G-NSVF: A Greedy Algorithm for Non-Slicing VLSI Floorplanning

B. N. B. Ray, Sony Snigdha Sahoo(✉), and Susil Kumar Mohanty

Department of Computer Science and Applications, Utkal University, Vani Vihar, Bhubaneswar 751004, Odisha, India

Abstract. Floorplanning is the first step in the physical design of VLSI. At this stage, the circuit is partitioned into blocks for packing them optimally within the chip. The metrics minimized in floorplan are overall interconnect wirelength, area of the chip, deadspace, etc. B* tree is a popular representation of floorplan as it captures both slicing and non-slicing floorplans. In this work, we have proposed a greedy algorithm for the initial floorplan, which can be used by simulated annealing placer that takes B* tree as the initial floorplan. The proposed algorithm in conjunction with B* tree when integrated into simulated annealing placer and experimented on MCNC benchmarks reduces the overall wirelength on an average by 11% and 69% as compared to random and prior greedy initial floorplans.

Keywords: B* Tree · Floorplanning · Greedy algorithm · Physical design

1 Introduction

Advancement in technology is leading to complex circuit design. VLSI floorplanning is an effective approach toward managing circuit design complexity. Given a set of circuit components or modules and a netlist specifying the interconnections between the modules, floorplanning ensures that none of the modules overlap each other and various metrics like area, total interconnect wirelength among modules are minimized.

Effective representation of such floorplans is important for ensuring the conversion between a representation and the corresponding floorplan. There are two basic floorplanning structures, namely slicing and non slicing floorplans. A brief comparison among the two has been given in [3]. Several representations have been proposed for representing non-slicing floorplans such as sequence pair [4], bounded slice line grid [5], O-tree [6], B*tree [1] etc. B* tree representation, as proposed by Chang et al. [1] has been proven to be an efficient representation of floorplan as it has many advantages over other representations. It has not only inherited all the best features of an ordered binary tree but also, is quite flexible in handling floorplanning problem with different kind of modules like hard, soft, preplaced, and rectilinear. It is very fast and can be easily implemented. The

P. K. Behera and P. C. Sethi (Eds.): CSI 2020, CCIS 1372, pp. 48–58, 2021.
https://doi.org/10.1007/978-981-16-2723-1_6

authors in [9] have introduced following greedy evaluation function V_i for each module v_i for $(1 \leq i \leq n)$, n being the number of modules.

$$V_i = \lambda_1 \times \frac{h_i \times w_i}{H \times W} + \lambda_2 \times \frac{h_i + w_i}{H + W} \tag{1}$$

Where h_i and w_i are height and width of module v_i and H and W are the height and width of the floorplanning region. The constants $0 \leq \lambda_1, \lambda_2 \leq 1$ are such that $\lambda_1 + \lambda_2 = 1$. On the basis of the decreasing values of V_i for each $1 \leq i \leq n$ as given by Eq. 1, they generated an initial floorplan and represented the floorplan as B* tree. Their experimental results on MCNC Benchmarks using HSA (Hybrid Simulated Annealing) algorithm show improvement in overall interconnection wirelength and deadspace.

In this paper, we have introduced a greedy heuristic for initial floorplan modifying Eq. 1, so that it can be used by simulated annealing placer that takes the initial floorplan as a B* tree. The experimental results for our initial greedy floorplan on MCNC benchmarks [13] using B* tree based simulated annealing placer [1] reduce the final wirelength of floorplan on average by 11% and 69% respectively compared to floorplans generated randomly and also by Eq. 1, respectively.

In short, our contributions in this paper are listed below:

- We have proposed a greedy heuristic for the initial floorplan as an input to the simulated annealing placer.
- We have also integrated it into annealing based placer that takes initial floorplan as B* tree.
- Experimental results on MCNC benchmarks for the proposed initial greedy floorplan on an average show 11% and 69% reduction in placement wirelength compared to the random floorplan and also floorplan due to Eq. 1 respectively. In addition to this, the new heuristics also reduces the dead-space of final placement almost 3× than that of existing schemes.

