論文 / 著書情報
Article / Book Information

Title	A Simulated Annealing Based Approach to Integrated Circuit Layout Design
Authors	Yiqiang Sheng, Atsushi Takahashi
Citation	Simulated Annealing - Single and Multiple Objective Problems, , , pp. 239-260
Pub. date	2012, 10
URL	http://www.intechopen.com/articles/show/title/a-simulated-annealing-based-approach-to-integrated-circuit-layout-design
Creative Commons	See next page.

License

Chapter 12

A Simulated Annealing Based Approach to Integrated Circuit Layout Design

Yiqiang Sheng and Atsushi Takahashi

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51126

1. Introduction

The optimization techniques for integrated circuit (IC) layout design are important. Generally speaking, the basic process of modern hardware engineering includes designing, manufacturing and testing. IC layout is an inevitable stage of designing before manufacturing. There are many applications which are directly related with layout optimization in practice, such as floor plan for very-large-scale integration (VLSI) design, placement for printed circuit board (PCB) design, packing for logistics management, and so on. In this research, we mainly focus on the optimization for three layout problems, which are 2D packing, 3D packing and 2D placement. The 2D/3D packing is to position different modules into a fixed shape, normally rectangular one, with area or volume minimization. The placement can be regarded as the packing problem with interconnect optimization. Since a general placement problem is NP-hard, there are no practical exact algorithms so far to be sure to find optimal solutions. As an alternative to get the optima, heuristics [1-6] are typically used to find near optimal solutions within a given runtime.

As product size keeps shrinking, product lifecycle keeps shortening and product complexity goes up, more electronic components will be integrated into a smaller IC chip or PCB with higher density and shorter time to market. At the same time, multi-objective optimization is common for IC/PCB layout in real product design, so another difficulty is the trade-off between conflicting objectives, such as low power and high performance. Pareto improvement for multiple objectives is one of the biggest challenges we have to face nowadays. The layout problem becomes much harder to find near-optimal or even acceptable solutions with high requirements. In order to improve the best cases and mitigate the worst cases of IC/PCB layout, it becomes increasingly critical and urgent to improve the quality of solution and reduce runtime.

Simulated annealing based algorithm with a good representation for 2D/3D packing is one of the most popular ways to improve the quality of solution. On the one hand, many

researches explored different representations [7-12], such as bounded-slice-line grid, sequence pair, FAST sequence pair, Q-sequence, selected sequence pair, etc. In order to code and decode 3D-packing problem, sequence pair for 2D packing is extended to sequence triple and sequence quintuple, and it has been proved that sequence triple could represent the topology of the tractable 3D packing and there are at least one sequence quintuple which can be decoded to a topology as an optimal packing for volume minimization. But the effectiveness to improve solution quality and reduce runtime is quite limited due to huge solution space and complex solution distribution, even if a very good representation is used. The experimental results within a short runtime are still far from near-optimal solutions in real implementation to solve the packing problem. So it is the right time to explore new algorithms in order to solve 2D/3D problem more effectively.

There are many significant shortcomings of traditional heuristics for IC layout optimization. Let us take simulated annealing (SA) and genetic algorithm (GA) as an example. For SA, firstly, some slight modifications of solution are repeated to get a good convergence. Therefore, the global search is inefficient in general. It is disadvantageous to solve the problem with huge solution space, such as VLSI design. Secondly, SA does not use the past experience, including past good solutions and past moves, and it is a big informational waste. To speed up SA, some researchers [4] proposed two-stage SA for VLSI design. But the search speed is still quite slow, and it is not seriously considered to avoid or reduce informational waste. For GA, it evaluates too many candidates in order to get next generations. The evaluation takes too much runtime. Besides GA selects the next generation according to a ranking function, which is not always necessary but takes much time. So it is possible to improve the solution quality or reduce runtime if we can overcome the mentioned shortcomings.

In this research, a simulated annealing based approach [13-14], named mixed simulated annealing (MSA), is proposed to improve solution quality and reduce runtime by overcoming the shortcomings of inefficient global search and informational waste. In mixed simulated annealing, a special crossover operator is designed to use a part of information from past good solutions and get higher improving efficiency, and the solutions gotten by the crossover are much better than random solutions. To evaluate the effectiveness and the reliability of the proposed mixed simulated annealing, we apply it to three mentioned optimization problems, i.e. 2D packing, 2D placement and 3D packing, and get considerable improvement for all three problems. The experimental results show the runtime, the packing ratio of area for 2D packing, the packing ratio of volume for 3D packing and two more objectives (low power and short maximal delay) for 2D placement are improved considerably by using MCNC, ami49_X and ami98_3D benchmarks. For example, the runtime of mixed simulated annealing with sequence quintuple representation is up to 4 times faster than that of 2-stage SA with the same representation, and the packing ratio of volume is improved by up to 12% within 100s runtime.

2. Simulated annealing for integrated circuit layout

Based on the theory of statistical mechanics and the analogy between solid annealing and optimization problem, S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi [1] proposed simulated

annealing algorithm in 1983. The annealing is to heat up a solid with a very high temperature and then to cool it down slowly until it reaches or approaches its minimum energy state. Each state of solid represents a feasible solution of problem. The energy of the state is the value of cost function to evaluate the solution. The state with the lowest energy corresponds to the optimal solution with the best value of cost function. SA is a stochastic algorithm with iterative improvement. Each iterative step consists of changing current solution to a new solution, named a move to neighbourhood. The acceptance probability of new solutions depends on the current temperature, which is scheduled from the highest temperature to the lowest temperature. An important point we have to mention here is that, if the physical process is to cool the solid down very quickly, it is known as quenching, instead of annealing. The difference between normal simulated annealing and simulated quenching is the parameter setting of temperature scheduling.

In detail, let S be the solution space with neighbourhood structure. For any solution S belongs to S, we define the cost function C(S), i.e. the total cost function (C_t) for multi-objective placement problem. A non-optimal solution S is defined by local optimum, if it can not reach better solution by moving to any neighbouring solution S′. That is to day, for any neighbour solution (S′) of local optimal solution (S), the inequality C(S) < C(S′) is always satisfied. The depth D(S) of local optimal solution is defined by the maximum value such that D(S) + C(S) > C(S′). The maximum depth of local optimal solution in S is denoted by d(S). Let $X(T_i)$ be a variable of the cost function C(S) at each temperature T_i, where i is 0, 1, 2, … . Let C_{opt} be the minimum cost function. According to [2], the equality $\lim_{i\to\infty} X(T_i) = C_{opt}$ is satisfied with the following conditions: (1) The solution space S is finite and irreducible; (2) There exists an equilibrium distribution for the transition probability matrix; (3) $T_i \geq T_{i+1}$ and $T_i > 0$ for all i; (4) $\lim_{i\to\infty} T_i = 0$; (5) $\sum_{i:\in(0,\infty)} [\exp(-d(S)/T_i)] = \infty$.

In real implementation with a given finite runtime, we are using a fast geometric simulated quenching scheduling ($T_{k+1} = qT_k$, $0<q<1$) with repeated inside loop (p times) to enhance the efficiency of standard SA [3] as the following equation.

$$T_i = T_0 \cdot q^{\lfloor i/p \rfloor} \tag{1}$$

where i is the iterative step, T_i is the variable temperature at the i^{th} step, T_0 is the initial temperature when i = 0, p is the inside loop number and q the temperature coefficient near but less than 1.

