A Simulated Annealing Approach with Sequence-Pair Encoding Using a Penalty Function for the Placement Problem with Boundary Constraints

Satoshi TAYU
School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa, Japan

Abstract— The module placement is one of the most important problem in the VLSI design. A practical VLSI placement problem often includes some constraints. In this paper, we propose a penalty function approach for the efficient simulated annealing search on the solution space of constrained problems. We apply the penalty function approach to the placement problem with boundary constraints. Experimental results show that our proposed method can accomplish more effective simulated annealing search than the conventional method proposed in [3] for two modules sets, an MCNC benchmark ami49 and a randomly generated module set.

I. Introduction

The module placement problem has been investigated in connection with the optimization problem of VLSI physical design and it is one of the most important subproblems. This problem is to place given modules on the plane without overlaps and its objective is to minimize the area of the minimum rectangular bounding box. There are several approaches to the problem and they have made a tremendous progress in VLSI technology. However, as the number of modules in recent VLSI systems becomes larger, it becomes harder to obtain a placement with small area. In order to solve this problem effectively, the stochastic methods such as the simulated annealing approaches and genetic algorithms come to be employed. Simulated annealing approaches, using one of encoding schemes such as bounded slice-line grid (BSG) [5], sequence-pair [4], and ordered tree (O-tree) [7], are widely applied to the module placement problem.

A sequence-pair is an ordered pair of permutations of modules and used as an encoding scheme in the simulated annealing approach to a placement problem. In a simulated annealing approach, we consider a solution space consisting of placements obtained from a sequence-pair by some decoding scheme and search on the solution space by traversing neighboring solutions. In the simulated annealing approach we use *the moves*, choosing scheme of neighborhoods, proposed in [1, 4].

In the simulated annealing approach, if infinite computation time is allowed and the transition probability matrix satisfies some conditions, we can reach an optimum with asymptotic probability one [2] while, in practical implementation with finite computation time, we often fall into some local optimum or other solutions. In addition, the practical VLSI design problem often includes some constraints and those constraints may avoid us reaching better solutions. Hence, we need to find an effective way to escape from worse local optimums with high probability in limited computation time implementation.

In this paper, we propose a penalty function approach to the problem with some constraints and we apply it to the placement problem with boundary constraints. As seen in the experimental results, we can find that our proposed method generates better solutions than the method proposed in [3], which also uses the simulated annealing search, though the computation time of one iteration stage, choosing a neighbor solution and generating and evaluating its placement, of our method takes $O(n\log n)$ computation time and that of the conventional method in [3] takes $O(n^2)$ computation time.

II. The Simulated Annealing with the Sequence Pair Encoding

A. The Sequence-Pair Encoding

A module m_i ($1 \leq i \leq n$) is a rectangle with its width w_i and height h_i and it must be located in the *xy*-plane such that its bounding line segments have the vertical or horizontal direction. A variable r_i represents whether the rotation is applied to module m_i ($r_i = 1$) or not ($r_i = 0$) (see Figs. 1 (a) and (b)). A module m_i is said to be *placed* if *x*- and *y*-coordinates and r_i are given, where (x_i, y_i) corresponds to the coordinates of the left bottom corner of m_i as shown in Fig. 1. We call $\vec{r} = (r_1, r_2, \ldots, r_n) \in \{0,1\}^n$ *a rotation vector* of modules. In order to represent the horizontal and vertical widths of placed modules, we use two functions 'hor' and 'ver' defined as $\mathrm{hor}(m_i) = w_i$ and $\mathrm{ver}(m_i) = h_i$ if $r_i = 0$ (Fig. 1 (a)) and $\mathrm{hor}(m_i) = h_i$ and $\mathrm{ver}(m_i) = w_i$ if $r_i = 1$ (Fig. 1 (b)). Two placed modules m_i and m_j are said to *overlap* each other if the following four inequalities hold: $x_i < x_j + \mathrm{hor}(m_j)$, $x_j < x_i + \mathrm{hor}(m_i)$, $y_i < y_j + \mathrm{ver}(m_j)$, and $y_j < y_i + \mathrm{ver}(m_i)$. A sequence of tuples $P = ((x_1, y_1, r_1), (x_2, y_2, r_2), \ldots, (x_n, y_n, r_n))$ is called *a placement (of a module set $M = \{m_1, m_2, \ldots, m_n\}$)* if no two modules overlap. We let $A(P)$ denote the area of the minimum rectangular bounding box including all modules,