The remaining of the paper is organized as follows. Section 2 gives a brief review of B* tree representation and simulated annealing algorithm for solving the floorplan problem. Section 3 begins with the problem statement, followed by our greedy approach. The implementation details are given in Sect. 4. The paper concludes with its future scope in Sect. 5.

2 B* Tree Representation

B* tree can be constructed along the same lines as that of the DFS procedure [1]. As proposed in [1] "first, the left subtree is recursively constructed starting from the root and then the right subtree. Let R_i be the set of modules located on the right-hand side and adjacent to a module b_i denoting the node n_i. The left child of the node n_i corresponds to the lowest module in R_i that has not been visited. The right child of n_i represents the module located above and adjacent to b_i, with its x-coordinate equal to that of b_i and its y-coordinate less than that

of the top boundary of the module on the left-hand side and adjacent to b_i, if any [1]. The B*-tree maintains the geometric relationship between two modules as follows. If node n_j is the left child of node n_i, module b_j must be located on the right-hand side and adjacent to module b_i in the admissible placement; i.e., $x_j = x_i + w_i$. Besides, if node n_j is the right child of n_i, module b_j must be located above and adjacent to module b_i, with the x-coordinate of b_j equal to that of b_i; i.e., $x_j = x_i$. Also, since the root of T represents the bottom-left module, the x- and y-coordinates of the module associated with the root(x_root,y_root) = (0,0)". Primitive operations like insertion, the search can be performed in constant time, and deletion can be done in linear time on a B*tree. Apart from these, three new operations have been proposed for perturbing a B* tree in [8], namely rotation of a block, movement of a block from one place to another, and swapping of two blocks. These operations can be carried out in O(h) time, where h denotes the height of the B*tree.

2.1 Simulated Annealing Based B* Tree Representation

VLSI floorplanning being an NP-hard problem, various heuristic methods have been suggested for dealing with it. They can be categorized as constructive methods and iterative methods. The constructive methods use heuristic information for constructing the floorplan, whereas, the iterative ones make use of metaheuristic strategies, such as genetic algorithm, simulated annealing, and tabu search for obtaining good solutions. Floorplan algorithms used in [10] and [11] are also based on the simulated annealing. Simulated annealing (SA) is one of the widely used techniques used for approximating global optimization in a large search space [9]. However classical SA process has a significant draw-back of excessive running time. Thus, several annealing schemes for controlling the temperature changes during the annealing process have been proposed to reduce the running time of SA while searching for desired solutions more efficiently. One of the most successful among them has probably been the annealing schedule used by TimberWolf [7]. It provides not only the relative positions of the modules, but also their aspect ratios and pin positions. In [12], a fast simulated annealing algorithm has been proposed, which is significantly different from the existing simulated annealing schemes. It tries to speed up the annealing process. Hybrid simulated annealing approach in [9] constructs the initial B* tree using a new greedy approach, and a new operation on B* tree has also been proposed for exploring the search space. It leads to a much quicker optimal solution.

2.2 Original Algorithm

According to [9] "given an initial floorplan encoded in B* trees, at each temperature, the local search method finds a locally optimal solution through systematically examining those B* trees obtained by rotating a module by 90°, moving a module to another place or swapping two modules". If one of the operations leads

to a floorplan with a smaller cost, the floorplan is accepted. The pseudo-code of the local search algorithm is presented in Algorithm 1 [9].

Algorithm 1: Algorithm for Local Search

I is an initial configuration
T is the Temperature
$mt = 0$, $uphill = 0$, $reject = 0$
$N^{max} = k \times m$
An empty list $LIST$
while $((uphill^{max})$ and $(mt < 2 \times N^{max}))$ **do**
    Randomly use one of the three operations to generate a new configuration J
    Calculate $cost(J)$, and update the cost to the list $LIST$
    $mt = mt + 1$
    $\Delta C = cost(J) - cost(I)$
    if $(\Delta C \leq 0)$ **then**
        $I = J$
        $localbest = J$
    else
        Randomly generate a number t $(0 \leq t \leq 1)$
        if $(t < e^{\frac{\Delta C}{T}})$ **then**
            $I = J$
            $uphill = uphill + 1$
        else
            $reject = reject + 1$
        end if
    end if
end while
$reject\_rate = \frac{reject}{mt}$
return configuration I, $reject\_rate$, $localbest$

After the local search, simulated annealing is used as a global search method to explore the search space [9].