As shown in Figure 1, it is a typical flow chart of SA, which is used for layout optimization in this research. The initial solution is randomly produced or simply follows past layout designs. The temperature scheduling is used to change the current temperature (T). The parameters of the temperature scheduling include the starting temperature T_0, the ending temperature T_e, a temperature coefficient and an inside loop number. One of moving methods is selected with given probabilities, for example, the same probability for each moving method in real experiment (near 33% in the case of three moving methods). A new solution is tried by using the current selected moving method. The new solution is evaluated by a cost function (C) and compared with the old one. The new solution is

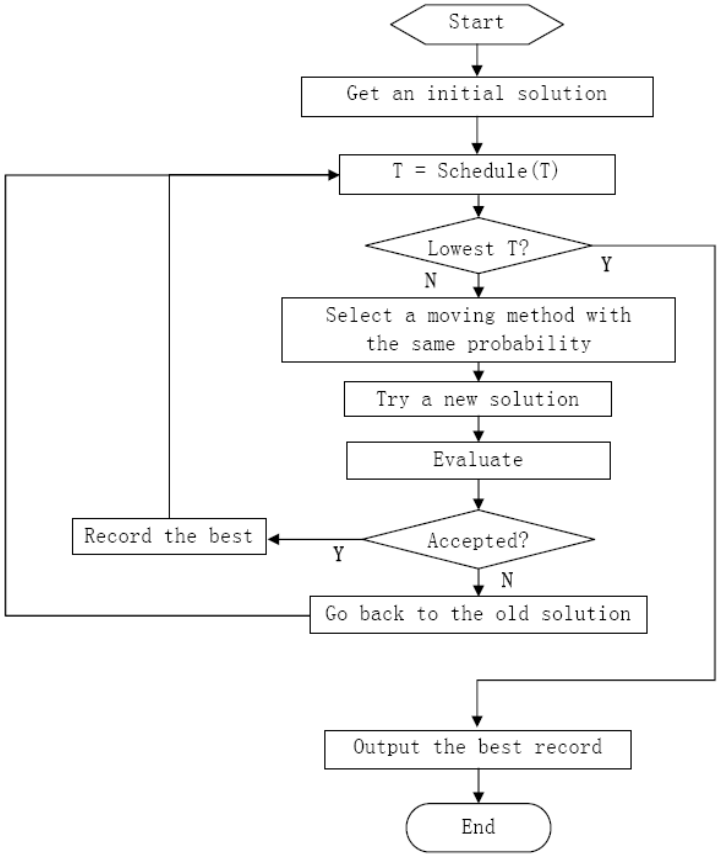

Figure 1. A typical flow chart of simulated annealing

accepted with a calculated acceptance probability $P = \exp[-\Delta C/T]$, which depends on the difference of cost function (ΔC) and the current temperature (T). The probability P is between 0 and 1. The temperature coefficient between 0 and 1 is set to control the speed of temperature reduction. The inside loop number is set to control the repeated moves for each T. If the new solution is improved ($\Delta C < 0$), then $P = 1$, and the best recorder will be implemented: If the new solution is better than the current best, the best record will be replaced. If rejected, the current solution will go back to the old one and continues the next temperature scheduling until reaching the lowest temperature Te. The output is the latest best record. The real implementation of SA algorithm depends on four basic definitions: (1) solution representation, (2) moving methods, (3) cost function, (4) temperature scheduling.

For the solution representation, 2D/3D topology for IC layout is defined by the orthogonal coordinate system, as shown in Figure 2, and is represented by sequence pair for 2D general cases, sequence triple for 3D simple cases or sequence quintuple for 3D general cases. Each

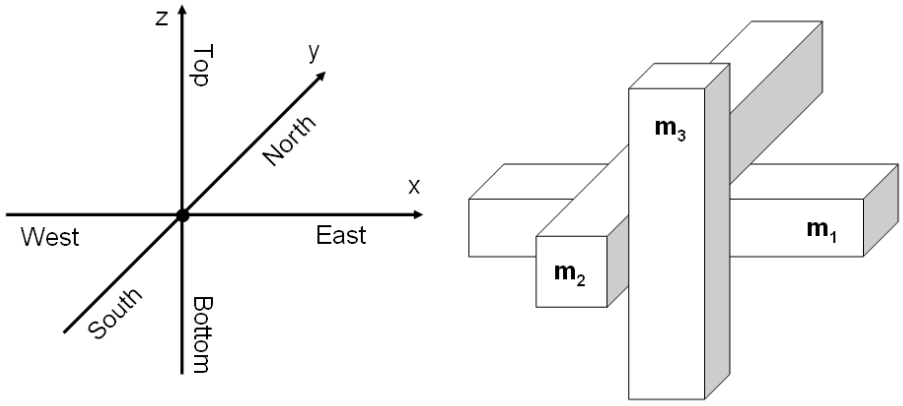

Figure 2. Orthogonal coordinate system and 3D packing topology represented by $\Gamma^1(m_2,m_1,m_3)$, $\Gamma^2(m_1,m_3,m_2)$ and $\Gamma^3(m_1,m_2,m_3)$ according to relative location

layout in 3D general cases is regarded as a set of the relations of relative location between boxes, i.e. "Top-Bottom", "North-South" and "West-East" (TB-, NS- and WE-) relations. The coding and the decoding are based on TB-, NS- and WE- relation corresponding to the order of modules. In Figure 2, box m_2 is on the west of box m_3, i.e. WE-relation, since the x-coordinate of any part of box m_2 is always smaller than or equal to that of any part of box m_3. Similarly box m_1 is on the north of box m_3, i.e. NS-relation, and box m_2 is on the top of box m_1, i.e. TB-relation. The 3D packing can be represented by $\Gamma^1(m_2,m_1,m_3)$, $\Gamma^2(m_1,m_3,m_2)$, and $\Gamma^3(m_1,m_2,m_3)$ according to the coding rule. The detail of representation will be introduced in section 5. The solution space of all mentioned representation is finite, instead of infinite solution space of original layout problem. All solutions decoding by the mentioned representations are feasible.

For the moving methods, let us take 2D placement as example. Three basic moving methods with small changes are designed to change the current solution by using the sequence-pair representation. The "rotation" changes the orientation of a module. The "exchange" exchanges the order of two modules in all sequences. The "move" changes the order of a module in one of sequences. The detail of each moving method will be discussed in section 6 and section 7.

For the cost function in the case of 2D placement, the total value (C_t) includes the dynamic power function (C_p), the maximal delay function (C_d) and the bounding area function (C_a). The estimation for each cost function will be discussed in section 8.

For the temperature scheduling, the starting temperature T_0, the ending temperature T_e, a temperature coefficient and an inside loop number are set according to the size of module number and the requirement of solution quality. As a reference, a set of parameters in our experiment is set as follows: T_0 = 100000, T_e = 10, Inside loop number p = 500, Temperature coefficient q = 0.98.

3. How to improve traditional simulated annealing

There are at least two shortcomings which impact the search speed of traditional SA. (1) Inefficient global search: In order to assure a final convergence effectively, the moving methods with relatively small changes should be used, so the global search within a short runtime is quite limited. Even using higher temperature, it is still slow to explore the huge search space of layout problem. (2) Informational waste: It does not use the information of past experience, including past solutions and past moves. It is quite possible to get a very good configuration of past solutions at the beginning but to lose it at last.

First of all, to overcome the shortcoming of inefficient global search, a two-stage algorithm is considered as follows. The first stage is named rough search, and the second stage is named focusing search. The rough search tends to big changes, such as crossover from genetic algorithm, to improve global search ability, while the focusing search tends to small changes, such as exchange, move and rotation, to get final convergence and reach better near-optimal solution.

Secondly, to overcome the shortcoming of informational waste, a special crossover operator from genetic algorithm, which reuses the information of past solutions, is considered. Comparing with random operator, the crossover operator has a search direction, which is based on the configuration from past good solutions, by use a part of configuration of the current best to reduce the informational waste. Besides, a guide with the probabilities to select running method adaptively according to the short-term improving speed is also considered.

