IPSJ Transactions on System LSI Design Methodology Vol.6 94–100 (Aug. 2013)

[DOI: 10.2197/ipsjtsldm.6.94]

Regular Paper

A New Variation of Adaptive Simulated Annealing for 2D/3D Packing Optimization

Yiqiang Sheng[1,a)] Atsushi Takahashi[1,b)]

Received: November 22, 2012, Revised: March 8, 2013,
Accepted: April 26, 2013, Released: August 5, 2013

Abstract: 2D/3D packing optimization is facing big challenges to get better solution with less runtime. In this paper, we propose a new variation of adaptive simulated annealing (ASA) to solve packing problem. In the traditional ASA, the parameters that control temperature scheduling and random step selection are adjusted according to search progress. In the proposed ASA, a guide with adaptive probabilities is used to automatically select moving methods, including crossover to improve its efficiency. The interesting point is the traditional SA with crossover is inefficient, while the proposed ASA with crossover is efficient due to the adaptive guide. Based on the experiment using MCNC, ami49_X and ami98_3D benchmarks, the computational performance is considerably improved. In the case of area minimization, the results gotten by the proposed ASA are normally better than the published data of 2D packing. In the case of volume minimization for 3D packing, the results gotten by the proposed ASA are better than the data of traditional ASA and SA.

Keywords: layout optimization, adaptive simulated annealing, crossover, 3D packing, 2D packing

1. Introduction

Simulated annealing (SA) [1] is one of the most popular optimization techniques [2], [3], [4], [5], [6], [7], [8] to get near optimal solutions for NP-hard problems. As the solution space keeps increasing in practice, the optimization techniques are facing big challenge to get better solution within a short runtime. Many improved variations of SA, such as adaptive simulated annealing (ASA) [3], are proposed to speed up standard SA. In traditional ASA, the parameters that control temperature scheduling and random step selection are adjusted according to search progress in order to find near optimal solutions more efficiently.

However, there are several shortcomings of traditional ASA and SA to be overcome. For example, the configuration of past solutions is not used even in ASA. It is a big informational waste. The basic idea to improve traditional ASA is to propose an adaptive guide with probabilities to select diverse moving methods, including a special crossover operator to reuse the configuration of past solutions. There are at least three difficulties when SA with crossover. Firstly, SA-based algorithm has only one state at a time, but a crossover needs at least two states at a time. Secondly, the acceptance probability is unstable. Most of states after a normal crossover have very low acceptance probability. Thirdly, since a crossover tends to have a big change each time, it is hard to approach a local optimum, so it is also hard to get effective global convergence. After overcoming all mentioned difficulties, a new variation of ASA is proposed. The traditional SA with crossover is inefficient, while the proposed SA with crossover is efficient due to adaptive guide.

To evaluate the effectiveness of the proposed ASA, we apply it to 2D/3D packing optimization. To solve the problem more efficiently, researches explored many representations, such as BSG [9], sequence pair (SP) [10], O-tree [11], B*-tree [12], CBL [13], FAST-SP [14], Q-sequence [15], Selected SP [16], etc. It has been proved that SP could represent a general 2D topology and there is at least one optimal solution decoding by sequence pair for area minimization, so we are using SP representation to solve 2D packing problem in this paper. The 3D packing problem [17] in 3D layout design is to position different modules into a fixed rectangular box with volume minimization. The SP is extended to sequence quintuple to code and decode 3D packing. It is proved that sequence quintuple could represent the topology of tractable 3D packing and there is at least one sequence quintuple which can be decoded to a topology as an optimal 3D packing for volume minimization.

The contributions of this paper are as the following. As a new variation of SA, adaptive simulated annealing with crossover operator (ASA_X) is proposed to improve the efficiency of simulated annealing by overcoming the shortcomings of traditional SA and ASA. ASA_X is used to improve 2D/3D packing optimization. The experimental results using MCNC, ami49_X and ami98_3D benchmarks show the proposed ASA_X obtains stable improvement of computational performance for both objectives: area and volume. The runtime is reduced considerably. The results of area minimization are normally better than the published data.