Algorithm 2: Simulated Annealing Algorithm

Initial B^* tree B (By Algorithm 1)
$best = B$, $count = 0$
$conv\_rate = 1$, min
An initial temperature T
while $((reject\_rate < conv\_rate)$ and $(actual\_T > term\_T))$ **do**
    $count = count + 1$
    Optimize I using the Algorithm 1
    if $(cost(localbest) < cost(best))$ **then**
        $best = localbest$
    end if
    Update T and $actual\_T$
    if $(count > min)$ **then**
        $conv\_rate = 0.95$
    end if
end while

3 Problem Formulation

Let $M = \{v_1, v_2, \cdots, v_n\}$ be a set of modules with height h_i and width w_i and N be the Net-list for specifying interconnection among the modules. A floorplan F assigns M onto a plane in such a manner that none of the modules overlap each other. Area A of a floorplan F is a measure of the area of the smallest rectangle that surrounds all the modules. Wirelength W is the interconnection cost, and it is a measure of the total wirelength for maintaining the connection specified

by N. The cost of a floorplan as given in [9] is given by the following equation.

$$f = w \times \frac{area(F)}{norm\_area} \; + \; (1-w) \times \frac{wirelength(F)}{norm\_wirelength}$$

where w and (1-w) are weights such that $w \in [0, 1]$ assigned for minimizing area and interconnection. This cost function has been adopted in [9]. A floorplan is said to be compact if no modules can be moved left or down without moving other modules. A compact floorplan can be represented using B* tree. The solution space is composed of all the B* trees that can be constructed when modules are given. As there exists a unique B* tree for every compacted floorplan, redundant solutions in the search space are eliminated.

3.1 Our Proposed Greedy Algorithm

This section proposes a modified heuristic cost function for the initial floorplan representation of the B* tree.

For generating the initial floorplan, the heuristic cost function defined by Eq. 1 has been modified as below. For each module v_i, the evaluation function say f_{v_i} is defined as

$$f_{v_i} = \lambda_1 \times \frac{h_i \times w_i}{H \times W} + \lambda_2 \times \frac{h_i + w_i}{H + W} + \frac{no\_of\_pins_i}{total\_no\_of\_pins} \tag{2}$$

where $0 \leq \lambda_1, \lambda_2 \leq 1$ are non negative weights such that $\lambda_1 + \lambda_2$=1. and have been set to 0.5 each as has been adopted in [9]. And $no\_of\_pins_i$ is the number pins of each module v_i, while $total\_no\_of\_pins$ is the summation of pins of each of the modules.

This cost function is calculated for every module. Modules are then packed in decreasing order of their heuristic values.

The heuristic function given by Eq. 1 normalizes area as VLSI floorplanning has always focused on minimization of area and wirelength. But the proposed approach has considered area, wirelength and pin density of each module together. A module with larger f_{v_i} is packed earlier i.e. modules are packed in decreasing order of f_{v_i} values.

3.2 Greedy Algorithm

The following Greedy_initial_placer() algorithm places modules according to decreasing value of f_{v_i}.