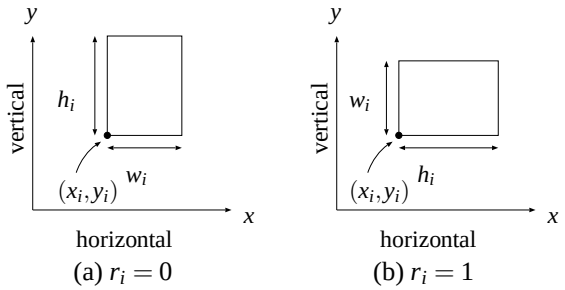


Fig. 1. An example of a placement.

i.e., $A(P) = V(P)H(P)$ for

$$V(P) = \max_{1\le i\le n}(x_i + \mathrm{hor}(m_i)) - \min_{1\le i\le n} x_i \quad \text{and}$$
$$H(P) = \max_{1\le i\le n}(y_i + \mathrm{ver}(m_i)) - \min_{1\le i\le n} y_i.$$

A placement P is said to be *optimal* if no other placement P' satisfies $A(P') < A(P)$. The objective of the placement problem is to compute a placement P with small $A(P)$.

A sequence-pair $\langle \Gamma_+, \Gamma_- \rangle$ is an ordered pair of permutations Γ_+ and Γ_- of n modules. The sequence-pair encoding scheme is used to represent some left-right and below-above restrictions between modules in the placement. Let $f_+(m_i)$ and $f_-(m_i)$ be the positions of m_i appearing in Γ_+ and Γ_-, respectively. For example, if $\Gamma_+ = (m_2, m_3, m_4, m_1)$ then $f_+(m_1) = 4$, $f_+(m_2) = 1$, $f_+(m_3) = 2$, and $f_+(m_4) = 3$. $\langle \Gamma_+, \Gamma_- \rangle$ prescribes the following restrictions R1 and R2 to the placement: For two modules m_i and m_j,

R1 if $f_+(m_i) < f_+(m_j)$ and $f_-(m_i) < f_-(m_j)$ (resp. $f_+(m_i) > f_+(m_j)$ and $f_-(m_i) > f_-(m_j)$), then $x_i + \mathrm{hor}(m_i) \le x_j$ (resp. $x_j + \mathrm{hor}(m_j) \le x_i$), i.e., m_i is *at the left* (resp. *right*) of m_j, and

R2 if $f_+(m_i) > f_+(m_j)$ and $f_-(m_i) < f_-(m_j)$ (resp. $f_+(m_i) < f_+(m_j)$ and $f_-(m_i) > f_-(m_j)$), then $y_i + \mathrm{ver}(m_i) \le y_j$ (resp. $y_j + \mathrm{ver}(m_j) \le y_i$), i.e, m_i is *below* (resp. *above*) m_j.

Note that m_i and m_j have exactly one of R1 and R2 relationships and those relationships are transitive.

$S = \langle \Gamma_+, \Gamma_-, \vec{r}\ \rangle$, a sequence-pair together with a rotation vector, is useful to represent some restrictions of a placement and we use it as a code. We now define a mapping P from such codes S to placements. The coordinates (x_j, y_j) of the left bottom corner of module m_j can be computed by the following equations:

$$x_j = \begin{cases} \max\{x_i + \mathrm{hor}(m_i) \mid f_+(m_i) < f_+(m_j), \\ \quad f_-(m_i) < f_-(m_j)\} \\ \quad \text{if at least one such } i \text{ exists and} \\ 0 \quad \text{if otherwise, and} \end{cases}$$

$$y_j = \begin{cases} \max\{y_i + \mathrm{ver}(m_i) \mid f_+(m_i) > f_+(m_j), \\ \quad f_-(m_i) < f_-(m_j)\} \\ \quad \text{if at least one such } i \text{ exists and} \\ 0 \quad \text{if otherwise.} \end{cases}$$

By definition, $P(S)$ is one of the smallest area placements satisfying the restrictions prescribed by S. Using some binary tree structure, such a placement $P(S)$ can be obtained from S in $O(n \log n)$ computation time[6]. It is known that there exists a solution S such that $P(S)$ is optimal [4].

