Improvement of Simulated Annealing Search –Based on Tree Representations–

Takaaki Banno Kunihiro Fujiyoshi
Department of Electrical Electronic Engineering
Tokyo University of Agriculture and Technology
2-24-16 Nakacho Koganei-shi Tokyo, 184-8588 Japan
banno@fjlab.ei.tuat.ac.jp , kfujiyos@fjlab.ei.tuat.ac.jp

Abstract— Placement problem for LSI layout is often referred to "Rectangle packing problem." For this problem, several representations of rectangle packing were proposed and packings are searched by Simulated Annealing based on a representation. To search suboptimal solutions by Simulated Annealing, it is necessary to define appropriate MOVE operations. In this paper, we make a conjecture that a degradation of search efficiency arises if adjacent solution, which was made by one MOVE, needs several times of MOVEs to restore. Therefore, we restrict MOVE operations so that they can restore any solution in constant times and confirmed the effectiveness by experiments.

I. Introduction

Placement problem for LSI layout is often referred to "Rectangle packing problem," which is to place the given rectangles in a smallest rectangle. For this problem, several representations of rectangle packing were proposed and packings are searched by Simulated Annealing based on a representation. Simulated Annealing (SA)[1] is a method simulating a physical phenomenon called annealing, by cooling down the temperature according to schedule, SA searches in solution space for suboptimal solution stochastically. To search suboptimal solutions based on the representation by SA, it is necessary to define appropriate MOVE operations.

For packing representation, O-Tree[3] was proposed by P.-N.Guo et al. An O-Tree consists of an ordered tree and can represent any bottom-left packing. Note that a bottom-left packing is the packing compacted so that any blocks can neither move down nor left. The size of the solution space is the smallest among known representations, and an O-Tree can be decoded into the corresponding packing in $O(n)$ time, where n is the number of rectangles. They encoded an O-Tree as a bit string, and defined MOVE operations on the bit string and got dense packing by SA. But by this MOVE operation, adjacent solution which is made by one MOVE may differ greatly from the former solution and it might cause the degradation of search efficiency.

To perturb O-Tree, Ukibe et al. used a MOVE operation based on tree representation[4]. By this MOVE operation, adjacent solution which is made by one MOVE will be more similar to former solution, thus improvement of the search efficiency was expected, but experiment results were not so good. We make a conjecture that a search efficiency by a MOVE operation is degraded if any one MOVE can't restore an adjacent solution made of solution f by one MOVE to solution f.

On the other hand, there is a problem called "3D packing problem," which is to pack all the given rectangular boxes in a rectangular box of minimum volume without any overlap. In the future, this problem will be applicable to very-large-scale integrations, which consist of many layers of active device. For this problem, several 3D packing representations have been proposed and packings are searched by SA based on a representation.

Kawai et al. proposed a representation called Double Tree and Sequence (DTS)[6]. DTS consists of unordered trees and sequence. Search efficiency of DTS was best among representations which can represent any rectangular box packing. But by the MOVE operations used in experimental comparisons in [6], any one MOVE can't restore a certain adjacent solution made of solution f by one MOVE to solution f.

Therefore, in this paper, we restrict MOVE operations so that a certain series of j MOVEs can restore any adjacent solution made of solution f by one MOVE to solution f, where j is an integer not bigger than a given fixed integer, and confirm the effectiveness by experiments.

II. O-Tree[3]

O-Tree[3] is a representation using "ordered tree," and it can represent any bottom-left packing. "Ordered tree" is a rooted tree that has an order in children nodes that have the same parent node. Except for the root node, each node corresponds to a rectangle and is called by the rectangle's name. The root node corresponds to a pseudo rectangle with zero width on y-axis. Note that we define a node in left side in figures as a parent and nodes in right side as children, and the DFS(Depth First Search)

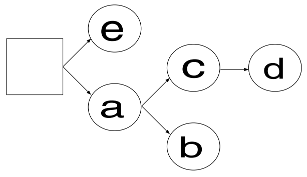

Fig. 1. Example of O-Tree

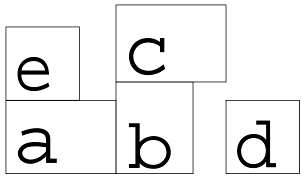

Fig. 2. packing corresponding to O-Tree shown in Fig.1

traversal is done with giving priority to lower node. The following constraints should be satisfied for any rectangle p,

x-constraints: If node p is a child of node q,

$$x(p) = x(q) + w(q),$$

where $x(p)$ denotes the x-coordinate of the left edge of rectangle p and $w(q)$ denotes the width of rectangle q.