[1] Department of Communications and Computer Engineering, Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[a)] sheng@eda.ce.titech.ac.jp
[b)] atsushi@eda.ce.titech.ac.jp

2. Traditional Simulated Annealing

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 neighborhood. The acceptance probability of new solutions depends on the current temperature, which is scheduled from the highest temperature to the lowest temperature. Let $\hat{S}$ be the solution space with neighborhood structure. For any solution S belongs to $\hat{S}$, we define the cost $C(S)$. The flow chart of standard SA is as follows. The initial solution is randomly produced. The parameters of temperature scheduling include the starting temperature T_0, the ending temperature T_e. A solution is updated by using a moving method. The new solution is evaluated by cost function and then it is compared to the old solution. The new solution is accepted with the probability $P = exp[-\Delta C/T]$, which depends on the cost difference ΔC and the current temperature T. If rejected, the current solution goes back to the old one. The terminal condition is reaching the lowest temperature T_e.

A non-optimal solution S is defined by local optimum if it cannot reach better solution by moving to any neighboring solution S′. That is to say, for any neighbor solution (S′) of local optimal solution (S), the inequality $C(S) \le C(S')$ is always satisfied. The depth $D(S)$ of local optimal solution S is defined by $Max[C(S')-C(S)]$. The maximum depth of local optimal solutions in $\hat{S}$ is denoted by d. Let S_i be i^{th} solution explored by simulated annealing, and T_i be the temperature when S_i is explored. Let C_{opt} be the minimum cost. The equality $lim_{i\to\infty} C(S_i) = C_{opt}$ is satisfied [1] with the following conditions: (1) The solution space $\hat{S}$ is finite and irreducible; (2) There exists an equilibrium distribution for the transition probability matrix; (3) $T_i \ge T_{i+1}$ and $T_i > 0$ for all i; (4) $lim_{i\to\infty} T_i = 0$; (5) $\sum_{i:\in(0,\infty)}[exp(-d/T_i)] = \infty$.

There are at least two shortcomings which impact the search speed. (1) Inefficient global search within a short runtime: In order to get a final convergence effectively, the moving methods with small changes should be used, so the global search within a short runtime is limited. It is normally inefficient to solve the problem with huge solution space. (2) Informational waste: SA does not use the experience of past moves and the configuration of past solutions. It is a big informational waste.

3. A New Variation of Simulated Annealing

The basic idea to propose the new variation of adaptive simulated annealing (ASA) is to use an adaptive guide to select moving methods and a special crossover operator as one of moving methods. The proposed ASA_X is an iterative improvement method and a stochastic algorithm. The guide is designed to speed up the search process by increasing the selection probability of moving methods which improve the solutions more in short term. The crossover is designed to reduce the informational waste by using the configuration of past solutions.

The flow chart of the proposed ASA is as shown in **Fig. 1**. The parameters of temperature scheduling are adjusted according to a faster annealing [1], [2]. The new features are as follows. The adaptive guide is added. One of moving methods is selected by the latest guide with adaptive probabilities, which are initially with the same probability for each moving method. By using the selected moving method, the old solution is replaced by a new solution, if improvement happens. The adaptive guide is updated after a given number of trials are applied. If the new solution is better than the current best, the best record is updated. The output is the latest best record.

To realize the idea of guide, a series of probabilities are adjusted automatically to select moving methods. The frequency of improvement $f_k(j)$ of j^{th} moving method in k^{th} iteration loop is defined by

$$f_k(j) = \frac{t_{improve}(j)}{t_k(j)}$$

where $t_k(j)$ is the number of trials using j^{th} moving method in k^{th} iteration loop and $t_{improve}(j)$ is the number of improved trials among $t_k(j)$ trials. The amplitude of improvement $a_k(j)$ of j^{th} moving method in k^{th} iteration loop is defined by

$$a_k(j) = \sum_{i=1}^{t_k(j)} Max\,[0, -\Delta C_k\,(j,i)]$$

where $\Delta C_k(j,i)$ is the difference of cost at i^{th} trial of j-th method in k^{th} iteration loop. The improvement ratio $s_k\,(j)$ is defined by