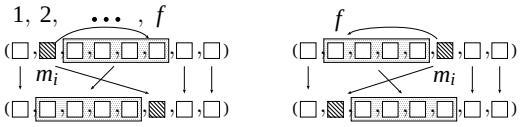
B. The Neighborhood Structure

For convenience, we often regard S as a placement, i.e., a solution, instead of $P(S)$. For some different codes S and S', $P(S) = P(S')$. In this case, we regard that S and S' are different solutions. In the simulated annealing, we search the solution space consisting of such placements S's. A neighborhood structure of the solution space can be represented by a set of operations to compute a neighborhood S' from a solution S. In this paper, we consider a solution space whose neighborhood structure is provided by the following three operations:

IN+ (−): (insert) choose a module m_i and a position f in Γ_+, put m_i to f and the other modules between $f_+(m_i)$ and f are shifted by 1 toward $f_+(m_i)$ (see Figs. 2 (a) and (b)).

FX: (full-exchange) choose two modules m_i and m_j and then exchange m_i and m_j in both Γ_+ and Γ_-.

RT: (rotate) choose one modules m_i and rotate it, i.e., $r_i := 1 - r_i$.



(a) IN+ for $f_+(m_i) < f$ (b) IN+ for $f_+(m_i) > f$

Fig. 2. Operations IN± and FX.

In other words, in our simulated annealing approach, a neighborhood is obtained from a current solution S by applying one of above operation to S, where each operation is chosen with probability $1/3$, here. The neighborhood structure with such probabilities corresponds to "the transition probability matrix" in [2].

C. The Simulated Annealing

Let $\mathbb{S}$ be the solution space with a neighborhood structre on which we search by the simulated annealing for minimizing the cost $C(S)$ of a solution $S \in \mathbb{S}$. Note that, in the area minimization problem mentioned above, $C(S) = A(S)$. A non-optimal solution S is said to be *locally optimal* if we cannot reach better solution from S by traversing neighboring solutions without visiting any worse solution S', i.e., S' with

INPUT: a module set, temperature schedule T, and finish temperature $T_e \geq 0$. (the initial solution S_0 is sometimes given in an input.)

OUTPUT: A solution S.

begin
construct an initial solution S randomly.
set $i := 0$, $S := S_0$, and $T := T_0$.
while $T > T_e$ **do**
repeat t times **do**
choose a neighborhood S' of S
choose p with $0 < p < 1$ randomly.
if $p \leq \exp\left(-\frac{C(S')-C(S)}{T}\right)$ (†)
$S := S'$.
end if
end repeate
$i := i+1$, $T := T_i$.
end while
end

Fig. 3. Simulated Annealing Algorithm.

$C(S') > C(S)$. For a locally optimal solution S, *the depth* $D(S)$ *of* S is the maximum value such that we cannot reach better solution from S by traversing only neighboring solutions S' with $C(S') < D(S)+C(S)$. The maximum depth of a locally optimal solution S is called *the depth of* $\mathbb{S}$ and denoted by $d(\mathbb{S})$.

Let $X(T_i)$ be the random variable representing the cost, the value of the objective function $C(S)$ of the current solution S, at each T_i in the algorithm and let C_{opt} be the minimum cost of a solution. Then, a simulated annealing search described in Fig. 3 with a temperature schedule $T_0, T_1, \ldots$ on a solution space $\mathbb{S}$ guarantees to reach an optimal solution with asymptotic probability one, i.e., $\lim_{i\to\infty} X(T_i) = C_{\text{opt}}$, if the followings five conditions are satisfied [2]:

(a) the solution space $\mathbb{S}$ is finite and irreducible,

(b) there exists an equilibrium distribution for the transition probability matrix,

(c) $T_i \geq T_{i+1}$ and $T_i > 0$ for all i,

(d) $\lim_{i\to\infty} T_i = 0$ (and $T_e = 0$), and

(e) $\sum_{k=0}^{\infty} \exp\left(\frac{-d(\mathbb{S})}{T_k}\right) = \infty$.