y-constraints: Let $\Psi(p)$ be a set of nodes which are traversed before node p in DFS traversal and whose intervals overlap with p's interval. Note that p's interval is the range from the x-coordinate of the left edge of rectangle p to that of right edge.

$$\begin{aligned} y(p) &= \max_{q \in \Psi(p)} \{y(q) + h(q)\} \quad \text{if } \Psi(p) \neq \{\}, \\ y(p) &= 0 \qquad\qquad\qquad\qquad \text{otherwise,} \end{aligned}$$

where $y(p)$ denotes the y-coordinate of the bottom edge of rectangle p and $h(q)$ denotes the height of rectangle q

An example of O-Tree and its corresponding packing is shown in Fig.1 and Fig.2.

A. *Known MOVE operations*

To make an adjacent solution for SA, following two sets of MOVE operations were used.

1: "Bit representation"[3]

2: "Node Move with leaving descendant (NM)"[4]

A.1. Bit Representation[3]

To represent an O-Tree of n nodes by "bit representation," we use a $2n$ bit string T to identify the branching structure of the tree and a permutation π as the names of n nodes in DFS order on the O-Tree. The bit string T is a realization of the tree structure. We write a "0" when we pass an edge away from the root and a "1" when we pass an edge toward the root in the tree in the DFS order. For example, an O-Tree shown in Fig.3 is represented by $T = (00101101)$ and $\pi = (abcd)$. Packing shown in Fig.4 is obtained from the O-Tree shown in Fig.3. After we transform an O-Tree to a "bit representation," we change tree by following two ways.

"Change of bit string" : Exchange a "0" and a "1" in bit string T. Note that when we pick up a bit string of any length from the head, the number of 0s must be not less than that of 1s.

"Change of permutation" : Exchange two elements in a permutation π.

If a bit representation is used for MOVE operations, a certain MOVE can restore any adjacent solution made of solution f by one MOVE to solution f. For example, when we change bit string $T = (00101101)$ of the O-Tree shown in Fig.3 to $T' = (00001111)$ by one "Change of bit string" operation, "Change of bit string" can restore T' to T by one MOVE.

"finite-restorable MOVE operations" : In this paper, if a certain series of j MOVEs can restore any adjacent solution made of solution f by one MOVE to solution f, where j is an integer not bigger than a given fixed integer, we call such MOVE operations as "**finite-restorable**."

When we change bit string $T = (00101101)$ of the O-Tree shown in Fig.3 to $T' = (00001111)$ by "Change of bit string"operation, packing and O-Tree obtained by T' and π will change massively as shown in Fig.5 and Fig.6. Thus, "Change of bit string " may cause big change in O-Tree and we are afraid that it cause degradation of search efficiency.

A.2. Node Move with Leaving Descendants[4]

"NM,"[4] which was used by Ukibe at el. is a MOVE operation based on tree, and therefore it doesn't make big change in one MOVE, but search efficiency was not so good.

We represented an O-Tree by a binary tree in well known way[7] like B*-Tree[8]. An example is shown in Fig.7. In this figure, we regard nodes in upper side as younger brothers, and in lower side as elder brothers.

We use following two ways for MOVE operations.

"NM": On tree, choose a node to move, and say the node s and the parent of the node r.

Put each child of s to the parent of s, so that the node becomes a child of r and is in the position of s with maintaining the brother relation.

Put s to a node t so that s becomes the youngest child of t (t can't be r, nor s itself).

An example is shown in Fig.8. In Fig.8, node a is chosen to move, and a's children e and c are put to a's parent b. a is moved to the root's youngest child.

