2-Stage Simulated Annealing with Crossover Operator for 3D-Packing Volume Minimization

Yiqiang Sheng[#1], Atsushi Takahashi[*2], Shuichi Ueno[#3]

[#] *Department of Communication and Integrated Systems*
Tokyo Institute of Technology, Tokyo, 152-8852, Japan
[1] sheng@lab.ss.titech.ac.jp
[3] ueno@lab.ss.titech.ac.jp

[*] *Division of Electrical, Electronic and Information Engineering*
Osaka University, Suita-shi, 565-0871, Japan
[2] atsushi@si.eei.eng.osaka-u.ac.jp

***Abstract*—The 3D packing for VLSI physical design is facing big challenges to get better solution quality with less computational time. In this paper, we propose 2-stage simulated annealing with crossover operator (2-SA-X) to solve a general rectangular 3D-packing problem by using sequence-*k*-tuple representation, where *k* is defined by 3 and 5. The basic ideas of this research are to reuse the information of past solution by integrating the crossover operator from genetic algorithm and to improve the global search ability by using two different stages. The first stage mainly focuses on the global search by moving methods with big changes, including the crossover, while the second stage focuses on local search by the moving methods with small changes. Based on the experiment using ami98_3D benchmark, the computational performance of 3D packing is considerably improved. The paper shows how much the 3D-packing ratio of volume and the computational time can be improved by using the proposed 2-SA-X algorithm, comparing with normal 2-stage simulated annealing (2-SA) without the crossover operator.**

***Keywords*—3D packing, 2-stage simulated annealing, sequence-*k*-tuple representation, VLSI physical design, CAD technique**

I. INTRODUCTION

Simulated annealing (SA) algorithm [1] is an important tool to solve design automation problems in VLSI computer-aided design, but there are several significant disadvantages of traditional SA, such as its slow search speed. In [2], James M. Varanelli etc proposed 2-stage SA to speed up traditional SA for VLSI CAD. Based on the mentioned 2-stage SA, the main consideration of this paper is to use the crossover operator from genetic algorithm to speed up 2-stage SA algorithm further to solve 3D packing problem with sequence-*k*-tuple representation [3].

The 3D-packing technique has many meaningful applications on VLSI/PCB design [4] and logistics management. A general packing problem in VLSI/PCB physical design is to position different modules into a fixed shape, normally rectangular box, with 3D volume minimization. As VLSI design keeps going through higher complexity and IC manufacturing technology continues to scale to smaller dimensions, 3D packing problem is becoming increasingly important nowadays. At the same time, 3D-packing technique is facing big challenges to get better solution quality with less computational time. Since a general packing problem is NP-hard with infinite solution space, there is no exact algorithm to get optimal solutions in practice so far. As an alternative to get the optimal solutions, heuristic approaches [5-10] are used to find near optimal solutions. SA [1][2][3][5] is one of the most popular heuristic algorithms for 2D and 3D placement. Many researches explored different representations for 2D placement, such as BSG[11], SP[12], O-tree[13], B*-tree[14], CBL[15], FAST-SP[16], Q-sequence[17], Selected SP[18] etc. The SP representation for 2D packing can be extended to sequence-*k*-tuple representations to code and decode the 3D-packing problem. It is proved [3] that sequence triple (k=3) could represent the topology of tractable 3D packing and there are at least one sequence quintuple (k=5) which can be decoded to a topology as an optimal 3D packing for volume minimization.

The main contributions of this paper are as follows. (1) In this paper, we propose a new algorithm, which is 2-stage simulated annealing with crossover operator (2-SA-X). Also we show how to solve a general rectangular 3D packing problem with sequence-*k*-tuple representation, where k can be 3 or 5, by using normal 2-stage simulated annealing (2-SA) and the proposed 2-SA-X methodology. (2) By comparing the computational performance of 3D-packing with two different representations (i.e. k=3 and k=5) for two different heuristics (i.e. 2-SA and 2-SA-X), this paper shows how to select a proper heuristic with a proper representation effectively. (3) By using ami98_3D benchmark, the paper shows how much the packing ratio of volume and the computational time can be improved by using the new algorithm.

