyoshi and Y. Kajitani: "VLSI module placement based on rectangle-packing by the sequence-pair," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, V01. 15, N0. 12, 1996, pp. 1518-1524.

[13] Koji kiyota, Kunihiro fuiiyoshi, "Simulated Annealing Search Through General Structure Floorplans Using Sequence-Pair", Symposium On Circuits And Systems, Geneva, Switzerland , IEEE, 2000, pp. 77-80.

[14] Yun-Chih Chang, Yao-Wen Chang, Guang-Ming Wu, and Shu-Wei Wu,"B*-Trees: A New Representation for Non-Slicing Floorplans" (c) ACM, 2000.

[15] Tung-Chieh Chen, and Yao-Wen Chang," Modern Floorplanning Based on B*-Tree and Fast Simulated Annealing" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, VOL. 25, NO. 4, APRIL 2006, pp. 637-650.

[16] Jai-Ming Lin and Yao-Wen Chang, " TCG: A Transitive Closure Graph-Based Representation for General Floorplans", IEEE transactions on very large scale integration (VLSI) systems, VOL. 13, NO. 2, 2005, pp. 288-292.

[17] J.-M. Lin and Y.-W. Chang, "TCG-S: Orthogonal Coupling of P*-admissible Representations for General Floorplans," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2004, pp.968 - 980.

[18] B. H. Gwee, and M. H. Lim, "A GA with heuristic-based decoder for IC Floorplanning", The VLSI journal, 1999, pp. 157-172.

[19] X. G. Wang, L. S. Yao, and J. R. Gan, "VLSI Floorplanning Method Based on Genetic Algorithms", Chinese Journal of Semiconductors, 2002, pp. 330-335.

[20] P. -N Guo, C. -K. Cheng, T., Yoshimura, "An O-tree representation of non-slicing floorplan and its applications", Proc. 36th ACWIEEE Design AutoNation Cont., 1999, pp.268-273.

[21] Liang Huang, Yici Cai, Xianlong Hong, "A Parallel VLSI Floorplanning Algorithm Using Corner Block List Topological Representation", IEEE, 2004, pp. 1208-1212.

[22] Koichi Haua, Shin'ichi Wakabayashi, Tetsushi Koide, "Solving the Rectangular Packing Problem by an Adaptive GA Based on Sequence-Pair", ASP-DAC, 1999, pp. 181-184.

[23] Ning Xu', Xian-Long Hong'.She-Qin Dong', he-Bang Yu'," TSCSP: Tabu Search Algorithm for VLSI Module Placement Based on the Clustering Sequence-Pair", IEEE, 2003.

[24] Chikaaki KODAMA, Kunihiro FUJIYOSHI, and Teppei Koga, "A Novel Encoding Method Into Sequence-Pair ", ISCAS, IEEE, 2004, pp. V329 – V332.

[25] Pradeep Fernando and Srinivas Katkoori, "An Elitist Non-Dominated Sorting based Genetic Algorithm for Simultaneous Area and Wirelength Minimization in VLSI Floorplanning" In 21st International Conference on VLSI Design, 4-8 January 2008, Hyderabad, India, IEEE Computer Society, 2008, pp. 337-342.

[26] Dipanjan Sengupta, Andreas Veneris, Steve Wilton, Andre Ivanov, Res Saleh, "Sequence Pair Based Voltage Island Floorplanning", Proceedings of the 2011 International Green Computing Conference and Workshops, IEEE computer society Washington, 2011, pp. 1-6.

[27] Zhen Chen, Jinzhu Chen, Wenzhong Guo, Guolong Chen, "A Coevolutionary Multi-Objective PSO algorithm for VLSI Floorplanning" 8th International Conference on Natural Computation (ICNC), IEEE, 2012, pp. 712-728.