Note that, if $C(S') \leq C(S)$, the inequality (†) in Fig. 3 is always true.

In a practical implementation, we have to simulate the algorithm in finite computation time. In this paper, we set $T_i = r^{\lfloor i/t \rfloor} T_0$ for some integer t, real number $r = 1-\varepsilon$ for some positive $\varepsilon \ll 1$, and $T_e > 0$ as many other researchers do in the area minimization.

III. Boundary Constraint and its Penalty Function

A. Sequence Pair coding for Boundary Constraints

In the placement problem with boundary constraints, some modules are required to placed along some specified boundaries. The module set of the problem is divided into five subsets F, L, R, B, and T. A module $f \in F$ is said to be *free* and it has no boundary constraint. A module in L (resp., R, B, T) is said to be *left* (resp., *right*, *bottom*, *top*) and it must be placed along the left (resp. right, bottom, top) boundary. In the sequence pair coding of a solution satisfying the above boundary constraints, for $l \in L$ and $x \neq l$, either $f_+(l) < f_+(x)$ or $f_-(l) < f_-(x)$ holds. For $l \in L$, let *the V-value of* l be the number of modules x violating l, i.e., x has the restriction to be at the left of l, and denoted by

$$\text{vio}(l) \quad = \quad \#\{x | f_+(x) < f_+(l) \text{ and } f_-(x) < f_-(l)\}.$$

Similarly, for $r \in R$, $b \in B$, and $t \in T$, we define their V-value by

$$\begin{aligned} \text{vio}(r) &= \#\{x | f_+(x) > f_+(r) \text{ and } f_-(x) > f_-(r)\}, \\ \text{vio}(b) &= \#\{x | f_+(x) > f_+(b) \text{ and } f_-(x) < f_-(b)\}, \text{ and} \\ \text{vio}(t) &= \#\{x | f_+(x) < f_+(t) \text{ and } f_-(x) > f_-(t)\}, \end{aligned}$$

respectively. A solution S is said to be *feasible* if every constrainted module m satisfies $\text{vio}(m) = 0$. If S is feasible, we can obtain a feasible placement having the same area with $P(S)$ by relocating modules of $R \cup T$ in $P(S)$.

B. The Conventional Method

A transformation from an arbitrary sequence-pair to a feasible one, called SQ_Transformer, is proposed in [3]:

SQ_Transformer

Step 1 (LB-procedure): In the sequence Γ_+, place all modules in L to the left of the leftmost modules $b \in B$ in Γ_+, e.g.,

$$(\ldots, l_1, *, b_1, *, l_2, *, b_2, b_3, l_3, \ldots) \rightarrow (\ldots, l_1, *, l_2, l_3, b_1, *, *, b_2, b_3, \ldots),$$

where $l_i \in L$ and $b_i \in B$.

Step 2 (TR-procedure): In the sequence Γ_+, place all modules in R to the right of the rightmost module $t \in T$.

Step 3 (T-procedure): For all $t \in T$ (from the right to the left in Γ_+), find all modules x (from the left to the right in Γ_+) which appear at the left of t in Γ_+, if x appears at the left of t in Γ_- then x is placed at the right of t in Γ_-.

Steps 4-6 (B,L,R-procedures): B-,L-, and R-procedures are defined similarly to the T-procedure.

Step 7 : Output the final sequence-pair.

Steps 1 and 2 take $O(n)$ computation time and Steps 3-6 take $O(n^2)$ computation time [3], where n is the number of modules. Thus, the total computation time of the method is $O(n^2)$. We let $\mathrm{SQ}(S)$ be the feasible solution obtained from S by applying the SQ_transformer. In the simulated annealing in [3], a solution S is evaluated by the area of $\mathrm{SQ}(S)$ at each step, i.e., the cost function of S is given by $C(\mathrm{SQ}(S))$.