$$s_k(j) = \frac{a_k(j) f_k(j)}{\sum_{j=0}^{l-1} a_k(j) f_k(j)}$$

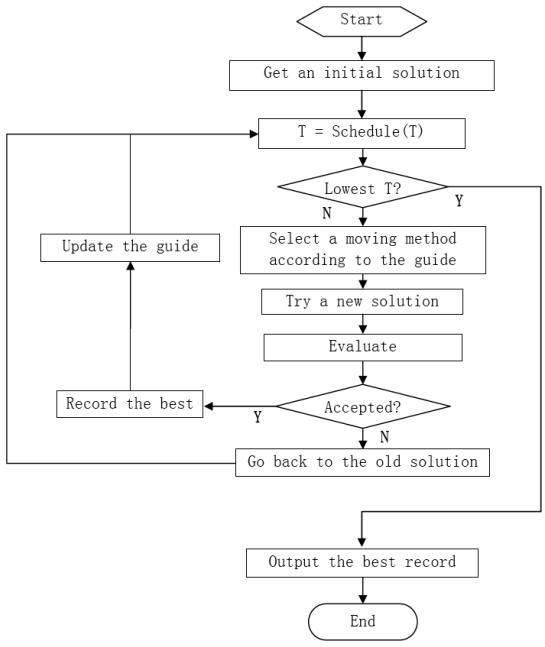

Fig. 1 The flow chart of the proposed algorithm.

where l is the number of moving methods. In this research, there are four different moving methods, so $l = 4$. We only pick up $-\Delta C > 0$ to calculate a_k this time. The guide is not updated after an iteration loop if no improvement happens during the iteration loop. The initial probability p_0 is same for each moving method. The next probability p_{k+1}, which is defined by $(s_k + p_k)/2$, is adapted according to the current probability p_k and the improvement ratio s_k.

To realize the idea of crossover, there are three difficulties to be overcome. Firstly, SA-based algorithm has only one state at a time. We can not use a normal crossover operator like genetic algorithm, which is based on the population of many solutions. A special crossover operator is designed to generate a new solution from the current solution and the best solution so far. The margin and center of the new solution inherit the margin of the current solution and the reversed center of the best solution, respectively. The reason to reverse the best solution is to get a different solution even two given solutions are same. Secondly, the acceptance probability is unstable. Because one of two states is from the best so far, it has more chance to get a better solution by crossover. As a result, the acceptance probability of crossover is improved. Thirdly, it is hard to approach a local optimum by using crossover due to its big move at a time. An adaptive guide to adjust the selection probability of crossover is needed to improve the acceptance probability and the final convergence. We will discuss the crossover further in part V.

In short, there are two main changes for ASA_X: (1) ASA_X changes the probabilities of selecting different moves based on history; (2) ASA_X adds a new crossover operator that reuses best-so-far solutions. There are four moving methods with individual adaptive probability. Each method's probability is updated according to the improvement amplitude and the improvement frequency from the past iterations. The crossover operator generates the new solution by composing the boundary of current solution with the center of best-so-far solution.

4. Application to 2D Packing and 3D Packing

Since 2D packing is a special case of 3D packing, let us start the formulation of 3D case. A general 3D rectangular packing problem is formulated as follows. Let $M = \{m_1, m_2, \ldots, m_n\}$ 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. Let $(x_i, y_i, z_i, r_{xy-i}, r_{yz-i}, r_{zx-i})$ of module m_i be the location and rotation on 3D orthogonal coordinate system, 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. 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 $M = \{m_1, m_2, \ldots, m_n\}$ with height, length and width $\{(h_1, l_1, w_1), (h_2, l_2, w_2), \ldots, (h_n, l_n, w_n)\}$. The constraint is no overlap between m_i and m_j, where $i \ne j$. The output is a set of location and rotation of modules $\{(x_1, y_1, z_1, r_{xy-1}, r_{yz-1}, r_{zx-1}), (x_2, y_2, z_2, r_{xy-2}, r_{yz-2}, r_{zx-2}), \ldots, (x_n, y_n, z_n, r_{xy-n}, r_{yzn}, r_{zx-n})\}$. The objective is to minimize the volume of bounding box.

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 $(A[i], A[i+1], \ldots, A[j])$, where $A = (A[0], A[1], \ldots, A[n-1])$. Let $\Gamma^i = (\Gamma^i[0], \Gamma^i[1], \ldots, \Gamma^i[n-1])$ be a sequence $(1 \le i \le v)$, where v is the number of sequences. Let $F^i(m_j)$ be the order of m_j in sequence 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 $(F^1(m_j), F^2(m_j), \ldots, F^v(m_j))$.

The original 2D/3D packing problem is with infinite solution space. The coding and decoding method is needed to connect the problem and its representation. The solution space of a good representation should be finite. The solutions after a good representation should be feasible and be better to include at least one optimal solution.