"Exchange of nodes": Exchange two elements in O-Tree

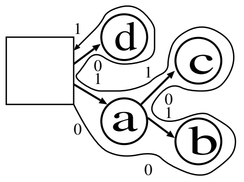

Fig. 3. O-Tree whose code is $T = (00101101)$, $\pi = (abcd)$

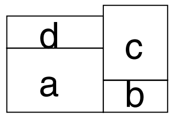

Fig. 4. packing corresponding to O-Tree shown in Fig.3

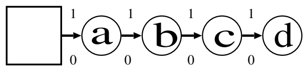

Fig. 5. O-Tree whose code is $T' = (00001111)$, $\pi = (abcd)$

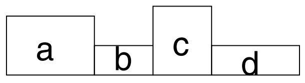

Fig. 6. packing corresponding to O-Tree shown in Fig.5

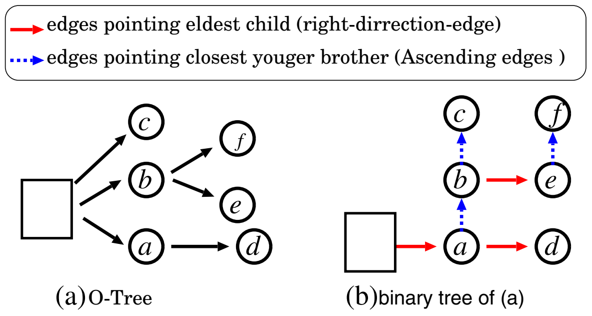
edges pointing eldest child (right-dirrection-edge)
edges pointing closest youger brother (Ascending edges)
c
f
b
e
a
d
(a)O-Tree
(b)binary tree of (a)

Fig. 7. (a) O-Tree (b) binary tree representation of the O-Tree. Ascending edge starting from each node points closest younger brother and right-direction-edge points eldest child.

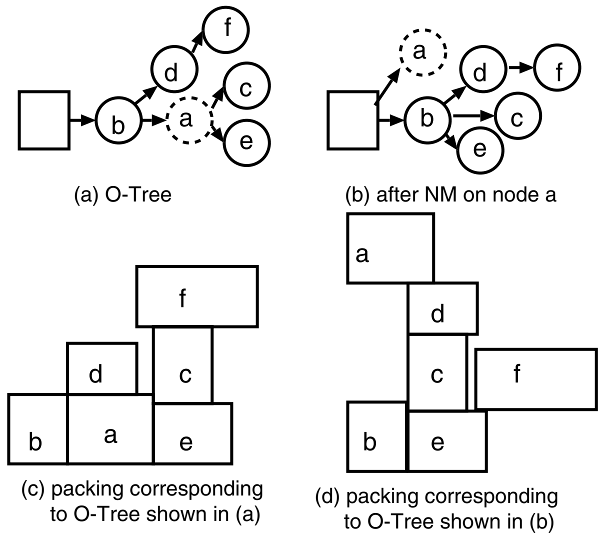
(a) O-Tree
(b) after NM on node a
(c) packing corresponding to O-Tree shown in (a)
(d) packing corresponding to O-Tree shown in (b)

Fig. 8. NM operation and change on O-Tree and packings

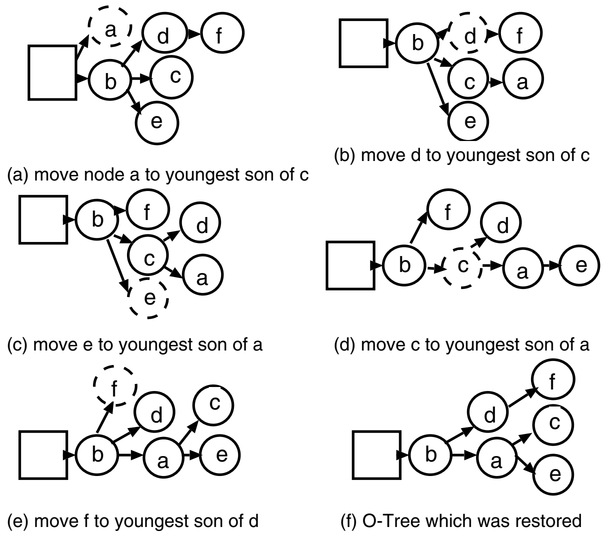
(a) move node a to youngest son of c
(b) move d to youngest son of c
(c) move e to youngest son of a
(d) move c to youngest son of a
(e) move f to youngest son of d
(f) O-Tree which was restored

Fig. 9. Several MOVEs to restore O-Tree shown in Fig.8(b) to O-Tree shown in Fig.8(a)

However, if we changed tree by "NM," if the node to move left descendants, several MOVEs are required to restore solutionIf the number of descendants of the node to move were k, and the number of younger brothers and those descendants were l, we need at most $k+l+1$ MOVEs to restore. Thus, relations between solutions are not symmetric by the MOVE operations including "NM" in solution space. An example is shown in Fig.9, where four "NM" operations are needed to restore the tree which we have changed in Fig.8.

conjecture1 : We make a conjecture that a degradation of search efficiency arises if adjacent solution is not "finite-restorable."