In real implementation of 2-stage SA, the temperature scheduling of the second stage is same with that of the first stage using a geometric scheduling ($T_{k+1} = qT_k$, $0<q<1$) with repeated inside loop (p times) as the equation (1). The only difference is the parameter setting. In the second stage, T'_i (instead of T_i in the first stage) is the variable temperature at the i^{th} step in the second stage, T'_0 (instead of T_0) is the initial temperature of the second stage, p' (instead of p) is the inside loop number of the second stage and q' (instead of q) the temperature coefficient of the second stage. The detailed temperature scheduling setting depends on the requirement of runtime. As a reference, two sets of the parameters are set separately as follows: $T_0 = 100000$, $T_e = 100$, $p = 1000$, $q = 0.98$, $T'_0 = 1000$, $T'_e = 1$, $p' = 1000$, $q' = 0.98$. For different benchmarks, the inside loop number can be increased from 1000 to 2000, 5000, etc. Also the temperature coefficient can be closer to 1, such as 0.99, 0.995, etc. The parameter setting with a given runtime is adjusted by selecting the initial temperature, the final temperature, the temperature coefficient and the inside loop number.

4. Mixed simulated annealing

By overcoming the mentioned shortcoming, we proposed a mixed simulated annealing (MSA) to speed up traditional simulated annealing (SA) and 2-stage SA. The basic idea is to improve the global search ability and to speed up the search process by a special crossover operator, which uses the information of past solutions. Just like SA and 2-stage SA, MSA is

an iterative improvement method and a stochastic algorithm. The main difference between 2-stage SA and MSA is the special crossover operator to use a part of configuration of the current best and to reduce the informational waste. Although we can get a rough solution by producing solutions randomly, it is with low improving efficiency. The proposed crossover operator has a search direction, which is based on the configuration from past good solutions, to get high improving efficiency.

The intuitive comparison shows several intuitive advantage of MSA comparing with traditional SA and 2-stage SA. For global search ability, 2-stage SA is better than traditional SA due to big changes in the first stage, and MSA is even better than 2-stage SA due to the crossover operator and even bigger changes in the first stage. Traditional SA and 2-stage SA do not use past experience, while MSA is using past good solutions by using the crossover operator.

5. Application to integrated circuit layout optimization

The detailed IC design process includes system specification, architectural design, functional design, logic design, circuit design, layout design and verification. The layout is near the last stage of IC design, and it is a critical stage of electronic product development. As one of the key steps of IC layout, the placement has big impact on the overall quality of IC chip.

5.1. Problem definition

Let us start with the formulation of 3D placement. Let $M = \{m_1, m_2, ..., m_n\}$ denote the modules or blocks to be placed, where n is the number of modules. Each m_i, where $1 \le i \le n$, has height h_i, length l_i and width w_i. The packing volume is defined by the minimum bounding rectangular parallelepiped including all modules. For the placement, we need to optimize the interconnect networks. Let $N = \{n_1, n_2, \ldots, n_l\}$ be the set of interconnect nets between modules, where l is total net number. Let len_i denote the estimated wire length of each net n_i, $1 \le i \le l$. Let P_i denote the estimated dynamic power, i.e. the interconnect power of net n_i. Let $(x_i, y_i, z_i, r_{x\text{-}i}, r_{y\text{-}i}, r_{z\text{-}i})$ be the location and rotation on 3D orthogonal coordinate system for each module m_i, $1 \le i \le k$, where (x_i, y_i, z_i) means the coordinates of the below-rear-left corner of module m_i, and $(r_{x\text{-}i}, r_{y\text{-}i}, r_{z\text{-}i})$ denotes the rotation (0, 1) of m_i on yz-, zx- and xy-plane. $r_{z\text{-}i} = 1$ is the normal state of modules, while $r_{z\text{-}i} = 0$ is rotated by 90 degree.

In short, The input is a set of modules $M = \{m_1, m_2, ...\}$ with height, length and width $\{(h_1, l_1, w_1), (h_2, l_2, w_2), ...\}$ and a net list $N = \{n_1, n_2, ...\}$. The constraint is no overlap between m_i and m_j, where $i \ne j$. The output is a set of location and rotation for each module $\{(x_1, y_1, z_1, r_{x\text{-}1}, r_{y\text{-}1}, r_{z\text{-}1}), (x_2, y_2, z_2, r_{x\text{-}2}, r_{y\text{-}2}, r_{z\text{-}2}), ...\}$ such that: (1) Minimize the power consumption; (2) Minimize the maximal delay; (3) Minimize the volume of bounding box.

The 3D-packing problem is a special case of 3D placement with no consideration of power and delay. The 2D placement problem is regarded as a special case of 3D placement with z=0, as shown in Figure 3. The 2D-packing problem is a special case of 3D packing with z=0. All of the mentioned three problems are formulated so far.

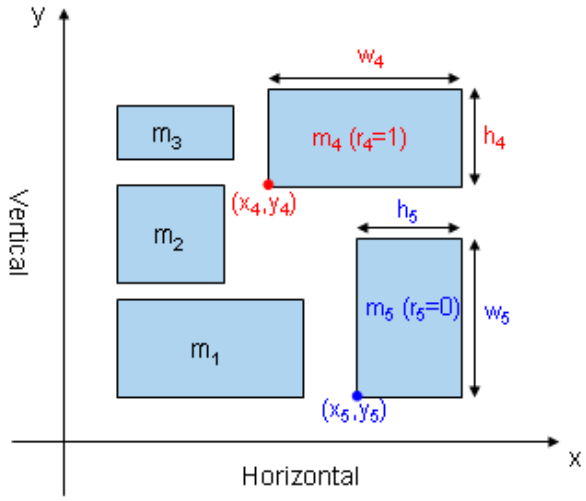

Figure 3. 2D placement as a special case of 3D placement

5.2. Problem representation

The original packing or placement is with infinite solution space. The coding and decoding method is generally needed to connect the problem and its representation. The solution space of a good representation should be finite. And the solutions after a good representation should be feasible and be better to include at least one optimal solution. Sequence quintuple can be used to represent a general 3D-packing topology, but the solution space of sequence quintuple is quite large. Furthermore, sequence quintuple representation is simplified to sequence triple representation, which can be decoded to a relatively simple 3D topology. In the case of 2D-packing topology, sequence pair, which can be simplified from sequence triple, is used to represent a general 2D packing in this research. As an example, the coding from Figure 3 to Figure 6 can be gotten by using the coding-decoding transition method in Figure 4 and Figure 5, which are based on North-South and West-East relation corresponding to the order of modules in sequence pair as follows.

In order to get a positive sequence Γ+, each West-South corner of module connects to the West-South corner of the whole layout, and each East-North corner of module connects to the East-North corner of the whole layout without any intersection as shown in Figure 4. We can get a sequence (m_3,m_2,m_4,m_1,m_5) corresponding to (0,1,2,3,4) from left side (i.e. two red points in Figure 4) to right side (i.e. two blue points in Figure 4). The positive sequence Γ+ is changed from the West-North corner to the East-South corner, i.e. from 0 to n-1 in Γ+, where n is the total number of modules. By using this coding method, we can get the whole sequence Γ+ as shown in Figure 5, which is Γ+ (m_3,m_2,m_4,m_1,m_5). Similarly, in order to get Γ-, each West-North corner of module connects to the West-North corner of the whole layout, and each East-South corner of module connects to the East-South corner of the whole layout

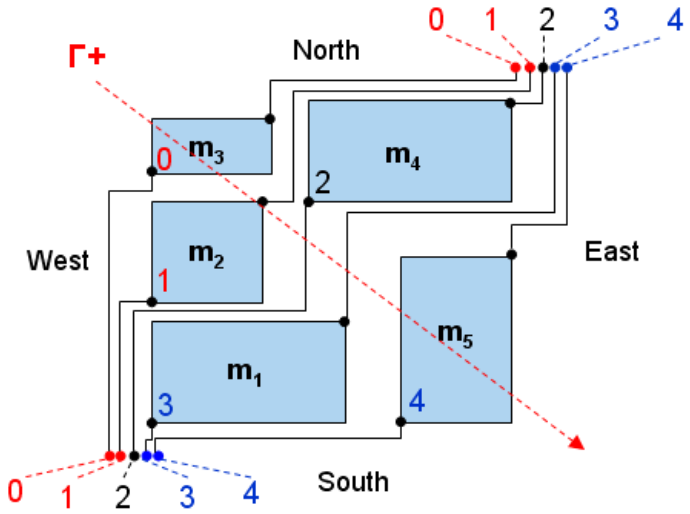

Figure 4. Coding-decoding transition method to get Γ+ (m_3,m_2,m_4,m_1,m_5) using the West-South and East-North corners of module to connect with those corners of layout