Sequence pair (SP) (Γ^1, Γ^2) represents a general 2D topology [10]. Two sequences generate a finite solution space which includes at least one optimal solution of 2D topology for area optimization by decoding. It is regarded as a set of the relations of relative location between modules, i.e., "North-South" and "West-East" (NS- and WE-) relations. Let $(m_i \text{ N } m_j)$ and $(m_i \text{ W } m_j)$ denote NS- and WE-relations. SP defines $(m_i \text{ W } m_j)$ when

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

It defines $(m_i \text{ N } m_j)$ when

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

The rule of symmetry to be followed is that $(m_i \text{ N } m_j)$ is the same relation as $(m_j \text{ S } m_i)$. That is to say, the topology should be reversely decoded if the order of labeling is reversed. 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 $(n!)^2 2^n$.

SP representation for 2D packing is extended to 3D packing [17] as sequence triple (ST) representation, which consists of three sequences Γ^1, Γ^2 and Γ^3, and sequence quintuple (SQ) representation, which consists of five sequences Γ^1, Γ^2, Γ^3, Γ^4 and Γ^5. The ST and SQ representation define 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. For example, 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.

ST consists of three sequences Γ^1, Γ^2 and Γ^3. The coding and the decoding are based on TB-, NS- and WE- relation corresponding to the order of modules in ST. 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 $(n!)^3 6^n$.

Three or even four sequences are not enough to represent a general 3D packing topology. It means that ST does not cover all kinds of topology of 3D packing. That is to say, the topology

with the minimum volume might not be covered. For example, the packing in **Fig. 2** cannot be represented by ST. SQ generates a finite solution space which includes at least one optimal solution of 3D packing for volume minimization by decoding. For example, the packing in Fig. 2 is represented by SQ where $\Gamma^1 = (m_1, m_5, m_6, m_2, m_4, m_3)$, $\Gamma^2 = (m_6, m_5, m_2, m_3, m_4, m_1)$, $\Gamma^3 = (m_2, m_4, m_5, m_1, m_3, m_6)$, $\Gamma^4 = (m_5, m_4, m_1, m_6, m_3, m_2)$ 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 SQ. 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 $(n!)^5 6^n$.

Three moving methods including rotation, exchange and move are defined as follows. Firstly, the rotation changes the orientation of a module. When a rotation is applied to module m_i, r is changed to $1 - r$, where r is one of $(r_{xy-i}, r_{yz-i}, r_{zx-i})$. Secondly, the exchange moving method exchanges the order of two modules in all sequences, e.g. in sequence pair (Γ^1, Γ^2), $F_1(m_i)$, $F_2(m_i)$, $F_1(m_j)$, and $F_2(m_j)$ are changed to $F_1(m_j)$, $F_2(m_j)$, $F_1(m_i)$, and $F_2(m_i)$, respectively. Thirdly, the move changes the order of a module in one sequence Γ^i. When a move is applied to module m in Γ^i, $F^i(m)$ is changed to another value, say j, and the orders of modules whose order is between $F^i(m)$ and j are shifted accordingly. For example, if the operation is to move m_5 to the first position in Γ^- in **Fig. 3**, the move leads to $F_-(m_1) = F_-(m_1) + 1$ and $F_-(m_2) = F_-(m_2) + 1$.

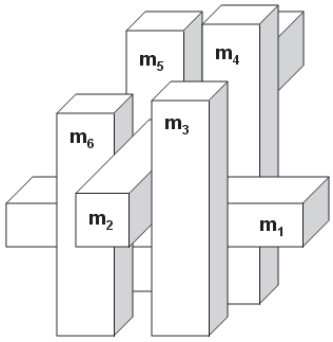

Fig. 2 A 3D packing topology that cannot be represented by sequence triple.

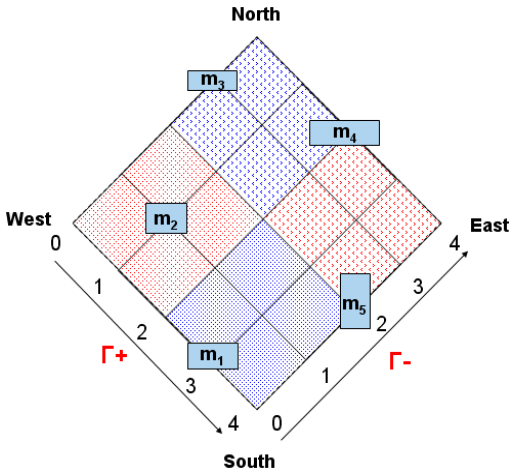

Fig. 3 An example of two sequences Γ^+ and Γ^-, which are selected randomly from $(\Gamma^1, \Gamma^2, \Gamma^3)$ or $(\Gamma^1, \Gamma^2, \Gamma^3, \Gamma^4, \Gamma^5)$.