B. *Improved MOVE Operations and Experiments*

From **conjecture1**, we restrict "NM" so that they can be "finite-restorable," and carry out experiment to compare effectiveness.

"Node Move with leaving at most l descendants (NM_l)":

This MOVE operation is similar to "NM." The only difference is that a node to move is restricted that the total number of descendants (including younger brothers and their descendants) must be less than or equal to l.

To confirm the effectiveness, we implement SA search program based on O-Tree representation. We use

BIT: "Change of bit string" or "Change of permutation"

ExNM_∞: "NM" or "Exchange of nodes"

ExNM_1: "NM_1" or "Exchange of nodes"

ExNM_0: "NM_0" or "Exchange of nodes"

for MOVE operations. We use MCNC-benchmark ami33 (33 rectangles), ami49 (49 rectangles) and GSRC-benchmark n600 (600 rectangles). For each input sets, we

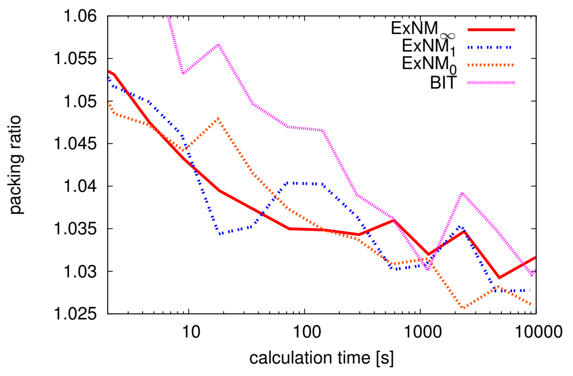

Fig. 10. comparison of search efficiency on ami33 (33 rectangles)

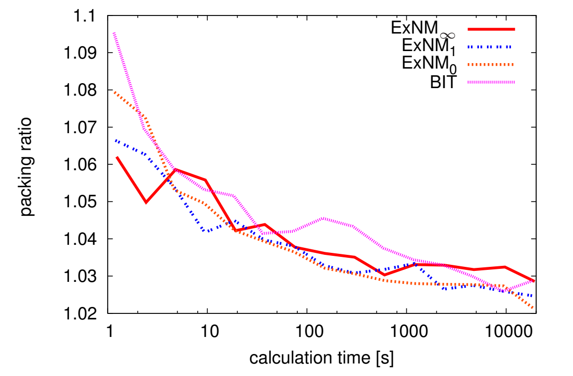

Fig. 11. comparison of search efficiency on ami49 (49 rectangles)

fixed the initial and final temperatures appropriately by pre-runs and carried out experiments for several cooling ratios. For each cooling ratio, SA searches are carried out five times with distinct seeds of pseudorandom numbers. Results are shown in Fig. 10–12, where the x-axis is the CPU time, and the y-axis is the packing ratio. The packing ratio is obtained by dividing the area of the bounding rectangle by the total area of all rectangles. Note that these figures do not show the process of SA. These show averages of five final solutions obtained by using SA with the same annealing schedules (initial and final temperatures and cooling ratio) and with distinct seeds.

From these figures, if we took more than a certain amount of time, results of ExNM_0 and ExNM_1 are better than that of ExNM_∞, that means SA with "finite-restorable" MOVE operations obtained better results. And from Fig.10–11, if we took a short time, results of ExNM_0, ExNM_1, and ExNM_∞ are better than that of BIT.

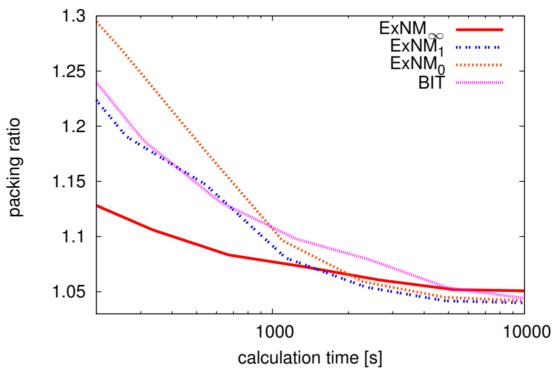

Fig. 12. comparison of search efficiency on n600 (600 rectangles)

III. Double Tree and Sequence[6]

DTS is a representation using "unordered tree." "Unordered tree" is a rooted tree that doesn't have order in child nodes that have the same parent node. DTS consists of two unordered trees, x-tree and y-tree, and one permutation, z-order.

x-constraints: X-tree is an unordered tree, where each node of the tree corresponds to a rectangular box, except that the root node of the tree represents a pseudo rectangular box whose x-size is zero, but the y-size and z-size are infinite. The x-tree constraint is "the x-coordinate of the left plane of the box corresponding to a node on x-tree must be the same as that of the right plane of the box corresponding to its parent node." y-tree is similar to x-tree.

z-constraints: If z-order $= (\ldots a \ldots b \ldots)$ and the rectangular projection of rectangular box a on the xy-plane overlaps that of b, "a is below b."