The rest of the paper is organized as follows. Section II explains the problem formulation. Section III describes the sequence-*k*-tuple representation. Section IV shows the detail on how to solve 3D packing problem. Section V presents the experimental data and the comparison results. Finally, Section VI concludes this paper.

II. PROBLEM FORMULATION

A general 3D rectangular packing problem can be formulated as follows. 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 $1 \leq i \leq n$, has height h_i, length l_i and width w_i. Let $(x_i, y_i, z_i, r_{xy-i}, r_{yz-i}, r_{zx-i})$ be the location and rotation on 3D orthogonal coordinate system for each module m_i, $1 \leq i \leq k$, where (x_i, y_i, z_i) means the coordinates of the bottom-south-west corner of module m_i, and $(r_{xy-i}, r_{yz-i}, r_{zx-i})$ represents the rotation (0, 1) of m_i on xy-, yz- and zx- plane as shown in Figure 1. If $r_{xy-i} = 1$, the height is the vertical length and the width is the horizontal length on xy- plane. If $r_{xy-i} = 0$, the height will be the horizontal length and the width will be the vertical length, which is rotated by 90 degree on xy- plane. In short, the input is a set of modules $M = \{m_1, m_2, ...\}$ with height, length and width $\{(h_1, l_1, w_1), (h_2, l_2, w_2), ...\}$. The constraint is no overlap between m_i and m_j, where $i \neq j$. The output is a set of location and rotation for each module $\{(x_1, y_1, z_1, r_{xy-1}, r_{yz-1}, r_{zx-1}), (x_2, y_2, z_2, r_{xy-2}, r_{yz-2}, r_{zx-2}), ...\}$. The objective is to minimize the volume of bounding box.

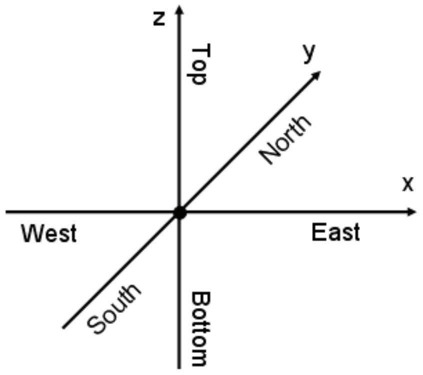

Figure 1: Orthogonal coordinate system for 3D-packing topology

III. Representation

To represent 3D packing problem, let us start with 2D packing problem, since both 2D and 3D packing representations are coding and decoding methods to get finite solution space, instead of the infinite solution space of the original problem. The sequence pair (SP) representation [12] can be used to represent a general 2D packing problem. The SP representation can be extended from 2D (k=2) to 3D as a more general sequence-k-tuple representation [2], where k can be 3 or 5.

The sequence-k-tuple representation defines the orthogonal coordinate system (x, y, z) for 3D-packing topology, which can be regarded as a set of the relations of relative location between boxes, i.e. "top", "bottom", "north", "south", "east" and "west" (TB-, NS- and WE-) relations as shown in Figure 1. For example, as shown in Figure 2, box m_2 is on the west of box m_3, 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, and box m_2 is on the top of box m_1.

In case of k=2, the sequence-k-tuple representation consists of two sequences Γ^+ and Γ^-. Let $(\Gamma^+[0], \Gamma^+[1], ..., \Gamma^+[k-1])$ and $(\Gamma^-[0], \Gamma^-[1], ..., \Gamma^-[k-1])$ be the components of Γ^+ and Γ^-, respectively. Let $f_+(m_j)$ and $f_-(m_j)$ be the order of m_j in sequences Γ^+ and Γ^-, respectively. The two sequences Γ^+ and Γ^- generate a finite solution space which includes at least one optimal solution of 2D packing for area optimization by decoding. The coding and the decoding are based on TB- and WE- relation corresponding to the order of modules in sequence pair. For a given packing with n modules, the solution space is $(n!)^2$ for k=2. If the rotation of the module is not fixed, then the solution space will increase to $(n!)^2 2^n$.