5. Crossover Operator

A special crossover operator is designed as follows. Two sequences are derived from the current solution and the best solution, and the remaining sequences are derived from the current solution. Two sequences (Γ^+, Γ^-) out of three sequences $(\Gamma^1, \Gamma^2, \Gamma^3)$ or five sequences $(\Gamma^1, \Gamma^2, \Gamma^3, \Gamma^4, \Gamma^5)$ are picked up randomly for 3D as shown in Fig. 3. 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 $n/2 - 1$, where n is the number of modules. The child of SP (Γ_c^+, Γ_c^-) is given by $\Gamma_f^+[0, i] + \Gamma_m'^+ + \Gamma_f^+[n - i - 1, n - 1]$ and $\Gamma_f^-[0, i] + \Gamma_m'^- + \Gamma_f^-[n - i - 1, k - 1]$, where $\Gamma_m'^+$ and $\Gamma_m'^-$ are the inverse of $\Gamma_m^+ - \Gamma_f^+[0, i] - \Gamma_f^+[n - i - 1, n - 1]$ and the inverse of $\Gamma_m^- - \Gamma_f^-[0, i] - \Gamma_f^-[n - i - 1, k - 1]$, respectively.

To make it clearer, let us take an example to explain the crossover. As shown in **Fig. 4**, 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. 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. Assume that $i = 1$. Then, the child is the layout represented by $\Gamma^+ = (m_3, m_2, m_4, m_1, m_5)$ and $\Gamma^- = (m_5, m_2, m_1, m_3, m_4)$ as the right layout of **Fig. 5**, 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 Fig. 5, 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 center of right picture of Fig. 5.

6. Experiment and Comparison

A set of experiments was implemented by using the proposed ASA_X, in comparison to traditional SA and ASA. We are using MCNC, ami49_X and ami98_3D benchmarks. The ami49_X is produced by duplicating ami49 circuit X times. The ami98_3D is produced by inheriting the height and width of ami49_2, which is

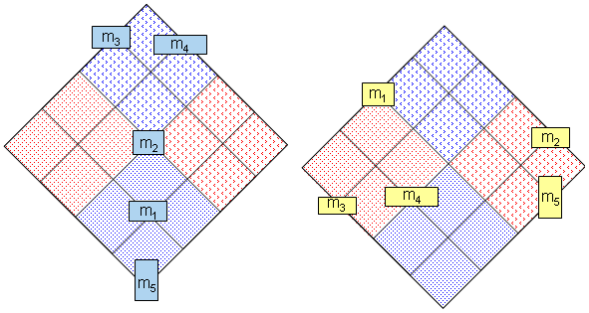

Fig. 4 Crossover operator as a moving method: the current solution and the best-so-far solution (Father and Mother).

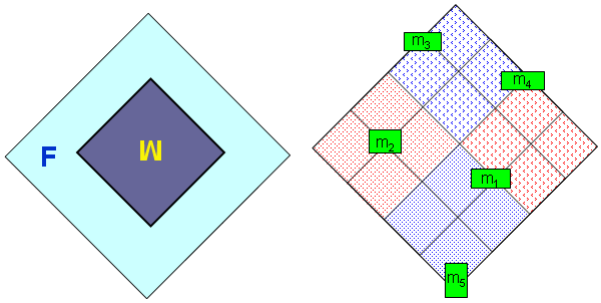

Fig. 5 Crossover as a moving method: the next solution (Child).

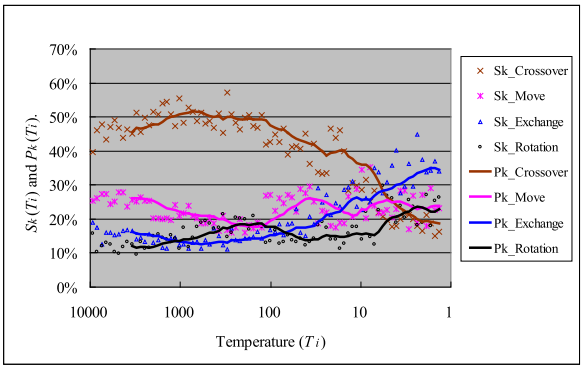

Fig. 6 The improvement ratio S_k and the selection ratio P_k according to temperature scheduling.