C. Proposed Method

C.1 Some Conditions for the Penalty Function

We first describe our strategy to obtain a feasible solution by simulated annealing search using a penalty function on the solution space including infeasible solutions. The extended objective function C used in our simulated annealing includes the penalty function pen multiplied by a sufficient large coefficient c, i.e.,

$$C(S) \quad = \quad C_0(S) + c\,\mathrm{pen}(S), \tag{1}$$

where $C_0(S)$ denotes the original objective function of the problem which is $A(S)$ in area minimization problem. The value c should be determined according to the temperature schedule and the instance.

It is desirable that, if $C(S)$ is *relatively small* and S is infeasible, then S may have a neighboring solution S' with $C(S') < C(S)$, where we assume that the simulated annealing search will reach a solution with relatively small value. Also, the reachability in the sub-solution space consisting of only feasible solutions should be guaranteed. For satisfying those properties, we will consider a penalty function pen and the neighborhood structure satisfying the following conditions:

Condition 1: The penalty function is a mapping from solutions to non-negative integers i.e., $\mathrm{pen} : \mathbb{S} \to \mathbb{N}$, and satisfies that, for all $S \in \mathbb{S}$, $\mathrm{pen}(S) = 0$ iff S is feasible.

Condition 2: For each infeasible solution S, there exists a neighboring solution S' such that $\mathrm{pen}(S') < \mathrm{pen}(S)$.

Condition 3: There exists at least one code S whose placement is optimal and $C(S) < C(S')$ for every non-optimal solution S' if c is sufficiently large.

Condition 4: For arbitrary feasible solutions S and S', it is reachable from S to S' with traversing only feasible solutions.

Since $\mathbb{S}$ is finite, Conditions 1 and 2 guarantee the existence of a path $(S_0, S_1, \ldots, S_k)$ from an arbitrary infeasible solution S_0 to some feasible one S_k such that traversing only neighboring solutions on $\mathbb{S}$ satisfying $\mathrm{pen}(S_i) < \mathrm{pen}(S_{i+1})$. In a practical implementation, we may need this for only solutions S_k having relatively small $C(S_k)$, where we could not describe the term *relatively small* explicitly. Also, we claim that

Claim 1 *In the experiment, c should be set to satisfy that, for a solution S having relatively small cost,* $C(S) - C(S') \leq c$ *for every neighboring solution* S' *of S.* ■

The best value of c may depend on the neighborhood structure and the temperature schedule.

C.2 Penalty Function

We now define a penalty function $\mathrm{pen}(S)$ of a solution S of the problem. The penalty function of a solution which we use in this paper is defined by

$$\mathrm{pen}(S) \quad = \quad \sum_{m \in L \cup R \cup B \cup T} \mathrm{vio}(m).$$

A module $m \in L \cup R \cup B \cup T$ is called *a violated module* if $\mathrm{vio}(m) > 0$. $\mathrm{pen}(S)$ can be calculated in $O(n \log n)$ computation time by using a binary tree structure similar to red-black tree but has little different property and, in order to enumerate the number of violations, a new inserted module is always added to the tree as a leaf. We omit the details here but we give the outline of the calculation. The insertion of modules to the binary tree is done by left to right order appearing in Γ_+. The modules are sorted in the tree according to the order of the position in Γ_-. At every insertion, each module m in the tree has four values, $d_L(m)$, $d_R(m)$, $d_B(m)$, and $d_T(m)$, where, for every $X \in \{L, R, B, T\}$, $d_X(m)$ denotes the number of descendants which are included in X. When a module is inserted to the tree, the number of violations between m and already inserted modules is enumerated. d_X's are updated in an insertion of a module and are recalculated in the reconstruction of the tree.

For the objective function given in (1), Conditions 1 and 3 hold, clearly. Condition 2 is supported by Theorem 1 if neighboring relationships provided by IN$\pm$ and FX operations are included in the neighborhood structure.

Theorem 1 *For an arbitrary solution S with* $\mathrm{pen}(S) > 0$, *there exists a neighboring solution* S' *through* IN$\pm$ *or* FX *operations such that* $\mathrm{pen}(S') < \mathrm{pen}(S)$.