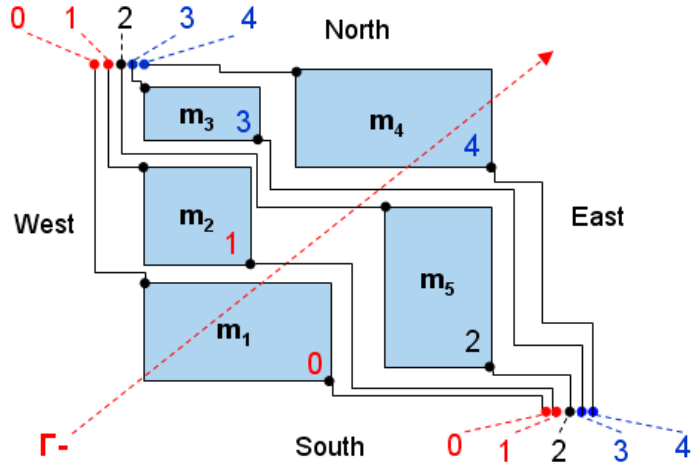

Figure 5. Coding-decoding transition method to get Γ- (m_1,m_2, m_5,m_3,m_4) using the West-North and East-South corners of module to connect with those corners of layout

without any intersection. As shown in Figure 5, we can get (m_1,m_2, m_5,m_3,m_4) as Γ- using the same method to get Γ+. The negative sequence Γ- is changed from the West-South corner to the East-North corner, i.e. from 0 to n-1 in Γ-. So far, we get the coding of a general 2D-packing topology based on North-South and West-East relation corresponding to the order of modules in sequence pair, as shown in Figure 6.

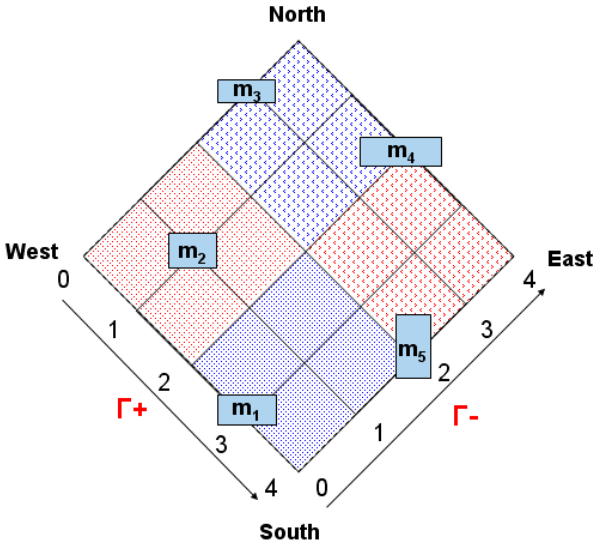

Figure 6. Intuitive image using slant grid (0,1,2,3,4) for Γ^+ and Γ^- to explain the relation between solution representation and 2D-packing topology

Generally, a 3D-packing topology is defined by the orthogonal coordinate system (x, y, z) in sequence triple and sequence quintuple representations. It is regarded as a set of the relations of relative location between boxes, i.e. "Top-Bottom", "North-South" and "West-East" (TB-, NS- and WE-) relations. Let (m_i T m_j) denote that m_i is on the top of m_j. Similarly, (m_i N m_j) and (m_i W m_j) denote NS- and WE-relations. The rule of symmetry to be followed is that (m_i T m_j) is the same relation as (m_j B m_i). That is to say, the topology should be reversely decoded if the order of labeling is reversed.

We define the notation of sequence pair, sequence triple and sequence quintuple as follows. Let ($\Gamma^i[0]$, $\Gamma^i[1]$, ..., $\Gamma^i[n\text{-}1]$) be the components of Γ^i. Let $F^i(m_j)$ be the order of m_j in sequences Γ^i. For example, if $\Gamma^i[l]$ is m_j, then $F^i(m_j) = l$. So the order of m_j can be represented by ($F^1(m_j)$, $F^2(m_j)$, ...). In general, let A+B be the sequence which is the concatenation of A and B, and A-B be the sequence obtained from A by removing all the elements in B, where A and B are sequences. Let us denote $A[i, j]$, where $i<j$, as the sequence ($A[i]$, $A[i+1]$, ..., $A[j]$), where A = ($A[0]$, $A[1]$, ..., $A[n\text{-}1]$).

In case of sequence pair, the two sequences generate a finite solution space which includes at least one optimal solution of 2D packing for area optimization by decoding. Sequence pair defines (m_i W m_j) when

$$F^1(m_i) < F^1(m_j) \text{ and } F^2(m_i) < F^2(m_j)$$

It defines (m_i N m_j) when

$$F^1(m_i) < F^1(m_j) \text{ and } F^2(m_i) > F^2(m_j)$$

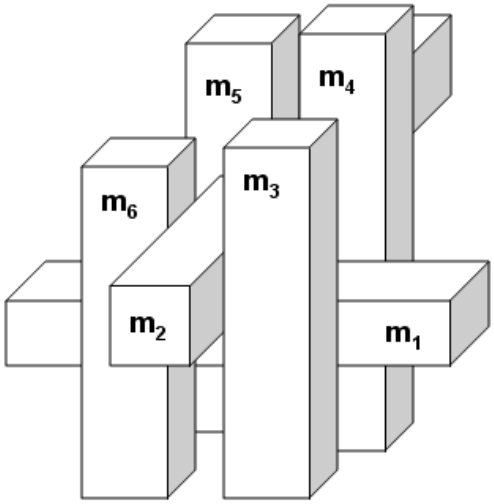

Figure 7. A 3D-packing topology decoded from sequence quintuple

For a given packing with n modules, the solution space is $(n!)^2$. If the rotation of the module is not fixed, then the solution space will increase to $(n!)^2 2^n$.

In case of sequence triple, it consists of three independent sequences Γ^i, where $1 \le i \le 3$. The coding and the decoding are based on TB-, NS- and WE- relation corresponding to the order of modules. sequence triple defines (m_i W m_j) when

$$F^2(m_i) > F^2(m_j) \text{ and } F^3(m_i) < F^3(m_j).$$

It defines (m_i N m_j) when

$$F^1(m_i) < F^1(m_j),\ F^2(m_i) < F^2(m_j) \text{ and } F^3(m_i) < F^3(m_j)$$

It defines (m_i T m_j) when

$$F^1(m_i) < F^1(m_j),\ F^2(m_i) > F^2(m_j) \text{ and } F^3(m_i) > F^3(m_j)$$

For a given packing with n modules, the solution space is $(n!)^3$. If the rotation of the module is not fixed, then the solution space will increase to $(n!)^3 2^{3n}$.

However, sequence triple does not cover all kinds of topology of 3D packing. As shown in Figure 7, (m_4 N m_1) and (m_1 N m_6) lead to (m_4 N m_6). At the same time, (m_6 W m_2) and (m_2 W m_4) lead to (m_4 W m_6). The pair (m_4, m_6) is conflicting with the rule of uniqueness, i.e. each pair of modules should be assigned with a unique topology. That means the packing can not be represented by sequence triple. As a result, the topology with the minimum volume might not be covered by sequence triple.

In case of sequence quintuple, it consists of five sequences Γ^i, where $1 \le i \le 5$. Sequence quintuple generates a finite solution space which includes at least one optimal solution of 3D packing for volume optimization by decoding. sequence quintuple defines (m_i W m_j) when

$$F^1(m_i) < F^1(m_j) \text{ and } F^2(m_i) < F^2(m_j)$$

It defines (m_i N m_j) when

$$F^3(m_i) < F^3(m_j) \text{ and } F^4(m_i) < F^4(m_j)$$

It defines (m_i T m_j) when

$$F^5(m_i) < F^5(m_j)$$

where m_i and m_j is overlapping in the projected xy-plane after WE- and NS- decoding. For a given packing with n modules, the solution space is $(n!)^5$. If the rotation of the module is not fixed, then the solution space will increase to $(n!)^5 2^{3n}$.