Algorithm 3: Greedy_initial_placer()

1: **for** i = 1 **to** n **do**
2: Module v_i, calculate $f_{v_i} = \lambda_1 \times \frac{h_i \times w_i}{H \times W} + \lambda_2 \times \frac{h_i + w_i}{H + W} + \frac{no\_of\_pins_i}{total\_no\_of\_pins}$
3: **end for**
4: Sort f_{v_i} in decreasing order $\forall_{i=1}^{n} v_i$
5: Let $M' = \{v'_1, v'_2, \cdots, v'_n\}$ be the permutation of $M = \{v_1, v_2, \cdots, v_n\}$ resulting from step 4
6: Call *Place_Module(M')* to place modules in layout area

The procedure *Place_Module(M′)* places modules in the prescribed order of M' within the chip and returns the initial B* tree B for the simulated annealing placer. While placing modules in the chip, the procedure *Place_Module(M′)* uses the following operations, as discussed in [9]. If B is the set of modules placed in the chip, then the feasible region of the chip is the free space in the chip where if a module is placed, then it neither overlaps with any modules of B nor it oversteps the floorplan area.

1. Determine points in the feasible region and sort them in their increasing order of y-coordinates and then choose the point in that order.
2. Then choose the module from M' and put at the bottom left corner of that feasible point and include the module in set B. Then delete placed module from the set M'.
3. If at that feasible point determined in step 1, no module can be placed, then exclude that feasible point and select the next feasible point in the sorted order and then goto step 2.
4. Continue to step 2 and step 3 until M' is empty.

At the end of step 4, the initial greedy B* for the annealing placer is returned. The pseudo-code *Place_Module(M′)* is described by Algorithm 4.

Algorithm 4: Place_Module(M')

```
1: M' = {v'_1, v'_2, ··· , v'_n}
2: Initialize B* tree B = Nil
3: while (M' ≠ φ) do
4:     Execute steps 1), 2), 3) for selecting a feasible point p_i and a module v'_i;
5:     Put v'_i at point p_i, then remove v'_i from M' and add it to B
6: end while
7: return B* tree B
```

3.3 Time Complexity Analysis

In Algorithm 3, the step 4 involves $O(nlogn)$ comparisons for sorting f_{v_i} $(1 \leq i \leq n)$, the heuristic values f_{v_i}. The procedure *Place_Module(M′)* (Algorithm 4) at step 6 uses $O(nlogn)$ comparisons for sorting y-coordinates of n feasible points and $O(n)$ time to place n modules in B* tree B. Thus the running time of Greedy_initial_placer() Algorithm $= O(nlogn) + O(nlogn) + O(n) = O(nlogn)$.

After the modules are packed according to the above Algorithm 3 (see Fig. 1), it is then converted into B* tree, as shown in Fig. 2. Then Algorithm 1 and Algorithm 2 are used to find the solution of the floorplan.

Before we discuss the effect of various initial floorplans on the quality of the final floorplan, we use the following additional notations.

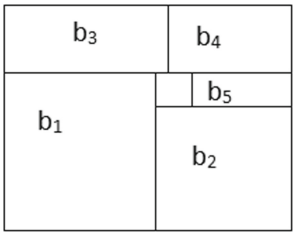

Fig. 1. Floorplan

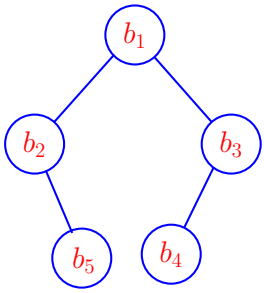

Fig. 2. B* Tree of the Floorplan

- B*tree WL(B* tree Deadspace): final floorplan wirelength (Dead space) due to random initial floorplan given as an input to B* tree-based simulated annealing placer [1]
- HSA WL(HSA Deadspace): final floorplan wirelength (Deadspace) due to initial floorplan of Eq. 1
- Proposed WL(Proposed Deadspace): final floorplan wirelength (Deadspace) due to initial floorplan of Eq. 2.