Proof. We only show in the case that $S = \langle \Gamma_+, \Gamma_-, \vec{r} \rangle$ has at least one violated module in $L \cup T$. We can similarly show it in case that a violated module exists in $R \cup B$. Let $l_1 \in L \cup T$ be the leftmost violated module in Γ_+. Without loss of generality, we assume $l_1 \in L$. Let x be the rightmost module in Γ_+ such that any module appearing between x and l_1 in Γ_+ does not appear at the left of l_1 in Γ_-. Then,

$$\begin{aligned} \Gamma_+ &= (\ldots, x, \ldots, y, \ldots, l_1, \ldots), \\ \Gamma_- &= (\ldots, x, \ldots, l_1, \ldots, y, \ldots). \end{aligned}$$

Therefore, we have the following:

Proposition 1 *Every module y between x and* l_1 *in* Γ_+ *appears after* l_1 *in* Γ_-. ■

Let y be an arbitrary module between x and l_1 in Γ_+. Since $l_1 \in L$ is the leftmost violated module in Γ_+, $y \notin L$.

If $x \notin T$, let $S' = \langle \Gamma'_+, \Gamma_-, \vec{r} \rangle$ be the solution obtained by inserting l_1 immediately before x in Γ_+, i.e., $\Gamma'_+ := (\ldots, l_1, x, \ldots, y, \ldots)$. Since $x \notin T$, and $y \notin L$, V-values of x and y do not increase and we have the theorem.

If $x \in T$, let S' be the solution obtained by the full-exchange operation of l_1 and x. For the full-exchange operation, V-values of other modules do not increase. Also, the V-value of l_1 becomes smaller. From Proposition 1, the V-value of x does not increase. Therefore, the total sum of V-values decreases, i.e.,

$\mathrm{pen}(S') < \mathrm{pen}(S)$. ■

We next show that Condition 4 holds under our neighborhood structure. Note that the rotation vector of a solution is nothing to do with the feasibility and we need not to consider the rotation vector. For $X \in \{L,R,B,T,F\}$, let $X = \{m_1^X, m_2^X, \ldots, m_{|X|}^X\}$, Φ_X be a sequence $(m_1^X, m_2^X, \ldots, m_{|X|}^X)$, and Φ_X^r be the reverse of Φ_X. From the following, the neighborhood structure satisfies Condition 4:

Theorem 2 *Let S be an arbitrary feasible solution and $S' = \langle \Gamma'_+, \Gamma'_-, \vec{r} \rangle$ with $\Gamma'_+ = (\Phi_L, \Phi_T, \Phi_F, \Phi_R, \Phi_B)$ and $\Gamma'_- = (\Phi_L^r, \Phi_B, \Phi_F, \Phi_R^r, \Phi_T)$. Then, there exists a path $(S_1, S_2, \cdots, S_k)$ with $S_1 = S$ and $S' = S_k$ such that every S_i is feasible, and S_i and S_{i+1} are neighborhood in $\mathbb{S}$ for i with $1 \le i \le k-1$.*

Proof. We apply IN+ operations to all modules in $L \cup T$ according to the order appearing in Γ_+ so that every module in $L \cup T$ is placed at the left of every module in $F \cup R \cup B$ without traversing any infeasible solution. See the following example, where $l_i \in L$, $t_i \in T$, and $\alpha_i \notin L \cup T$:

$$
\begin{array}{c}
(\alpha_1, \alpha_2, \underline{l_1}, \alpha_3, \alpha_4, \underline{t_1}, \alpha_5, \underline{l_2}, \ldots) \\
\Downarrow \\
(\underline{l_1}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \underline{t_1}, \alpha_5, \underline{l_2}, \ldots) \\
\Downarrow \\
(\underline{l_1}, \underline{t_1}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \underline{l_2}, \ldots) \\
\Downarrow \\
(\underline{l_1}, \underline{t_1}, \underline{l_2}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \ldots) \\
\vdots
\end{array}
$$