6. Moving methods for SA, 2-stage SA and MSA

According to section 5.2, we design the following moving methods. First of all, three basic moving methods, which are named by rotation, exchange and move, are used in the focusing search. Based on the basic methods, the group rotation and the group exchange are also used as two of moving methods in the rough search. They are repeatedly used the rotation and exchange operator with a given number. The groups are randomly selected modules for rotation or pairs of modules for exchange.

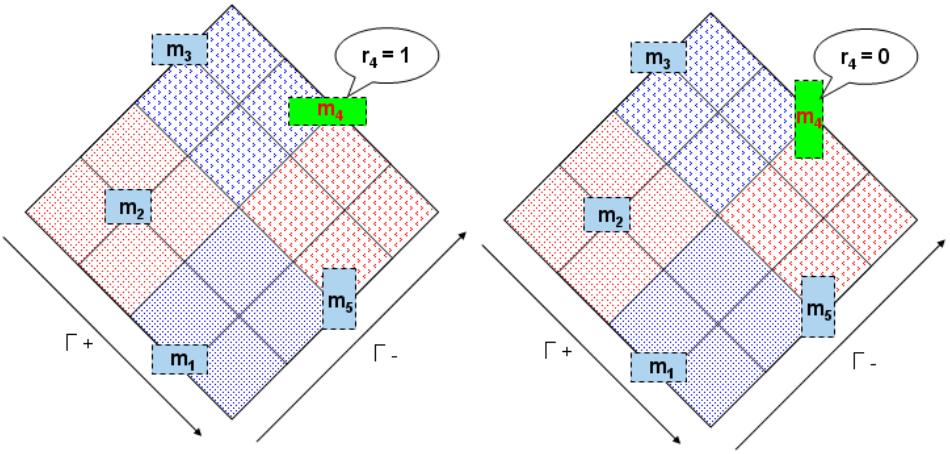

Figure 8. An example of layout before and after "rotation" in focusing search

In detail, the rotation changes the orientation of a module. When a rotation is applied to module m_i, r_i is changed to 1 - r_i. As an example shown in Figure 8, if a rotation is applied to module m_4, r_4 is changed to 1 –r_4. With respect to 3D packing, r_i is randomly selected from $r_{x\text{-}i}$, $r_{y\text{-}i}$, and $r_{z\text{-}i}$.

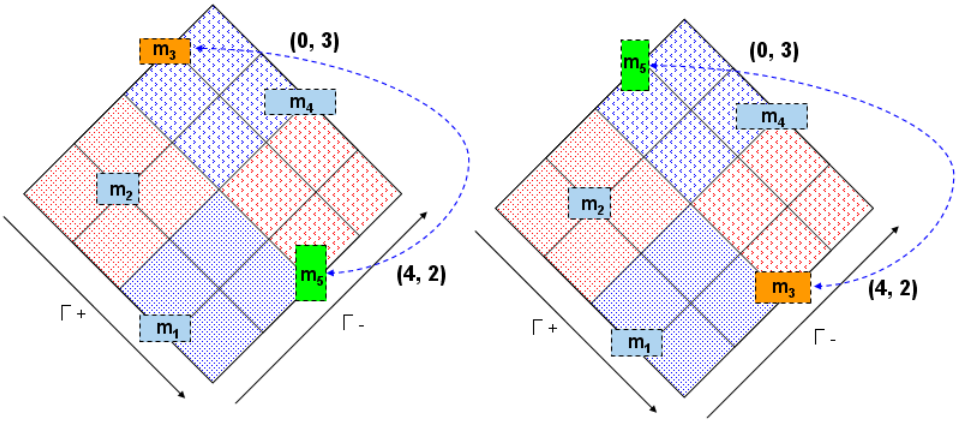

Figure 9. An example of layout before and after "exchange" in focusing search

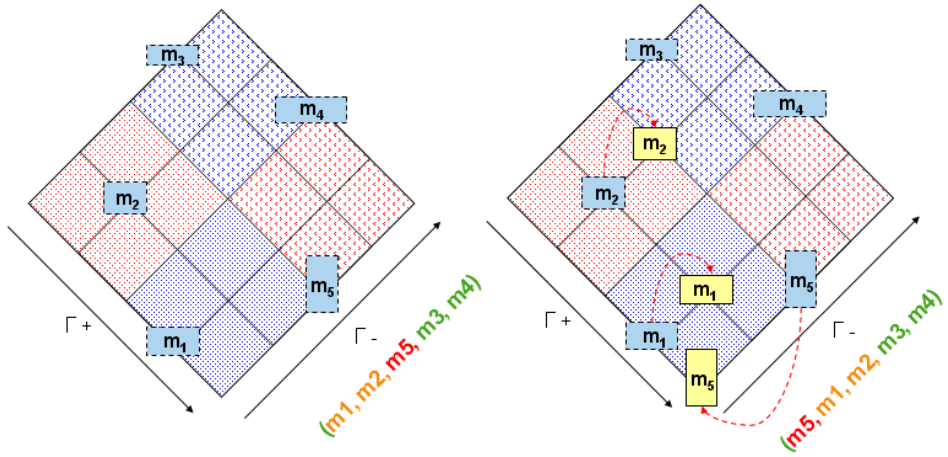

Figure 10. An example of layout before and after "move" in focusing search

The exchange moving method exchanges the order of two modules in Γ^i, where Γ^i corresponds to all sequences, i.e. the sequence pair (Γ^+, Γ^-), the sequence triple (Γ^1, Γ^2, Γ^3) or the sequence quintuple (Γ^1, Γ^2, Γ^3, Γ^4, Γ^5). When an exchange is applied to module m_i and m_j with sequence triple representation, $F^1(m_i)$, $F^2(m_i)$, $F^3(m_i)$, $F^1(m_j)$, $F^2(m_j)$ and $F^3(m_j)$ are changed to $F^1(m_j)$, $F^2(m_j)$, $F^3(m_j)$, $F^1(m_i)$, $F^2(m_i)$and $F^3(m_i)$, respectively. In the case of sequence pair, $F_+(m_i)$, $F_-(m_i)$, $F_+(m_j)$, and $F_-(m_j)$ are changed to $F_+(m_j)$, $F_-(m_j)$, $F_+(m_i)$, and $F_-(m_i)$,

respectively. For example, if the modules m_3 and the module m_5 are operated by the exchange in Figure 9, then ($F_+(m_5)$, $F_-(m_5)$) is changed from (4, 2) to (0, 3), and ($F_+(m_3)$, $F_-(m_3)$) is changed from (0, 3) to (4, 2).

The move changes the order of a module in Γ^i. When a move is applied to module m_i in Γ^i, $F^i(m_i)$ is changed to another value, say j, and the orders of modules whose order is between $F^i(m_i)$ and j are shifted accordingly. For example, , if the operation is to move m_5 to $F_-(m_5) = 0$ in Γ^-, the move will lead to $F_-(m_1) = F_-(m_1) + 1$ and $F_-(m_2) = F_-(m_2) + 1$, i.e.$\Gamma^-(m_1, m_2, m_5, m_3, m_4)$ is changed to $\Gamma^-(m_5, m_1, m_2, m_3, m_4)$ as shown in Figure 10.

7. Crossover operator for MSA

Besides, a special crossover operator is designed to generate a new solution from the current solution and the best solution so far in the rough search based on the representation in section 5.2. The margin and centre of the new solution (child) inherit the margin of the current solution (father) and the reversed centre of the best solution (mother), respectively. The reason to reverse the best solution is to get a different solution even two given solutions are the same solution.