In case of k=3, the sequence-k-tuple representation is called sequence triple, which consists of three different sequences Γ^i, where $1 \leq i \leq 3$. Let $(\Gamma^1[0], \Gamma^1[1], ..., \Gamma^1[n-1])$, $(\Gamma^2[0], \Gamma^2[1], ..., \Gamma^2[n-1])$ and $(\Gamma^3[0], \Gamma^3[1], ..., \Gamma^3[n-1])$ be the components of Γ^1, Γ^2 and Γ^3. Let $F^1(m_j)$, $F^2(m_j)$ and $F^3(m_j)$ be the order of m_j in sequences Γ^1, Γ^2 and Γ^3. For example, the packing in Figure 2 can be represented by $\Gamma^1(m_2,m_1,m_3)$, $\Gamma^2(m_1,m_3,m_2)$ and $\Gamma^3(m_1,m_2,m_3)$. The coding and the decoding are based on TB-, NS- and WE- relation corresponding to the order of modules in sequence triple. For a given packing with n modules, the solution space is $(n!)^3$ for k=3. If the rotation of the module is not fixed, then the solution space will increase to $(n!)^3 2^{3n}$.

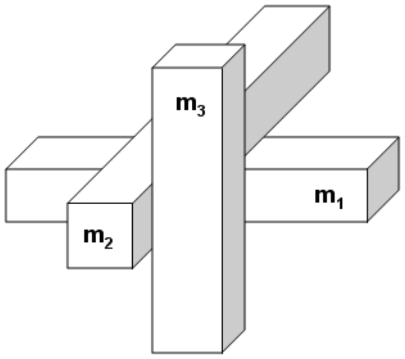

Figure 2: A 3D-packing topology that sequence triple is enough

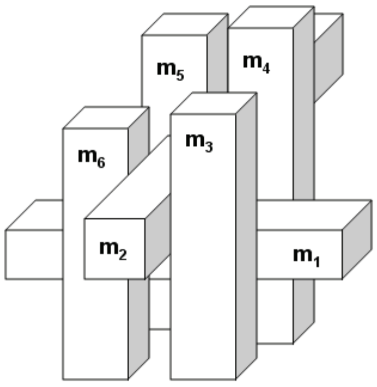

Figure 3: A 3D-packing topology that sequence triple is not enough

In case of k=5, the sequence-k-tuple representation is called sequence quintuple, which consists of five different sequences Γ^i, where $1 \leq i \leq 5$, i.e. Γ^1, Γ^2, Γ^3, Γ^4 and Γ^5. The sequence quintuple has larger solution space than the sequence triple. The reason why we need sequence quintuple is as follow. As shown in Figure 3, it is an example of 3D-packing topology that can not be represented by the sequence triple, i.e. k=3 are not enough. It means the sequence triple does not cover all kinds of topology of 3D packing. That is to say, the topology with the minimum volume might not be covered. That is the reason why the representation needs to be extended to a larger system, named the sequence quintuple. The five sequences generate a finite solution space which includes at least one optimal solution [3] of 3D packing for volume optimization by decoding. For example, the packing in Figure 3 can be represented by $\Gamma^1(m_1, m_5, m_6, m_2, m_4, m_3)$, $\Gamma^2(m_6, m_5, m_2, m_3, m_4, m_1)$, $\Gamma^3(m_2, m_4, m_5, m_1, m_3, m_6)$, $\Gamma^4(m_5, m_4, m_1, m_6, m_3, m_2)$ and $\Gamma^5(m_6, m_5, m_4, m_3, m_2, m_1)$. The coding and the decoding are based on TB-, NS- and WE- relation corresponding to the order of modules in the sequence quintuple. For a given packing with n modules, the solution space is $(n!)^5$ for k=5. If the rotation of the module is not fixed, then the solution space will increase to $(n!)^5 2^{3n}$.

For a general sequence-k-tuple representation, where k=2, 3 or 5, let us define notations as follows. Let ($\Gamma^i[0]$, Γ^i [1], ..., Γ^i [n-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 = ($A[0]$, $A[1]$, ..., $A[n-1]$).