An example of DTS corresponding to the packing shown in Fig.14 is shown in Fig.13.

To implement DTS with SA, following two MOVE operations were used.[6]

"NM" : On x-tree or y-tree, move nodes in the similar way to O-Tree. Difference is that it is not necessary to move a node to the position of the youngest child, because DTS is an unordered tree, so it doesn't have brother relations.

"Exchange": Exchange two elements in the x-tree or y-tree or z-order.

A. *Improved MOVE Operations and Experiments*

For the same reason as O-Tree, we restrict "NM," so that it can be "finite-restorable," and carry out experiment to compare the effectiveness.

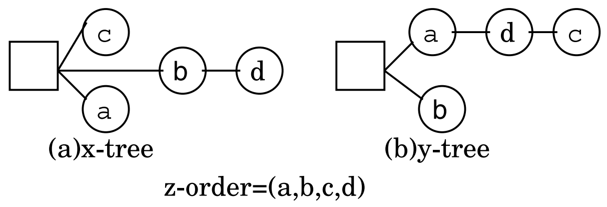

Fig. 13. An example of DTS

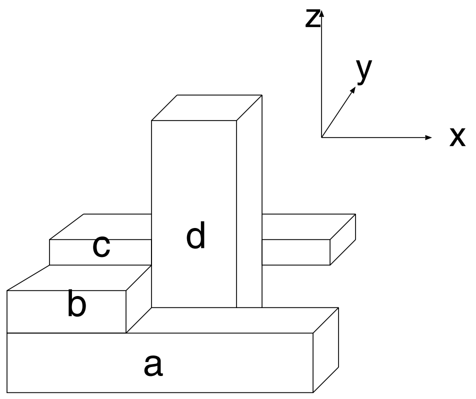

Fig. 14. 3D packing corresponding to DTS shown in Fig.13

"Node Move with leaving at most k descendants (NM$_k$)":

On x-tree or y-tree, move nodes in the similar way to O-Tree. Note that k is the number limit of descendants. Difference is that it is not necessary to put node to position of the youngest child, because DTS is an unordered tree, it doesn't have brother relations.

To confirm the effectiveness of SA search, we use

ExNM$_\infty$: "NM" and "Exchange"

ExNM$_3$: "NM$_3$" and "Exchange"

ExNM$_0$: "NM$_0$" and "Exchange"

for the MOVE operations. We use the benchmark "Beasley & OKP" okp4 (61 rectangular boxes), okp5 (97 rectangular boxes). Moreover, we use input sets that consist of 200 and 300 rectangular boxes of random integer sizes (between 5 and 100), respectively. For each input set, we fixed the initial and final temperatures appropriately by pre-runs and carried out experiments for several cooling ratios. For each cooling ratio, SA searches are carried out five times with distinct seeds of pseudorandom numbers. Results are shown in Fig.15–18, where the x-axis is the CPU time, and the y-axis is the volume ratio. The volume ratio is obtained by dividing the volume of the bounding rectangular solid boxes by the total volume of all rectangular boxes. Note that these figures do not show the process of SA. These show averages of five final solutions obtained.

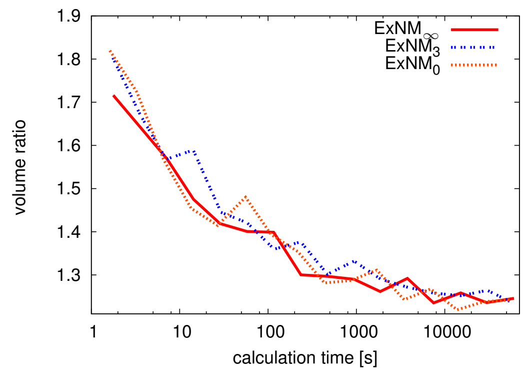

Fig. 15. comparison of search efficiency on okp4 (61 rectangular boxes)