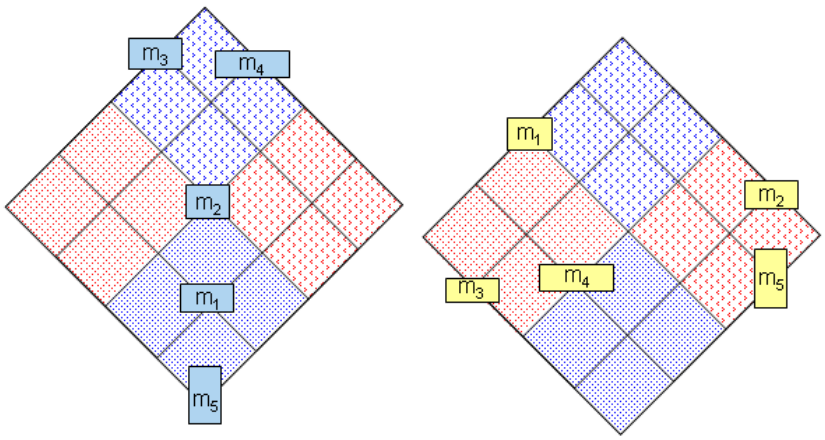

Figure 11. An example of two layouts before the crossover operator: the current layout (F: father) and the best layout so far (M: mother)

For the detail of crossover, two sequences Γ^+ and Γ^- are selected randomly from (Γ^1, Γ^2, Γ^3) or (Γ^1, Γ^2, Γ^3, Γ^4, Γ^5) for 3D packing. Let us denote the father as (Γ_f^+, Γ_f^-) which is selected from the current solution. The mother (Γ_m^+, Γ_m^-) is from the best solution so far. A number i is an integer randomly produced between 1 and $k/2$–1. The child of sequence pair (Γ_c^+, Γ_c^-) is given by $\Gamma_f^+[0, i] + \Gamma'^+_m + \Gamma_f^+[(k–i–1), k–1]$ and $\Gamma_f^-[0, i] + \Gamma'^-_m + \Gamma_f^-[(k–i–1), k–1]$, where Γ'^+_m and Γ'^-_m are the inverse of $\Gamma_m^+ – \Gamma_f^+[0, i] – \Gamma_f^+[(k–i–1), k–1]$ and the inverse of $\Gamma_m^- – \Gamma_f^-[0, i] – \Gamma_f^-[(k–i–1), k–1]$, respectively.

To make it clearer, let us take an example to explain the crossover operator. As shown in Figure 11, the left layout is represented by $\Gamma^{+}(m_3,m_4,m_2,m_1,m_5)$ and $\Gamma^{-}(m_5,m_1,m_2,m_3,m_4)$ as the father, which is the capital "F" in Figure 12. The right one is $\Gamma^{+}(m_1,m_3,m_4,m_2,m_5)$ and $\Gamma^{-}(m_3,m_4,m_1,m_5,m_2)$ as the mother, which is the reversed capital "M" in Figure 12. If we assume the i be 1, the child will be the layout $\Gamma^{+}(m_3,m_2,m_4,m_1,m_5)$ and $\Gamma^{-}(m_5,m_2,m_1,m_3,m_4)$ as the right layout of Figure 12, where $\Gamma^{+}(m_3, \ldots, m_5)$ and $\Gamma^{-}(m_5, \ldots, m_4)$ are from the father as the margin of left picture of Figure 12, and $\Gamma^{+}(\ldots,m_2,m_4,m_1,\ldots)$ and $\Gamma^{-}(\ldots,m_2,m_1,m_3,\ldots)$ are from mother with an inverse order as the centre of left picture of Figure 12.

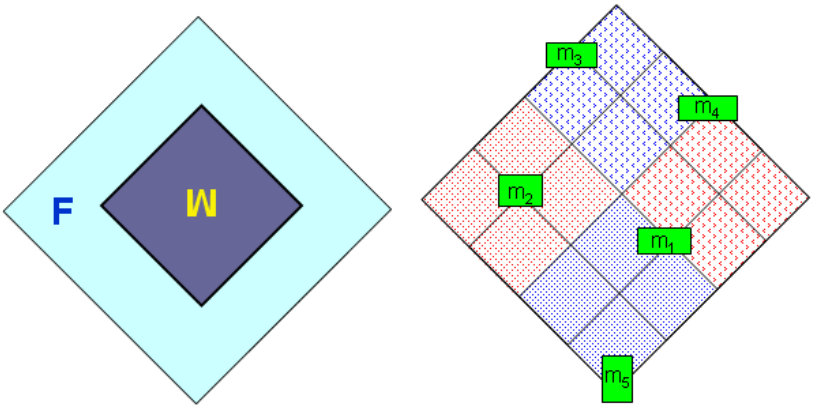

Figure 12. The layout (child) after the crossover operator between the current layout (F: father) and the best layout so far (M: mother) with an inverse order

8. Objectives and cost function

To solve multi-objective problem, we are using the total cost function, which includes area of bounding rectangle for 2D case, volume of bounding box for 3D case, interconnect power and maximal delay. Especially, the interconnect power and the maximal delay are two typical conflicting objectives, which need to experiment carefully to satisfy the requirements in real product design.

For the multi-objective optimization of 2D placement in this research, three different objectives are defined by one formula as follow.

$$C_t = \alpha \cdot C_p + \beta \cdot C_d + \gamma \cdot C_a \tag{2}$$

where $\alpha+\beta+\gamma=1$, and C_t is the total cost function, which includes the power function C_p, the delay function C_d and the area function C_a. And α, β, γ can be user-defined. As mentioned, C_p and C_d are normally conflicting in real implementation. That is to say, good C_p may lead to bad C_d, so we have to consider the trade-off between C_p and C_d using a lot of random values of α and β.

For power estimation, the dynamic power of a net n_i is proportional to $C(i)$, $V_{dd}(i)^2$, $f(i)$ and $S(i)$, where $C(i)$ is the capacitance of a net, $V_{dd}(i)$ is the voltage of power supply, $f(i)$ is the clock frequency, and $S(i)$ is switching probability of the net. Normally $C(i)$ is proportional to the length of net, so let Len_i represent its value. In case of no information, let us assume that $V_{dd}(i)$ and $f(i)$ are same for each net and $S(i)$ is randomly defined from 0 to 1. So the interconnect power is simplified as the function of Len_i and $S(i)$.

For performance estimation, the maximal delay among all nets is used. The delay is defined by the wire length of nets. To get the wire length estimation for each net, the half perimeter wire length (*HPWL*) is used for the approximation of wire length. Given any net n_i, connected with modules $\{m_1, m_2, ..., m_s\}$, *HPWL* is half perimeter of the minimum bounding box for all centres of module m_i, where i is an integer from *1* to s. In case of r_i=1, $HPWL[n_i]$ is given by $max[x_i + h_i/2] - min[x_i + h_i/2] + max[y_i + w_i/2] - min[y_i + w_i/2]$. So $HPWL[n_i]$ is gotten from (h_i, w_i, n_i), (x_i, y_i, r_i). The power and the delay are estimated so far.

For the objective of 2D packing, the area estimation is the minimum bounding rectangle including all modules, which is the total height H multiplied by the total width W. In practical implementation, we use a relative value as the cost function of area, i.e. the bounding area divided by the area of total modules, because any value with unit would not be scalable to use the experiments by diverse benchmarks.

For the objective of 3D packing, instead of 2D case, the volume estimation is given by the minimum bounding rectangular parallelepiped including all modules, which is the total height H multiplied by the total width W and multiplied by the total length L. In real implementation, the cost function is also using the relative value of volume.

9. Experiment and comparison

To evaluate the effectiveness and reliability of the proposed MSA in practice, a set of experiments was implemented, comparing with traditional SA and 2-stage SA. In the case of 2D packing and placement, we are using ami49_X and MCNC benchmarks. The ami49_X is produced by duplicating ami49 circuit X times. In the case of 3D packing, ami98_3D benchmark is produced by inheriting the height and width of 2D ami49_2 benchmark and randomly getting the length between the given minimum and maximum dimensions. The implementation for 3D packing is to compare MSA with the mentioned 2-stage SA. MSA is implemented in Python environment on 2.16GHz PC with 3.00GB memory. For a fair comparison, SA and 2-stage SA is also implemented at the same machine. The maximum runtime is within 14,400s (4 hours) each time.