IV. 2-Stage Simulated Annealing without and with Crossover

A. Simulated Annealing (SA)

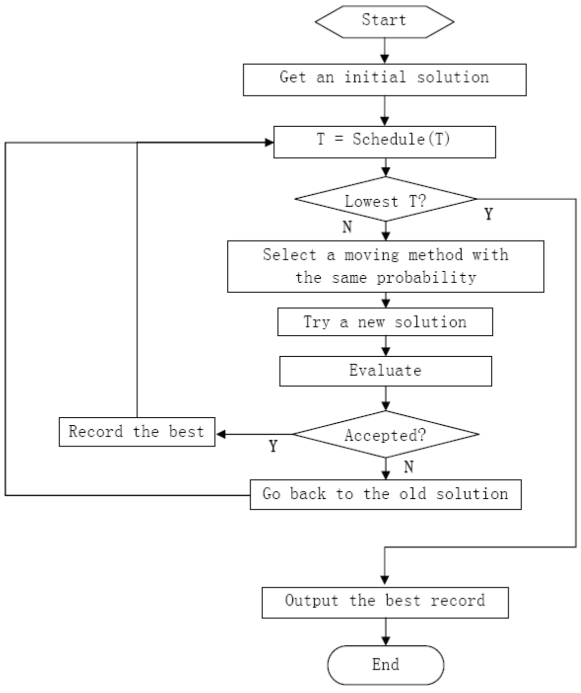

Figure 4: Flow chart of traditional SA

Simulated annealing (SA) [1] is one of the most popular heuristics in the field of VLSI/PCB physical design. In [3], H. Yamazaki etc. use the traditional SA to solve 3-packing problem. The basic flow chart of traditional SA is as shown in Figure 4. The input is a set of module with the length, the width and height. An initial solution is randomly produced. The key point of SA is the temperature scheduling. 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. After temperature scheduling, a moving method is selected with the same probability, i.e. near 33.3% for three different moving methods. And then a new solution is tried by using the selected moving method. Then the new solution is evaluated and compared with the old solution. And the solution will be accepted with a calculated acceptance probability $P = exp[-\Delta C/T]$ according to the evaluation and the current temperature. If the solution is improved, the new solution will be compared with the best solution so far to decide if updating the best record or not. If non-improved and rejected, the current solution goes back to the old solution and continues the next temperature scheduling until reaching the lowest temperature T_e. The output will be the best record.

B. 2-Stage Simulated Annealing (2-SA)

Some researches extend the traditional SA to 2-stage simulated annealing (2-SA). In 2-SA, the flow chart is controlled from the first stage, named rough search in this paper, to the second stage, named focusing search here, as shown in Figure 5. The 2-SA starts with the rough search using the first temperature scheduling. After reaching the lowest temperature of the first scheduling, it is changed to the focusing search and starts the second temperature scheduling. In focusing search, different parameters of temperature scheduling are used. Besides, different moving methods are selected with the same probability during both the rough search and the focusing search.

In real implementation in this paper, an initiation solution is randomly produced. The moving methods are divided into rough moving methods and focusing moving methods. The 2-SA tries a new solution by using a rough moving method at the first stage and using a focusing moving method at the second stage. And then the solution is evaluated by the cost function (C). The solution is accepted with a probability $P = exp[-\Delta C/T]$, where T is the temperature, P is between 0 and 1. If $\Delta C < 0$, then $P = 1$. The parameters of temperature scheduling are denoted as follows. The initial value and the ending value of temperature are T_0 and T_e during rough search. For focusing search, T'_0 and T'_e are used to replace T_0 and T_e. A temperature coefficient between 0 and 1 is set to control the speed of temperature reduction. An inside loop number is set to control the repeated moves for each T. In case of $\Delta C < 0$, the "Best Record Module" is implemented: When the new solution is better than the current best, the record will be updated.

C. Moving Methods of 2-SA

Different moving methods are designed to try new solutions. The rough moving methods tend to have big changes, while the focusing moving methods allow moving with small changes. For the rough search of 2-SA, two moving methods are used: group rotation and group exchange. The group rotation and group exchange are the rotation of randomly selected modules and the exchange of randomly selected pairs of modules, respectively. The rotation of a module and the exchange of a pair of modules are explained in the focusing runs.