4 Performance of the Proposed Greedy Approach

In order to study the efficacy of the proposed greedy strategy against random initial floorplan and floorplan due to Eq. 1, we implemented the proposed greedy algorithm and the greedy algorithm of Eq. 1 in C++. We integrated them into the publicly available B* tree-based simulated annealing placer [1]. For simulation purposes, we have used the MCNC floorplan benchmark suite [13]. All experiments were conducted on a Linux machine with a Core i5 processor, with a speed of 2.3 GHz and 4GB RAM. We have conducted three sets of experiments on MCNC benchmarks. In our first set of experiments, we presented the random floorplan of benchmark circuits as input to the B* tree-based simulated annealing placer as used in [1]. For the second set of experiments, we repeated the first experiment providing benchmark circuits as input to the annealing placer based on a greedy strategy of Eq. 1. And for the third set of experiments, we provided benchmark circuits as input to the placer based on our greedy strategy given by Eq. 2. The experimental results for wirelength and deadspace for various circuits for these three sets of experiments are presented in Table 1 and Table 2, respectively.

In Table 1, the names of circuits are put in the first row, starting from column 2 through column 6. The second row of the table presents final wirelengths (B* tree WL) of floorplans in millimeter for various circuits from column 2 through column 6 based on random floorplan. The second row presents final floorplans wirelengths (HSA WL) for the greedy algorithm on the framework of Eq. 1. And the third row shows the final placement wirelength on the basis of our

Table 1. B* Tree, HSA and proposed algo wirelength result

Circuit	ami33	ami49	apte	hp	xerox	Norm_WL
B*tree WL	178.139	11460.6	1058.36	317.894	1162.141	1.116
HSA WL	170.317	8818.91	1112.05	1136.8	1098.19	1.69
Proposed WL	170.317	8818.91	1018.5	267.51	1151.44	1

Table 2. B* Tree, HSA, proposed algo deadspace result

Circuit	ami33	ami49	apte	hp	xerox	Norm_DS
B*Tree Deadspace	5.56	46.89	2.03	23.87	6.21	2.93
HSA Deadspace	6.06	29.95	2.03	24.69	4.19	2.92
Proposed Deadspace	6.06	29.95	3.44	2.23	7.04	1

Table 3. Computation statistics for random initial floorplan

Circuit	ami33	ami49	apte	hp	xerox
No of modules	33	49	9	11	10
Height	1.715	39.074	14.918	3.766	7.966
Width	0.714	1.708	3.186	3.08	2.59
Area	1.22451	66.738	47.53	11.599	20.632
Wirelength	178.139	11460.6	1058.36	317.894	1162.141
Total Area	1.156	35.445	46.562	8.83	19.350
Deadspace	5.56	46.89	2.03	23.87	6.21
CPU Time	2.39	30.74	1.94	1.34	0.27
Last CPU Time	2.37	27.89	0.18	1.12	0.23

Table 4. Computation statistics for initial floorplan based on Greedy HSA

Circuit	ami33	ami49	apte	hp	xerox
No of modules	33	49	9	11	10
Height	0.679	29.624	3.186	21.476	2.59
Width	1.813	1.708	14.918	0.546	7.798
Area	1.23103	50.5978	47.5287	11.7259	20.197
Wirelength	170.317	8818.91	1112.05	1136.8	1098.19
Total Area	1.156	35.445	46.562	8.83	19.350
Deadspace	6.06	29.95	1.89	24.69	4.19
CPU Time	2.77	29.07	0.14	4.22	0.29
Last CPU Time	2.74	24.05	0.21	0.37	0.29

Table 5. Computation statistics for proposed Greedy initial floorplan

Circuit	ami33	ami49	apte	hp	xerox
No of modules	33	49	9	11	10
Height	0.679	29.624	13.182	2.016	2.604
Width	1.813	1.708	3.658	4.48	7.994
Area	1.23103	50.5978	48.2198	9.032	20.816
Wirelength	170.317	8818.91	1018.5	267.51	1151.44
Total Area	1.156	35.445	46.562	8.83	19.350
Deadspace	6.06	29.95	3.44	2.23	7.04
Last CPU Time	2.74	24.05	0.09	0.14	0.20

greedy algorithm given by Eq. 2. From Table 1, it is clear that for all circuits, the proposed greedy initial floorplan consistently reduces the final placement wirelength. Column 7 of Table 1 displays the normalized wirelengths (Nor_WL) of B* tree WL, HSA WL, and Proposed WL. From column 7, one can see that the proposed greedy approach, on an average reduces the final placement wirelength by 11% and 69%, respectively, as compared to the random floorplan and greedy floorplan due to Eq. 1. This is also evident from the bar graph shown in Fig. 3, where the x-axis represents benchmarks and y-axis wirelength. In Fig. 3, blue bar, red bar and gray bar correspond to B*tree WL, HSA WL, and Proposed WL, respectively.