For area optimization of 2D packing, let γ be 1 and $\alpha+\beta$ be 0 in the cost function. As shown in Table 1, the best, average and worst cases among 50 trials are gotten. The comparison of solution quality and runtime between SA and MSA is gotten. MSA reduced near 21% runtime with better solution quality. As shown in Table 2, a near log-linear trend of average improvement rates from SA to MSA is gotten. That means MSA should be more suitable for the placement with a larger number of modules.

For interconnect optimization of 2D placement, let γ be 0 and $\alpha+\beta$ be 1 in the cost function. The experiment is using ami49_X benchmarks. To get the figures, α is randomly produced from 0.1 to 0.9. 240 solutions are tested for comparison. For all tested ami49_X with X from 1 to 12, block number from 49 to 588, and net number from 408 to 4896, the improved results are gotten. Figure 13 shows that MSA obtains at least 13% Pareto improvement with the constraint of less than 108.2% maximal delay. To get the worst cases, we tested more 120 solutions with α given by 0.1, 0.3, 0.5, 0.7, and 0.9. As shown in Figure 14, MSA got near 6% worst-case mitigation on average for the interconnect power with no degradation of maximal delay.

Benchmarks	**Best (mm^2)**	**Average (mm^2)**	**Worst (mm^2)**	**Runtime (s)**
apte	47.08	47.36	47.67	3.2
xerox	19.80	20.50	21.21	1.5
hp	9.03	9.17	9.34	2.3
ami33	1.18	1.23	1.29	17
ami49	36.91	37.79	38.83	37
ami49_2	73.58	75.48	77.38	142
ami49_4	147.3	151.1	155.8	547

Table 1. Area optimization by MSA for 2D packing

Benchmarks	**Solution(mm^2)**		**Runtime (s)**		**Improvement (%)**	
	SA	MSA	SA	MSA	**Solution**	**Runtime**
apte	47.38	47.36	4.1	3.2	0.04%	22%
xerox	20.51	20.50	1.9	1.5	0.05%	21%
hp	9.18	9.17	2.7	2.3	0.11%	15%
ami33	1.24	1.23	22	17	0.52%	23%
ami49	37.96	37.79	45	37	0.48%	18%
ami49_2	75.98	75.48	194	142	0.71%	27%
ami49_4	152.2	151.0	720	547	0.88%	24%

Table 2. Average improvement of area for 2D packing

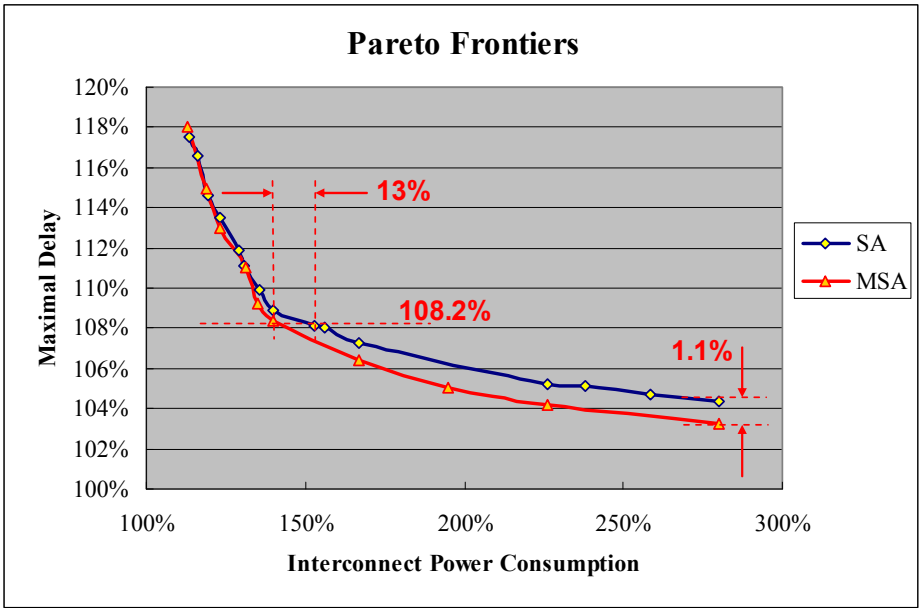

Figure 13. Pareto frontiers and its improvement by MSA for 2D placement (sequence pair)

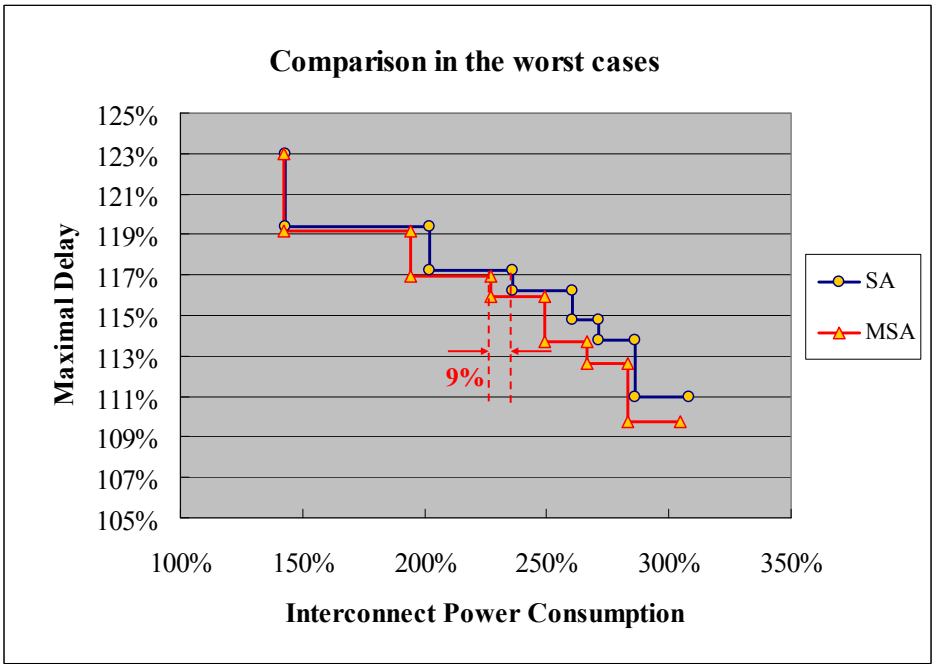

Figure 14. Worst-case mitigation by MSA for 2D placement (sequence pair)

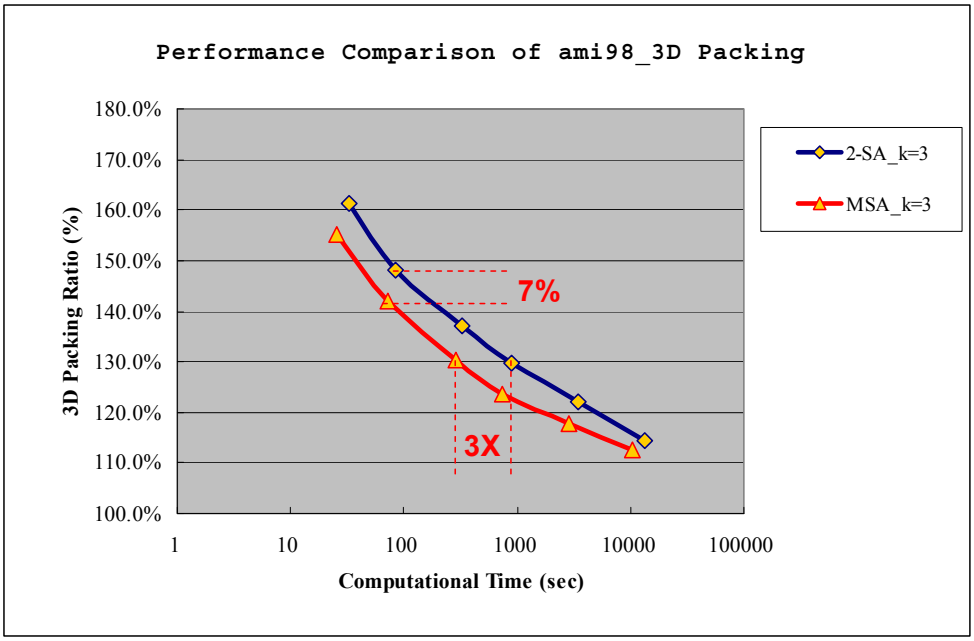

Figure 15. Performance improvement by MSA for 3D packing (sequence triple)