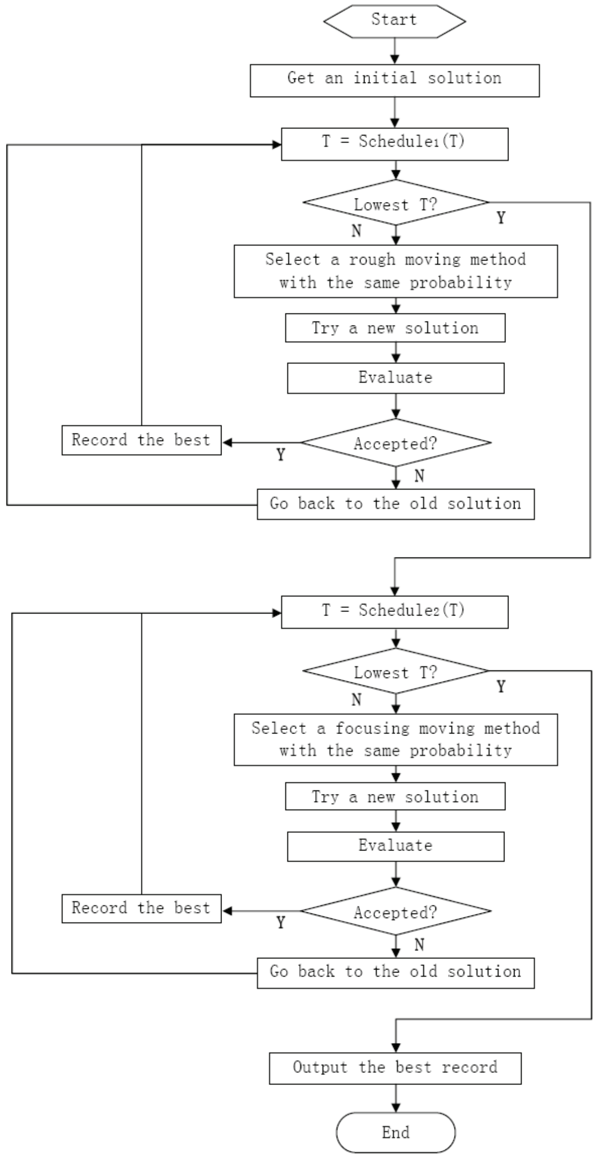

Figure 5: Flow chart of 2-SA and 2-SA-X

For the focusing search, three focusing moving methods (rotation, exchange and move) are used. The rotation moving method changes the orientation of a module. When a rotation is applied to module m_i, r_i is changed to $1 - r_i$, where r_i is randomly selected from $r_{xy\text{-}i}$, $r_{yz\text{-}i}$, and $r_{zx\text{-}i}$. The exchange moving method exchanges the order of two modules in Γ^i, where Γ^i corresponds to all sequences, i.e. the sequence triple $(\Gamma^1, \Gamma^2, \Gamma^3)$ or the sequence quintuple $(\Gamma^1, \Gamma^2, \Gamma^3, \Gamma^4, \Gamma^5)$. When an exchange is applied to module m_i and m_j with the 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. 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.

D. 2-Stage Simulated Annealing with Crossover (2-SA-X)

The basic idea to design 2-stage simulated annealing with crossover (2-SA-X) is to speed up the search and improve the global search ability by the crossover operator, which reuses the information of past solutions. The flow chart of 2-SA-X is totally same as 2-SA, while 2-SA-X is using the crossover as one of rough moving methods. The new solution is generated by the crossover between the current solution and the best solution so far. The margin and center of the new solution (child) inherit the margin of the current solution (father) and the reversed center of the best solution (mother), respectively. The reason why reverses the best solution is to get a different solution even two given solutions are same.

In details, the crossover operator works as follows. First, two sequences Γ^+ and Γ^- are selected randomly from $(\Gamma^1, \Gamma^2, \Gamma^3)$ or $(\Gamma^1, \Gamma^2, \Gamma^3, \Gamma^4, \Gamma^5)$. 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 SP (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.