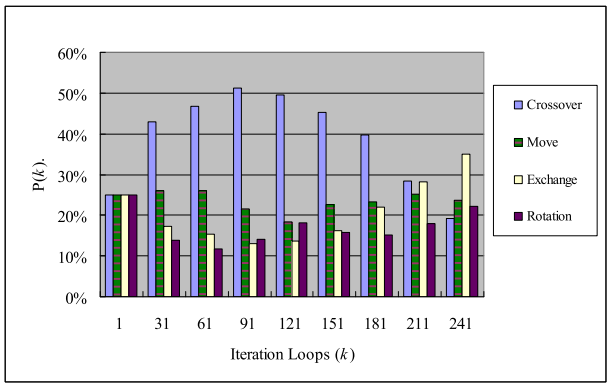

Fig. 7 Guide with probabilities $P(k)$ to select each moving method according to the improvement ratio.

produced by duplicating ami49 circuit twice, and randomly getting the length between the given minimum and maximum dimensions.

For fair comparison, all algorithms are implemented in Python environment on 2.16GHz PC with 3.00GB memory. A fast annealing of temperature scheduling decreasing exponentially according to $T_i = T_0 \exp(-ci^{1/D})$ [1], [2] is used in this paper, where c and D are two constant parameters and T_0 is the initial temperature. This annealing schedule is faster than fast Cauchy annealing, where $T_i = T_0/i$, and much faster than Boltzmann annealing, where $T_i = T_0 \ln i$. The temperature scheduling is often simplified to an easy-to-calculate version, that is, $T_{i+1} = aT_i$ $(0 < a < 1)$, where T_i is the temperature at i^{th} loop which contains p trials. a is the temperature coefficient between 0 and 1, and a is normally near 1.

To confirm the effectiveness of guide and crossover during the whole search process, the improvement ratio s_k of each moving method according to temperature (T from T_0 to T_e) is gotten by the experiment using ami98_3D, as shown in **Fig. 6**. In experiments, $a = 0.99$, $p = 1000$. Based on the experiment, it is confirmed that the improvement ratio s_k of crossover is normally bigger than that of any other moving method, when the temperature is large enough. During the initial stage, the crossover is the most efficient moving method, but it is not efficient at the final stage.

The selection ratio p_k is a relative value, which is calculated by improvement ratio and the previous selection ratio, and the total probability 100% is maintained. The initial probability p_0 is set to 25%. The number of trials in each iteration loop is $t = 300$. The selection ratio (p_k) is shown in **Fig. 7**.

For 3D packing optimization, the average results of 50 trials are gotten. All experiments are implemented within 4,000s each time using ami98_3D benchmark. We test ASA with and without crossover as shown **Table 1**. When we are using the crossover operator, the average improvement of 3D packing using SQ representation is near 9.6% with runtime from 20 s to 4,000 s. In short, both the guide and the crossover improve the efficiency of search process.

For the detail of 2D packing, as shown in **Table 2**, the best, average and worst cases of area minimization among 50 trials are gotten within 10 minutes each time using MCNC and ami49_X benchmark by SP representation. In the best case of area minimization, the results gotten by ASA_X are normally better than the published data. The comparison of solution and runtime between ASA and ASA_X on average is as shown in **Table 3**. The ASA_X reduced near 20% runtime with better solution. A near log-linear trend of average improvement rates from ASA to ASA_X is gotten. That means ASA_X should be more suitable for the packing problem with a larger number of modules.

Table 1 Performance comparison of 3D packing by adaptive simulated annealing with and without crossover.

Runtime (s)	ASA	ASA_X	Improvement (%)
20	178.1%	165.7%	12.5%
40	169.0%	148.6%	20.4%
200	151.2%	140.2%	11.0%
400	142.3%	136.5%	5.9%
2000	132.8%	129.3%	3.5%
4000	123.9%	119.6%	4.3%

Table 2 Area optimization by ASA_X for 2D packing with SP representation.

Benchmarks	Best (mm^2)	Average (mm^2)	Worst (mm^2)	Runtime (s)
apte	46.92	47.33	47.61	2.9
xerox	19.8	20.48	21.19	1.1
hp	8.95	9.17	9.31	1.5
ami33	1.18	1.23	1.28	15
ami49	36.67	37.68	38.69	29
ami49_2	73.01	75.17	77.12	121
ami49_4	146.4	150.3	155.1	519

Table 3 Average improvement of 2D packing with SP representation.