[28] Samsuddin, AbAl-Hadi Ab Rahman, Andaljayalakshmi G, "A Genetic Algorithm Approach to VLSI Macro Cell Non-Slicing Floorplans Using Binary Tree", Proceedings of the International Conference on Computer and Communication Engineering, IEEE, 2008.

[29] Fubing Mao, Ning Xu, Yuchun Ma, "Hybrid Algorithm for Floorplanning Using B*-tree Representation", Third International Symposium on Intelligent Information Technology Application, IEEE, 2009, pp. 228-231.

[30] Chen, J., & Chen, J., "A hybrid evolution algorithm for VLSI floorplanning", Int. Conf. Comput. Intell. Software Eng. (CiSE), IEEE, 2010, pp. 1-4.

[31] Jianli Chen, Wenxing Zhu, "A Hybrid Genetic Algorithm for VLSI Floorplanning", International Conference on Intelligent Computing and Intelligent Systems (ICIS), IEEE, 2010, pp. 128-132.

[32] S. Anand · S. Saravanasankar · P. Subbaraj, "Customized simulated annealing based decision algorithms for combinatorial optimization in VLSI floorplanning problem", Springer , 2011.

[33] Yiding Han, Koushik Chakraborty, Sanghamitra Roy, Vilasita Kuntamukkala, "A GPU Algorithm for IC Floorplanning: Specification, Analysis and Optimization", 24th Annual Conference on VLSI Design, IEEE, 2011.

[34] Jai-Ming Lin and Yao-Wen Chang, "TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans", Proc. DAC, IEEE, 2001, pp. 764-769.

[35] Guolong Chen, Wenzhong Guo, Hongju Cheng, Xiang Fen and Xiaotong Fang, "VLSI Floorplanning Based on Particle Swarm Optimization" Proceedings of 3rd International Conference on Intelligent System and Knowledge Engineering, IEEE ,2008, pp. 1020-1025.

[36] Guolong Chen, Wenzhong Guo, Yuzhong Chen, "A PSO-based intelligent decision algorithm for VLSI Floorplanning", Soft Computing, VOL. 14, NO. 12, Springer, 2009, pp. 1329-1337.

[37] Pei-Ning Guo, Toshihiko Takahashi, Chung-Kuan Cheng, "Floorplanning Using a Tree Representation", IEEE transactions on computer-aided design of integrated circuits and systems, VOL. 20, NO. 2, 2001, pp. 281-289.

[38] Hiroshi Ninomiya, Kimihiko Numayama and Hideki Asai, "Two-staged Tabu Search for Floorplan Problem Using O-Tree Representation", IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16-21, 2006.

[39] Maolin Tang and Xin Yao, "A Memetic Algorithm for VLSI Floorplanning", IEEE transactions on systems, man, and cybernetics—part b: cybernetics, VOL. 37, NO. 1, 2007, pp. 62-69.

[40] Maolin Tang, Raymond Y.K. Lau," A Parallel Genetic Algorithm for Floorplan Area Optimization", Seventh International Conference on Intelligent Systems Design and Applications,IEEE, 2007, pp.801-806

[41] H. H. Chan, I. L. Markov, "Practical Slicing and Non-slicing Block-Packing without Simulated Annealing", ACM/IEEE Great Lakes Symp. on VLSI, 2004, pp. 282-287.

[42] Tang, X., Wong, D.F., "FAST-SP: a fast algorithm for block placement based on sequence pair", In Proceedings of the ASP-DAC, 2000, pp. 521–526.

[43] Chang-Tzu Lin, De-Sheng Chen, Yi-Wen Wang., "GPE: A New Representation for VLSI Floorplan Problem", Proc. ICCD, IEEE, 2002, pp. 531 -533.

[44] Adya, S.N., Markov, I.L., "Fixed-outline floorplanning: enabling hierarchical design", Transactions on Very Large Scale Integration (VLSI) Systems, IEEE Transactions , VOl. 11, Issue: 6 , 2003, pp. 1120-1135.