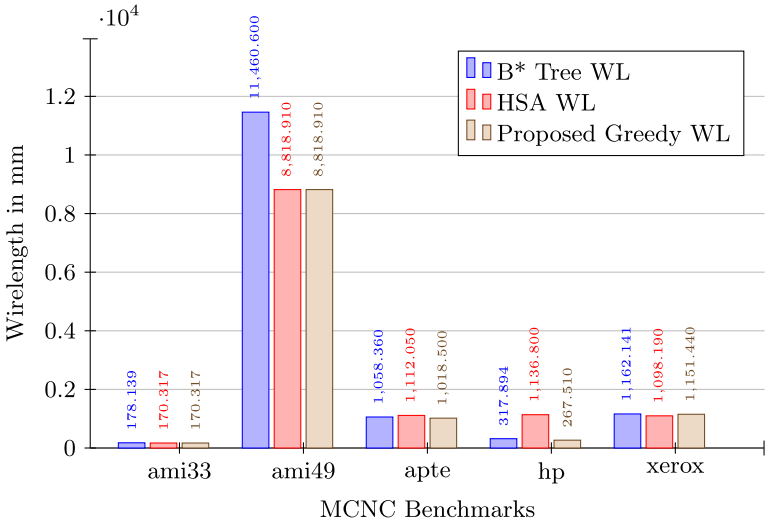

Fig. 3. Plot representing B* Tree wirelength, Greedy HSA wirelength and that obtained in the Proposed Greedy Approach (Color figure online)

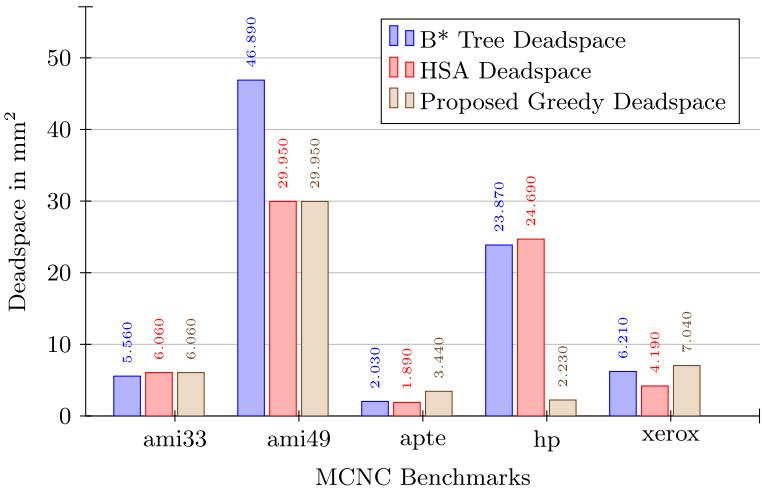

Fig. 4. Plot representing B* Tree deadspace, Greedy HSA deadspace and that obtained in the Proposed Greedy Approach (Color figure online)