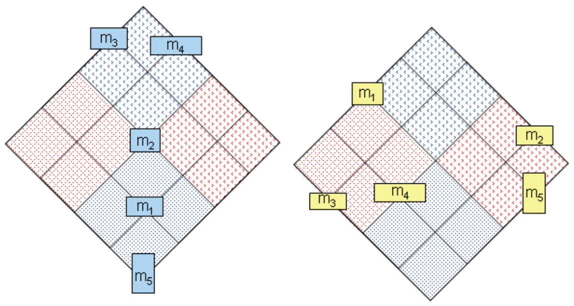

Figure 6: Crossover as a moving method: Father and Mother

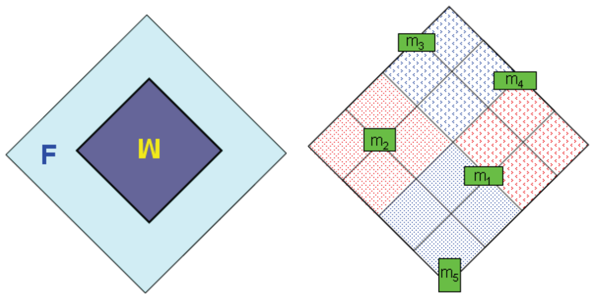

Figure 7: Crossover as a moving method: Child

To make it clearer, let us take an example to explain the crossover moving method. The father is $\Gamma^+(m_3,m_4,m_2,m_1,m_5)$ and $\Gamma^-(m_5,m_1,m_2,m_3,m_4)$ as shown in Figure 6. The mother 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 shown in Figure 6. 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 $\Gamma^{-}(m_5,m_2,m_1,m_3,m_4)$, where $\Gamma^{+}(m_3, ..., m_5)$ and $\Gamma^{-}(m_5, ..., m_4)$ are from the father as the margin, and $\Gamma^{+}(...,m_2,m_4,m_1,...)$ and $\Gamma^{-}(...,m_2,m_1,m_3,...)$ are from mother with an inverse order as the centre as shown in Figure 7.

Since 2-SA-X shares the same structure with 2-SA, it is an iterative improvement method and a stochastic algorithm with two basic stages. 2-SA-X inherits the advantage of genetic algorithm to reuse the good configuration of past solutions. The detailed comparison among SA, 2-SA and 2-SA-X is shown in Table I. It shows several intuitive advantage of the proposed 2-SA-X methodology comparing with traditional SA and 2-SA.

TABLE I
COMPARISON AMONG SA, 2-SA AND 2-SA-X

Comparison		**SA**	**2-SA**	**2-SA-X**
Informational reuse	Past solutions	No	No	Yes
	Past moves	No	No	No
Search within a short runtime	Global	Bad	Good	Even Better
	Local	Good	Good	Good
Local optimum	Approach	Easy	Easy	Easy
	Escape	Slow	*Fast /Slow	*Fast /Slow

*Fast/Slow means that it is fast to escape local optimum during high temperature, while it is slow to escape local optimum during low temperature.

E. Cost Function and Parameter Setting

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 the real implementation, the cost function (C) is using the relative value of volume, which is the volume of the minimum bounding box divided by the total volume of all modules.

The parameters of temperature scheduling depend on the requirement of computational time. As a reference, the parameters used in experiments can be set as follows: T_0 = 10000, T_e = 100, T'_0 = 1000, T'_e = 1, Inside loop number = 1000, Temperature coefficient = 0.98. To get the required solution quality and computational time, the parameter setting normally adjusts the temperature coefficient and the inside loop number. For example, the temperature coefficient can be closer to 1, such as 0.99, 0.995, etc. Also the inside loop number can be increased from 1000 to 2000, 5000, etc.

V. Experiment and Comparison

The ami98_3D 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. By using the ami98_3D benchmark, a set of experiments is implemented to evaluate the proposed 2-SA-X algorithm, comparing with 2-SA algorithm. Since the only difference between 2-SA-X and 2-SA is a crossover operator, any improvement from 2-SA to 2-SA-X will be due to the proposed crossover operator. The 2-SA-X algorithm is implemented in Python environment on 2.16GHz PC with 3.00GB memory. For a fair comparison, 2-SA is also implemented at the same machine. The maximum computational time is within 14,400s (4 hours) each time, which depends on the parameter setting. We are using sequence-k-tuple representation for 3D-packing problem, where k is 3 or 5.