Bench-marks	Solution(mm^2)		Runtime (s)		Improvement (%)	
	SA	ASA_X	SA	ASA_X	Solution	Runtime
apte	47.38	47.33	3.6	2.9	0.11%	19%
xerox	20.51	20.48	2.0	1.1	0.16%	45%
hp	9.18	9.17	2.8	1.5	0.11%	46%
ami33	1.24	1.23	19	15	0.86%	21%
ami49	37.96	37.68	42	29	0.79%	31%
ami49_2	75.98	75.17	189	121	1.15%	36%
ami49_4	152.2	150.3	712	519	1.37%	27%

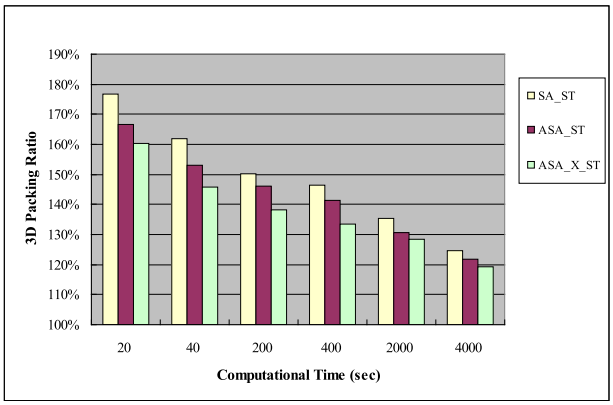

Fig. 8 Comparison of computational performance among SA, ASA and ASA_X using ST representation.

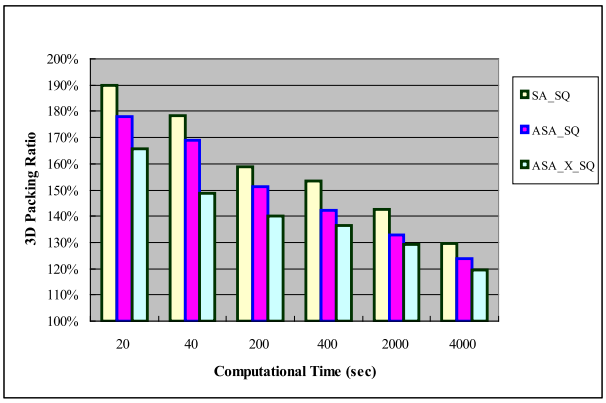

Fig. 9 Comparison of computational performance among SA, ASA and ASA_X using SQ representation.

Figure 8 shows the comparison of volume minimization using ST representation. The proposed ASA_X outperforms SA with the improvement of 3D packing ratio between 5% and 16%. The average improvement from SA to ASA is near 12%. The ASA_X outperforms ASA with the improvement between 2% and 8%. The average improvement of volume minimization from ASA to ASA_X is near 6%.

As shown in **Fig. 9**, it is the comparison of volume minimization using SQ representation. The proposed ASA_X outperforms SA with the improvement of 3D packing ratio between 10% and 30%. The average improvement from SA to ASA is near 19%. The ASA_X outperforms ASA with the improvement between 4% and 20%. The average improvement of volume minimization from ASA to ASA_X is near 10%.

7. Conclusion

In this paper, a new variation of adaptive simulated annealing with crossover (ASA_X) is proposed to solve 2D/3D packing problem. A guide with adaptive probabilities is used to automatically select moving methods. A special crossover is designed to use the information of past solutions and get high improving efficiency. Based on the experimental results, the proposed ASA_X obtains stable improvement for two objectives: area and volume. The proposed ASA_X reduced computational runtime with the better solution for area and volume minimization. The proposed ASA_X has potential to improve more NP-hard problems, such as 3D packing with rectilinear boxes, effectively.