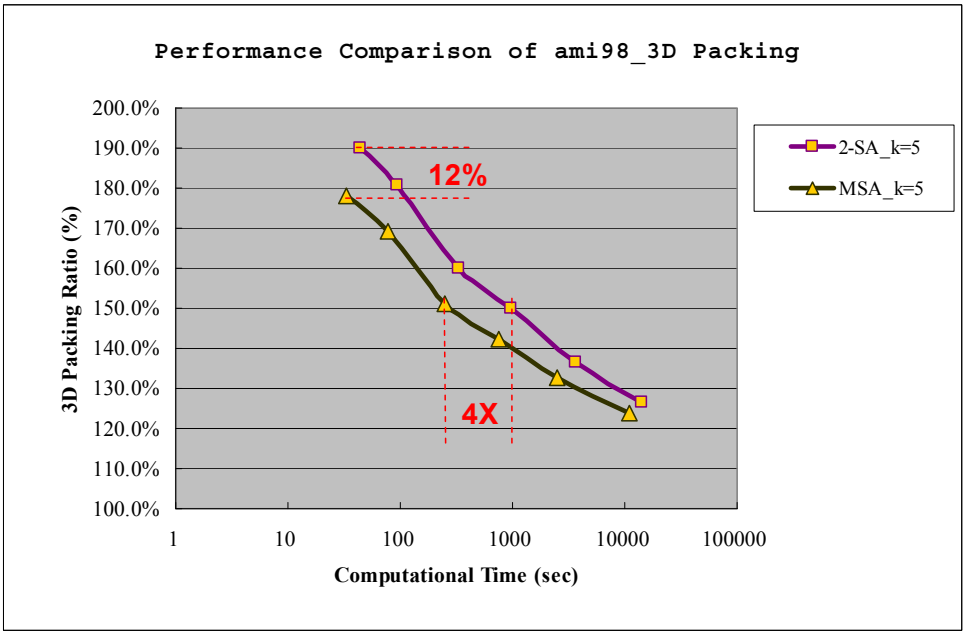

Figure 16. Performance improvement by MSA for 3D packing (sequence quintuple)

For volume optimization of 3D packing, we compare the computational performance of volume ratio using 2-stage SA and MSA. The results show considerable improvement from 2-stage SA to MSA. The improvement of packing ratio is between 2% and 7% for sequence triple representation. The improvement for runtime is up to 3 times, as shown in Figure 15. With regard to sequence quintuple representation, the experiment also shows the improvement from 2-stage SA to MSA. The improvement of packing ratio is between 3% and 12%. The improvement for runtime is up to 4 times with the sequence quintuple representation, as shown in Figure 16. The packing ratio of volume is improved by near 7% with less than 100s runtime, if we select MSA with sequence triple representation, instead of 2-stage SA with the same representation. The packing ratio of volume is improved by near 12% with less than 100s runtime, if we select MSA with sequence quintuple representation, instead of 2-stage SA with the same representation. In short, the overall solution quality and the runtime of MSA algorithm are better than these of 2-stage SA algorithm.

10. Conclusion and future work

In summary, the optimization techniques for integrated circuit (IC) layout design with large solution space are facing big challenges to get better solution quality with less runtime. In this research, a new simulated annealing based approach, named mixed simulated annealing (MSA), is proposed to solve three typical layout design problems, which are 2D packing, 2D placement and 3D packing, by using sequence pair, sequence triple and sequence quintuple representations. A new crossover operator is designed to reuse the information of past solutions and get high improving efficiency. Based on experiment, MSA improved both the best and the worst cases of 2D placement for interconnect power and maximal delay. For area minimization, MSA reduced computational runtime with the better solution quality, and a near log-linear trend of average improvement rates from SA to MSA is gotten for both solution quality and runtime. The overall quality of packing by MSA is normally better than the published results. For the volume minimization of 3D packing, MSA improved the solution quality (up to 12% better) and the computational time (up to 4 times faster). For the future work, the proposed MSA has potential to be extended to more general problems, such as 2D/3D packing or placement with rectilinear boxes.

Author details

Yiqiang Sheng and Atsushi Takahashi
Department of Communications and Integrated Systems, Graduate School of Science and Engineering, Tokyo Institute of Technology, Tokyo, Japan

Acknowledgement

The authors would like to thank Prof. Kunihiro Fujiyoshi at Tokyo University of Agriculture and Technology, to thank Prof. Shuichi Ueno, Dr. Tayu, Mr. Shinoda, Mr. Inoue and Mr.

Zhao at Tokyo Institute of Technology, and to thank Mr. Yamada, Mr. Ando and Mr. Ukon at Osaka University for their discussion, comment and support.

11. References

[1] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pages 671–680, 1983.

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

[3] L. Ingber, "Simulated Annealing: Practice versus Theory," *Mathematical Computer Modelling*, vol.18, no.11, pages 29–57, 1993.

[4] J. Varanelli and J. Cohoon, "A two-stage simulated annealing methodology," *Fifth Great Lakes Symposium on VLSI*, pages 50-53, 1995.

[5] Y. Sheng, A. Takahashi and S. Ueno, "A Stochastic Optimization Method to Solve General Placement Problem Effectively," *Proceedings of Information Processing Society of Japan (IPSJ) DA Symposium*, vol. 2011, no.5, pages 27-32, 2011.

[6] Y. Sheng, A. Takahashi and S. Ueno, "RRA-Based Multi-Objective Optimization to Mitigate the Worst Cases of Placement," *The IEEE 9th International Conference on ASIC*, pages 357-360, 2011.

[7] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," *Proceedings of International Conference on CAD*, pages 484-491, 1996.

[8] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pages 1518-1524, 1996.

[9] X. Tang and D. F. Wong, "FAST-SP: a fast algorithm for block placement based on sequence pair," *Proceedings of IEEE Asia South Pacific Design Automation Conference*, pages 521-526, 2001.

[10] C. Zhuang, K. Sakanushi, L. Jin and Y. Kajitani, "An enhanced Q-sequence augmented with empty-room-insertion and parenthesis trees," *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pages 61-68, 2002.

[11] C. Kodama and K. Fujiyoshi, "Selected sequence-pair: an efficient decodable packing representation in linear time using sequence-pair," *Proceedings of IEEE Asia South Pacific Design Automation Conference*, pages 331-337, 2003.

[12] H. Yamazaki, K. Sakanushi, S. Nakatake and Y. Kajitani, "The 3D-Packing by Meta Data Structure and Packing Heuristics," *IEICE Transaction on Fundamentals of Electronics*, Communications and Computer Sciences, vol. E83-A, no. 4, pages 639-645, 2000.

[13] Y. Sheng, A. Takahashi and S. Ueno, "An Improved Simulated Annealing for 3D Packing with Sequence Triple and Quintuple Representations," *IEICE Technical Report on VLSI Design Technologies*, Vol.111, No.324, pp.209-214, 2011.

[14] Y. Sheng, A. Takahashi and S. Ueno, "2-Stage Simulated Annealing with Crossover Operator for 3D-Packing Volume Minimization," *Proceedings of the 17th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, pages 227-232, 2012.

INTECH