TABLE II
IMPROVEMENT OF VOLUME OPTIMIZATION IN CASE OF K=3

2-SA (k=3)		2-SA-X (k=3)		Improvement (%)	
Packing Ratio	Runtime	Packing Ratio	Runtime	Packing Ratio	Runtime
161.2%	33	155.2%	25	6%	22%
148.1%	84	142.1%	71	6%	15%
137.1%	327	130.3%	284	7%	13%
129.7%	889	123.5%	749	6%	16%
122.1%	3475	117.7%	2818	4%	19%
114.4%	13272	112.6%	10522	2%	21%

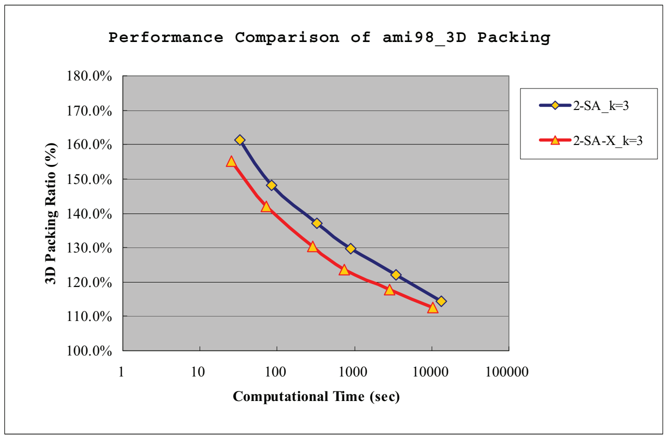

Figure 8: Performance comparison between 2-SA and 2-SA-X (k=3)

The research compares the computational performance of volume optimization for 3D packing using 2-SA and 2-SA-X with k=3 and k=5 representations. Table II shows the improvement of volume optimization from 2-SA to 2-SA-X with regard to k=3. The improvement of packing ratio is between 2% and 7%. The improvement of computational time is between 13% and 22%. It means 2-SA-X outperforms 2-SA with the sequence triple representation, as shown in Figure 8.

Table III shows the improvement of volume optimization from 2-SA to 2-SA-X with regard to k=5. The improvement of packing ratio is between 3% and 12%. The improvement of computational time is between 17% and 31%. It means that 2-SA-X also outperforms 2-SA with the sequence quintuple representation, as shown in Figure 9. That is quite meaningful information on how to select more effective heuristic with a proper representation for 3D placement or floorplanning in VLSI/PCB physical design. Also it shows how much the packing ratio of volume and the computational time can be improved by a right selection of heuristics and representations with respect to ami98_3D benchmark. For example, the packing ratio of volume can be improved by near 7% with less than 100s computational time, if we select 2-SA-X with *k*=3, instead of 2-SA with *k*=3. And the packing ratio of volume can be improved by near 12% with less than 100s computational time, if we select 2-SA-X with *k*=5, instead of 2-SA with *k*=5. Besides, it also shows how the computational performance depends on different requirements of the packing ratio and the computational time. In short, the overall solution quality and the computational time of 2-SA-X methodology are better than these of 2-SA methodology.

TABLE III
IMPROVEMENT OF VOLUME OPTIMIZATION IN CASE OF *K*=5

2-SA (k=5)		2-SA-X (k=5)		Improvement (%)	
Packing Ratio	Runtime	Packing Ratio	Runtime	Packing Ratio	Runtime
190.2%	44	178.1%	33	12%	24%
181.0%	95	169.0%	79	12%	17%
159.8%	332	151.2%	253	9%	24%
150.1%	976	142.3%	752	8%	23%
136.7%	3614	132.8%	2476	4%	31%
126.6%	14205	123.9%	10941	3%	23%