Table 2 presents the results for deadspace of final floorplans for various circuits by initial floorplans generated randomly by Eq. 1 and by Eq. 2 (proposed greedy algorithm). Here also column 7 presents normalized deadspace of final floorplans by three approaches. From Table 2, it is evident that the deadspace of final floorplans due to the random floorplan and HSA based floorplan are almost 3× more than that of the proposed greedy floorplan. This fact is also illustrated by bar graph shown in Fig. 4, where the x-axis represents benchmarks and y-axis deadspace. In Fig. 4, blue bar, red bar and gray bar correspond to B*tree Deadspace, HSA Deadspace, and Proposed Deadspace, respectively. Table 3, Table 4, and Table 5 show computation statistics (such as Height, Width, Area, Wirlength, Deadspace, and CPU time of final floorplan) for random, greedy HSA and proposed greedy initial floorplans. The consistent decrease in wirelength and deadspace of the final floorplan for the proposed greedy initial floorplan may be attributed to the fact that the proposed strategy is more amenable to generate initial floorplan that is close to the optimal, which is the biggest advantage of the algorithm. It is interesting to explore the nature of the cost function surface on the basis of various greedy approaches to the initial floorplan.

5 Conclusion and Future Scope

In this work, we have proposed a greedy heuristic for the initial floorplan, which can be used as an input by any simulated annealing placer. The proposed greedy floorplan, when given as an input to simulated annealing placer based on B* tree representation, shows a good reduction in wirelength as well as dead space compared to the random floorplan and prior art on the greedy floorplan. Experimental results on MCNC benchmarks for the proposed greedy approach on an

average reduces final floorplan wirelength by 11% and 69% as compared to random and prior greedy initial floorplans, respectively. The deadspace produced by random and prior greedy floorplans are almost 3× more than that of the proposed greedy approach. In the future, we plan to study the effect of the proposed greedy floorplan on the thermal aware floorplan.

References

1. Chang, Y.C., Chang, Y.W., Wu, G.M., Wu, S.W.: B* Tree: a new representation for non slicing floorplans. In: ACM/IEEE Design Automation Conference, pp. 458–463 (2000)
2. Chen, J., Zhu, W., Ali, M.M.: A Hybrid Simulated Annealing Algorithm for Non-slicing VLSI Floorplanning (2011)
3. Chan, H.H., Adya, S.N., Markov, I.L.: Are floorplan representations important in digital design?. In: Proceedings of the International Symposium on Physical Design, pp. 168–173 (2000)
4. Murata, H., Fujiyoshi, K., Nakatake, S., Kajitani, Y.: Rectangle-packing based module placement. In: Proceedings of the ICCAD, pp. 472–479 (1995)
5. Nakatake, S., Fujiyoshi, K., Murata, H., Kajitani, Y.: Module placement on BSG-structure and IC layout applications. In: Proceedings of the ICCAD, pp. 484–491 (1996)
6. Guo, P.-N., Cheng, C.-K., Yoshimura, T.: An O-tree representation of non-slicing floorplan and its applications. In: Proceedings of the DAC, pp. 268–273 (1999)
7. Sechen, C., Sangiovani-Vincentelli, A.L.: The timber wolf placement and routing package. IEEE J. Solid-State Circuits **SC-20**(2), 510–522 (1985)
8. Chen, J., Liu, Y., Zhu, Z., Zhu, W.: An adaptive hybrid memetic algorithm for thermal-aware non-slicing VLSI floorplanning. Integr. VLSI J. **58** (2017). https://doi.org/10.1016/j.vlsi.2017.03.006
9. Chen, J.Z., Ali, W., Montaz, A.: A hybrid simulated annealing algorithm for non slicing VLSI floorplanning. IEEE Trans. Syst. Man Cybern Part C: Appl. Rev. **41**, 544–553 (2011). https://doi.org/10.1109/TSMCC.2010.2066560
10. Sur-Kolay, S.: Studies on nonslicible floorplans in VLSI layout design. Ph.D. dissertation, Department of Computer Science and Engineering, Jadavpur University, Calcutta (1991)
11. Wong, D.F., Liu, C.L.: Floorplan design for rectangular and L-shaped modules. In: Proceedings of the International Conference on Computer Aided Design, pp. 520–523, November 1987
12. Chen, T.C., Chang, Y.-W.: Modern floorplanning based on fast simulated annealing, pp. 104–112 (2005). https://doi.org/10.1145/1055137.1055161
13. The MCNC Benchmark Problems for VLSI Floorplanning. http://www.mcnc.org