References

[1] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: Optimization by Simulated Annealing, *Science*, Vol.220, No.4598, pp.671–680 (1983).
[2] Varanelli, J. and Cohoon, J.: A two-stage simulated annealing methodology, *Fifth Great Lakes Symposium on VLSI*, pp.50–53 (1995).
[3] Stefankovic, D.: Adaptive Simulated Annealing: A Near-optimal Connection between Sampling and Counting, *The 48th Annual IEEE Symposium on Foundations of Computer Science*, pp.183–193 (2007).
[4] Ho, S.Y., Lin, Y.K. and Chu, W.C.: An orthogonal simulated annealing algorithm for large floorplanning problems, *IEEE Trans. VLSI Systems*, Vol.12, No.8, pp.874–877 (2004).
[5] Tang, M. and Yao, X.: A Memetic Algorithm for VLSI Floorplanning, *IEEE Trans. Systems, Man and Cybernetics*, Vol.37, No.1, pp.62–69 (2007).
[6] Drakidis, A., Mack, R.J. and Massara, R.E.: Packing-based VLSI module placement using genetic algorithm with sequence-pair representation, *IEE Proceedings on Circuits, Devices and Systems*, pp.545–551 (2006).
[7] Sheng, Y., Takahashi, A. and Ueno, S.: A Stochastic Optimization Method to Solve General Placement Problem Effectively, *Proc. Information Processing Society of Japan* (*IPSJ*) *DA Symposium*, Vol.2011, No.5, pp.27–32 (2011).
[8] Sheng, Y., Takahashi, A. and Ueno, S.: 2-Stage Simulated Annealing with Crossover Operator for 3D-Packing Volume Minimization, *Proc. 17th Workshop on Synthesis And System Integration of Mixed Information technologies* (*SASIMI*), pp.227–232 (2012).
[9] Nakatake, S., Fujiyoshi, K., Murata, H. and Kajitani, Y.: Module placement on BSG-structure and IC layout applications, *Proc. International Conference on CAD*, pp.484–491 (1996).
[10] Murata, H., Fujiyoshi, K., Nakatake, S. and Kajitani, Y.: VLSI module placement based on rectangle-packing by the sequence-pair, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.15, No.12, pp.1518–1524 (1996).
[11] Guo, P.N., Cheng, C.K. and Yoshimura, T.: An O-tree representation of non-slicing floorplan and its applications, *Proc. Design Automation Conference*, pp.268–273 (1999).
[12] Chang, Y.C., Chang, Y.W., Wu G.M. and Wu, S.W.: B*-trees: A new representation for non-slicing floorplans, *Proc. Design Automation Conference*, pp.458–463 (2000).
[13] Hong, X., Huang, G., Cai, Y., Dong, S., Cheng, C.K. and Gu, J.: Corner Block List: An effective and efficient topological representation of non-slicing floorplan, *Proc. International Conference on Computer Aided Design*, pp.8–12 (2000).
[14] Tang, X. and Wong, D.F.: FAST-SP: A fast algorithm for block placement based on sequence pair, *Proc. IEEE Asia South Pacific Design Automation Conference*, pp.521–526 (2001).
[15] Zhuang, C., Sakanushi, K., Jin, L. and Kajitani, Y.: An enhanced Q-sequence augmented with empty-room-insertion and parenthesis trees, *Proc. Design, Automation and Test in Europe Conference and Exhibition*, pp.61–68 (2002)
[16] Kodama, C. and Fujiyoshi, K.: Selected sequence-pair: An efficient decodable packing representation in linear time using sequence-pair, *Proc. IEEE Asia South Pacific Design Automation Conference*, pp.331–337 (2003).
[17] Yamazaki, H., Sakanushi, K., Nakatake, S. and Kajitani, Y.: The 3D-Packing by Meta Data Structure and Packing Heuristics, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E83-A, No.4, pp.639–645 (2000).

Yiqiang Sheng received his M.E. degree from Nankai University, Tianjin, China, in 2003. He worked at Sanyo Electric Co., Ltd. in Osaka, Japan, from 2003 to 2008 and at Nokia Corporation in Tokyo, Japan, from 2008 to 2010. He did research at Osaka University, Osaka, Japan, from 2010 to 2011. He is currently a Ph.D. candidate at Tokyo Institute of Technology since 2010. His current research is focusing on heuristics for VLSI/PCB physical design optimization.

Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He had been with Tokyo Institute of Technology as a research associate from 1991 to 1997, and as an associate professor from 1997 to 2009, and from 2012. He had been with Osaka University as an associate professor from 2009 to 2012. He visited University of California, Los Angeles, U.S.A., as a visiting scholar from 2001 to 2002. He is currently with Department of Communications and Computer Engineering, Graduate School of Science and Engineering, Tokyo Institute of Technology, as an associate professor since 2013. His research interests are in VLSI layout design and combinational algorithms. He is a member of IEEE, IEICE, and IPSJ.

(Recommended by Associate Editor: *Tetsushi Koide*)