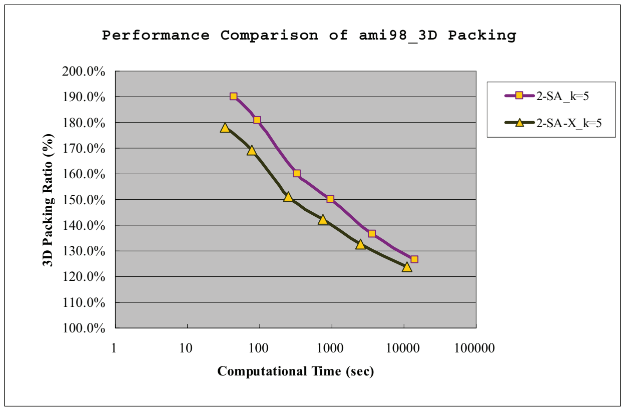

Figure 9: Performance comparison between 2-SA and 2-SA-X (*k*=5)

VI. Conclusion

In this paper, 2-stage simulated annealing with crossover operator (2-SA-X) is proposed to solve 3D-packing problem with sequence-*k*-tuple representation. 2-SA-X is designed by integrating a normal 2-stage simulated annealing and the crossover operator from genetic algorithm. Based on the experiment, 2-stage SA with crossover improved both the solution quality and the computational time, comparing with 2-stage SA without crossover (2-SA). For the future work, the proposed 2-SA-X and the sequence-*k*-tuple representations have potential to be extended to more general 3D packing problems, such as 3D packing with rectilinear boxes.

Acknowledgment

The authors would like to thank Prof. Kunihiro Fujiyoshi at Tokyo University of Agriculture and Technology, to thank Dr. Tayu, Mr. Zhao and Mr. Shinoda at Tokyo Institute of Technology, and to thank Mr. Yamada, Mr. Ando and Mr. Ukon at Osaka University for their discussion, comment and support.

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] J. Varanelli and J. Cohoon, "A two-stage simulated annealing methodology," Fifth Great Lakes Symposium on VLSI, pages 50-53, 1995.
[3] 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.
[4] J. Cong, T. Kong, J. Shinnerl, M. Xie and X. Yuan, "Large Scale Circuit Placement," ACM Transaction on Design Automation of Electronic Systems, vol. 10, no. 2, pages 389-430, 2005.
[5] S. Y. Ho, Y. K. Lin and W.C. Chu, "An orthogonal simulated annealing algorithm for large floorplanning problems," IEEE Transactions on VLSI Systems, vol. 12, no. 8, pages 874-877, 2004.
[6] A. Drakidis, R. J. Mack and R. E. Massara, "Packing-based VLSI module placement using genetic algorithm with sequence-pair representation," Proceedings of IEE on Circuits, Devices and Systems, pages 545-551, 2006.
[7] M. Tang and X. Yao, "A Memetic Algorithm for VLSI Floorplanning," IEEE Transactions on Systems, Man and Cybernetics, vol. 37, no. 1, pages 62-69, 2007.
[8] P. Subbaraj, S. Sankar and S. Anand, "Parallel Genetic Algorithm for VLSI Standard Cell Placement," Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies, pages 70-84, 2009.
[9] F. Mao, N. Xu and Y. Ma, "Hybrid Algorithm for Floorplanning Using B*-tree Representation," Proceedings of Third International Symposium on Intelligent Information Technology Application, pages 228-231, 2009.
[10] Y. Sheng, A. Takahashi and S. Ueno, "RRA-Based Multi-Objective Optimization to Mitigate the Worst Cases of Placement," Proceedings of IEEE 9th International Conference on ASIC, pages 357-360, 2011.
[11] 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.
[12] 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.
[13] P. N. Guo, C. K. Cheng and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," Proceedings of the Design Automation Conference, pages 268-273, 1999.
[14] Y. C. Chang, Y. W. Chang, G. M. Wu and S. W. Wu, "B*-trees: a new representation for non-slicing floorplans," Proceedings of the Design Automation Conference, pages 458-463, 2000.
[15] X. Hong, G. Huang, Y. Cai, S. Dong, C. K. Cheng and J. Gu, "Corner Block List: An effective and efficient topological representation of non-slicing floorplan," Proceedings of the International Conference on Computer Aided Design, pages 8-12, 2000.
[16] 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.
[17] 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
[18] 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.