We next apply IN+ operation to all modules in $R \cup B$, we can obtain a sequence so that every module in $R \cup B$ is placed at the right of every module in $F \cup T \cup L$. Applying IN− operations to the current sequence-pair by the similar fashion, we can obtain a sequence-pair with

$$
\begin{aligned}
\Gamma_+ &= \left(\boxed{L \cup T}, \boxed{F}, \boxed{R \cup B} \right) \text{ and} \\
\Gamma_- &= \left(\boxed{L \cup B}, \boxed{F}, \boxed{R \cup T} \right),
\end{aligned}
$$

where, for a partition $M_1, M_2, \ldots, M_m$ of M,

$$
\left(\boxed{M_1}, \boxed{M_2}, \ldots, \boxed{M_m} \right)
$$

denotes a sequence such that, for all pair of $m \in M_i$ and $m' \in M_j$ with $i < j$, m appears before m' in it. More applications of IN± operations to it, we can obtain a sequence-pair with

$$
\begin{aligned}
\Gamma_+ &= \left(\boxed{L}, \boxed{T}, \boxed{F}, \boxed{R}, \boxed{B} \right) \text{ and} \\
\Gamma_- &= \left(\boxed{L}, \boxed{B}, \boxed{F}, \boxed{T}, \boxed{R} \right).
\end{aligned}
$$

Then, applying some FX operations to it, we can obtain a sequence-pair with

$$
\begin{aligned}
\Gamma_+ &= \left(\Phi_L, \Phi_T, \boxed{F}, \Phi_B, \Phi_R \right) \text{ and} \\
\Gamma_- &= \left(\Phi_L^r, \Phi_B, \boxed{F}, \Phi_T^r, \Phi_R \right).
\end{aligned}
$$

Lastly, by applying IN± operations to the modules of F (and RT operations to some modules), the desired solution S_0 is obtained. Since infeasible solutions are not generated in each transformation, we have the theorem. ■

It should be noted that, since we start with a randomly generated solution (sequence pair with a rotation vector) and it may be infeasible solution, the probability with the final placement being feasible mainly depends on the temperature schedule (including the length of the schedule) and the value c. Of cause, it also depends on the problem instance, i.e., the module set and constrained modules. Clearly, if we use a temperature schedule with small length, we can obtain a feasible solution with probability almost 0, where the length of a temperature schedule $T_0, T_1, \cdots$ with T_e is $\max\{k | T_{k+1} \le T_e\}$. So, c should be set according to the temperature schedule and the instance.

IV. Experimental Results

In this section, we give the experimental results of the simulated annealing searchs on $\mathbb{S}$ under conventional and our proposed methods under the same temperature schedule. We apply the simulated annealing approach to BenchMark ami49 and a randomly generated module sets M_{100}, where a module m_i in M_{100} is generated satisfying that $10 \le w_i, h_i \le 100$, $\max\{h_i/w_i, w_i/h_i\} \le 3$. The number of each boundary constrained modules is set to the same value, i.e., $|L| = |R| = |T| = |B|$. For ami49, we examine two cases $|L| = 4$ and 6 and, for M_{100}, $|L| = 6$ and 9, where we choose them randomly but, in case of ami49, we avoid choosing largest two modules as boundary constrained ones. The neighborhood structure is composed by the three operations RT, FX, and IN±, and each operation is chosen with same probability. We apply simulated annealing approach 100 times to ami49 and M_{100} with parameters $T_0 = 10^5$, $T_e = 10$, $r = 0.98$, $t = 10n$, and $c = A_M/10$ for ami49 and $T_0 = 10^5$, $T_e = 1$, $r = 0.98$ $t = 10n$, and $c = A_M/100$ for M_{100}, where n is the number of modules. The value c should be set according to the temperature schedule and the instance, while, in our experiment, c is normalized by only A_M.

TABLE I
Experimental results for ami49.