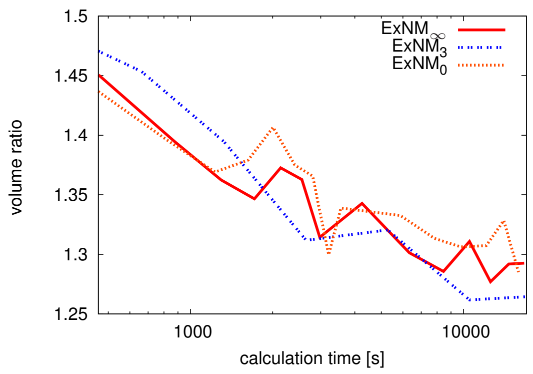

Fig. 16. comparison of search efficiency on okp5 (97 rectangular boxes)

In Fig.16, 17, 18, if we took more than a certain amount of time, result of ExNM$_0$ and ExNM$_3$ are better than that of ExNM$_\infty$, that means"finite-restorable" MOVE operations obtained better results.

IV. Conclusion

In this paper, for Simulated Annealing, we make a conjecture that a search efficiency by a MOVE operation is degraded if any one MOVE can't restore an adjacent solution made of solution f by one MOVE to solution f. Therefore, we prefigured that there should be the improvement of search efficiency, when we restrict MOVE operations, so that a certain series of j MOVEs can restore any adjacent solution made of solution f by one MOVE to solution f, where j is an integer not bigger than a given fixed integer. Then we carried out experiments for O-Tree, which is representation for rectangle packing, and DTS, which is representation for rectangular boxes, and made sure of validity.

Our future work is the theoretical verification of Simu-

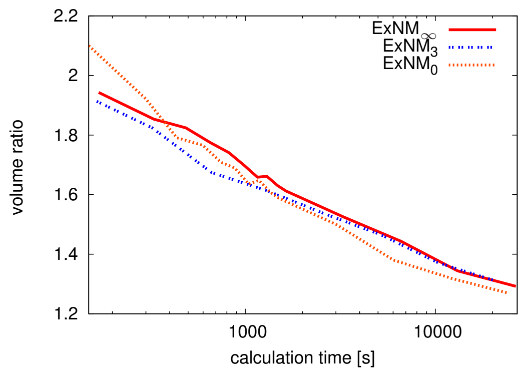

Fig. 17. comparison of search efficiency on 200 rectangular boxes

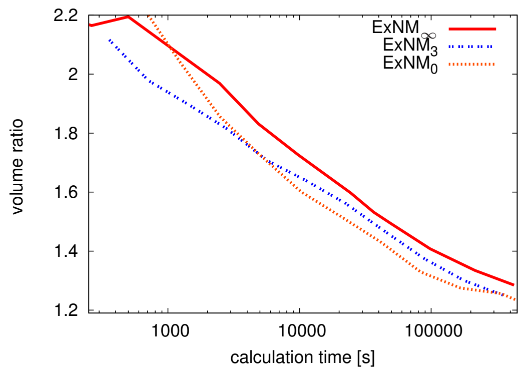

Fig. 18. comparison of search efficiency on 300 rectangular boxes

lated Annealing solution space.

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi: "Optimization by Simulated Annealing," Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.

[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani: "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," IEEE Trans. CAD, vol.15, no.12, pp.1518–1524, 1996.

[3] P. N. Guo, T. Takahashi, C. K. Cheng, and T. Yoshimura: "Floorplanning using a tree representation," IEEE Trans. CAD, vol. 20, no. 2, pp. 281–289, 2001.

[4] H. Ukibe, and K. Fujiyoshi: "Arbitrary Convex and Concave Rectilinear Block Packing Based on O-Tree Representation," APCCAS, pp.1554–1557, 2008.

[5] H. Yamazaki, K. Sakanushi, S. Nakatake, and Y. Kajitani: "The 3D-packing by meta data structure and packing heuristics," IEICE Trans. Fundam., vol.E83-A, no.4, pp. 281–289, 2000.

[6] K. Fujiyoshi, H. Kawai,and K. Ishihara: "A Tree Based Novel Representation for 3D-Block Packing,"IEEE Trans. CAD, vol. 20, no. 2, pp. 759–764, 2009.

[7] R. Sedgewick : "Algorithms," Addison-Wesley Publishing Company, 1989.

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

[9] T. Banno, and K. Fujiyoshi : "Improvement of Simulated Annealing Search Based on Tree Representations," IEICE Tech. Rep. CAS2014-51, pp. 1–6, 2014. (in Japanese).