$\|L\|$	4		6		0
	conv.	prop.	conv.	prop.	—
avg.	1.0479	1.0442	1.0582	1.0529	1.0351
best	1.0376	1.0332	1.0467	1.0398	1.0248
worst	1.0587	1.0600	1.0785	1.0791	1.0509
run time	31	18	35	18	16

We apply our method 100 times to ami49 and M_{100}. We show the average ratio $A(S)/A_M$ and average runtime (sec) for ami49 in Table I for each case, where the conventional method is shown by "conv" and our proposed method by "prop." $|L| = 0$ implies the experimental result of the placements without boundary constraints. The best placement for $|L| = 6$ is shown in Fig. 4. In the experimental result, we can

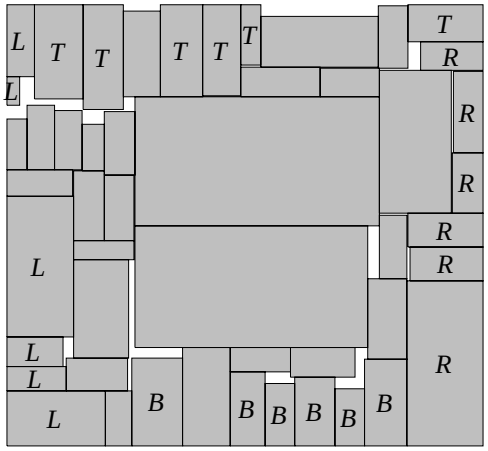


Fig. 4. The best placement of ami49 with $|L| = 6$.

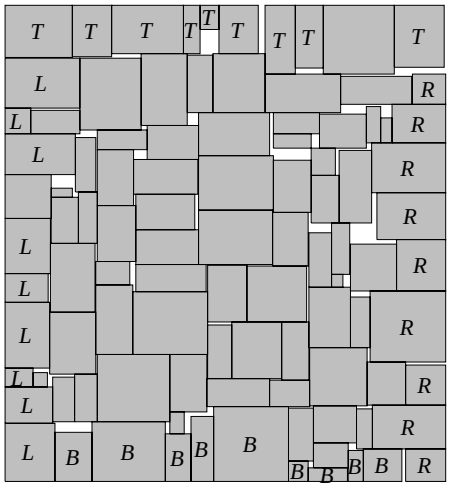


Fig. 5. The best placement of M_{100} with $|L| = 9$.

TABLE II
EXPERIMENTAL RESULTS FOR M_100.

$\|L\|$	6		9		0
	conv.	prop.	conv.	prop.	—
avg.	1.0479	1.0442	1.0582	1.0529	1.0362
best	1.0376	1.0332	1.0467	1.0398	1.0305
worst	1.0587	1.0600	1.0785	1.0791	1.0445
run time	176	101	214	102	85

find that our method generates better solutions than the conventional method.

We show the results in Table II and the best placement for $|L| = 9$ in Fig. 5. From those results, we can see that, if $|L|$ become large, our methods become more effective than the conventional method.

V. CONCLUSION

In this paper, we proposed a penalty function approach for stochastic optimization methods of a problem with some constraints. Since the penalty function is just in the objective function, this approach may be available for optimization problems with many kinds of codes and stochastic methods. However, the theoretical analyses for the penalty function approach are remained in the future work. We applied the penalty function approach to the placement problem with boundary constraints and constructed an efficient searching scheme by simulated annealing search using sequence pair encoding.

REFERENCES

[1] K. Fujiyoshi, T. Ohomura, and K. Ijiri, "Solution space by sequence-pair suitable for simulated annealing search," technical report, IEICE, 2000. (in Japanese).

[2] B. Hajek, "Cooling schedules for optimal annealing," Mathematics of Operations Research, vol. 13, no. 2, pp. 311–329, 5 1988.

[3] J. Lai, M.-S. Lin, T.-C. Wang, and L.-C. Wang, "Module placement with boundary constraints using the sequence-pair representation," Proc. Asia and South Pacific Design Automation Conference, pp. 515–520, Feb. 2001.

[4] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," IEEE Trans. Computer-Aided Design, vol. 15, no. 12, pp. 1518–1524, Dec. 1996.

[5] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani, "Module packing based on the bsg-structure and ic layout application," IEEE Trans. Computer-Aided Design, vol. 17, no. 6, pp. 519–530, Jun. 1998.

[6] T. Takahashi, "An algorithm for finding a maximum-weight decreasing sequence in a permutation, motivated by rectangle packing problem," technical report, IEICE, 1996. (in Japanese).

[7] T. Takahaashi, "A new encoding scheme for rectangle paking problem," Proc. Asia and South Pacific Design Automation Conference, pp. 175–178, Dec. 2000.